

Claudia P. Ayala Martínez

Systematic Construction Of Goal-Oriented COTS Taxonomies

Doctoral Thesis

In partial satisfaction of the requirements for the
Degree of PhD in Software

Advisor: Dr. Xavier Franch Gutiérrez

Barcelona, Spain. February 2008



Departament de Llenguatges i Sistemes Informàtics
Universitat Politècnica de Catalunya

To my family
with love and gratitude

Abstract

The process of building software systems by assembling and integrating pre-packaged solutions in the form of Commercial-Off-The-Shelf (COTS) software components - hereafter COTS- has become a strategic need in a wide variety of application areas. In general, COTS are software that provide a specific functionality, available in the market to be purchased, interfaced and integrated into other software systems. Nowadays there is an increasingly huge marketplace of COTS; therefore, one of the most critical activities in COTS-based development is the selection of the components to be integrated into the system under development. Selection is basically composed of three activities: searching of candidates from the marketplace, evaluating them with respect to the system requirements and deciding the COTS to be used. Most of the different existing methods for COTS selection focus their efforts on evaluating and deciding, letting aside the problem of searching components in the marketplace. This lack of proposals is a serious drawback that makes the whole selection process highly risky, and often expensive and inefficient.

This dissertation introduces the GOTHIC (**Goal-Oriented Taxonomy and reuse Infrastructure Construction**) method that provides and integrated strategy for facilitating COTS components searching and reuse. It has been elaborated following an iterative process based on action-research premises to identify the actual challenges related to COTS searching. Then, possible solutions were envisaged and implemented by several industrial and academic case studies in different domains. Successful results were recorded to articulate the synergic GOTHIC method solution. Thus, the research in this thesis contains seven main contributions integrated in the GOTHIC method as a whole.

C1. Better understanding of challenges for finding COTS. An explorative survey of problems and issues related with COTS selection both in industrial practice and literature was performed stating a set of critical challenges to select COTS.

C2. A goal-oriented strategy for dealing with COTS marketplace evolvability. Case studies demonstrated the feasibility of using goal-oriented approaches in this context. We deal with their adaptation and provide prescriptive support in their use.

C3. Provision of a well-defined process for constructing goal-oriented taxonomies. It ensures the completeness and correctness of taxonomies, as well as their management and evolution as a result of COTS marketplace progress and/or different users' needs.

C4. A systematic approach for dealing with the unstructured, incomplete and vast amount of COTS related information. It supports COTS selectors to decide what information sources to use according to their specific quality project needs.

C5. A COTS domain analysis strategy for recording the informational dimensions required to select COTS. Identification of required dimensions and suitable models for supporting the COTS domain reuse infrastructure creation.

C6. Identification of reusable artifacts throughout the GOTHIC method to build a reusable knowledge base. In addition, a specific strategy for populating the COTS knowledge base was explored and a software prototype was implemented.

C7. Preliminary insights of the GOTHIC method feasibility and effectiveness in industrial settings. Preliminary empirical results have been obtained from some Norwegian companies and academia.

Acknowledgments

I would like to express my sincere gratitude to all who have made possible the achievement of this goal. Throughout its accomplishment I have learnt a lot not only as a professional and researcher but also as a person.

First of all, I would like to thank my advisor Dr. Xavier Franch for the trust he placed on me and the time, patience, guidance and invaluable advises he has given me. His engagement and knowledge have inspired me a lot. Thank you Xavi!

To all members of the GESSI group for sharing their work with me; especially to Professor Pere Botella for giving me the opportunity to form part of the group and sharing his knowledge with me.

To Professor Reidar Conradi from the Norwegian University of Science and Technology (NTNU), for giving me the opportunity to learn the working philosophy of his group. To all members of the Software Engineering group at NTNU for their kind hospitality and cooperation during my research stay there, and for impressing upon me the importance of empirical research as a fundamental part of the Software Engineering field, mainly to Professor Reidar Conradi, Dr. Carl-Fredrik Sørensen and Dr. Jingyue Li for their invaluable feedback and advises.

Thanks to all the reviewers that have contributed with their comments in different stages of this work. To the researchers and students I have collaborated with throughout this thesis. I acknowledge their invaluable contribution.

I devote this project to my family, for supporting me in the tough and happy moments no matter the distance. For their love and inspiration; for teaching me that no goal is too lofty given perseverance, hard work and dedication; for their example, unity, courage and strength of character. Especially to my grandmother Sara and my mother Pilar for always encouraging me to follow my goals and instilling me in the values that have made me the person I am today.

Special thanks to my Spanish family (my family in-law) for their love and support; for always looking after me and make me feel as at home. To my husband Carlos, for his love and continuous support, and for being my best friend and companion along the same path.

To the people, colleagues and friends who have given me a hand whenever I needed.

This work has been possible thanks to the Mexican Research Council (Consejo Nacional de Ciencia y Tecnología - CONACyT) and the Agència de Gestió d'Ajuts Universitaris i de Recerca (European Social Fund). It was performed in the context of the Spanish MEC TIN2004-07461-C02-01 project.

Barcelona, Spain; January 2008

Claudia P. Ayala Martínez

Vita

Claudia P. Ayala Martínez was born on August 6, 1978, in Tuxtla Gutiérrez, Chiapas, México.

In February, 2001 she graduated as a Computer Science Engineer from the Technical Institute of Tuxtla Gutiérrez (Instituto Tecnológico de Tuxtla Gutiérrez, Chiapas; México -ITTG-).

In 2001, she was awarded as one of the best Engineering graduated students in Mexico, and also as the best graduated student from ITTG, the highest honour a student can receive at ITTG.

In 2001 she entered to the Technological Research Formative program of the Electrical Research Institute (Instituto de Investigaciones Eléctricas -IIE-) in Cuernavaca, Morelos; México and worked as a researcher in the same institution.

At the end of 2002 she was selected for a grant by the Mexican Research Council (CONACyT) and entered to the doctoral program at the Technical University of Catalunya (UPC). In 2005 she received the Advanced Studies Diploma (Diploma de Estudios Avanzados -DEA-).

During doctoral studies she was also the recipient of some competitive grants including the International Research Support from the Generalitat de Catalunya (European Social Fund), which enabled her to make a research stay at the Norwegian University of Science and Technology (NTNU) in Trondheim, Norway.

Since the end of 2003 she belongs to the GESSI (Software Engineering for Information Systems) research group from the Technical University of Catalunya (UPC) and has participated in COTS related projects.

Contents

Abstract	i
Acknowledgments	iii
Vita	v
List of Figures	xi
List of Tables	xiii
Chapter 1. Introduction	1
1.1 The Problem: Searching COTS in the Marketplace	2
1.2 Importance of the Problem	4
1.3 Research Goal	5
1.3.1 Additional Objectives	6
1.4 Research Context	7
1.5 Methodological Approach	9
1.6 Publications in Relation to this Thesis	14
1.7 Contributions	17
1.8 Organization of the Document	19
Chapter 2. Related Work	23
2.1 Introduction	23
2.1.1 COTS Definition	25
2.2 State-of-the-art	27
2.2.1 COTS Selection Approaches	27
2.2.1.1 Analyzing COTS Selection Approaches	32
2.2.2 Software Reuse Infrastructures and Knowledge Bases	34
2.2.2.1 Classification and its Central Role in Implementing Reuse Infrastructures	35
2.2.2.2 Kinds of Software Classification Schemas	37
2.2.3 COTS Classification Approaches	40
2.2.3.1 Analyzing COTS Classification Approaches	42
2.2.4 COTS Related Search Engines	44
2.2.4.1 Analyzing COTS Related Search Engines	45
2.3 State-Of-The-Practice	47
2.3.1 COTS Location and Reuse	47
2.3.2 Available COTS Catalogues and Repositories	48
2.4 Relevant Approaches Supporting the Solution Addressed by this Thesis	51
2.4.1 Goal-Oriented Approaches	51
2.4.1.1 Goal-Based Requirements Analysis Method (GBRAM)	53
2.4.1.2 The <i>i*</i> Goal-Modeling Approach	55
2.4.2 Software Quality	57

2.4.2.1 Software Quality Models	58
2.4.2.2 COTS Evaluation by Quality Models	61
2.4.2.3 ISO/IEC 9126-1 Software Quality Standard.....	62
Chapter 3. Research Method.....	67
3.1 Research Design	68
3.2 Formative Case Studies	70
3.2.1 Business Applications (BA).....	71
3.2.2 Software Applications Development (SAD) and Requirements Engineering Support Tools (REST).....	71
3.2.3 Real Time Synchronous Communication Tools (RTSC)	72
3.3 Formative Research Stages Used to Develop the GOTHIC Method	72
3.3.1 First Stage: Proposal of an Initial Version of the Method	72
3.3.2 Second Stage: Validation and Improvement of the Resulting Method in Academic Cases.....	74
3.3.3 Third Stage: Improvement of the Method to support suitable reuse and Empirical Evaluation	77
3.4 Answers to Research Questions.....	81
3.5 Threats of Validity Discussion	82
Chapter 4. The GOTHIC Method	85
4.1 GOTHIC: A Method to Build a COTS Domain Reuse Infrastructure Based on Goal-Oriented Taxonomies	86
4.2 Characteristics of the Proposal	89
4.3 Intended Improvements	90
4.3.1 Improvement on the Effectiveness of COTS Marketplace Organization	90
4.3.2 Improvement on Managing COTS Marketplace Characteristics.....	91
4.3.3 Improvement on COTS Information Rendering	92
4.3.4 Improvement on Managing and Reusing COTS Related Issues.....	93
4.4 Intended Audience	93
4.5 Applicability of the Proposal	93
Chapter 5. Exploration of Information Sources.....	97
5.1 Background.....	98
5.2 Capturing Information Quality Dimensions for COTS Selection	99
5.2.1 Identifying COTS Selection Information Problems	100
5.2.2 Determining IQ in the COTS Selection Context	101
5.2.3 Determining a Measurable Framework for Assessing IQ in the COTS Selection Context.....	104
5.3 A Systematic Approach for Managing and Reusing COTS Information Sources	108
5.3.1 Heuristics to Support IQ Assessment in the COTS Selection Context.....	109
5.3.2 A Conceptual Model for Systematically Supporting COTS Selectors Decision-Making	109

5.4 Summary and Discussion	111
Chapter 6. COTS Domain Analysis	113
6.1 Background.....	114
6.2 Domain Analysis for Supporting COTS Selection: Dimensions.....	115
6.3 Domain Analysis for Supporting COTS Selection: Models.....	117
6.4 A Unified Model for COTS Domains	121
6.4.1 Integrating all the COTS domain models into the ISO/IEC 9126-1	121
6.4.2 Transforming the Models into the ISO/IEC 9126-1 Framework.....	123
6.5 Domain Analysis-Based COTS Selection	124
6.6 Case Study Example	126
6.7 Summary and Discussion	130
Chapter 7. Goal-Oriented Core of GOTHIC.....	133
7.1 Background.....	134
7.2 Activity 3: Identification, Refinement and Statement of Goals	134
7.2.1 Goal Identification and Refinement.....	135
7.2.1.1 Supporting Mechanisms, Techniques and Models	135
7.2.2 Statement of Goals.....	138
7.3 Activity 4: Establishment of Dependencies.....	138
7.3.1 Use of i^* SD models	139
7.3.2 Identifying COTS dependencies.....	139
7.4 Activity 5: Goal Taxonomy Structuring	149
7.5 Summary and Discussion	153
Chapter 8. Goal Taxonomy Validation and Management.....	155
8.1 Predicates and Functions	155
8.1.1 Conditions Over Taxonomies	156
8.2 Four-Step Process for Transformation Rules Application.....	157
8.2.1 Transformation Rules in Step 1	158
8.2.1.1 Heuristics, Combined Rules and Stop Condition for Step 1.....	160
8.2.2 Transformation Rules in Step 2	161
8.2.2.1 Heuristics, Combined Rules and Stop Condition for Step 2.....	164
8.2.3 Transformation Rules in Step 3	165
8.2.3.1 Heuristics, Combined Rules and Stop Condition for Step 3.....	166
8.2.4 Transformation Rules in Step 4	167
8.3 Goal-Oriented Taxonomies Formulation, Evaluation and Management.....	169
8.4 Applying the Goal-Oriented Taxonomy Validation and Management Process ...	169
8.4.1 Validating and Manipulating the Goal-Oriented Taxonomy Obtained for the RTSC Case Study	170
8.4.2 Validating and Manipulating an Existing COTS Classification Schema	171
8.5 Summary and Discussion	175

Chapter 9. Knowledge Base Management	177
9.1 The Experience Factory (EF) and Learning Software Organization (LSO)	
Paradigms	178
9.2 The GOTHIC's Knowledge Base as an EF + LSO	180
9.3 The Context of Use of the GOTHIC's Knowledge Base	181
9.4 GOTHIC's Knowledge Base Population and Maintenance	182
9.5 Software Tools Supporting Some GOTHIC's Activities and their Deliverables..	185
9.5.1 Existing Software Tools Supporting GOTHIC's Activities	186
9.5.1.1 DesCOTS System	186
9.5.1.2 REDEPEND-REACT Tool.....	189
9.5.2 New Software Tools Developed for Supporting GOTHIC's Activities	190
9.5.2.1 OTS-Wiki Prototype	190
9.5.2.2 Information Quality (IQ) Tool	194
9.6 Summary and Discussion	195
Chapter 10. Method Evaluation	197
10.1 Preliminary Industrial Evaluation of GOTHIC	198
10.1.1 Short/Medium Term Empirical Evaluation	199
10.1.1.1 Academic Seminar	200
10.1.1.2 Explorative Survey.....	201
10.1.1.3 Industrial Seminar	203
10.1.2 Short/Medium Term Qualitative Results	205
10.2 Limitations of the GOTHIC Method.....	206
10.3 The comparative of GOTHIC with similar approaches.....	206
Chapter 11. Conclusions & Future Work	209
11.1 Contributions of the Approach	209
11.1.1 Reliability and Effectiveness related Contributions	210
11.1.2 Reusability related Contributions	211
11.1.3 Contributions in collaboration with other members of the GESSI group....	212
11.2 Future Work	213
11.2.1 Major Future Research Lines.....	213
11.2.2 A Multidisciplinary Intended Project	214
List of Abbreviations	217
Glossary	219
References.....	221
Annex 1. Heuristics Supporting GOTHIC Activities	245
Annex 2. IQ COTS Reference Model	251
Annex 3. Domain Model for the RTSC Case Study	259

List of Figures

Fig. 1.1	Relationship among thesis chapters, GOTHIC activities and publications	21
Fig. 2.1	Evolution of COTS selection practices [Moh+07]	28
Fig. 2.2	Components reuse environment.....	34
Fig. 2.3	Existing kinds of software classification techniques	38
Fig. 2.4	Overview of GBRAM activities	54
Fig. 2.5	Excerpt of an <i>i*</i> model for an academic tutoring system.....	56
Fig. 2.6	Conceptual model of the ISO/IEC 9126-1 standard	64
Fig. 3.1	Research phases, research questions and corresponding studies	70
Fig. 3.2	Research phases and their corresponding publications.....	80
Fig. 4.1	High-level activities of the GOTHIC method.....	86
Fig. 4.2	Conceptual model for goal-oriented COTS taxonomies: overview.....	90
Fig. 4.3	Overview of the COTS marketplace structuring.....	94
Fig. 5.1	An excerpt of the COTS IQ Reference model	110
Fig. 6.1	Overview of the ISO/IEC 9126-1-based quality model for COTS.....	121
Fig. 6.2	Conceptual model excerpt for the GOTHIC domain model.....	125
Fig. 6.3	Excerpt of some domain models constructed for the RTSC case	126
Fig. 6.4	Some dependencies among RTSC Tools and other types of tools	127
Fig. 7.1	<i>i*</i> SD model progressively constructed to assess the RTSC domain.....	140
Fig. 7.2	Example of the refinement and establishment of COTS dependencies	141
Fig. 7.3	Overview of dependencies identifying among actors.....	143
Fig. 7.4	Example of discovering other environmental actors	145
Fig. 7.5	Overview of the <i>i*</i> SD model obtained.....	147
Fig. 7.6	Partial View of the goal-oriented hierarchy for the RTSC case	153
Fig. 8.1	Set of Transformation Rules	159
Fig. 8.2	Heuristics driving transformation rules application in Step 1	160
Fig. 8.3	Heuristics driving transformation rules in Step 2	164
Fig. 8.4	Heuristics driving transformation rules application in Step 3	166
Fig. 8.5	Heuristics driving transformation rules application in Step 4	168
Fig. 8.6	Conceptual model of the goal-oriented core of GOTHIC	169
Fig. 8.7	The 4-Step goal-taxonomy validation process for the RTSC case	171
Fig. 8.8	An excerpt of the BA Gartner classification.....	172
Fig. 8.9	The 4-step goal-taxonomy manipulation process for the BA case	174
Fig. 9.1	The Experience Factory paradigm	179
Fig. 9.2	GOTHIC experience base and knowledge reuse artifacts	181
Fig. 9.3	OTS-Wiki Portal Main Interactions	183

Fig. 9.4 <i>i*</i> SD model summarizing the GOTHIC population strategy.....	185
Fig. 9.5 Overview of the DesCOTS System	187
Fig. 9.6 Snapshot of a business application taxonomy with Taxonomy Tool.....	188
Fig. 9.7 QM snapshot: defining a method.....	189
Fig. 9.8 REDEPEND-REACT snapshot: constructing an <i>i*</i> SD model	190
Fig. 9.9 Goal-based Scenarios designed to reach the OTS-Wiki High-Level goals	191
Fig. 9.10 Scenario excerpts for enabling OTS-Wiki high-level goals	192
Fig. 9.11 A snapshot of the OTS-Wiki prototype	193
Fig. 9.12 A snapshot of the Information Quality Tool.....	195

List of Tables

Table 1.1	Research Questions of this Research	9
Table 1.2	Shaw’s characterization of Research Questions related to Research Questions in this Thesis	10
Table 1.3	Shaw’s characterization of Software Engineering Research Results	11
Table 1.4	Characterization of High-Level Research Results of this Thesis	11
Table 1.5	Shaw’s characterization of Software Engineering Research Validation	12
Table 1.6	Main Validation Efforts Pursued in this Thesis	13
Table 1.7	High-level contributions of this thesis, associated publications and chapters	19
Table 2.1	The Fundamental change among custom-development and CBSD	24
Table 2.2	Different definitions of COTS, surveyed by [Mor-Tor02]	25
Table 2.3	Activities and Roles in COTS Selection	27
Table 2.4	Comparing some representative methods dealing with COTS selection	32
Table 2.5	Relevant drawbacks of current reusable software classification schemas	40
Table 2.6	Summary of some COTS classification approaches	42
Table 2.7	Summary of some COTS Related Search Engines	45
Table 2.8	Summary of organizations types related to COTS selection activities	48
Table 2.9	Some representative COTS related repositories and catalogues	49
Table 2.10	Assessment of the role-related challenges for supporting COTS Selection	51
Table 2.11	Some relevant goal-oriented approaches as surveyed in [Kav-Lou05]	53
Table 2.12	Some relevant quality models approaches as surveyed in [Car05T]	60
Table 2.13	ISO/IEC 9126-1 quality model top level hierarchy	63
Table 3.1	Availability of Case Studies Data	71
Table 3.2	Research Questions Revisited with respect to the obtained results	81
Table 3.3	Strategies for dealing with threats to validity [Rob02]	82
Table 4.1	High-level Inputs and Outputs of GOTHIC Activities	89
Table 5.1	COTS Related Information Sources Types	100
Table 5.2.	Basic IQ dimensions to as suggested by [Wan-Str96]	101
Table 5.3.	Excerpt of COTS selection IQ needs & facts elicited from COTS selectors	103
Table 5.4	Excerpt of the GQM approach used to guide the information storage and metrics definition	105
Table 5.5	Excerpt of the ISO/IEC 9126-1 framework for stating COTS IQ	106
Table 5.6	Excerpt of the Information Sources prioritization in the RTSC case	108
Table 6.1	Domain analysis practices for representing COTS dimensions	118
Table 6.2	High-level characteristics and subcharacteristics describing COTS Non-technical factors	120

Table 6.3	Excerpt of the quality model for the RTSC case	128
Table 6.4	Excerpt of a non-technical factor decomposition for the RTSC case.....	128
Table 6.5	Excerpt of the unifying model for the RTSC case.....	129
Table 7.1	Glossary of GBRAM heuristics used in GOTHIC's Activity 3	135
Table 7.2	Different types of actors related to COTS domains.....	136
Table 7.3	An excerpt of actors and goal identification	137
Table 7.4	A scenario excerpt of the RTSC case study.....	137
Table 7.5	Example of the identification and refinement process.....	138
Table 7.6	An example of a goal-schema.....	138
Table 7.7	Excerpt of the identification and coupling of system actors.....	144
Table 7.8	Excerpt of system actor's identification & coupling with domain goals....	150
Table 7.9	Excerpt of a possible goal-oriented taxonomy for the RTSC case	151
Table 7.10	Example questions and answers attained to taxonomy nodes	152
Table 8.1	Predicates and functions over taxonomies.....	156
Table 8.2	Predicates, semantics and abbreviations used over goals	156
Table 8.3	Effect of transformation rules on the stop condition for Step 1.....	161
Table 8.4	Effect of transformation rules on the stop condition for Step 2.....	165
Table 8.5	Effect of transformation rules on the stop condition for Step 3.....	166
Table 8.6	Ttransformation rules and their applicability to the 4-steps process	168
Table 8.7	Validating the RTSC goal-classification schema	171
Table 8.8	Discovering goals process for an existing 'ad-hoc' classification hierarchy	172
Table 8.9	Goals bound to some nodes of the BA taxonomy	173
Table 8.10	Transforming the Gartner classification into a goal taxonomy	173
Table 10.1	Issues regarding the use of GOTHIC as perceived by researchers.....	201
Table 10.2	Excerpt of explorative survey results from some Norwegian companies .	202
Table 10.3	Issues regarding the use of GOTHIC in industrial organizations.....	204
Table 10.4	Research Questions Revisited.....	205
Table 11.1	Intended benefits to the different COTS selection roles.....	212
Table A1.1	Glossary of heuristics and their codes in GOTHIC	245
Table A1.2	Heuristics that Support GOTHIC Activities.....	246
Table A2.1	COTS IQ Reference Model	253

Chapter

1

Introduction

Nowadays, an alternative paradigm to the traditional software development lifecycle consists on building systems by integrating pre-packaged solutions, usually known as *Commercial-Off-The-Shelf* (COTS) components –hereafter COTS–; and migrating existing systems towards COTS-Based Systems (CBS) [Mey-Obe02], [Car-Lon00], [Keil-Tiw05]. It has become an economic and strategic need for developing large and complex software systems. As a consequence, a vast marketplace of COTS is actually available and steadily growing.

In general, COTS are software components that provide a specific functionality and are available in the market to be purchased, interfaced, and integrated into other software systems. More precisely: “*A COTS product is a [software] product that is: (1) sold, leased, or licensed to the general public; (2) offered by a vendor trying to profit from it; (3) supported and evolved by the vendor, who retains the intellectual property rights; (4) available in multiple, identical copies; and (5) used without source code modification by a consumer*” [Mey-Obe02].

The potential benefits of using this technology are mainly its reduced costs and shorter development time [Obe-Bro97], which is due to the fact that components can be procured instead of being developed from the scratch. Hence, as the size and complexity of systems grow, the use of COTS has become the standard way of developing software [Rei+03], [Bhu-Boe07].

However, COTS-Based Systems Development (CBSD) also introduces new risks and challenges. It basically changes the focus of the development-centric approach assumed in traditional software development by a procurement-centric approach characterized by a constant, iterative trade-off among risks, user requirements, system architecture and marketplace availability [Bro+00], [Kon-Hut07]. As a result, many challenges, ranging from technical to legal issues, must be faced for adapting the traditional software engineering activities with the aim of exploiting the benefits of using COTS [Mor+02], [Crn-Lar02].

In particular, one of the most critical activities in CBSD is the selection of the COTS themselves: if the wrong COTS is selected, the risk of a project failure increases dramatically [Bas-Boe01], [Vit+03a], [Crn+05], [Bhu-Boe07].

Several authors describe the selection process as a set of different phases and strategies, e.g., [Fin+96], [Obe-Bro97], [Kun-Bro99], [Moh+07]. All of them agree that the high-level selection process is basically composed of three main activities:

- 1) *Searching COTS candidates from the marketplace,*
- 2) *Evaluating them with respect to the system requirements*
- 3) *Deciding the best COTS from a set of competing alternatives*

The ever-growing nature of the COTS marketplace, both in the variety of market segments available and the components offered therein [Vit+03b], as well as the lack of available and well-suited information about the components, make difficult to obtain and efficient and quality-assured search and evaluation [Li06], [Bhu-Boe07].

In this context, researchers and practitioners have been dealing with COTS selection for quite a time and different selection methods have been proposed (e.g. CAP, CARE, CEP, CRE, OTSO, PECA, PORE, QESTA, Scarlet, STACE, Storyboard, etc.), see Chapter 2. However, most of them have been proposed for driving COTS evaluation and deciding the COTS, letting aside the problems related to search COTS and information about them in the marketplace. This lack of support affects the whole selection process, making it highly risky, often expensive and inefficient: no matter how good is the evaluation process, selection may be wrong if the candidates chosen to be evaluated are not the right ones [Neu-Stu07]. Trying to deal with this gap, **this research focuses precisely on the improvement of the COTS searching phase.**

1.1 The Problem: Searching COTS in the Marketplace

It is well-known that one of the essential problems in software reuse is organizing collections of reusable components in order that component re-users are able to find and understand them, in other words, for effective search and retrieval [Pri91]. If this process fails, the reuse can not happen [Fra-Pol94] or even worse, wrong components may be selected.

Storing, searching and retrieval of reusable components have been usually supported by component repository systems. Each of these activities relies on the existence of a systematic method of organizing the components in the repository so re-users can match existing reusable parts to their current needs. However, although reusable component repositories have been an active research area for more than a decade [Fra-Kan05], the special nature of the COTS has taken the original concept of reuse into a completely different arena. COTS marketplace nature drastically changes the “classical” component repository reuse schema to a global reuse approach where the marketplace acts as a global, dynamic, distributed, and heterogeneous set of repositories; and where nobody has knowledge or control of the available repositories [Clar+04], [Wan-Hom06].

In this context, indexing and representing COTS so that they can be found and understood are two important issues endangered by various facts:

1. *Uncontrolled COTS marketplace basis.* COTS marketplace acts as a continuous “product conveyor belt” [Bro+98]. Lots of potential vendors or providers offer their products without any direct control [Man+07], [Lau-Ped05] and no one has great influence over the speed, content, or variety of products on the product belt.
2. *Growing size of the COTS marketplace:* New and improved products and technologies are continuously offered [Bro+00], [Bhu-Boe07]. Thus, existing market segments offer more and more products, and new market segments are continuously emerging [Ulk-Sep04]. Mobile technologies are a good example of both situations.
3. *Rapid changes in the COTS marketplace:* New versions of existing products are released every few months [Yan+05], [Mer06]. Moreover, market segments frontiers move slightly over the years, making products to offer services that initially were seen as belonging to different segments [Fra05]. For instance, current mail server systems usually provide instant messaging facilities, even video-conferencing services.
4. *Type of descriptions available for COTS:* Given the commercial nature of the marketplace, it is common that COTS suppliers tend to highlight strengths and hide weaknesses of their licensed components; therefore, sometimes the trustworthiness of the COTS information available is unclear. So, COTS re-users are faced with the problem of ensuring that COTS perform the functionality they claim [Moh+04], [Ast+06], [Ast+06b].
5. *Lack of standards for COTS descriptions.* Currently, the amount of information available about COTS is widespread, vast and still growing. Component providers and brokers do not have a standard for describing components, which results in a variety of documentation styles very difficult to compare [Tau+04]. This fact does not allow performing automated or at least assisted search [Cec+06], [Ast+06]. A study conducted in [Ber+03] evidenced that the required COTS information is highly incomplete, widespread and unstructured, becoming very difficult to obtain.
6. *Dependencies among COTS:* Although COTS are attempting to simulate the “plug and play” capability of the hardware world, in reality they are not designed to work isolated, but in collaboration with others. Therefore many dependencies among them exist, either for enabling, enhancing, or complementing their functionality [Fra-Mai03]. However, commonly, these dependencies are not explicitly declared. It may result in several over-costing and integration problems [Bhu-Boe07], [Don+05]. For instance, an organization selecting a document management tool would discover quickly that they need to acquire a document imaging tool for enabling the functionality of scanning and storing paper documents.
7. *Lack of COTS reuse support.* As mentioned above, the special nature of COTS has taken the original concept of reuse into a completely different arena [Aya-Fra06a], and actually there is a lack of efficient mechanisms to effectively record and reuse COTS related issues.

This list is representative of the most important reasons why traditional component repositories and associated searching and retrieval approaches break down in the COTS context [Aya-Fra06a].

Consequently some practical questions without efficient answers arise when carrying out a particular COTS searching process, such as:

- *What kinds of components are available and which of them could be useful to solve this particular problem?*
- *Which are the relationships among the market segments corresponding to these kinds and which are their implied needs?*
- *How to find and process the information referred to the components of those kinds which facilitate to perform an informed evaluation?*

Therefore, **this thesis focuses on how to provide an effective way to address these practical questions in order to effectively exploit the benefits of the COTS technology and deal with the risks implied in its use.**

1.2 Importance of the Problem

CBSD is an economic and strategic need in a wide variety of different application areas in both public and private companies. According to studies performed by Gartner Group, at least 70% of all new software applications developed in 2003 by major corporations involved COTS. At this respect, Michael Blechar, vicepresident of Internet & e-Business Technologies at Gartner added: "*COTS can be a major enabler of productivity and savings. Through 2004, IS organizations that are mature in CBSD methods and that use a model-driven or pattern-based application development framework containing a large inventory of business components have the potential to be 5 to 10 times more productive and responsive than those that do not.*" [Gri02]. In addition, from previous empirical studies in small and medium companies [Li06] we know that roughly half of all software projects make use of COTS in 2006 and this percentage is rising rapidly.

Thus, whilst software engineers are becoming increasingly aware of the contribution that such technology is making to the software industry, business, and society in general, they have also recognized that fostering its adequate (re)use is actually a crucial task for progressing towards improvements in a great variety of application areas [Sim-Dil06].

In summary, although the risks inherent to COTS usage are as large as their potential return, they are foreseen as the standard way of developing software systems [Rei+03]. According to Gartner Group, COTS are expected to constitute over 90% of the software applications running in major corporations. As a result of this tendency, increasingly amounts (dozens of thousands) of COTS are now accessible in the marketplace. Nevertheless, there is still a lack of well-suited approaches to enable an effective COTS searching from the marketplace [Cec+06], making difficult to obtain a quality-assured selection and the full potential of large-scale software reuse. A critical consequence of performing the searching related activities poorly is that the whole COTS selection process is damaged and confidence on the results of the process diminishes [Req+05], [Cec+06], [Neu-Stu07].

Many studies report that an inadequate COTS selection process is a critical and common cause of IT projects failure [Vit+03a], [Bas-Boe01]. For instance, the study detailed in [Hau02] shows that more than 30% of ERP¹ projects are abandoned before the end of the project, 20% of the remaining projects exceed budgets, and more than 20% do not keep the project deadlines. They conclude that ERP projects face the same risks as any other IT project. Furthermore, the study presented in [Bas+00] concludes that the main key inhibitor of CBSD is the difficulty of finding components and information about them.

All this empirical evidence is a compelling reason for understanding that research efforts are required to avoid these project failures, supporting the COTS searching and reuse activities that are currently lacking of appropriate support.

1.3 Research Goal

This thesis is focused on improving COTS selection processes in the context of organizations that regularly select and reuse COTS by offering a solution to deal with some open issues related to categorizing and searching such components from the marketplace and their adequate reuse.

This dissertation proposes a method called GOTHIC (Goal-Oriented Taxonomy and reuse Infrastructure Construction) to deal with some COTS marketplace challenges by building a goal-oriented domain reuse infrastructure. Such domain reuse infrastructure can evolve as the marketplace does, and may be used in COTS searching processes to obtain the appropriate criteria for locating the most appropriate kind of components. The general objective of this work can be stated as:

“To provide support to the effective construction of a reliable and understandable representation of the COTS marketplace for enabling COTS search, especially for coarse-grained COTS”

The following paragraphs further analyze this general objective with some descriptions intended to refine each one of its aspects.

What do we mean by “effective construction”?

Searching COTS from the marketplace is an activity endangered by various threats, as those mentioned in Section 1.1 and the lack of experience in how to arrange the marketplace to deal with these problems. These factors make the process of COTS marketplace arrangement and representation difficult to rely only on common sense. Thus by effective construction we mean:

- To construct COTS marketplace classification schemas as a way of representation of the marketplace’s contents by following a rigorous and systematic method, with well-defined steps and objectives, leading to the identification of the appropriated elements of the classification schema.

¹ Enterprise Resource Planning systems are one of the most COTS solutions used in industry, designed to handle virtually all of an organization business computing needs.

- To provide useful mechanisms to adapt and evolve the classification schemas to the COTS marketplace evolution and specific organizational needs for which they are intended.

What do we mean by “reliable and understandable representation of the COTS marketplace”?

By *reliable* we mean the construction of a classification schema founded on a deep understanding of the COTS domain and its environment. All the relevant market features, components characteristics, technologies, regulations and standards of the domain should be identified as well as their relationships and dependencies among them. The outcome of this analysis should be formally represented and recorded to create a reliable COTS domain knowledge base for supporting marketplace representation’s evolution.

By *understandable* we mean classification schemas which are based on a clearly stated, complete and unambiguous quality framework, which includes all the required information to perform an informed selection. We also mean classification schemas which, by providing a formal representation of all the relevant information for performing COTS selection, can be clearly understandable by COTS re-users and providers in order to enable adequate storing, searching and retrieval mechanisms.

What do we mean by “coarse-grained COTS”?

COTS available in the market differ widely among each other. They may range from simple libraries which provide a limited and clear set of functions, to components whose functionality is so broad and complex that their understanding requires extensive exploration. We call this kind of COTS, *coarse-grained COTS*. They are the focus of our attention, because the risks involved in selecting this kind of COTS are critically greater than the ones involved in smaller COTS that are in addition usually more easily identifiable and comparable.

1.3.1 Additional Objectives

In the course of this work we have identified some issues, which although not part of the initial objectives, have been explored up to a certain extent in our research. In some ways they extend or complement the original goals of this project, thus we have concluded that it is worth to include them as significant objectives. These issues are:

The identification of artifacts to support the reusability of components information and the domain knowledge gained in the classification schemas construction as well as in each COTS selection experience

As our work progressed, it became evident that to deal with COTS marketplace evolution some reusable artefacts must exist to support the evolution of the classification schemas. Moreover, to be effective, the classification schema must operate within the context of a domain reuse infrastructure environment aimed at promoting the reuse at various levels. At this respect, several works (e.g. [Mor+00], [Cla+04], [Moh+04], [Wan-Hom06]) advocate that having a repository for COTS

products with enough information about them is becoming a necessity for improving the CBSD practice.

Several of the deliverables produced in the classification schema construction, or parts of them were reusable. Consequently, we expanded our original objective to build a COTS domain reuse infrastructure from the classification schema construction process that not only provides an adaptable, evolvable, understandable and reliable structure of the COTS marketplace but also is instrumented to reuse information and knowledge gained in each selection process experience. Thus, the method relies on some artifacts and mechanisms aimed to this.

The search for alternative strategies for populating and maintaining the COTS reuse infrastructure

By the nature of the COTS reuse infrastructure constructed with GOTHIC, we realized that the applicability of the method was actually intended for medium and large size organizations that can assume the related cost of constructing and maintaining a COTS domain reuse infrastructure. However, we have expanded the method to build a generic and flexible domain reuse infrastructure, and also explored some strategies to make feasible the open and global use of such generic infrastructure to afford the related benefits to all COTS re-users, including software organizations of any size. Although to deeply explore and evaluate these open and global strategies are not into the goals of this thesis, they are considered as future work. Thus, as a matter of fact most of our findings with regard to these issues are labelled as “intended populating and maintenance strategies”, and their further exploration is considered as future work related in Chapter 11.

The provision of tool support to the method

COTS classification schemas construction requires the management of several of issues and their relationships at several levels. This is not an easy task. So it can be considered really desirable to provide some tool-support for some method activities. Although it is not part of the deliverables of this thesis, we briefly present some tools and proof-of-concept prototypes developed in our project whose construction was partially guided by the concepts presented here.

1.4 Research Context

The research in this thesis has been conducted within the GESSI² (Software Engineering for Information System Group) research group from the Technical University of Catalunya (UPC).

The GESSI group conducts research in many fields of software engineering, with particular emphasis on supporting procurement and implementation of COTS components, requirements engineering, software metrics, construction of quality models for software domains, software process modelling, enactment, and software certification.

² <http://www.lsi.upc.es/~webgessi/index.html>

This thesis is specifically involved in the COTS selection and procurement research line which has been progressing through several projects the group has carried out and is currently carrying out.

Some of the most representative projects related with this research are:

- ♦ PRISMA (Academic Record Management System) [PRI],
- ♦ DALI (Methodologies and tools for the Development, Acquisition, evaluation and Integration of software components) [DAL], and
- ♦ UPIC (towards a Unified approach to the Procurement and Implementation of information system) [UPI].

The PRISMA project was a technology transfer project whilst DALI and UPIC refer to financed research projects. The author of this thesis has been directly involved in the last two projects.

In the PRISMA project, the group was engaged to assess the development of an academic record management information system which was planned to include some strategic business functionalities. To select the required COTS, several problems were experienced. As a result, several research works were envisaged and some of them were further explored through the subsequent DALI project, which goal was to supply a methodological and technological platform for component-based software development.

Although the DALI project's goal was successfully attained, some other problems out of the original scope of DALI were also evident, as those mentioned in Section 1.1. Consequently, given the importance of the former findings, the current UPIC project was envisaged to study, extend and apply methods, models, techniques and languages to help the effective and efficient procurement and implementation processes for COTS in the context of the information systems organizations.

The work presented in this thesis is part of the UPIC project. Some members of the group are engaged in complementary issues to reach the overall goal of the UPIC project, as the formal representation of quality aspects driving COTS selection, the evaluation of architectures and alternatives, etc. whilst this thesis is devoted to improve the COTS searching process. The work presented here will be complemented with their results.

On the other hand, to reach a preliminary evaluation of the GOTHIC method, the author of this dissertation performed a research stay at the Norwegian University of Science and Technology (NTNU). They have a recognized worldwide expertise in empirical research and industrial evaluation issues. The evaluation process was performed within the European ITEA project, Norwegian COSI (Co-development using inner & Open Source in Software Intensive products) [COS] which aims to enable the Norwegian IT sector to fully exploit the benefits and advantages of COTS and Open Source Software (OSS) components. Some of the industrial partners of this project are IKT-Norge, eZ systems, Keymind and Linpro.

Furthermore, as a result of our collaboration from this preliminary evaluation, several future research lines are envisaged with the GOTHIC proposal as a core of a project having as potential partners to some academic and industrial organizations as Hewlett

Packard, NTNU, Politecnico di Torino, Centre de Recherche Public Henri Tudor (CRPHT), Beijing University of Technology (BJUT) and UPC among others (see Chapter 11).

1.5 Methodological Approach

Software engineering research is often motivated by problems that arise in the production and use of real world software. Shaw provides a way of characterizing software engineering research, in terms of what she describes as types of research questions, research results and validation techniques [Sha01].

To provide an overview of the research performed in this thesis, this section characterizes the research questions, research results and validation efforts taken to perform this thesis in the terms proposed by Shaw [Sha01]. Chapter 3 further details the research approach taken.

In general, this research was elaborated following an iterative process based on action-research premises jointly with case studies. Such combined approach was chosen because it permitted a flexible design of research questions through the research process. Moreover, it allows trying several action plans to solve some COTS searching related problems. Research questions and possible solutions were envisaged and evaluated on a 3 iterative research stages implemented by several industrial and academic case studies in different domains (see Chapter 3).

Table 1.1 presents the research questions driving this research.

Table 1.1 Research Questions of this Research

Research Questions of this Research	
RQ1: <i>What are the actual challenges of COTS selection processes?</i>	
	RQ1.1- What are the actual challenges of COTS searching processes?
RQ2: <i>How can we support COTS searching challenges?</i>	
	RQ2.1- Can goal-oriented approaches be used to produce useful results for dealing with COTS searching challenges?
	RQ2.2- How can we characterize COTS in the marketplace?
	RQ2.3- How can relevant information related to COTS be gathered, evaluated and synthesized?
	RQ2.4- How can such information be maintained for its reuse in different COTS selection processes?

According to Shaw [Sha01], software engineering research answers questions about: 1) methods or means of development; 2) methods for analysis or evaluation; 3) details of designing, analyzing, or evaluating a particular instance; 4) generalizations or characterizations over whole classes of systems or techniques; 5) exploratory issues concerning existence or feasibility. Table 1.2 lists these types of software engineering research questions and characterizes the research questions of this thesis.

Table 1.2 Shaw’s characterization of Research Questions related to Research Questions in this Thesis

Question Type	Examples	RQ
Method or Means of development	How can we do / create / modify / evolve (or automate doing) X? What is a better way to do / create / modify / evolve X?	RQ2
Method for analysis or evaluation	How can I evaluate the quality/correctness of X? How do I choose between X and Y?	RQ2.3
Design, Evaluation or Analysis of a particular instance	How good is Y? What is property X of artifact/method Y? What is a (better) design implementation, maintenance, or adaptation for application X? How does X compare to Y? What is the current state of X / practice of Y?	RQ1 RQ1.1
Generalization or Characterization	Given X, what will Y (necessarily) be? What, exactly, do we mean by X? What are its important characteristics? What is a good formal/empirical model for X? What are the varieties of X, how are they related?	RQ1 RQ2
Feasibility study or Exploration	Does X even exist, and if so what is it like? Is it possible to accomplish X at all?	RQ2.1 RQ2.2 RQ2.4

As observed from Table 1.2, the different studies performed as part of this thesis yield to different kinds of research questions proposed by Shaw. The first two types of research questions proposed by Shaw produce methods of development or of analysis that the authors investigated in one setting, but can presumably be applied in other settings. The third type of research question deals explicitly with some particular system, practice, design or other instance of a system or method; these may range from narratives about industrial practice to analytic comparisons of alternative designs. Generalizations or characterizations explicitly rise above the experiments, case studies or examples performed. Finally, researches that deal with an issue in a completely new way are sometimes treated differently from researches that improve on prior art, so feasibility or exploration as a separate category.

According to Shaw, the tangible contributions of software engineering research may be procedures or techniques for development or analysis; they may be models that generalize from specific examples, or they may be specific tools, solutions, or results about particular systems.

Table 1.3 lists the types of research results mostly reported in software engineering and provides specific examples [Sha03]. Table 1.4 illustrates the high-level research activities performed in this thesis, their relationship with research questions and the characterization of research results as proposed by Shaw.

There are several kinds of evidence supporting research results [Sha01]. It is essential to select a suitable form of validation for the type of research result and the method used to obtain the result. Different types of results have value, and more

rigorous results emerge only over time, either through cumulative evidence or by building rigorous experiments on a base of more informal experience.

Table 1.3 Shaw's characterization of Software Engineering Research Results

Research Results	Abbreviation	Research Approach or Method
Procedure or Technique	PoT	New or better way to do some task. It includes techniques for implementation, representation, management and analysis. A technique should be operational –not advice or guidelines, but a procedure.
Qualitative or Descriptive Model	QDM	Structure or taxonomy for a problem area. Well-organized interesting observations.
Empirical model	EMO	Empirical predictive model based on observed data.
Analytic Model	AMO	Structural model that permits formal analysis or automatic manipulation.
Tool or notation	ToN	Implemented tool that embodies a technique. Formal language to support a technique or model (should have a calculus, semantics, or other basis for computing or doing reference)
Specific solution, prototype, answer or judgment	SPA	Solution to application problem that shows application of SE principles. Careful analysis of a system or its development. Result of a specific analysis, evaluation, or comparison
Report	REP	Interesting observations, rules of thumb, but no sufficiently general or systematic to rise to the level of a descriptive model.

Table 1.4 Characterization of High-Level Research Results of this Thesis

High-Level Research Activities Performed in this Thesis	RQ Related	Type of Result
To conduct a careful analysis of the COTS selection processes state-of-the-art and state-of-the-practice, in order to understand and organize the problem area.	RQ1	QDM
To invent new ways to deal with the COTS searching detected challenges to enable its effectiveness and reuse. They range from specific solutions to notations, prototypes and tools implementation.	RQ2	PoT SPA ToN
Report interesting observations about such processes.	RQ2	REP
To create and defend generalizations from real examples, drawn to the performed case studies called “formative” (see Chapter 3).	RQ2	PoT
To embody results in a prescriptive method.	RQ2	SPA

The validation techniques in terms of Shaw’s characterizations are illustrated in Table 1.5.

Table 1.5 Shaw’s characterization of Software Engineering Research Validation

Type of Validation	Examples
Analysis	I have analyzed my result and find it satisfactory through rigorous analysis, e.g. For a formal model ...rigorous derivation and proof For an empirical model ...data on use in controlled situation For a controlled experiment ...carefully designed experiment with statistically significant results
Evaluation	Given the stated criteria, my result... For a descriptive model ...adequately describes phenomena of interest ... For a qualitative model ...accounts for the phenomena of interest ... For an empirical model ...is able to predict ... because ..., or ...generates results that fit actual data ... Includes feasibility studies, pilot projects
Experience	My result has been used on real examples by someone other than me, and the evidence of its correctness/usefulness/effectiveness is ... For a qualitative model ...narrative For an empirical model or tool ...data, usually statistical, on practice For a notation or technique ...comparison of systems in actual use
Example	Here’s an example of how it works on ... For a technique or procedure ...a “slice of life” example based on a real system ... For a technique or procedure ...a system that I have been developing ... For a technique or procedure ...a toy example, perhaps motivated by reality.
Persuasion	I thought hard about this, and I believe passionately that ... For a technique ...if you do it the following way, then ... For a system ...a system constructed like this would ... For a model ...this example shows how my ideas works
Blatant assertion	No serious attempt to evaluate result. This is highly unlikely to be acceptable.

The action-research process driving our research in order to develop the GOTHIC method required early validation while under development. Hence, we distinguish among two kinds of validation: *formative* and *summative*. This distinction is key in that the formative validation involves the evolution of the research work simultaneously coupled with validation (i.e., its central role was shaping the GOTHIC method by integrating successful results of the different research stages); whilst the summative evaluation is addressed to validate the method developed during the formative evaluation.

Table 1.6 summarizes the main validation approaches followed in this thesis in terms of the Shaw characterization of research validation. It also provides the Chapters where detailed information about these validation efforts are discussed.

Table 1.6 Main Validation Efforts Pursued in this Thesis

Validation Type	High-Level Activities	Chapter
Evaluation	♦ Implementation of formative and preliminary summative research validation efforts to evaluate the proposed method.	Chapter 3 and 10 respectively
	♦ Diverse results of the method have been evaluated by researchers and industrial practitioners to provide evidence of their usefulness.	Chapter 3 Chapters 5 to 9
	♦ Preliminary summative evaluation efforts have been performed in Norwegian industries.	Chapter 10
Experience	♦ Academic and industrial case studies were carried out and useful lessons learned were summed up into heuristics.	Chapters 5 to 9
Example	♦ Several case studies motivated by reality were performed and the obtained results were evaluated by experts.	Chapter 3
	♦ Implementation of two software tools supporting some issues of our proposed method.	Chapter 9
	♦ A proof of concept prototype of the GOTHIC reuse infrastructure.	Chapter 9 Chapter 10
Persuasion	♦ Persuasion on the grounds of the formative and preliminary summative evaluation results. Current results are discussed in their corresponding chapters.	Chapter 11

In the software engineering area, it is very hard to industrially validate methods such as this. It is because its further summative evaluation necessarily requires the implementation and use of large scale repositories and analysis that are not feasible to be obtained in a short period of time (i.e., it would not be reasonable to argue a full industrial validation of the ideas presented in this thesis within the terms of the PhD studies). Therefore, in this thesis dissertation, as with all research topics for which a critical mass is an issue, for summative evaluation we provide some feasibility and effectiveness insights by means of arguments that extrapolate from academic cases and post-mortem summaries of industrial cases in order to preliminarily answer research questions. Of course, the whole industrial summative evaluation of the method is one of our main goals but it is considered as future work (see Chapter 11).

From Table 1.6 it can be noted that regardless the research validation phase (i.e., formative or summative), throughout this thesis work several kinds of research validation efforts have been carried out.

1.6 Publications in Relation to this Thesis³

Many aspects of this thesis have been published at different levels:

Journals

- P1** [Fra+07] Franch, X.; Grau, G.; Mayol, E.; Quer, C.; Ayala, C.; Cares, C.; Navarrete, F.; Haya, M.; Botella, P.: “Systematic Construction of *i** Strategic Dependency Models for Socio-Technical Systems” *International Journal of Software Engineering and Knowledge Engineering (IJSEKE)*. Volume 17, No. 1, February 2007.
- P2** [Aya+05c] Ayala, C., Botella, P., Franch, X.: “Construction of a Taxonomy for Requirements Engineering Commercial-Off-The-Shelf Components” *Journal of Computer Science and Technology, Special Issue on Software Requirements Engineering* Vol. 5, No. 2, August 2005.

Conference Proceedings

- P3** [Aya-Fra08] Ayala, C., Franch, X.: “Assessing What Information Quality Means in OTS Selection Processes”. International Conference on Composite-Based Software Systems (ICCBSS 2008). To appear. IEEE Society Press
- P4** [Aya+07] Ayala, C., Sørensen, C.F., Conradi, R., Franch, X., Li, J.: “Open Source Collaboration for Fostering Off-The-Shelf Components Selection”. In *IFIP International Federation for Information Processing*, Volume 234, Open Source Development, Adoption and Innovation. (OSS 2007). June 2007, pp. 17-30.
- P5** [Aya-Fra06c] Ayala, C.; Franch, X.: “Domain Analysis for Supporting Commercial Off-The-Shelf Components Selection”. In *Proceedings of the 25th International Conference on Conceptual Modelling (ER 2006)*. Tucson, Arizona, USA. Lecture Notes in Computer Science. Vol. 4215/2006. pp. 354-370.
- P6** [Aya-Fra06a] Ayala, C., Franch, X.: “A Goal-Oriented Strategy for Supporting Commercial Off-The-Shelf Components Selection” In *Proceedings of the 9th International Conference on Software Reuse (ICSR)*. Torino, Italy. June 2006. Lecture Notes in Computer Science. Vol. 4039-2006. pp. 1-15.
- P7** [Aya-Fra05] Ayala, C.; Franch, X.: “Transforming Software Package Classification Hierarchies into Goal-Based Taxonomies” In *Proceedings of the 16th International Conference on Database*

³ All these publications can be downloaded from <http://www.lsi.upc.edu/~cayala/>

and Expert Systems Applications (DEXA 2005). Copenhagen, Denmark. 22-26 August 2005. Lecture Notes in Computer Science. Vol. 3588/2005. pp. 665-675.

- P8** [Gra+05] Grau, G.; Franch, X.; Mayol, E.; Ayala, C.; Cares, C.; Carvallo, J.P.; Haya, M.; Navarrete, F.; Botella, P.; Quer, C.: “*RiSD*: A Methodology for Building *i** Strategic Dependency Models” In Proceedings of the 17th International Conference on Software Engineering and Knowledge Engineering (SEKE’05). 14-16 July, 2005. Taipei, Taiwan, Republic of China. pp. 259-266.
- P9** [Aya+05a] Ayala, C.P., Botella, P., Franch, X.: “On Goal-Oriented COTS Taxonomies Construction” In Proceedings of the 4th International Conference on COTS-Based Software Systems (ICCBSS 2005). Bilbao, Spain. Lecture Notes in Computer Science. Vol. 3412/2005. pp. 90-100.

Doctoral Symposium

- P10** [Aya06] Ayala, C.: “Systematic Construction of Goal-Oriented COTS Taxonomies” In Proceedings of the 3rd Doctoral Consortium at the 18th Conference on Advanced Information Systems Engineering (CAISE 2006). 5-9 June 2006, Luxembourg.

Workshops

- P11** [Aya-Fra06b] Ayala, C., Franch, X.: “Overcoming COTS Marketplace Evolvability and Interoperability”. In Proceedings of CAISE Forum at 18th Conference on Advanced Information Systems Engineering (CAISE 2006) June 2006, Luxembourg.
- P12** [Aya+05b] Ayala, C.; Cares, C.; Carvallo, J.P.; Grau, G.; Haya, M.; Salazar, G.; Franch, X.; Mayol, E.; Quer, C.: “A Comparative Analysis of *i**-Based Goal-Oriented Modelling Languages” In Proceedings of the International Workshop on Agent-Oriented Software Development Methodology (AOSDM 2005), at the SEKE Conference. July 2005. Taipei, Taiwan; China. Pp.43-50.
- P13** [Aya+04c] Ayala, C.; Botella, P.; Franch, X.: “Construcción de una Taxonomía de Componentes COTS Orientados a la Gestión de Requisitos” In Proceedings of the VII Workshop on Requirements Engineering. Tandil, Argentina. December 2004. ISBN 950-658-147-9. pp. 214-225.
- P14** [Aya+04b] Ayala, C.; Cares, C.; Carvallo, J.P.; Grau, G.; Haya, M.; Salazar, G.; Franch, X.; Mayol, E.; Quer, C.: “Análisis Comparativo de Lenguajes de Modelado Orientados a Objetivos basados en *i**” In Proceedings of the Jornadas Iberoamericanas de Ingeniería

del Software e Ingeniería del Conocimiento (JIISIC'04). Madrid, Spain. 2004. Pages: 527-540.

- P15** [Aya+04a] Ayala, C.; Botella, P.; Franch, X. "Goal-Based Reasoned Construction of Taxonomies for the Selection of COTS Products" In Proceedings of the 8th World Multiconference on Systemics, Cybernetics and Informatics (SCI 2004). July 18-21, Orlando, Florida, USA.

Technical Reports

- TR1** [Aya-Fra07] Ayala, C., Franch X.: "A Systematic Approach to Manage Information Quality for Supporting Software Package Selection" Research Report Universitat Politècnica de Catalunya, Departamento de Lenguajes y Sistemas Informáticos.
- TR2** [Aya-Fra06TR-a] Ayala, C., Franch, X.: "Domain Analysis for Supporting Commercial Off-The-Shelf Components Selection" (Extended Version) Research Report LSI-06-16-R. Universitat Politècnica de Catalunya, Depto. de Lenguajes y Sistemas Informáticos. http://www.lsi.upc.edu/dept/techreps/llistat_detallat.php?id=916
- TR3** [Aya-Fra06TR] Ayala, C., Franch, X.: "A Process for Building Goal-Oriented COTS Taxonomies" Research Report LSI-06-7-R. Universitat Politècnica de Catalunya, Departamento de Lenguajes y Sistemas Informáticos. http://www.lsi.upc.edu/dept/techreps/llistat_detallat.php?id=907
- TR4** [Aya05PT] Ayala, C. "Systematic Construction of Goal-Oriented COTS Taxonomies". Thesis Project presented to fulfill the requirements of the "Diploma de Estudios Avanzados" and PhD. Thesis Evaluation. July 2005.
- TR5** [Aya+05TR] Ayala, C.P., Botella, P., Franch, X. "Goal-Based Reasoning in the Construction of Taxonomies for COTS Components" Technical Report LSI-05-58-R. Universitat Politècnica de Catalunya, Departamento de Lenguajes y Sistemas Informáticos.
- TR6** [Aya+04TR] Ayala, C. P., Botella, P., Franch, X.: "Towards the Definition of a Taxonomy for the COTS Product's Market" Technical Report LSI-04-3-R. Universitat Politècnica de Catalunya, Departamento de Lenguajes y Sistemas Informáticos. 2004.

Master Thesis Projects

- MT1** [Aas-Lar07] Aaslund, K., Larsen S.: "OTS-Wiki: A Web Community for Fostering Evaluation and Selection of Off-The-Shelf Software Components" Master Thesis. Department of Computer and Information Science, Norwegian University of Science and

Technology (NTNU). Spring 2007. Supervisor: Professor Reidar Conradi. Co-advisors: Dr. Carl-Fredrik Sørensen and Claudia Patricia Ayala Martínez.

<http://www.idi.ntnu.no/grupper/su/su-diploma-2007/dipl07-larsen-aaslund.pdf>

MT2 [Ger07]

Gerea, M.: "Selection of Open Source Components: A Qualitative Survey in Norwegian IT Industry". Master Thesis. Department of Computer and Information Science, Norwegian University of Science and Technology (NTNU). Supervisor: Professor Reidar Conradi. Advisor: Dr. Carl-Fredrik Sørensen. Spring 2007. <http://www.idi.ntnu.no/grupper/su/su-diploma-2007/dipl07-gerea.pdf>

Undergraduate Students Projects

PFC1 [Mes07]

Messegue, F. "Eina de suport per al anàlisi de dominis". Supervisor: Dr. Xavier Franch. January 2007. In catalan. <http://www.lsi.upc.edu/~cayala/Papers/IQToolDocumentation.pdf>.

PFC2 [Ger06]

Gerea, M.: "Selection and Evaluation of COTS and Open Source Components", 15th Dec. 2006, 81 p. part of course TDT4735 Depth Project in Software Engineering, Department of Computer and Information Science, Norwegian University of Science and Technology (NTNU). Supervisor: Professor Reidar Conradi. Advisor: Dr. Carl-Fredrik Sørensen. <http://www.idi.ntnu.no/grupper/su/fordypningsprosjekt2006/gereafordyp06.pdf>

1.7 Contributions

The primary contributions of this thesis are related with the COTS searching and reuse areas. In general, since most of current COTS selection methods do not deal with COTS searching and reusability issues, the GOTHIC method proposed in this thesis advances in the state-of-the-art by providing support to build a reliable COTS domain reuse infrastructure supported by highly evolvable and adaptable goal-oriented COTS taxonomies. It not only deals with some open issues related to categorizing and searching COTS but also it is aimed at supporting reusability of both COTS related information and the knowledge gained in each selection process. It impacts positively on the accuracy and reliability of COTS selection processes and therefore improves the whole CBSD practice.

The GOTHIC method was conceived as a set of interrelated and synergic strategies to deal with some problems mentioned in Section 1.1. The high-level contributions that lead to the GOTHIC method conception as a whole could be summarized as follows⁴:

⁴ In cases when contributions are obtained in collaboration with external people, I explicitly indicate what has been my participation.

- ♦ **C1.** *Better understanding of challenges for finding COTS.* An explorative study of actual problems and issues related with COTS selection processes in both industrial practice and literature was performed, leading to the identification of a set of critical challenges to select COTS. I performed the literature survey, whilst the qualitative industrial survey was performed as part of course TDT4735 Depth Project in Software Engineering by Marinela Gereá and subsequently as her Master Thesis at NTNU with the main advice of Professor Reidar Conradi and Dr. Carl Fredrik-Sørensen. During my research stay at NTNU, I actively participated throughout this study by providing my COTS-related experience in all preparation of the study, brainstorming sessions and results' analysis.
- ♦ **C2.** *A goal-oriented strategy for dealing with COTS marketplace evolvability.* Performed case studies demonstrate the feasibility of using goal-oriented approaches in this context. We deal with their adaptation to this context and provide prescriptive support in their use.
- ♦ **C3.** *Provision of a well-defined process for constructing goal-oriented taxonomies.* This process ensures the completeness and correctness of taxonomies, as well as their management and evolution as a result of COTS marketplace progress and/or different users' needs.
- ♦ **C4.** *A systematic approach for dealing with the unstructured, incomplete and vast amount of COTS related information.* It supports COTS selectors to decide what information sources to use according to their specific quality project needs. Our results were also implemented as a software tool. Such software tool was implemented by Fernando Messegue, as a final degree project at UPC under my advice and supervision of Dr. Xavier Franch.
- ♦ **C5.** *A COTS domain analysis strategy for recording the informational dimensions required to select COTS.* The required dimensions were identified, as well as suitable models for supporting the COTS domain reuse infrastructure creation.
- ♦ **C6.** *Identification of reusable artifacts throughout the GOTHIC method to reach the Experience Factory [Bas+94] and Learning Software Organization [Ruh+01] paradigms.* In addition, a specific strategy for populating the COTS knowledge base was explored and a software prototype was implemented. Such software prototype was implemented by Kristian Aaslund and Simon Larsen, as a Master Thesis project at NTNU co-advised by Dr. Carl Fredrik-Sørensen and me; and under the supervision of Professor Reidar Conradi.
- ♦ **C7.** *Preliminary insights of the GOTHIC method feasibility and effectiveness.* Preliminary empirical results have been obtained from some Norwegian companies and academia.

Table 1.7 summarizes these high-level contributions and relates them with the corresponding publications and chapters. Moreover, contributions are further described throughout this thesis' chapters, and are then summarized in Chapter 11.

Table 1.7 High-level contributions of this thesis, associated publications and chapters

Contribution	Chapter	Related Publication
GOTHIC as a whole	Chapter 4	P6, P10, TR4
C1	Chapter 2	PFC2, MT2
C2	Chapter 7	P1, P2, P8, P9, P11, P12, P13, P14, P15, TR5, TR6
C3	Chapter 8	P7, TR3
C4	Chapter 5	P3, TR1 PFC1
C5	Chapter 6	P5, TR2
C6	Chapter 9	P4, MT1
C7	Chapter 10	To be submitted

1.8 Organization of the Document

The thesis document is structured in the following 11 Chapters:

- ♦ **Chapter 1. *Introduction*.** It provides an introduction to the work, the objectives of the thesis and an overview of the proposal.
- ♦ **Chapter 2. *Related Work*.** It presents an overview of the state-of-the-art and state-of-the-practice of COTS selection, as well as a brief introduction to the most interesting features related to the disciplines used to solve the problem presented in this thesis.
- ♦ **Chapter 3. *Research Method*.** It presents the research process used to produce the method introduced in this thesis. It briefly presents the case studies which served as the conceptual origin for the GOTHIC method.
- ♦ **Chapter 4. *The GOTHIC method*.** This chapter provides an overview of the GOTHIC method proposed in this thesis, including an introduction to its seven main activities and their objectives. Additionally some of its high-level contributions, intended audience and applicability conditions are discussed. It is also stated in [Aya-Fra06a].

From Chapter 5 to 9, the GOTHIC method activities are further described. Each chapter describes the background and the strategy used to deal with the problems mentioned in Section 1.1. Salient features and artifacts produced are illustrated through examples from the case studies discussed in Chapter 3.

It is worth to mention that the focus in this thesis and the GOTHIC method is on the activities, not in the sequence of activities; it means that although the method activities are described as sequential, they are in fact concurrent and overlapping.

The content of subsequent Chapters is as follows:

- ♦ **Chapter 5.** *Activity 1: Exploration of Information Sources.* It describes the approach to systematically tackle the problems related with the vast amount of unstructured, incomplete, evolvable and widespread COTS information that highly increases the risks of taking a wrong decision when selecting them. This activity is also detailed in [Aya-Fra07].
- ♦ **Chapter 6.** *Activity 2: COTS Domain Analysis.* This chapter discusses the domain analysis strategy for recording the information needed to describe COTS market segments as required for effective COTS. A summary of this chapter can be found in [Aya-Fra06c].
- ♦ **Chapter 7.** *Activity 3, 4, 5: Goal-Oriented Core of GOTHIC.* This chapter tackles the Identification, Refinement and Statement of Goals, Establishment of Dependencies, and Goal Taxonomy Structuring activities. In summary it details the application of goal-oriented approaches (e.g., GBRAM and i^*) and some considerations for their usage in the GOTHIC method. It is also explained in [Aya-Fra06b].
- ♦ **Chapter 8.** *Activity 6: Goal Taxonomy Validation and Management.* This chapter deals with the process of making goal-taxonomies schemas flexible to different intended needs and marketplace evolution patterns as well as assuring their completeness and correctness. Several transformation rules intended for this purpose, have been identified and validated in our industrial and academic experiences. A former version of these rules was published in [Aya-Fra05].
- ♦ **Chapter 9.** *Activity 7: Knowledge Base Management.* In this chapter, the advantages of the existence of several reusable artefacts identified in our GOTHIC process to reach the Experience Factory [Bas+94b] and Learning Software Organization [Ruh01] paradigms are illustrated. Also the intended strategies to populate and maintain the knowledge base repository obtained with GOTHIC [Aya+07]. Some tools developed and/or used to support the method activities and knowledge management are introduced.
- ♦ **Chapter 10.** *Method Evaluation.* This chapter discusses the main current evaluation efforts for the method presented in this thesis.
- ♦ **Chapter 11.** *Conclusions & Future Work.* This chapter summarizes the contributions of the thesis and details the future work.

Fig. 1.1 shows the relationship among the thesis chapters, the GOTHIC method activities and the publications related to this thesis dissertation.

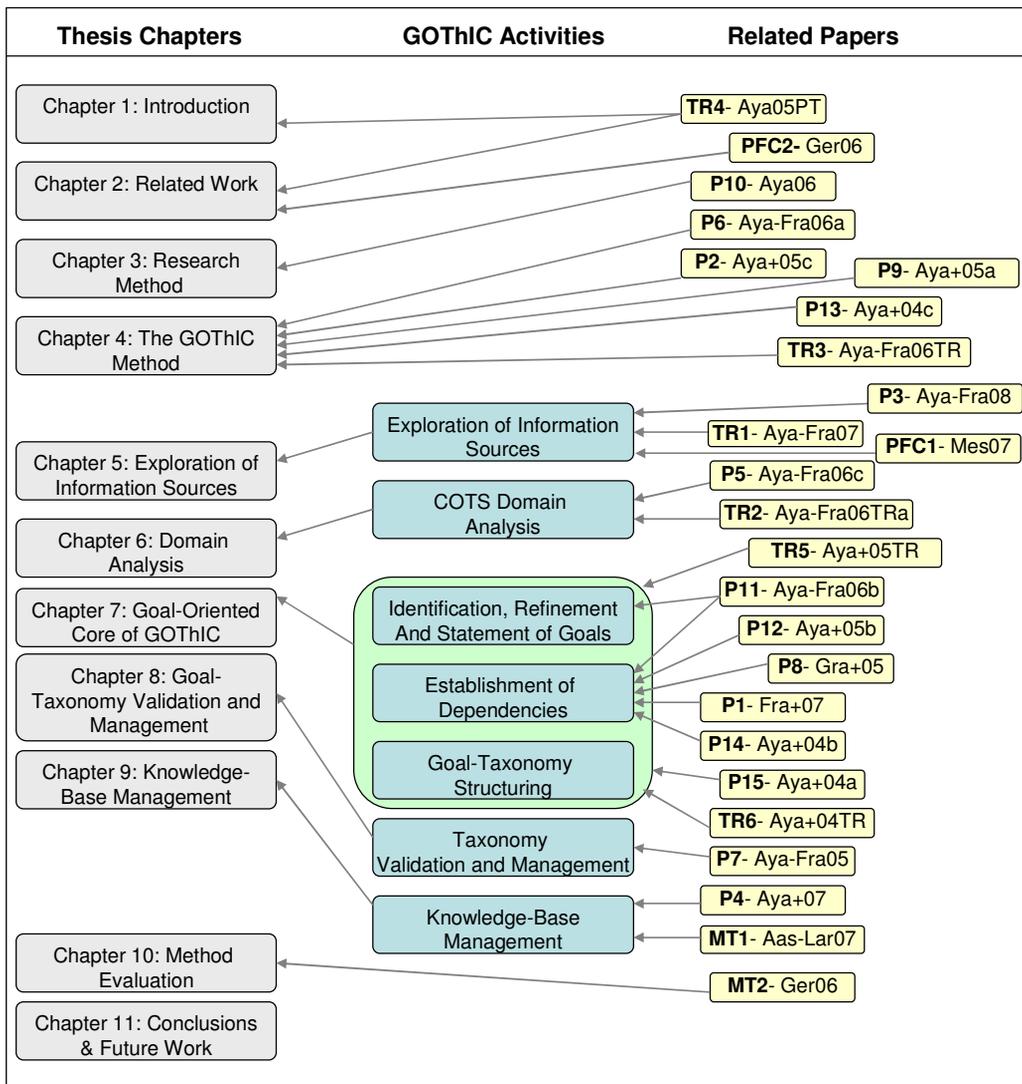


Fig. 1.1 Relationship among thesis chapters, GOTHIC activities and publications

Chapter

2

Related Work

The main purpose of this chapter is to position the work of this thesis by surveying the State-Of-The-Art and State-Of-The-Practice on COTS selection and searching processes. Moreover, the main topics used in shaping the method proposed in this thesis are discussed.

The Chapter is divided in 4 sections: Section 2.1 clarifies several definitions used through this thesis. Section 2.2 discusses the State-Of-the-Art in COTS selection processes and the wide range of works proposed to support COTS selection related issues. Section 2.3 tackles the State-Of-The-Practice and highlights the main industrial problems detected. Tables are used to summarize and compare the different approaches.

Finally, as this thesis builds upon existing work, in Section 2.4 some of the main approaches used to develop the GOTHIC method are briefly introduced, whilst many other important approaches are discussed as background in their corresponding method's activities description (Chapters 5-9).

2.1 Introduction

Traditionally, organizations developed systems from scratch with control over all or most of the pieces, following established process models in software engineering (e.g., Waterfall, Spiral, or Iterative process models). Regardless of which process model an organization used, they performed requirements, design, architecture, construction, integration and test activities. However, the use of COTS changes the focus of the development-centric approach assumed in traditional software development (i.e., custom development) by a procurement-centric approach [Bro+00], [Crn+05], [Kon-Hut07].

Table 2.1 provides a general overview of this fundamental software development change. It can be observed that the nature, timing, and order of activities performed and

the processes used differ accordingly. Furthermore, it can be inferred that using COTS is not merely a technical matter for system integrators, but many changes must be faced.

While some of these changes are obvious, others are quite subtle. Not only requirements engineering activities must change to support simultaneous consideration of system requirements and the marketplace, but also numerous technical, procurement, organizational, management, and business activities must also be adapted to deal with the challenges and risks of efficiently using COTS and exploiting their benefits [Voa98], [Crn-Lar02], [Mora+07].

In this context, it is obvious that the models traditionally used in custom-development do not address the COTS related activities [Mor+02], [Crn+05], [Kon-Hut07] making essential to perform further studies for supporting them. It is especially true in the COTS selection arena which some current open issues are discussed and addressed by this thesis.

Table 2.1 The Fundamental change among custom-development and CBSD

Traditional Software Lifecycle	Custom Development	COTS-Based Systems Development
Requirements	Creation of system requirements to create a software system that meets these requirements (the engineers are producers).	Creation of a set of flexible requirements followed by the COTS marketplace exploration for selecting components that best fit these requirements. The engineers are consumers who then integrate the products.
Design	Analyze requirements to produce a description of the internal structure and organization of the system that will serve the basis for its construction.	To integrate the products into a software system that meets the requirements. It implies an iterative trade-off process of requirements analysis, architecture, COTS availability, prioritization and negotiation.
Construction	Coding the detailed design to implement the system requirements.	Some requirement functionalities that were not addressed by any COTS are usually developed in-house. In any case, usually glue code is used to mediate components interactions; as well as bridges or adaptors to smooth over incompatibilities in the component interfaces.
Testing	Integration and evaluation of the product quality by verifying its behaviour by a finite set of test cases.	Although COTS are tested by the component provider, they should be retested by the user to assure their suitability and their good system integration.
Maintenance	Modification to code and associated documentation due to a problem or need for improvement	Due to maintenance effects, COTS-Based systems undergo a technology refresh and renewal cycle that has many implications.

2.1.1 COTS Definition

The literature about COTS addresses several development issues and is very heterogeneous in terminology; therefore, we provide a brief discussion of the different definitions to clarify the concepts used in this dissertation.

Firstly, it is important to stand out that there is a lack of consensus about the COTS characteristics and their definition. The term COTS applies to a broad range of products, which exhibit different issues [Car-Lon00], [Tor-Mor04], [Moh+07]. From the literature, the term COTS results very generic, covering a large variety of products. However, all agree that COTS are a special class of reusable components.

Indeed, COTS can be either software or hardware or a mixture of both. In this thesis, we only focus on software COTS, however most of the issues and advices are equally applicable to hardware.

Table 2.2 Different definitions of COTS, surveyed by [Mor-Tor02]

Source	Definition
Vigder and Dean [Vig-Dea97]	Define COTS as pre-existing software products, sold in many copies with minimal changes; whose customers have no control over specification, schedule, and evolution; access to source code as well as internal documentation is usually unavailable; complete and correct behavioural specifications are not available.
Carney and Long [Car-Lon00]	This approach considers Origin and Modifiability as attributes to define COTS. The possible values for these attributes are: <i>Origin</i> : Independent Commercial Item, Special Version of Commercial Item, Component Produced by Contract, Existing Components from External Sources, Component Produced In-house. <i>Modification</i> : Extensive Reworking of Code, Internal Code Revision, Necessary Tailoring and Customization, Simple Parameterization, Very Little or no Modification.
Basili and Boehm [Bas-Boe01]	Specify that COTS has the following characteristics: a) the buyer has no access to the source code; b) the vendor controls its development, and; c) it has a non-trivial installed base. This definition is more restrictive and does not take into account some types of software products like software products developed for special purposes and not widely deployed, special version of commercial software products and open source software.
Software Engineering Institute (SEI) [SEI]	A COTS product is: sold, leased, or licensed to the general public; offered by a vendor trying to profit from it; supported and evolved by the vendor, who retains the intellectual property rights; available in multiple, identical copies; and used without source code modification.

Sometimes the terms software package, Non-Developmental Item (NDI), out-of-the-box product, and shrink-wrap solution are used as COTS synonymous [Fra-Tor05]. In other cases, other labels are used to capture the source of the component, as GOTS (Off-The-Shelf Software owned by the government) and MOTS (Modifiable Off-The-Shelf).

In [Mor-Tor02] a survey about the different meanings and coverage of the term COTS is presented. An excerpt of this survey is stated in Table 2.2.

In addition, in a more recent work, Torchiano and Morisio [Tor-Mor04] gives a more detailed, empirically based definition stated as: “*A COTS product is a commercially available or open source piece of software that other software projects can reuse and integrate into their own products*”.

This definition is very generic. In fact, the Open Source Initiative (OSI) formed and established the Open Source Definition (OSD). The OSD is a formalization of what it means to distribute software that is open source, namely [OSI]:

- Free distribution (i.e., license cannot restrict selling or giving away)
- Source code (included)
- Derived works (i.e., software can be modified and distributed by others)
- Integrity of the author’s source Code (i.e., know who gets credit for the source code)
- Distribution of license (i.e., forbidding the addition of further restrictive licensing)
- License must not be specific to a product (i.e., rights cannot depend on a particular distribution)
- License must not contaminate other software (e.g., both licensed software and OSS can coexist in the same distribution)

The complete text and rationale of the OSD may be found in [OSI]. Under OSI (strictly speaking) a software product is in fact open source if and only if it conforms to the OSD.

From these definitions we can realize that they are different in their coverage. It is common that researchers and practitioners use the same word with different meanings. Some of them use the term COTS covering freeware and Open Source Software (OSS) as well as other kinds of components (e.g., [Ber+06], [Moh+07b]); whilst others are more restrictive (e.g. [Tau+04], [Bhu-Boe05]).

Currently, the term OTS (Off-The-Shelf) is used to refer COTS and OSS.

In order to be precise, in this thesis we follow the SEI definition of COTS [Mey-Obe02]:

“A COTS product is a [software] product that is:

- (1) sold, leased, or licensed to the general public;*
- (2) offered by a vendor trying to profit from it;*
- (3) supported and evolved by the vendor, who retains the intellectual property rights;*
- (4) available in multiple, identical copies; and*
- (5) used without source code modification by a consumer.”*

Consequently, we consider a COTS-Based System (CBS) as a computer based application that integrates one or more COTS, while COTS-Based System Development (CBSD) as the processes that lead to the development of a CBS.

2.2 State-of-the-art

2.2.1 COTS Selection Approaches

This section discusses existing COTS selection approaches and summarizes their contribution to the evolution of COTS selection practices. Subsequently, the main open issues of existing COTS selection approaches with respect to the topics addressed by this thesis are highlighted.

COTS selection has been considered a relatively immature area [Rei+03]. As COTS research has progressed, several approaches have been carried out for dealing with diverse aspects of COTS selection.

Despite there is no commonly accepted method for COTS selection [Ruh03], [Moh+07] all methods share the next high-level, iterative and overlapping steps:

- ◆ *Searching COTS from the marketplace*
- ◆ *Evaluating candidate COTS with respect to the system requirements*
- ◆ *Deciding the best COTS from a set of competing alternatives.*

Some of the required roles to perform the COTS selection process and the activity of documenting the decision are sketched in Table 2.3.

Table 2.3 Activities and Roles in COTS Selection

Activity	COTS Users Roles
Searching Candidates COTS	<i>Market Watcher (MW)</i> explores the marketplace segments to find components and information about them that may match the established requirements.
Evaluating Candidate COTS	<i>Quality Engineer (QE)</i> measures the factors that are related to the requirements in the candidate components.
Deciding COTS Component(s)	<i>Selector (S)</i> takes the final decision based on the evaluation of the candidates and also taking into account other relevant information (mainly organizational).
Documenting the Decision	<i>Knowledge Keeper (KK)</i> stores and documents the produced information and the decisions taken in the process for their future use in forthcoming selection processes.

In [Moh+07] the improvements made to the COTS selection process over the last decade are highlighted. Fig. 2.1 shows such progression with respect to some representative approaches.

It is considered that the first widespread selection method was the OTSO (*Off-The-Shelf Option*) Method, it was proposed by Kontio in 1995 [Kon95]. The method was further elaborated in [Kon96] and [Kon+96]. This method defines the basic structure of COTS selection methods and serves as a basis for other approaches. OTSO comprises 3 phases: searching, screening, and evaluation. It provides specific techniques to define the criteria for searching, evaluating and comparing the cost and benefits of alternative products. The Analytic Hierarchy Process (AHP) [Saa-90] is used to consolidate the

evaluation results for decision-making. However, to perform the screening phase, no suitable techniques were provided to find the candidate COTS in the marketplace.

In 1997 several approaches were proposed, examples are the IusWare approach [Mor-Tsu97], PRISM (*Portable, Reusable, Integrated, Software Modules*) [Lin+97], and CISD (*COTS-based Integrated Systems Development*) [Tra-Liu97]. Some of the main concerns of these approaches were related with the need of formalization of the COTS selection activities (mainly evaluation) [Mor-Tsu97] and providing a generic architecture to be used during COTS evaluation [Lin+97].

The proposal of Tran and Liu [Tra-Liu97] is related with the fact that COTS are designed to meet the needs of a marketplace instead of satisfying the requirements of a particular organization; therefore, it is not possible to ensure that the COTS available will meet all stated requirements, and usually there is no single COTS that satisfies all of them. Hence, they realized the need of supporting the processes of selecting multiple homogeneous COTS.

1995	<ul style="list-style-type: none"> ♦ The basic structure of COTS Selection Process (CSP): OTSO
1996	<ul style="list-style-type: none"> ♦ Further elaboration of OTSO.
1997	<ul style="list-style-type: none"> ♦ Formalization of CSP ♦ Generic component architecture ♦ Multiple COTS selection
1998	<ul style="list-style-type: none"> ♦ Requirements Engineering process for CSP
1999	<ul style="list-style-type: none"> ♦ Studying the effects of social factors
2000-2001	<ul style="list-style-type: none"> ♦ Tailorability of the evaluation process: ♦ Further refinement for the requirements engineering process (ongoing project).
2002	<ul style="list-style-type: none"> ♦ Detailed tailorable process ♦ Use of screenshots and use-cases for requirements ♦ Multiple COTS selection
2003	<ul style="list-style-type: none"> ♦ Risk-driven evaluation ♦ Use of fuzzy theory and optimization techniques ♦ Use of models to decide the suitability of COTS
2004	<ul style="list-style-type: none"> ♦ Emphasis on non-functional requirements ♦ Using quality models during the evaluation.
2005-2006	<ul style="list-style-type: none"> ♦ Systematic handling of mismatches between COTS attributes and requirements
Future	<ul style="list-style-type: none"> ♦ ?

Fig. 2.1 Evolution of COTS selection practices [Moh+07]

In 1998, the importance of a suitable requirements engineering process for CBSD was further evidenced. In this context, the PORE (Procurement Oriented Requirements) [Mai-Ncu98] approach represented a key milestone. PORE [Mai-Ncu98] is a template-based approach supporting iterative evaluation and selection of COTS. Its model identifies four goals: (a) acquiring information from stakeholders, (b) analyzing the information to determine if it is complete and correct, (c) making the decision about product requirement compliance if the acquired information is sufficient, and (d)

selecting one or more candidate COTS. The elicitation of features of existing COTS and requirements engineering are conducted in parallel using an iterative process of requirements acquisition and product evaluation. PORE's iterative process selects products by rejection (i.e., the products that do not meet core customer requirements are selectively and iteratively rejected and removed from the candidate list). However, the method does not address how existing COTS are gathered. A prototype tool known as PORE Process Advisor was developed to support the PORE approach. The SCARLET approach [Mai+02], (formerly named BANKSEC) published in 2002, is the successor of the PORE method. It adapts PORE to the banking domain and also enables multiple selections. However, the problem of gathering suitable COTS from the marketplace is still opened.

Along 1999 approaches as STACE (*Social-Technical Approach to COTS Evaluation*) [Kun-Bro99] (further elaborated in [Kun03]) emphasized the social and organizational issues to COTS selection process as well as the importance of non-technical factors such as business and vendor capabilities during the evaluation process. However, the process of how to get this kind of information about the components was not addressed.

Subsequently, since the selection of COTS involves an extensive process of requirements analysis, prioritization, and negotiation; approaches as CAP (*COTS Acquisition Process*) [Och+01], CRE (*COTS-Based Requirements Engineering method*) [Alv-Cas01], CEP (*Comparative Evaluation Process Activities*) [Phi-Pol02], PECA (*Plan, Establish, Collect, and Analyze*) [Cor+02], and StoryBoard [Gre+02], emphasized the decisive importance of the evaluation process and its tailoring.

CAP [Och+01] addressed the concept of "tailorable evaluation process" for highlighting that the COTS evaluation process should be tailored based on the available effort for each project. For tailoring the process, they made use of expert's knowledge.

The CRE approach [Alv-Cas01] was developed to facilitate a systematic, repeatable, and requirements-driven COTS selection process. The method focuses on non-functional requirements to assist the process of selection and evaluation of COTS. It has four iterative phases: identification, description, evaluation and acceptance. The identification phase is based on a careful analysis of influencing factors; there are five groups of factors: user requirements, application architecture, project objectives & restrictions, product availability and organizational infrastructure. During the description phase, the evaluation criteria are elaborated in detail. In the evaluation phase, a particular COTS product is selected based on estimated cost versus benefits.

The CEP approach [Phi-Pol02] claimed that the more reliable the source of data is, the higher confidence on the results is. Therefore they introduce the use of a Confidence Factor (CF). Any estimate made in the evaluation process should be adjusted based on the CF value of the source based on which these estimations are made.

The PECA approach [Cor+02] from the Software Engineering Institute (SEI) [SEI] proposes a COTS selection process which can be tailored by means of proposed guidelines. It is important to mention that SEI has been greatly contributing to the advancement on COTS selection practices since many years ago. Examples are [Bro+00], where they published a set of guidelines for developing CBD processes; the APCS (*Assembly Process for COTS-based Systems*) approach [Carn+03], a generic process framework for developing software systems based on COTS; and quantitative

methods for software selection an evaluation [Ban06]; as well as many others COTS related works that can be consulted at [SEI].

The StoryBoard approach [Gre+02] suggests incorporating use-cases and screen-captures during the requirements engineering process to help customers understand their requirements, and thus acquire more appropriate COTS.

Several other approaches making use of different techniques were put forward as the DBCS (*Domain-Based COTS Selection*) method [Leu-Leu03]; the composable process elements for developing CBS proposed by Boehm et al [Boe+03a] that makes use of the WinWin Spiral model [Boe+03b], the approach presented by Erol et al [Ero-Fer03], and the OPAL methodology and its associated tool [Kry+03] oriented to provide COTS selection support to some organizations in Luxemburg.

The DBCS [Leu-Leu03] makes use of specific domain models to decide the suitability of COTS products. This approach has as goal to reduce the amount of work required for the selection process by reusing a domain model of the domain. However, it does not tackle how the domain model should be constructed neither how candidate COTS are searched in the marketplace.

The WinWin Spiral model [Boe+03b] makes use of the risk-driven paradigm to identify, analyze and resolve risks in an iterative evaluation process, whilst the approach of Erol et al. [Ero-Fer03] suggests the use of fuzzy theory to quantify qualitative data, as well as optimization techniques to determine optimal solutions.

At this point, a common shortcoming of most COTS selection proposals was that they put more emphasis on functionality and cost factors than on non-functional requirements. Thus, some researcher and practitioners as Beus-Dukic and Bøegh [Beu-Bøe03] claimed that the role of non-functional requirements becomes more important than functional requirements in regard to COTS selection. It is worth noting that it is because COTS have their functionality already built-in [Car+03].

In this sense, some approaches made a great emphasis on managing non-functional requirements driving COTS selection by using quality models as [Fra-Car03] and [Bur+02]; this last addressing combined selection of COTS. The proposal of Franch and Carvalho [Fra-Car03] was further elaborated in the COSTUME (*COTS-based System qQuality Model dDevelopment*) method [Car+04c] and supported by a tool called DesCOTS system [Gra+04].

A special mention deserves a project started since 2001 by Chung et al. in order to define a more complete COTS selection approach called CARE (COTS-Aware Requirements Engineering) [Chu-Coo04]. It draws upon the ideas of existing methodologies including RUP [RUP] and PORE [Mai+02]. The goal is to define an agent- and goal-oriented methodology that supports the definition and selection of COTS from a technical view. Using goals, the traceability relationships among the produced artifacts are established and maintained (e.g. a soft-goal is refined and traced to specific system requirements). For each COTS, they capture: their goals (soft goals and hard goals) and their detailed specification. Thus, COTS are stored and maintained in a knowledge base, or repository. Departing from COTS descriptions, searches are enabled to determine which products appear to be potentially useful. However, despite pretending to fill the gap of previously presented methods and improve reuse by means

of this COTS descriptions repository, the CARE approach also share some obvious drawbacks with most existing proposals: 1) They assume that COTS candidates already exist as the system requirements are under development. 2) All of them assume that a COTS repository exists but any work addresses how to build such repository. 3) Finding COTS information to describe COTS in the repository is not an obvious task, and they do not address how to obtain it. 4) The maintainability and trustworthiness of the information in the repository is especially complex since COTS marketplace characteristics are not greatly dealt. 5) The searching process is not very efficient having to look for components in a widespread range of descriptions.

As it can be observed, although the COTS selection area has been greatly active in the last decade, in recent years a plethora of proposals and studies have been put forward to support COTS selection practices from many perspectives. Some examples are: [Ye-Kel04], [Dub-Fra04], [Sai+04], [Yeo-Mil04], [Wan-Far05] [Bhu-Boe05], [Jin+05], [Car05T], [Don+05], [Hen+05], [Req+05], [Bar+05], [Mic+05], [Man-And05], [Wan-Hom05], [Shy-Shi06], [Ban06], [Vaf+06], [Sas+06], [Zhe+06], [How-Lig06], [Car06], [Car-Fra06], [Crm+06], [Lin+07], [Moh+07b], [Bhu+07], [Man+07], [Rom-Ken07].

For instance, in 2005, based on empirical data gathered from five years of developing e-services applications, Bhuta and Boehm observed that projects that mitigated the risk of component interoperability earlier in the project development cycle were more successful during the integration phases than those projects that had neglected the component interoperability issues during their component selection. Thus, they developed and applied a method for component selection that focuses on piecewise evaluation, as well as the interoperability between the candidate components [Bhu-Boe05], [Bhu+07].

The use of quality models to drive COTS evaluation is being also explored (e.g., [Sed+03], [Car05T], [Raw-Mat06], [Ber+06], [Car+07a]); whilst several others selection methods are being also proposed describing useful activities to be performed (e.g. [Don+05], [Jin+05], [Sas+06], [Lin+07]).

Other works propose to add novel technologies emerging from other areas as decision support systems, method engineering, strategic contracting and procurement, simulation and formal reasoning. For instance, in [Coo+05], the use of finite value logic, fuzzy logic algorithms are being investigated for the selection of COTS; and [Moh+07b] introduces a process for handling mismatches between COTS and requirements using techniques as linear programming to identify near optimal solutions. It is called MiHOS (*Mismatch-Handling aware COTS Selection*) and aims at addressing COTS mismatches during and after the selection process. A tool support, namely MiHOS-SA which stands for MiHOS-Sensitivity Analysis is provided in order check the robustness of MiHOS's results against input errors.

On the other hand, a slight extent of industrial case studies, experiences and empirical studies have been published (e.g. [Jen03], [Rei+03], [Tor-Mor04], [Min04], [And04], Li+05], [Li+06], [Li06], [Kei-Tiw05], [Hsu-Wid06], [Sta-Lub06], [Don+06], [Car+07b]). However, it is evident that more empirical data which analyzes the advantages and drawbacks of selecting COTS in real software organizations is needed [Li06].

We may conclude that although many COTS selection issues still remain open, existing approaches, whilst of different effectiveness and suitability for different contexts, have contributed to the advance of COTS selection practices. However, more empirical data is needed to drive COTS selection research into the right path. A survey analysis is provided in the next subsection.

2.2.1.1 Analyzing COTS Selection Approaches

Table 2.4 Comparing some representative methods dealing with COTS selection

APPROACH		COMPARISON FACTORS											
Name	Year	SEARCH	IDENT	EVAL	SNG	MLT	REUSE	TAILOR	TS	CVR			
										MW	QE	S	KK
OTSO	95/96	-	-	√	√	-	-	-	-	-	*	√	-
IusWare	1997	-	-	√	√	-	-	-	-	-	√	*	-
PORE	1998	-	-	*	√	-	-	*	√	-	√	*	-
STACE	1999	-	*	√	√	-	-			-	√	√	-
CAP	2001	-	-	√	√	-	-	√	-	-	√	√	-
CRE	2001	-	-	√	√	*	-	-	-	-	√	√	-
CEP	2002	-	*	√	√	-	*	-	-	-	√	√	*
CARE	2001	-	*	*	√	-	*	-	√	-	√	√	*
PECA	2002	-	*	√	√	-	-	√	-	*	√	√	-
StoryBoard	2002	-	-	*	*	√	-	-	-	-	*	*	-
Combined Selection	2002	-	*	-	*	√	-	-	-	-	*	*	-
SCARLET	2002	-	-	*	√	√	-	*	√	-	√	*	-
DBCS	2003	-	-	*	√	-	*	*	-	-	*	√	-
WinWin	2003	-	-	√	√	*	-	-	-	-	*	√	-
COSTUME	2004	-	-	*	√	√	√	√	√	-	√	√	√
MiHOS	2007	-	-	√	√	-	-	-	√	-	√	√	-

(√) fully satisfies the criterion (*) partially, informally or implicitly satisfies the criterion
 (-) does not satisfy the criterion

To highlight the main open issues of existing COTS selection approaches, some of them are compared in terms of the following criteria:

- a. SEARCH: Support they offer to the process of searching COTS from the whole marketplace.
- b. IDENT: Support they offer to the identification of candidate COTS and information about them.
- c. EVAL: Evaluating candidate COTS
- d. REUSE: Reuse of knowledge gained in each selection process
- e. SNG: Suitability for single COTS selection
- f. MLT: Suitability for multiple COTS Selection
- g. TAILOR: Tailorability of the process to the COTS selection needs.
- h. TS: Availability of tool support to facilitate the application of the approach.

- i. CVR: Coverage to the COTS selection involved roles introduced in Table 2.3.

Table 2.4 shows an excerpt of some of the COTS selection methodologies and their coverage to the criteria exposed above. From such table, it is noticed that regardless of the different effectiveness and suitability for different COTS selection contexts and activities, the focus of most of the existing methods is on evaluating COTS alternatives with respect to the system requirements. Yet, it is necessary to address many issues regarding how to search and identify COTS from the marketplace and reuse knowledge about them.

In general, most of these methods address adequately some of the complex characteristics of the COTS selection process. However, their main focus is to apply analysis, evaluation and decision techniques in the selection process. Most of them assume that COTS candidates or a repository with enough information about them already exist but do not address their construction. Just a few of them informally address or suggest that support is needed to search, identify, and document COTS from the marketplace.

The major shortfalls of most COTS selection methods in the reviewed literature are:

- (1a) Lack of approaches that deal with all dimensions (i.e., functional aspects, non-functional, non-technical, quality, etc.) required to select COTS, as well as the lack of guidelines to support the users to tailor the method to their own processes.
- (2a) Lack of techniques and mechanisms to document COTS. Most current approaches assume that the knowledge required for evaluating COTS is available and reliable. However, it is not in practice [Ber+03]. COTS related information is highly heterogeneous and widespread making necessary to provide mechanisms to find and assess it.
- (3a) Absence of adequate mechanisms for helping the searching, identification, and documentation of candidate components in the global marketplace [Aya06], [Tau+04]. Even when some of the existing methods recognize the importance of searching candidate components to be evaluated, they do not specify how to search and identify these components; most of them assume that they already exist in a common place (i.e., a repository). (Section 2.2.3 describes different approaches trying to deal with some searching and identification issues).
- (4a) Most of the methods lack of knowledge management and reuse mechanisms. While most of the methods recommend documentation of the selection process, they do not address adequate mechanisms for recording and managing this body of knowledge. Thus, useful knowledge gained in previous experiences tends to be lost.

Furthermore, several works (e.g., [Mor+00], [Clar+04], [Moh+04], [Wan-Hom06]) advocate that having a repository for COTS products with enough information about them is becoming a necessity for improving the CBSD practice.

2.2.2 Software Reuse Infrastructures and Knowledge Bases

Systematic software reuse is an engineering strategy proposed to increase productivity and software quality and lead to economic benefits. Although COTS have been envisaged as reusable components, their efficient reuse imposes many critical tasks. It is well known that to reuse software components, re-users must be able to find and understand the components that best fit their needs. If the process fails, reuse can not happen [Fra-Pol94], or even worst erroneous components may be selected causing critical problems to the software development project.

In this context, how to index and represent COTS so that they can be found and understood are two important issues in enabling their efficient reuse [Bas+00], [Rav-Rot03].

There are two aspects in software reuse:

- ♦ **Developing for reuse.** It refers to developing components so that they can be reusable (e.g. the development of COTS).
- ♦ **Developing with reuse.** It refers to the process of developing systems by using reusable components (e.g. CBSD).

In this thesis we focus on developing with reuse.

A typical systematic software reuse environment can be summarized as follows: Reusable software components must be indexed and stored in a software repository (also called software component library or reusable software library), in such a way that they can be found and understood by re-users. Re-users search the repository for the components, and if they meet requirements, incorporate them into new applications. The structure of the repository is key in obtaining good location and retrieval results. Fig. 2.2 illustrates this process.

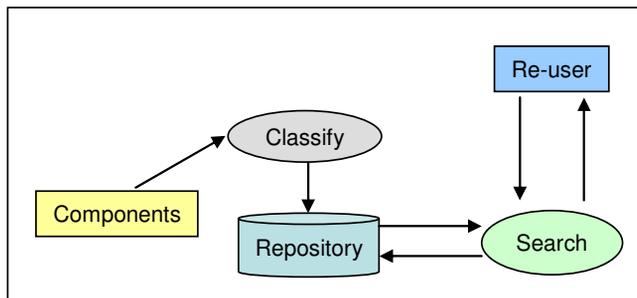


Fig. 2.2 Components reuse environment

Knowledge may also be reused and knowledge reuse is partly reflected in reuse of architectures, templates, processes or lessons learned from previous components selection. To be successful, many authors claim that reuse must embrace the reuse of components and experiences. This is the main idea of the so-called “*Experience Factory*” introduced by Basili et al [Bas+94b] which enhances organizational learning by promoting an organizational reuse infrastructure aimed at the storage and reuse of all sort of knowledge (experience and products) resulting from the activities performed in

the software lifecycle. Thus, having a knowledge base to store and retrieve information during different component selection processes involves a lot of advantages, for instance it reduces overall required evaluation time and effort whilst increase the reliability of results. More recently, such concept has been enlarged to promote the concept of *Learning Software Organizations* (LSO) [Ruh01] that are based on the same principle as the Experience Factory but enhancing the reuse of knowledge into the organizations.

Systematic software reuse has been a popular topic of debate and discussion for over 30 years in the software community. A considerable number of research lines and approaches have been put forward to support software reuse related activities. The reuse community initially concentrated its research on technical issues, such as repositories, search-based tools, and domain-specific languages; as well as information retrieval, information and knowledge acquisition, knowledge management and representation (see [Luc+04] for a comprehensible survey). Nevertheless, in practice, all these traditional approaches for building, maintaining, and browsing software reuse repositories have suffered from lack of domain-specific components, a heavy “fill-up” investment upfront, and under-critical information relevance later on [Fra-Fox95], [Mor+02b], [Li+04], [Luc+07]. This has been associated to some problems as the incomplete, unreliable, and too static characterization of components [Pou95], [Sea99], [Luc+04].

In the context of CBSD these low success rates are in addition related with the actual difficulty to deal with specific characteristics of the COTS marketplace and that traditional approaches are mostly oriented to intra-organizational environments where the reusing organization has control on the evolution of the functionality and assumptions of the assets, which is not the case of COTS reuse. Therefore, the need of new approaches supporting COTS reuse (i.e., extra-organizational reuse) has been widely recognized [Mor06].

2.2.2.1 Classification and its Central Role in Implementing Reuse Infrastructures

Classification (i.e., indexing) is central to the software reuse practice [Pri87]. A well-defined classification structure is essential to the design of an effective storing, searching and retrieval mechanism. Put simply, for reuse to be successful it must be easy to locate components with the same or similar functionality in order to reduce the probability of retrieving non-relevant components and make feasible their comparison in a component selection process.

Representation is an inherent problem of classification. The role of representation in a reuse environment is fundamental. Representation is defined as a language (textual, graphical, etc.) used to describe a set of objects. For instance, books in a library are represented by bibliographic records in a library catalogue. A representation allows operations that would be more difficult or impossible on the represented object itself. It means, it is much easier, for example, to sort a set of bibliographic records than to sort the same number of books.

Domain analysis is the process of acquiring and consolidating information about an application domain so that a reusable infrastructure can be designed reliably [Fra+98]. Both domain analysis and classification techniques have been used to derive

representations of reusable components in a domain in order to constitute a domain reuse infrastructure.

Classification Concept

The inherent concept of classification implies many terms that are often confused and used interchangeably as *taxonomy*, *typology*, *ontology*, *directory*, *cataloging*, *categorization* and *classification* are. For clarity, some of them are defined here [ClaSoc]:

- *Categorization*: is the process of associating a document with one or more subject categories. So the entry for a page on cross trainer shoes could go into Running, Manufacturing, and Sports Medicine. All of these are legitimate, depending on the context.
- *Typology*: is the study or systematic classification of types that have characteristics or traits in common. This idea is the basis for most typological constructions, particularly of stone artifacts where essential forms are often thought of as ‘mental templates’, or combinations of traits that are favoured by the maker.
- *Cataloging*: come from libraries, where specialists enter the metadata (such as author, date, title, and edition) for a document, apply subject categories to it, and place it into a class (such as a call number) for later retrieval. It tends to be used interchangeably with *Categorization*.
- *Directory*: is an organized set of links, like those on Yahoo or the Open Directory Project, which allows a web site to display the scope and focus of its content. A directory can cover a single host, a large multi-server site, an intranet, or the Web. At each level, the category names provide instant context information to users. Rather than a simple list, such as the results of a search, drilling down into the more and more specific categories (for example Shopping>Clothing>Footwear>Athletic) explains how the pages fit into the larger set of information.
- *Clustering*: is the process of grouping documents based on similarity of words, or the concepts in the documents as interpreted by an analytical engine. These engines use complex algorithms including Natural Language Processing, Latent Semantic Analysis, Bayesian statistical analysis, and so on.
- *Thesaurus*: is a set of related terms describing a set of documents. This is not hierarchical: it describes the standard terms for concepts in a *controlled vocabulary*. Thesauri include synonyms and more complex relationships, such as broader or narrower terms, related terms, and other forms of words.
- *Taxonomy*: is the organization of a particular set of information for a particular purpose. It comes from biology, where it's used to define the single location for a species within a complex hierarchic. Biologists have arguments about where various species belong, although DNA analysis can resolve most of the questions. In informational taxonomies, items can fit into several taxonomic categories.
- *Ontology*: is the study of the categories of things within a domain. It comes from philosophy and provides a logical framework for academic research on knowledge

representation. Work on ontologies involves schema and diagrams for showing relationships in Venn diagrams, trees, lattices, and so on.

Regardless the variety in this terminology, by classification, we mean ‘the ordering of entities into groups or classes on the basis of their similarity’ [Bai94].

Classification systems structure a body of knowledge that constitutes a field, allowing generalizing, communicating, and applying new findings. The advantages of using a classification structure in software reuse are many [Pri85]. Some of them are:

1. Classification provides a means of neatly organizing reusable objects and quickly retrieving them when needed.
2. It enables broadening and narrowing of searches, improving the recall and precision rate of the searches made. Recall is the number of relevant items retrieved over the number of relevant items in the database. Precision is the number of relevant items retrieved over the number of all items retrieved. Recall and precision are the classic measures of the effectiveness of an information retrieval system.
3. It also serves another purpose by acting as maps of knowledge content. We are able to comprehend the knowledge stored in a repository not by merely knowing how many objects it has, but also by distributing those numbers across various categories and sub categories.

Thus, classification schemas are used to create an index to assist in the physical storage of components and to provide input to search tools. The method classification is an important ingredient in determining the types of indices that can be used, the types of searches that can be conducted, and the type of tools that re-user can or must use [Pou-Ygl93].

2.2.2.2 Kinds of Software Classification Schemas

Various methods to classify and therefore represent reusable software in intra-organizational contexts have been proposed and implemented, including numerous formal and automated techniques [Kau05]. They can be classified into three broad families: library and information science, artificial intelligence and hypertext [Fra-Gan90].

In [Ye-Lo01] a hierarchy of current kinds of classification techniques is given. It is illustrated in Fig. 2.3.

Artificial intelligence based and hypertexts based techniques have been used only experimentally, examples may be found in [Fra-Pol94]. Most of the methods used in industry are those from the library and information science, therefore, in this document, we will only describe this kind of classification technique. Library and information science methods break into two main categories: controlled vocabulary and uncontrolled vocabulary.

A controlled vocabulary consists of a list of predefined terms that may be used to describe and classify reusable components. The advantage of this approach is that the search can make use of broader and narrower terms relationships within the controlled vocabulary, if such relationships are defined. A controlled vocabulary built in such a

way (i.e. a thesaurus or a glossary) represents a domain knowledge model for the domain of application. The disadvantage of this method is that it needs a lot of maintenance effort because the controlled vocabulary must be adapted regularly.

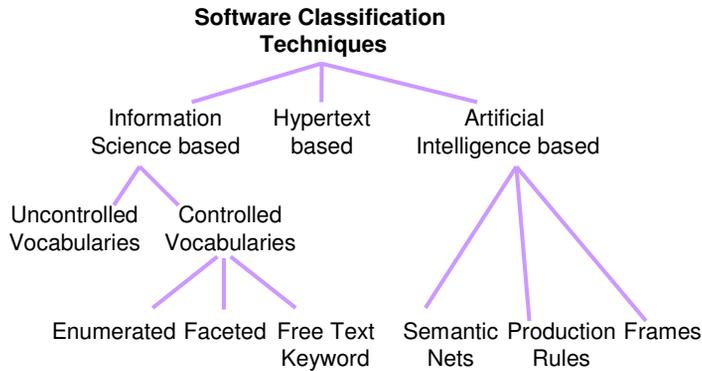


Fig. 2.3 Existing kinds of software classification techniques

Uncontrolled vocabularies do not place restrictions on the terms and syntax that can be used to classify and describe a reusable component.

Enumerated classification is a well-known retrieval method used by the Dewey Decimal system. In this method, information is placed in categories that are usually structured in a hierarchy of sub-categories. The appeal of a classification scheme is the ability to iteratively divide an information space into smaller pieces that reduces the amount of information that needs to be perused. The issues involved in using an enumerated classification include its inherent inflexibility and problems with understanding large hierarchies. There is a trade-off between the depth of a classification hierarchy and the number of category members. Some domains will lend themselves to many small classes. The effect is that users unfamiliar with its structure will become lost in the morass of possible classes [Hen97]. Other domains will have few categories, but must necessarily contain many members. In this case, selection of a class is only a first step in the retrieval process, as the user must then search a large number of category members for relevant information. Another issue is that once the hierarchy is in place, it gives only one view of the repository. Changes to that view may reverberate throughout the taxonomy, resulting in extensive redesign of class structures that can have consequences for the entire contents of the repository. Hence, the labor-intensive nature of enumerated classification remains a significant barrier to creating multiple structures. Enumerated classification requires users to understand the structure and contents of the repository to effectively retrieve information.

Faceted classification avoids enumerating component definitions in a hierarchy by defining attribute classes that can be instantiated with different terms [Pri85]. This is a variation of the relational model in which terms are grouped into a fixed number of mutually exclusive facets. Users search for components by specifying a term for each of the facets. Within each facet, classification techniques are used to help users to choose appropriate terms. This is very similar to the attribute-value structures used in a number of frame-based retrieval techniques in artificial intelligence [Bra+91], [Dev+91], [Pat+84], except that faceted techniques use a fixed number of facets (attributes) per

domain. Facets are more flexible than enumerated schemes because individual facets can be re-designed without impact on other facets. But some of the usability problems remain. While facets make it easy to synthesize and combine terms to represent components, it becomes hard for users to find the right combination of terms that accurately describe the information need, especially in large or complex information spaces [Fra-Gan90]. The method also requires that users know how the library and terms are structured, and have an understanding of the significance of each facet and the terms that are used in the facet [Cur89]. Field use of faceted retrieval systems has shown the need for training people to use facets effectively, and even more extensive training is necessary for designing faceted information domains [Pri91].

Free-text indexing methods use the text from a document for indexing. Document text is applied to a 'stop list' to remove frequently occurring words such as 'and' and 'the'. The remaining text is used as an index to the document. Users specify a query using key words that are applied to the indices to find matching documents. Since the process can be automated, no human classification effort is really required. However, human indexers are sometimes used to augment automatically extracted index terms. Matching criteria can range from Boolean match to more sophisticated methods, such as the vector model, that use statistical measures to rank retrieved information [Sal-McG83]. Free text methods are simple to build and retrieve from, but rely on regularities in linguistic texts that need large bodies of text to become statistically accurate. The non-linguistic nature of source code and the fact that clear and accurate documentation is not necessary for working code make these methods less attractive for software component repositories than for text documents. Free-text methods are most applicable to domains with extensive documentation, but it would be inaccurate to characterize most source code as being documented adequately for these methods. Although retrieval effectiveness of free-text methods have been questioned [Bla-Mar85], the low cost of building the repository coupled with adequate performance has made this approach popular in commercial text retrieval systems, and World Wide Web engines such as Yahoo! or Alta Vista.

In Table 2.5 we provide a summary of the mentioned reusable software classification schemas that have been used in intra-organizational contexts. The aim of such this table is to high-light relevant drawbacks that make these approaches inadequate to characterize COTS.

An identified problem is helping users understand reusable software components. This is important because if software engineers cannot understand components, they will not be able to reuse them. Current methods for representing reusable components are inadequate [Frak05]. A study of four common representation methods for reusable software components showed that none of the methods worked very well for helping users understand the components [Fra- Pol94].

In general existing approaches offer highly static and hard to understand classification schemas. Part of the problem is that most of them assume that an information space can be adequately represented with a single classification. But no classification is correct under all circumstances, and it is impossible in principle to identify all possible relevant and future features of the COTS marketplace.

We may conclude that although component reuse research is a very rich software engineering area (see [Par-Con07a] for a survey), most of the proposed works have assumed that reuse would take place in-house, with centralized repositories [Fra-Kan05] that are out of the COTS reuse reality.

As a result, the software engineering community has realized the need of more understandable and flexible approaches for characterizing and reusing COTS.

Next subsection provides an overview of the current proposals.

Table 2.5 Relevant drawbacks of current reusable software classification schemas

Classification Approach	Main Characteristics
Enumerated	<ul style="list-style-type: none"> • Inherent inflexibility and problems with understanding large hierarchies. • Users unfamiliar with the structure of the classification will become lost in the morass of possible classes. • Once the enumerated schema is in place, it only gives one view of the repository. • Changes to that view may reverberate throughout the taxonomy, resulting in extensive redesign of class structures that can have consequences for the entire contents of the repository.
Faceted	<ul style="list-style-type: none"> • It uses a fixed number of mutually exclusive facets. • Facets are a little bit more flexible than enumerated schemes because individual facets can be re-designed without impact on other facets. • It becomes hard for users to find the right combination of terms that accurately describe the information need, especially in large or complex information spaces • It requires that users know how the library and terms are structured, and have an understanding of the significance of each facet and the terms that are used in the facet.
Free-Text	<ul style="list-style-type: none"> • No considerable human classification effort is required. • It is more attractive for searching text documents than for software component repositories. Therefore, they are most applicable to domains with excessive documentation.

2.2.3 COTS Classification Approaches

From the increasing need of having COTS search and identification mechanisms to achieve more efficient and reliable selection processes, many works have been proposed to deal with classification of COTS by different classification mechanisms. Although the classification of reusable components have been an active area since several years ago e.g., [Pri91], [Gla-Ves95], the COTS classification area has recently emerged.

Several recent works arrange COTS by means of attributes for identifying relationships between characteristics of products and their impact on CBSD.

For instance, Carney and Long [Car-Lon00] propose the classification of COTS products using a bi-dimensional Cartesian space and report some examples that

populated this space. The dimensions they define are origin and modifiability. The origin dimension addresses the way the product is produced and they propose the following possible values: Independent Commercial Item, Special Version of Commercial Item, Component Produced by Contract, Existing Components from External Sources, Components Produced In-house. These values can describe products ranging from in-house components developed on purpose to commercial components ready to use with a large number of customers. The modification dimension describes to which extent the product either can or must be modified by the system developer that uses the component. This attribute has five possible values: Extensive Reworking of Code, Internal Code Revision, Necessary Tailoring and Customization, Simple Parameterization, Very Little or no Modification. Two of them assume access to code (extensive reworking, internal code revision), two (necessary tailoring, parameterization) imply some mechanisms built into the COTS to modify its functionality.

Morisio and Torchiano [Mor-Tor02] extended the work proposed in [Car-Lon00], proposing a classification schema for COTS products. They depart from the idea that different research works often adopt different implicit definition of COTS, thus making difficult comparing them and evaluating the applicability of proposed approaches. The purpose of their framework is twofold: first it is a way to precisely define the meaning of COTS, second it represents a way of specifying which sub-classes of products are addressed by a given work. The aim of their work is mainly classification. This proposal is similar to [Tor+02] and [Jac-Tor02] that emphasizes the assessment of the reuse of attributes.

In [Ack+02] a classification schema for business components is provided. Components are characterized into seven levels: Marketing, Task, Terminology, Quality, Interaction, Behaviour, and Interface; whilst in [Li+04] an empirical study on COTS classification is provided and in [Bia+03] a set of parameters characterizing COTS are identified and empirically assessed.

A special mention deserves the characterization of COTS for the business application market segment presented by some members of our GESSI research group [Car+04] since it was the origin of the research work exposed in this thesis. Such work proposed to characterize the business application COTS market segment by means of “characterization attributes” (this concept was taken from [Mor-Tor02], [Jac-Tor02]) to discriminate among different COTS categories and market segments. However, the research performed in such work was in the context of quality models reuse, and the rationale behind was not rigorous enough. Therefore, some key problems were identified among we can mention: 1) The classification schema was static and specific to the Business Applications market segment; 2) COTS marketplace characteristics were not taken into account; 3) The way to identify the discriminating characterization attributes (which capture the relevant information for discriminating categories) was not defined; it was based on common sense; 4) It was not defined how to deal with all COTS related information because the taxonomy presented departed from an existent taxonomy which was only restructured.

More recently, the work presented by Erofeev and Di Giacomo [Ero-Gia06] proposes an agile approach for COTS taxonomies development. However, many COTS marketplace characteristics are not dealt with, and the agile perspective of the method gives up many COTS reliability issues.

In [Cec+06] a more detailed survey of trends on COTS identification and retrieval approaches is provided.

2.2.3.1 Analyzing COTS Classification Approaches

In Table 2.6 we provide a summary of some of the most relevant COTS classification proposals compared in terms of the following criteria:

- a. Domain Specific: Describes if the approach is addressing a specific domain or it is used for general domains.
- b. Characterization Schema: Describes the attributes used to classify the components.
- c. Guided Construction: Describes if the approach provides methodological support for structuring the taxonomy.
- d. Classification Schema Evolution: Describes if the approach provides effective mechanisms to evolve the classification schema to deal with the constant growing and evolution of the COTS marketplace.
- e. Reuse: Describes if the approach tackles reuse of attributes. Satisfying this criterion does not necessarily imply the existence of any systematic technique.
- f. Population support: Describes if the approach addresses population strategies or not.

Table 2.6 Summary of some COTS classification approaches

Charact. Approach	Characterization Schema	Domain Specific	Guided Constr.	Evol.	Reuse	Pop. Support
[Car-Lon00]	Origin and Modifiability	-	-	-	-	-
[Mor-Tor02]	Categories of Source, Customization, Bundle, and Role.	-	-	-	-	-
[Tor+02]	Set of general attributes similar to those in the ISO 9126-1	-	-	-	*	-
[Jac-Tor02]	kind, architectural, level, and phase	*	-	-	*	-
[Ack+02]	Seven levels of attributes: Marketing, Task, Terminology, Quality, Interaction, Behaviour, and Interface	-	-	-	-	-
[Car+04]	Business Applications specific attributes	√	-	-	*	-
[Ero-Gia06]	Guided by a decision model	-	√	-	-	-

(√) fully satisfies the criterion (*) partially, informally or implicitly satisfies the criterion (-) does not satisfy the criterion

From Table 2.6, we can realize that the proposals cited before do not fully resolve the problems of classifying COTS neither for performing efficient searching and retrieval mechanisms, nor for reusing knowledge gained about COTS. Furthermore, they share some common characteristics that may be considered as drawbacks:

- (1b) Lack of guidance and methodological support to construct and/or adapt the classification schema. Although most of the approaches are developed for general COTS domains, there is no a clear explanation of how they can be used in different contexts and projects. Therefore, existing COTS classification schemas are often specific, and project bound.
- (2b) Lack of mechanisms to identify the properties that can help to organize COTS. To classify the items, the existing approaches rely on experience, knowledge, and observation and they rarely use knowledge engineering and requirements engineering techniques. This makes very difficult their general understanding, use, evolution and extension.
- (3b) Lack of mechanisms to deal with COTS marketplace evolution and growing. The categorization model used in the mentioned approaches is mainly static. However, the domain knowledge and the marketplace are constantly changing; new terms and components appear and meaning of old terms can alter. Thus, existing COTS classification schemas are very difficult to be evolved and extended.
- (4b) Non-efficient reuse mechanisms. Though the idea of classification schema assumes reuse, many of the approaches do not success in this issue. Most of them are too broad to be useful. On the other hand, there is a lack of category understanding and evolvability mechanisms that rapidly make the taxonomies obsoletes and/or unusable. Moreover, no mechanisms to reuse the knowledge gained about COTS are proposed.
- (5b) Absence of mechanisms to classify real items and populate the taxonomy. Existing approaches exemplify COTS classification by assuming that suitable component information already exists. None of them deal with population strategies and COTS related information finding and assessing.

Thus, while these proposals paid attention to the structure of the classification, there are not studies in deep to build these classifications taking into account the users requirements, evolution of the domain and trends of the marketplace.

Moreover, even less attention has been paid to the methodological aspects required to support its construction. In this context we argue that to be useful to support COTS marketplace structuring, more important than the concrete form that a classification schema takes, is the rationale behind its construction, (i.e., which are properties that may help to arrange it and how the classification schema can be searched). This is especially true when considering not just the construction of the classification schema, but its evolution.

2.2.4 COTS Related Search Engines

Many potential candidate COTS are available on the web. Finding an adequate component involves searching among abundant and heterogeneous information available and a need of analyzing it efficiently. Thus, the use of search systems is proposed to make more efficient the COTS selection processes.

Using classic search engines available on the web to search COTS related information (e.g., Google) is a very cumbersome task because: a) they are very generic and do not take into account the specific characteristics of COTS; b) users have to browse long document lists, analyze them and discriminate among the useful COTS related links. Hence, some automatic or semi-automatic search engines using different technologies have been proposed for finding and identifying COTS related issues in the web relying on available component catalogues supplied by related companies.

Representative examples of these tools are: Agora [Sea+98], SCB (*Software Commerce Broker*) [Aoy+98], IPSCoM (*Intelligent Portal for Searching Components*) [Agu05], MoreCOTS [Yan+06], and Sema-SC (*Semantic Component Selection*) [Sja-Beu06].

Agora [Sea+98] is a research prototype developed by SEI which attempts to create an indexed worldwide database of software components using JavaBeans and CORBA agents in conjunction with Web search technologies. It supports two basic processes: the location and indexing of components, and the search and retrieval of a component. The location and indexing of components is primarily an automated background task, while a human typically perform search and retrieval. There are exceptions in that an interface exists to allow a vendor to add a specific component to the index. The system combines Web search engines with an introspection process. Introspection, describes the capability of components to provide information about their own interfaces.

SCB (Software Commerce Broker) [Aoy+98] is a research prototype which attempts to collect information on software components worldwide over the Internet and provides a set of electronic catalogues of software components in a semi-formal specification language called SCL (Software specification and Commerce Language). Furthermore, SCB provides mechanisms on which customers in remote locations can play the component through the Web.

Although the IPSCoM project [Agu05] is still in the design phase, it aims at developing an open information portal for COTS software and non software components, in which several existing COTS repositories can be integrated making use of a generic ontology. The aim of this generic ontology is to provide (i) a standard for the definition of components that unifies the differences between different models (ii) a standard interface for component searching. As a result the ontology is able to hold information for each component regarding: General Information (e.g., name, version, language, etc.); Features (e.g., properties, methods and events); and Design (it describes how to construct a composite component connecting pre-existing components).

MoReCOTS [Yan+06] is a prototype of a specialized search engine for COTS marketed on the Web. It is based on meta-searching online specialized databases

maintained by five publishers of COTS catalogues (i.e., “Componentsource”, “Knowledgestorm”, “CXP”, “Enterprise SoftwareHQ”, and “Capterra5”). Thus, MoReCOTS is based on and inspired from components taxonomies available in those COTS catalogues. As a result, it provides (i) a directory containing a list of COTS categories, and (ii) a specialized search interface with specific search fields related to COTS characteristics.

Sema-SC [Sja-Beu06] is a semi-automated generic method for component identification and classification based on generic domain taxonomy and user generated semantic input. It was inspired by the semantic web community. Every query is semantically tailored to what is being looked for, arriving at better results than it is currently possible using available automated categorisation systems. It also relies on the information contained on available COTS related portals.

2.2.4.1 Analyzing COTS Related Search Engines

Table 2.7 Summary of some COTS Related Search Engines

COTS Search Engines	SM	Advanced Search		Main Search Result	Portals it relies on	AI	Dev. Stage	Lang.
		Bool	RS					
Agora	<i>c&k</i>	√	√	URL of the component	CORBA and JavaBeans portals	√	Prototype	English
SCB	<i>kw</i>	*	√	Component location	Set of specific portals it maintains	-	Prototype	English
IPSCom	<i>c&k</i>	√	√	General Information, Features and Design	Intended catalogs in the IPSCom ontology	√	Design	English
MoReCOTS	<i>ms</i>	√	√	Component information structured as suggested in [Sas+03]	5 COTS publishers portals	√	Prototype	English, French
Sema-SC	<i>c&k</i>	√	√	Component information	Generic Ontology to deal with several portals	√	Prototype	English

(√) fully satisfies the criterion (*) partially, informally or implicitly satisfies the criterion
 (-) does not satisfy the criterion

In Table 2.7 we provide a summary of some of the most relevant COTS related search engines assessed in terms of the following criteria:

- a. Search Method (SM). It includes four criteria which are: search by category (*c*); search by keywords (*kw*); search by category and keywords (*c&k*); and meta-searching (*ms*).
- b. Advanced Search. Describes if the approach provides any advanced mechanisms to search. They can be Boolean (Bool) operators or refined searches (RS).

- c. Main Search Result: Describes the main parameters the approach retrieves for describing the component.
- d. Portals it relies on. Describes the components repositories in which the approach relies on.
- e. Automatic Index (AI): It refers if the indexing of components is intended to be automatic or not.
- f. Development Stage. Refers to the stage of the development or the tool is.
- g. Language. Describes the language(s) supported by the approach.

In general, regardless their development stage and maturity (most of them are in prototyping or design stages), these kinds of tools are not widely succeeded in practice mainly because of their conception of “structured and centralized catalogues of COTS”, which is not true in practice (i.e., COTS marketplace is characterized by its widespread cataloguing nature, and non-standard descriptions) as well as the lack of homogeneous and trustworthy information available that hamper the acquiring of effective and quality assured COTS information.

The main open issues that current COTS searching tools face can be summarized as:

- (1c) Dependency on some COTS available catalogues. Current COTS related search engines provide only the facility to browse COTS from some specific electronic catalogues. Therefore, the searching space is focused only on the components available in the related catalogues, so important components that exist in other catalogues could be deterred.
- (2c) Fully dependency on available information provided by vendors or catalogues publishers. Studies confirm that the COTS information is usually incomplete and sometimes biased in order to highlight components characteristics [Ber+03], [Tau+04], [Ast+06]. Further discussion about the problems with using those catalogues is addressed in Section 2.3.2.
- (3c) Difficulty to deal with COTS marketplace heterogeneity. Each COTS catalogue available describes components following its own classification structure and description model (i.e., non-standard descriptions). Although most approaches try to deal with heterogeneity by using diverse techniques as: developing a general ontology, using semantic web technologies and description logics; they can only address a limited set of available catalogues. Therefore, the real applicability of most of these proposals have resulted scarce [Req+05].
- (4c) Finally, accuracy and information quality (i.e., trustworthiness, completeness, etc.) are not greatly ensured with the use of existing automatic or semi-automatic search engines. While COTS search engines achieve a high level of efficiency and low cost, they do not achieve the same level of information accuracy and reliability as manual approaches.

2.3 State-Of-The-Practice

Some empirical studies in companies using COTS show several relevant results about how they use such components [Tor-Mor04], [Li+04a], [Li+05], [Li06], [Hsu-Wid06]. Such studies reveal that companies do not normally use any formal process for selecting components. Instead, most of them are using an experience-based and/or hands-on trial-based selection processes.

In the first case, developers already have experience with some specific components or technology, and this experience is important in deciding which components to choose. In the second case, the World Wide Web (WWW) is used to find executable components and a few of them are then downloaded and further evaluated.

To specifically investigate how companies search COTS, we also performed an empirical study [Ger06] together with the Software Engineering Group at the Norwegian University of Science and Technology (NTNU).

Our results showed that the COTS searching and identification complexity is actually twofold:

- ♦ *How to know which kind of components are available and which of them could be useful to solve a specific problem?*
- ♦ *How to find and process the information referred to those components to perform an effective evaluation?*

As a result, it is evident that there is an increased need for organizing and obtaining suitable COTS information to achieve more efficient and reliable selection processes [Aya-Fra05].

From the answers of our respondents we found that the WWW is the most used means to find candidate components (i.e., search on available catalogues or specialized search engines) followed by colleague recommendations.

We also asked about the resources they usually use to locate components and information about them, as well as the perceived utility of such information for performing the different COTS selection activities.

2.3.1 COTS Location and Reuse

From our empirical study introduced before, we found that to satisfy COTS users practical needs, many kinds of organizations provide online COTS catalogues, defining categories of services, products, and knowledge, usually arranged in a hierarchical form. In some cases, they also provide additional resources as search engines, newsletters, forums, comments, rating of components, re-user opinions, white papers, case studies, lessons learned, etc.

Table 2.8 provides a list of representative types of organizations issuing COTS related topics.

Table 2.8 Summary of organizations types related to COTS selection activities

Organization Kind	Description	Examples
IT Consultant Companies	They are commonly dedicated to technological analysis and market monitoring. They play an important role in selling expert support for selecting COTS. However, the analyses they provide are often expensive and short-lived.	Gartner Forrester Co.
Commercial Web-Based companies	They range from general to domain-specific portals acting as marketing channels of components.	ComponentSource Knowledge Storm.
Professional Societies	They use hierarchies to organize COTS systems related knowledge	INCOSE
Portals with different registration procedures	Offer white reports, re-user opinions, or technical products from research projects.	eCots (currently not available) and OpenCores.
The academic world	Several proposals have been presented with the purpose of supporting COTS selection from the academia, see Section 2.2.1	CeBASE repository providing a “COTS lessons learned” database.
COTS vendors	They directly offer their products to the general public.	Any commercial firm
Open Based Portals	They act as a free indexing components information repository.	CMSmatrix.org WikiMatrix.org

2.3.2 Available COTS Catalogues and Repositories

By asking our respondents about the main online resources they use to select COTS, as well as performing an extensive online review and in the literature, we completed Table 2.9 that summarizes some of the most representative catalogues, repositories and services available, issued by different types of organizations as described in Table 2.8. Available catalogs are described by the following features:

- a. *Name*. Name of the approach or URL.
- b. (C) *Characterization*. Refers to the characterization schema it uses to classify COTS. It involves five features: by Name (*N*); by Software Categories (*SC*), by Platform (*P*); by IT Solutions (*ITS*); and by Editors (*E*).
- c. (RS) *Retrieval Schema*: Refers to the schema by which COTS and/or COTS related issues may be searched. It implies four criteria: by Browsing (*B*); Keyword (*KW*); List Selection (*LS*); Selection Wizard (*SW*).
- d. (IR) *Information Rendering*. Illustrates the way the information is presented. It is described in the next terms: Non-Structured (*S*); and Semi-Structured (*SS*).
- e. *Intended Objective*. Describes the main objective of the organization that supports the catalogue, repository or service.
- f. *Sponsor*. Describes the main sponsor of the approach.
- g. *Approx. Amount of Available Resources*. It describes the amount of components and in some cases the number of categories used to organize them.

Table 2.9 Some representative COTS related repositories and catalogues¹

Name	C	RS	IR	Intended Objective	Sponsor	Approx. amount of Available Resources
COTS Vendors	N	B, KS	NS	Marketing	Private Company	1 to more comp. each
eCots.org	SC	B, KS	NS	Research	European Research Funds	Over 550 Comp.
ComponentSource.com	SC,P	B, KS	NS	Marketing	Private Company	102 categories/ 3,369 Comp.
KnowledgeStorm.com	ITS	B, KS	NS	Marketing	Private Company	17,943 Resources (comp., papers, etc.)
CMSmatrix.org	N	LS	SS	Open and free collaboration for indexing Content Management Systems (CMS)	Free and Open	675 Comp./ 2670 re-users
CompareIM.com	N	LS	SS	Open and free collaboration for indexing Messaging systems	Free and Open	Over 400 resources
JdoCentral.com	N	B	NS	Interchange of Java components	Open Community and vendors	Over 4000 resources
Tucows.com	P	B, KS	NS	Promote freeware/shareware	Private Company	Over 40,000 software titles
Krugle.com	N	KS	NS	Search engine designed for developers	Private Company	
Forrester.com	ITS	B, KS	NS	Selling IT Strategic Support	Private Company	Ad-hoc resources
Gartner.com	ITS	B, KS	NS	Selling IT Strategic Support	Private Company	Ad-hoc resources
Inco.se.org	SC	B, KS	SS	Research Support in Systems Engineering	[INC] (non-profit)	1485 Comp.
CEBASE: http://fc-md.umd.edu/ll/index.asp	N	B, KS	NS	Sharing Lessons learned in using COTS	CEBASE Research	Over 100 experiences
Softguide.de	SC, N, P, E	B, KS	SS	Brokering COTS suppliers and COTS purchaser in Germany	GmbH & Co. KG, D-Wolfsburg	7500 COTS products and 5000 editors
CXP.fr	SC, N, P, E	B, KS	SS	Brokering COTS suppliers and COTS purchaser in France	Private Company	7 000 COTS products
Eoslist.com	SC	B, KS	NS	Maps the major OSS projects and companies delivering OS alternatives to COTS	Free and Open	350 projects
Wikimatrix.org	N	B, SW	SS	Open and free collaboration for indexing Wiki Tools	CosmoCode Company	85 comp.
forummatrix.org	N	B, SW	SS	Open and free collaboration indexing OTS-Forums management systems	CosmoCode Company	39 comp.
weblogmatrix.org	N	B, SW	SS	Open and free collaboration for indexing Weblog systems	CosmoCode Company	16 comp.

N: Name
SC: Software Categories
P: Platforms
ITS: IT Solutions

E: Editors
B: Browsing
KS: Keyword Search
SW: Selection Wizard

NS: Non-Structured
S: Structured
SS: Semi-Structured

¹ This survey is due to February 2007

Our analysis of these cataloguing and repositories initiatives, lead us to claim that although we can find many reuse repositories with sophisticated classification schemas and storage structures for COTS, they do not provide complete and efficient information to perform an informed COTS selection. Some common practical problems also related with the state-of-the-art problems are:

- ♦ *Uncontrolled repositories and cataloguing explosion:* Currently, there is an uncontrolled proliferation of cataloguing initiatives that describe the COTS software marketplace into different levels, from different points of view, objectives, meanings and scopes. Furthermore, each repository acts as the first one ever, leading to confusion by different descriptions and cataloguing of the same component.
- ♦ *Processes around these repositories are not clear:* The different and fuzzy processes and criteria used in each portal make difficult their use. Sometimes, the meaning of a particular domain is not clear without further examining the items, especially if the domain is absolutely unknown to the re-user. Consequently the understanding, use, evolution, extension, and customization of the categorization proposals are difficult.
- ♦ *Scarce information and updating:* Regardless the completeness, domain, and scope of the mentioned cataloguing initiatives, they contain only brief and unstructured descriptions of some inventoried components. Moreover, they face serious difficulties to maintain up-to-date information. Thus, structuring and discovering important information leads to rework, confusion, missing critical information and possibly deterring the use of some components.
- ♦ *Business-related nature of repositories:* Given the commercial nature of the marketplace, several sponsored repositories tend to highlight the strengths of some components and hide their weaknesses to give competitive advantage to their licensed components.
- ♦ *Non-suitable COTS Information Reuse:* The relevant decision-making processes in COTS selection are often intuitive and only sparsely documented in practice. Reuse includes not only finding and understanding components but also reusing the knowledge gained in each selection process to complete and improve COTS information in order to enhance future COTS selection processes. It is particularly important in the COTS context given the scarce nature of COTS-related information. Moreover, ad-hoc support analysis for selecting COTS offered by specialized companies is expensive and short-live above all for medium and small companies.

These drawbacks have been recognized as a barrier on the adoption of COTS in industrial projects, since they make the selection process highly risky and expensive for finding and managing component knowledge reuse. Some other studies supporting this claim are [Cla+04], [Req+05], [Cec+06], [Car-Fra06], [Wan-Hom06]. In Table 2.10 the assessment of role-related current practices and their related problems are summed up. They range from the need of having understandable taxonomies, a common COTS description metamodel embracing all the informational dimensions for evaluating COTS and a reuse infrastructure support able to deal with COTS marketplace characteristics.

Table 2.10 Assessment of the role-related challenges for supporting COTS Selection

Role	Current Practice	Problem
MW	<ul style="list-style-type: none"> • Proliferation of cataloguing initiatives • Some inventoried components in each catalogue • Widespread and unstructured components descriptions • Most catalogues do not have a clear rationale behind. 	<p>Understanding and using the categorizations may be difficult.</p> <p>Several descriptions of the same component.</p>
QE	<ul style="list-style-type: none"> • Lack of structured and widespread COTS descriptions • COTS providers do not provide structured and enough information for supporting evaluation and product quality assessment. 	<p>Complex discovering and structuring of critical information.</p>
S	<ul style="list-style-type: none"> • Difficulty to locate relevant information about the components 	<p>Hard requirements negotiation.</p> <p>Complex decision-making</p>
KK	<ul style="list-style-type: none"> • No support for organizations that continuously select COTS to reuse their knowledge or others knowledge about them. 	<p>Reuse of knowledge is usually tacit, leading to be lost if people are replaced.</p>

2.4 Relevant Approaches Supporting the Solution Addressed by this Thesis

This thesis builds upon existing work. In this section we briefly describe some approaches that played an important role in shaping the GOTHIC method solution. Several other important research approaches were related and they will be discussed in their corresponding method's activities description (Chapters 5-9).

2.4.1 Goal-Oriented Approaches

In the last years, the use of goals for supporting a variety of processes and disciplines, such as business process reengineering, organizational impact analysis and requirements engineering, is increasingly gaining importance.

Goals present several characteristics that make them attractive, e.g. expressiveness, stability and evolvability [Lam01].

As a consequence of this tendency, the software engineering community is currently addressing the problems associated with the formulation of business goals, plans, processes, etc., in order to achieve organisational objectives in an ever-changing organizational environment [Lou-Kav95].

From a methodological perspective, the intensive use of goals in these disciplines has led to a whole stream of research. We term all these efforts *goal-oriented approaches*.

Goal-based approaches resulted appropriate for handling the problem of structuring the COTS marketplace considering its rapid evolution and all the information related in a selection process.

Studies confirm that software development teams and stakeholders have a better understanding of the general goals they want to achieve than the functionality that should be exhibited by the desired system. It is because goals attempt to a language based on concepts in which most of the members of a development team and stakeholders are most comfortable and familiar [Ant97]. Moreover, they are evolutionary, more stable respect to changes in processes, and goal refinement provides a natural mechanism for structuring and exploring many alternatives

A goal is an objective that should be achieved and may be formulated at different levels of abstraction, ranging from high-level strategic to low-level technical concerns [Lam01]. Thus, goals capture, at different levels of abstraction, the various objectives covered by a market segment. Since a summary and comparison of several goal-driven approaches is available in [Gre94] and [Kav-Lou05], we will only discuss the selection of the approaches used as backbone of the GOTHIC method and describe some of their particularities.

Table 2.11 provides an overview of some relevant goal-oriented approaches. The *usage* view concerns the objectives of using goal modeling in RE, namely: (1) understanding the current organizational situation; (2) understanding the need for change; (3) providing the deliberation context within which the RE process occurs; (4) relating business goals to functional and non-functional system components and (5) validating system specification against stakeholders' goals.

The *subject* view looks at the notion of a goal and its nature. Initially, it seems difficult to discern a uniform notion of goal in RE. Indeed, the term goal has been used in different approaches to convey several meanings, including human tasks, problem solving outcomes, desired states of the world, and target concepts of human behaviour, policies and orientations for acting, and so on.

The *representation* view concerns the way goals are expressed. Goals can be expressed in a variety of formats, using more or less formally defined notations. We differentiate between *informal*, *semi-formal* and *formal* approaches. Informal approaches generally use natural language text to express goals; semi-formal use mostly box and arrow diagrams; finally, in formal approaches goals are expressed as logical assertions in some formal specification language.

Finally, the *development* view concerns the way that goal models are generated, and evolve. This view considers the dynamic aspects of goal driven approaches, i.e., the proposed way-of-working and the tool support provided for enacting this way-of-working.

Using the framework of usage, subject, representation, and development dimensions, it was possible to easily understand and accordingly select the best *fit for usage* goal-oriented approaches that may help us to reach our main objective that was the understandable and flexible structuring of COTS marketplace domains.

In this context, two main goal-oriented approaches were chosen to reach our objectives (although some others are also used and interrelated): The GBRAM method

in order to have a prescriptive method to elicit goals from several information sources; and the *i** modeling approach in order to formally record strategic dependencies existing among components in the marketplace. The rationale behind these elections is further discussed in next subsections.

Table 2.11 Some relevant goal-oriented approaches as surveyed in [Kav-Lou05]

Framework Components		Goal-oriented Approaches														
		Cognitive Task Analysis	<i>i*</i> (Strategic dependency Model)	Goal-Based Workflow	EKD	F ³ (OM)	ISAC	SIBYL	The reasoning loop model	REMAP	KAOS	GBRAM	Goal-scenario coupling	NFR framework	GSN	GQM
Usage	Understand current org. situation	✓	✓	✓	✓											
	Understand the need of change					✓	✓									
	Provide deliberation context within which RE occurs							✓	✓	✓						
	Relate business goals to system components										✓	✓	✓	✓		
	Evaluate system specs. against stakeholders goals														✓	✓
Subject	Enterprise goals	✓	✓	✓	✓						✓	✓	✓	✓		
	Process goals					✓	✓	✓	✓	✓						
	Evaluation goals														✓	✓
Representation	Formal		✓							✓	✓			✓		
	Semi-formal	✓		✓	✓	✓		✓	✓			✓	✓		✓	✓
	Informal						✓									
Development	Way-of-working		◆		◆		◆				◆	◆	◆	◆		
	Tool support	M	MF	M	M	MG	MG	M		MF	MF	MG	MG	MF	M	MG
✓ = deals with the issue ◆ = suggest a number of steps and associated strategies M = support for model construction, F = formal reasoning support, G = process guidance																

2.4.1.1 Goal-Based Requirements Analysis Method (GBRAM)

GBRAM was conceived by Annie I. Anton [Ant97] with the primary focus on the transformation of enterprise and system goals into requirements, more specifically to assist analysts in gathering software and enterprise goals from many sources and to support the process of discovering, identifying, classifying, refining and elaborating goals into operational requirements.

The method's chief contribution was the provision of heuristics and procedural guidance for identifying and constructing goals. In other words, it was developed for to identifying, elaborating, refining, and organizing goals for obtaining the software requirements document.

The high level phases of GBRAM briefly explained are:

- *Goal Analysis* that concerns the exploration of available information sources for goal identification followed by the organization and classification of goals.
- *Goal Refinement* that concerns the evolution of goals from the moment they are first identified to the moment they are translated into operational requirements for the system specification. It includes activities as refine, elaborate and operationalize of goals.

These high-level phases provide an overview of the GBRAM. Fig. 2.4 shows the activities with which an analyst is intimately involved when applying the GBRAM.

The ovals located within the dotted box on the upper right corner of the figure denote goal analysis activities. The ovals within the dotted box on the lower half of the figure denote the activities that take place during goal refinement. The box in the top left corner contains the possible inputs, which may vary in accordance with the documentation initially available to analysts. The output of GBRAM is always a software requirements document.

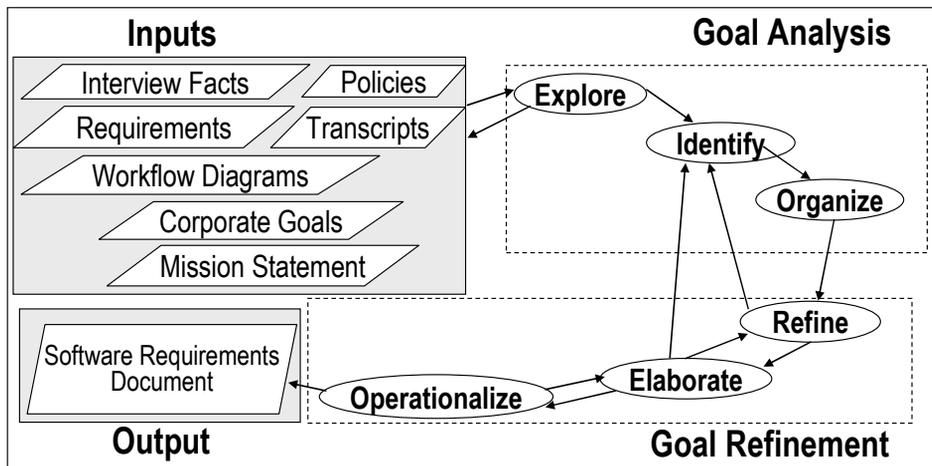


Fig. 2.4 Overview of GBRAM activities

The suitability to apply GBRAM to our purpose of constructing COTS marketplace taxonomies can be summarized as follows:

- It assumes that goals have not been previously documented or explicitly elicited; it supports the goal elicitation from all available sources of information. This is a very helpful aspect because in the COTS context, we have to simultaneously

evaluate widespread and heterogeneous information related with the domain for which we want to create the taxonomy.

- b) Some heuristics and guidelines from GBRAM could be reused in the COTS context, and many others can be added to support our specific task. This was proven to be useful for guiding the exploration, identification, and organization of goals towards a high probability of success while avoiding wasted efforts. (Annex 1 provides an excerpt of some useful heuristics used in our approach.)
- c) It offers a guide for applying an inquiry-driven approach to goal-based analysis; it was useful for building our intended decision tree taxonomies guided by questions-answers.
- d) It offers a dynamic framework to integrate other required techniques and mechanisms for formalizing the construction of a COTS marketplace reuse infrastructure. For instance, for constructing the models and artifacts required for the manipulation and evolution of the taxonomies or interoperability issues.

2.4.1.2 The *i** Goal-Modeling Approach

One of the most widespread goal-oriented modeling languages is the *i** notation proposed by Eric Yu in the first half of the 90's [Yu95]. *i** defines a repertory of intentional elements for building two types of models: the Strategic Dependency (SD) model and the Strategic Rational (SR) model, each one corresponding to a different abstraction level. Both models not only provide means for reasoning about goals but also about agents, therefore *i** is usually considered also an agent-oriented modeling language.

As mentioned above, the *i** language proposes the use of two models: a *SD model* that represents the intentional level of a system and *SR models* that represent its rational level (see Fig. 2.5 for an example).

A SD model consists of a set of nodes that represent *actors* and a set of *dependencies* among them. Actors may be specialized through the *is-a* relationship. Dependencies express that an actor (*dependor*) depends on some other (*dependee*) in order to obtain some objective (*dependum*). The dependum is an intentional element that can be a *goal*, a *task*, a *softgoal* or a *resource*. The dependor depends on the dependee to bring about a certain state in the world in goal dependencies; to carry out an activity in task dependencies; to perform some task that meets a softgoal in softgoal dependencies; and for the availability of an entity in resource dependencies.

The type of dependency also characterizes the actor of the dependency that has the responsibility to make decisions: in goal dependencies the dependee is expected to make any decisions required; in task dependencies the dependor decides how the task will be performed by the dependee; in softgoal dependencies the dependor makes the final decision, but does so with the dependee's know-how; in resource dependencies no decisions are required.

From our survey analysis of the problems related with searching COTS information, we realized the critical need to record COTS dependencies. *i** SD models provided us the way of doing it in a formal way whilst they resulted significantly understandable for almost any people. Moreover, their use was also greatly harmonized with the GBRAM approach and the techniques used to cover our research objectives. Chapter 6 provides details of our *i** approach adoption whilst Chapter 7 provides specific details of the way of using it as part of our intended GOTHIC method.

2.4.2 Software Quality

The business value of a software product results from its quality as perceived by both acquirers and end-users. Therefore, quality is more and more often seen as a critical attribute of the product, since its absence results in dissatisfied users and financial loss, and may even endanger lives [Sur-Abr03]. Software development organizations, in general, are not best equipped to deal with it: they do not have at their disposal the quality-related measurement instruments that would allow (or "facilitate") the engineering of quality throughout the entire software product life cycle, even less when CBSD is used [Car+07a].

Software quality has been one of the main goals of software engineering in the last decades. However, despite of the efforts to develop new and more powerful software process improvement techniques, such as CMM (Capability Maturity Model) [CMM93], Bootstrap [BOO93] or SPICE [EI+98], and the development of better metrics and product validation techniques, it remains an elusive target [Kit-Pfl96], [Sur-Abr03]. More recently this problem has been aggravated by some factors: the growing tendency of using COTS; the increasing COTS marketplace availability; and the continuous creation of new communication and marketing channels (e.g. web markets, search engines, etc.) which bridge the gap between providers and consumers of those products.

A key factor in the selection of COTS is the assessment of their quality. Traditionally two aspects have been considered when assessing software quality:

1. The software development process, which in some cases (e.g. when adopting a software improvement process like CMM or SPICE) can be further certified by a third party organization. However, even though the certification of the software development process used by the supplier can be considered a positive factor, it is a well known fact that it does not necessarily leads to better products [Voas98], [Kit-Pfl6].
2. The inspection of the final product to measure its compliance with respect to some quality characteristics. COTS are not bespoke software; they are acquired from external suppliers. Thus, they are black boxes (or at best very dark ones), customers seldom own their source code, and therefore product measurements can be applied and/or obtained only for those externally observable quality characteristics. Thus it is external product quality what really matters when it comes to decide to acquire a COTS or rather to construct a component from the scratch.

These two aspects have to be carefully considered when evaluating COTS quality. Furthermore, software quality characteristics are often difficult to check. This is partly due to their very nature, but we argue that there is another reason that can be mitigated, namely the lack of structured and widespread descriptions of COTS domains (i.e., categories of COTS, such as ERP systems, graphical or data structure libraries, etc.). This absence hampers the accurate description of COTS in the domain and the precise statement of quality requirements [Car+07a]. As a consequence, the whole component evaluation is damaged, and confidence on the result of the process diminishes. Hence, in order to assess the quality of a COTS, it is necessary to provide a framework to represent quality which can be used as the basis for its evaluation. As with all those concepts that require an understanding, it is necessary to create a model which is representative of the area of interest. Therefore, we claim that the assessment of COTS quality requires the existence of a COTS quality model.

A structured quality model for a given COTS domain will provide a hierarchy of software quality features and also metrics for computing their values. Once available, requirements over the domain, as well as components quality features, may be stated with respect to the quality model. Product quality can be evaluated and their characteristics can be compared with requirements based on a common description.

Therefore, we made use of existing work on the quality models area to support this claim. After several industrial and academic experiences, we have confirmed that the basic set of characteristics and subcharacteristics provided by the ISO/IEC 9126-1 standard is complete and accurate enough to be used as starting point in the process of COTS domains quality models construction.

Next subsections provide an overview of existing work and fundament our decision to use the ISO/IEC 9126-1 standard. Also, work in the field of metrics was applied, as the work done by Fenton et al [Fen-Pfl97], and the GQM (Goal-Question-Metric) approach [Bas+94].

2.4.2.1 Software Quality Models

Quality is a complex concept, for which no universal definition exists. Quality means different things to different people, thus it is highly subjective and context dependant. This fact has been corroborated in [Bas+04] where several top level software engineering practitioners, stated their personal and very diverse views on software quality.

According to Voas [Voa04] “We live in a world where beauty to one is a complete turnoff to another. Software quality is no different. We have the developer’s perspective, the end users’ perspective, the testers’ perspective, and so forth.” Because of this, it is really hard to get to an agreement on how to measure software quality. Kitchenham [Kit89] stated that quality is “hard to define, impossible to measure, easy to recognize”. Gilles [Gil97] stated "Quality is generally transparent when present, but easily recognized in its absence". These statements imply that quality is somehow perceptible. Thus, the problem is not quality being subjective, but how to correlate the different views on quality into a measurable quality framework, which can be

commonly agreeable at least in some context (e.g., a domain of knowledge, an organization, a project, etc.).

In the ISO standard 8402, a Software Quality Model is defined as:

“The set of characteristics and the relationships between them which provide the basis for specifying quality requirements and evaluating quality”.

Software quality models have been proposed in order to provide many benefits: they can be used as a base to define a commonly agreeable quality framework, which consolidates the different views on quality; they can be tailored to specific contexts; they provided a measurable base to the evaluation of software quality.

Since the late 70s, many proposals defining the hierarchy of a software quality model exist. Some examples are [McC+77], [Boe+78], [Bas92], [IEEE1661-98], and [ISO-9126-1] (see [Car05T] for a comprehensible survey).

Nearly all of these proposals share a hierarchical tree-like structure, composed of a set of high-level quality characteristics to which identifiable and measurable low-level quality features are linked. The number of layers of the hierarchy, as well as the number of fixed features differs largely among the different proposals.

Table 2.12 provides a comparative summary of existing quality models approaches.

We can observe that while those proposals paid attention to high-level quality features, very little has been devoted to the study of lower-level ones and their influence on the former, as well as the methodological aspects required to support the construction of the proposed quality models. Even less attention has been paid to the construction of quality models to support CBSD [Car05T].

Quality models are engineered to provide the basis for software evaluation; therefore metrics for the quality features have to be assigned. Earlier models were qualitative in nature. This was due to the fact that little work on the software metrics field existed. However, as work on this field evolved [Fen-Pfl97], [Zus97], quality model became a stronger framework to support software quality evaluation. Gilb’s [Gil88] idea that all of the quality features have to be made measurable, and Keller’s [Kel+90] introduction of qualitative and quantitative metrics at different levels, are some of the contributions supporting this attempt.

Nowadays, a great deal of applications for software quality models have been suggested, and in some cases explored. Although software quality models were originally tough to be used in software evaluation, many different and in some case creative applications have been proposed in the last decades.

Earliest works on software quality models such as Boehm’s [Boe+78], already outline some alternative applications. On the preface of this work the authors suggest that a well defined framework of various characteristics of software quality can be used to: (1) prepare the quality specification of a software product, (2) checking for compliance with quality specifications, (3) making proper design tradeoffs between development costs and operational costs and (4) software package selection.

Table 2.12 Some relevant quality models approaches as surveyed in [Car05T]

Model	Kinds of elements	Number of Layers	Metric	Construc. Approach	Over lapping	Relationship among elements	Measure applied to	Method
McCall	-Factors -Criteria -Metrics	4 (1 main subdivision, 1 layer of factors, 1 of criteria and 1 of metrics)	No	Fixed	Yes	Hierarchical	Criteria	No
Boehm	-High level characteristics -Primitive Characteristics -Metrics	4 (2 layers of high-level characteristics, 1 of primitive and 1 of metrics)	No	Fixed	Yes	Hierarchical metrics dependencies	Primitive characteristics	Yes
FURPS	-Quality Attributes -Metrics	3 (2 layers of quality attributes and one of metrics)	No	Fixed	No	Hierarchical	Quality attributes in the lowest levels	No
GQM	-Goals -Questions -Metrics	3 (goals, questions and metrics)	No	Custom	Lower-levels	Hierarchical	Questions	Yes
Gilb	-High-Level attributes -Low-Level attributes -Metrics	Not fixed (user definable)	No	Custom	Yes	Hierarchical	Low and High-level attributes	No
ISO/IEC 9126	-Characteristics -Subcharacteristics -Quality Attributes -Metrics	-2 upper layers (Charac. & Sub-charac.) -Attribute layers are not fixed, they are users definable -Metrics layers are associated to all attribute layers	Yes	Mixed	Lower-levels	Hierarchical	Quality Attributes	No
IEEE 1061-1998	-Factors -Subfactors -Metrics	Not fixed (user definable)	No	Custom	Yes	Hierarchical, conflicting and supporting relationships among factors	To any element in any layer (user definable)	Yes
Dromey	-High-level attributes -Quality carrying properties -Components	2 upper layers (High-level attributes and quality carrying properties), not fixed in lower levels	No	Custom	Yes	Links among components and high-level attributes	Not included	Yes
SQUID	-Quality Characteristics -Quality Attributes -Metrics	Upper layers are not fixed, they are user definable. 1 layer of quality attributes	No	Custom	Yes	Hierarchical links among external attributes and the internal affecting them	Quality Attributes	No

Although the use of software quality models to specify systems requirements has been an implicit goal since the early proposals (at least to specify non-functional requirements related to reliability, efficiency, security, etc.), some authors (e.g., Dromey [Dro96], Bøegh [Bøe+99] and Firesmith [Fir03]) have further explored this particular application and made some significant contributions.

Other authors have explored the possible applications of software quality models to support software architectural design. Representatives of this field are the works by Klein and Kazman [Kle-Kaz99] on the definition of architectural patterns, Lundberg [Lun+99] on the analysis of conflicting quality features in architectural design, and the works of Kazman and Klein [Kaz-Kle00] and Losavio [Los+03] on architectural tradeoff analysis.

To support software implementation has been the target of some works. Dromey [Dro95], [Dro96] considers that the only way to obtain quality products is to build quality into the products. He also considers that software quality is heavily influenced by the programming language used in product development. Thus, he has proposed a quality model aimed to the evaluation of the programming languages used in software development. In the SQUID method [Bøe+99] and the Prometheus approach [Tren-Pun03], quality models are used to set targets which are used to continuously evaluate the products through the development process.

2.4.2.2 COTS Evaluation by Quality Models

When selecting a COTS from a set of competing candidates, a comparison of the most significant characteristics has to be performed. This is an implicit activity in any procurement process, including software selection. This basic process allows to identify the mismatches (additional benefits or drawbacks) among the candidates and to determine the best cost/benefit rates. In today's growing electronic world, it is common to find web pages offering comparative tables of all sorts of products. Some examples in the software engineering domain can be found in [Ader03], [NPL], [TEC], [INC]. Comparative tables although usual in some cases are difficult to rely on, they are informally stated (not criteria for their definition are provided nor are metrics formally stated). In response to these problems some authors have proposed the use of software quality models as the most appropriated framework to compare the characteristics of software products. This is the case of the works by Kontio [Kon96] and more recently Rawashdeh [Raw-Mat06] on the evaluation and comparison of COTS and Olsina [Ols99], [Ols+99] on the evaluation of web portals.

A special mention deserves the work done by Carvallo [Car+07b] that includes non-technical factors into the evaluation framework.

COTS certification by third party independent organizations has been also explored [Voa98b], [Kelk+07], [Kes07]. For instance, Yacoub [Yac+00] developed a hierarchical reference model to guide the development of COTS certification criteria, for the *National Product Line Asset Center* (NPLACE) [NPL]. The categorization of this model includes the economical investment lifecycle and the product line engineering lifecycle. The risks are associated to quality features in the model. Specific quality features are identified based on the impact that they have in the specific project and their ability to be quantified. This framework is then used to derive a set of metrics relevant to the evaluation of the process and the products.

In [Ber+03] the authors define ISO/IEC 9126-based components quality models as a base for the evaluation of the information provided by product vendors. By examining manuals tutorial and other sources of information, authors try to evaluate the set of

features included in the quality models and provide the percentage of components whose quality features could be measured from the raw information included on them. After performing the experience authors concluded that there is a gap between the information provided, and the theoretical metrics defined in the model. The authors also propose some hints to improve software evaluation based on the experience.

In [Ber+06] the same authors provide an example for measuring COTS usability. Whilst Moraes et al [Mora+07] defines a COTS-based certification process based on experimental risk assessment and provides a useful survey of the work done about COTS certification based on standard quality models as the ISO/IEC 9126-1.

2.4.2.3 ISO/IEC 9126-1 Software Quality Standard

The most standardized and widespread of the software quality model approaches is surely the ISO-9126-1 standard promoted by the International Standards Organization (ISO) which aims the development of a standard framework for software quality [ISO9126].

It belongs to the first generation of software quality engineering standards for software products, which comprises: the ISO/IEC 9126 (*software quality*) and the ISO/IEC 14598 (*software product evaluation*) which remain closely related. The main idea behind these standards is the definition of a quality model and its use as a framework for software evaluation. In addition, the ISO/IEC 9126 maps to each of the phases of the life-cycle described in the ISO/IEC 15288.

The ISO/IEC 9126 consists on four parts: 9126-1 (quality model), 9126-2 (external metrics), 9126-3 (internal metrics), 9126-4 (quality in use metrics).

The ISO/IEC 9126-1 part specifically addresses quality models. A quality model is defined by means of 6 general characteristics of software, which are further refined into 27 subcharacteristics (see Table 2.13), but does not elaborate the quality model below this level, making thus the model flexible.

Subsequent refinement of the quality model implies that subcharacteristics are in turn decomposed into attributes, yielding to a multilevel hierarchy. Intermediate hierarchies of subcharacteristics and attributes may appear making thus the model highly structured.

Attributes represent the bottom of the hierarchy and are properties that the software products belonging to the domain of interest exhibit and are computed using some metric. Therefore, the model is to be completed based on the exploration of a particular software product and its application context; because of this, we may say that the standard is very versatile and may be tailored to domains of different nature. Fig 2.6 shows the ISO/IEC 9126 conceptual model.

Many applications that make use of this standard can be found, for instance [Ols99] [Ols+99], [Ban-Da02], [Los+03], [Sim-Bel03].

Table 2.13 ISO/IEC 9126-1 quality model top level hierarchy

Characteristic/ Subcharacteristic		Description
1	Functionality	
	1 Suitability	Presence and appropriateness of a set of functions for specified tasks
	2 Accuracy	Provision of right or agreed results or effects.
	3 Interoperability	Capability to the software products to interact with specified systems
	4 Security	Prevention to (accidental or deliberate) unauthorized access to data
2	5 Functionality Compliance	Adherence to application of functionality related standards or conventions
	Reliability	
	1 Maturity	Capacity to avoid failures as a result of faults in the software
	2 Fault Tolerance	Ability to maintain a specified level of performance in case of faults
	3 Recoverability	Capability to re-establish level of performance after faults
3	4 Reliability Compliance	Adherence to application of reliability related standards or conventions
	Usability	
	1 Understandability	Effort for recognizing the logical concept and its applicability
	2 Learnability	Effort for learning software application
	3 Operability	Effort for operation and operation control
4	4 Attractiveness	Capability of the product to be attractive to the user
	5 Usability Compliance	Adherence to application of usability related standards and conventions
	Efficiency	
	1 Time behaviour	Response and processing times; throughput rates
	2 Resource Utilization	Amount of resources used and the duration of such use
5	3 Efficiency Compliance	Adherence to application of efficiency related standards or conventions
	Maintainability	
	1 Analyzability	Identification of deficiencies, failure causes, parts to be modified, etc.
	2 Changeability	Capability to enable a specified modification to be implemented
	3 Stability	Capability to avoid unexpected effects from modifications
	4 Testability	Capability to enable for validating the modified software
	5 Maintainability Compliance	Adherence to application of maintainability related standards or conventions
6	Portability	
	1 Adaptability	Opportunity for adaptation to different environments
	2 Installability	Effort needed to install the software in a specified environment
	3 Coexistence	Capability to co-exist with other independent software in a common environment sharing common resources
	4 Replaceability	Opportunity and effort for using software in the place of other software
	5 Portability Compliance	Adherence to application of portability related standards or conventions

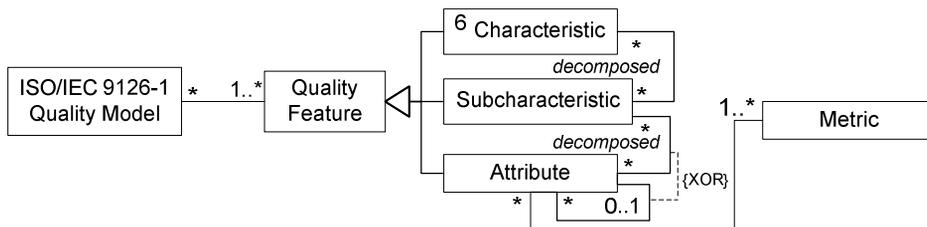


Fig. 2.6 Conceptual model of the ISO/IEC 9126-1 standard

Moreover, the ISO committee has been actively working during many years to enhance such generation of standards and they are currently in process of being replaced by a second generation of software quality standards that enhances their applicability. They are called SQuaRE, ISO/IEC 25000 (*Software Product Quality Requirements and Evaluation*) [SQU] which follows the same general concepts used in the ISO/IEC 9126-1 and ISO/IEC 14598 but enclosing them in a general umbrella.

In our proposed GOTHIC approach, the use of the ISO/IEC 9126-1 standard structure [ISO9126] played a crucial role for structuring all COTS domain related information in a unified framework in order to provide a reusable schema.

The reasons behind our decision were driven by the next considerations:

- Although several other project- or product-specific quality models and attribute catalogues (e.g., [Ader03], [Ber+03], [INC]) have been proposed, they are not so standardized, widely used and general-purpose. In fact, they are the result of many years of investigation on specific contexts such as the military, the banking, or the space ones.
- It represents a widespread used approach and most software engineers are familiar with its structure.
- The standard is highly customizable to different purposes and domains. Thus, it may be naturally adopted when building quality models for COTS in virtually all domains. However, in particular, the quality team may add new subcharacteristics specific to the domain, refine the definition of some existing ones, or even eliminate some (as explained in Chapter 6).
- It outlines a uniform framework well-suited for the integrated evaluation of all COTS selection related issues and their metrics. Also, work in the field of metrics has been applied, as [Fen-Pfl97], [Bas+04] (see Chapters 5 and 6 for details).
- It allows optimal reusability of product quality features throughout different COTS selection processes.

Summarizing, the use of the ISO/IEC 9126-1 quality standard structure was vital in our approach since it provided a very flexible structure whose hierarchy was tailored to embrace all the informational dimensions required to select and reuse COTS, as detailed in Chapters 5 and Chapter 6. This was achieved by adding, adapting, decomposing or discarding some of the characteristics and/or subcharacteristics proposed by its structure, and then completing the hierarchy with the features and metrics required for the evaluation of the specific COTS domain.

Our approach was complemented by using (when applicable) several subcharacteristics appearing in several COTS domains, as reported in [Car05-T]. It provides an ISO/IEC 9126-1-based catalogue that provides a set of subcharacteristics which appear over and over in different COTS domains. This catalogue has resulted from empirical experiences, providing a rich top-level hierarchy that can be tailored to new domains in the same way than the original ISO/IEC 9126-1. We have followed this research line and adopted some of these useful subcharacteristics in our approach related in Chapter 6.

Chapter

3

Research Method

The main purpose of this chapter is to present the process used to create the GOTHIC method. As mentioned in Section 1.4, the research in this thesis was originated from previous research and technology transfer projects. From these previous projects, several COTS selection related problems were experienced and some practical solutions were proposed, as those related in [Car05T]. However, to efficiently exploit the benefits of the former approaches, it was evident the crucial need of dealing with COTS searching open issues.

Some members of the group exposed some preliminary ideas of using taxonomies to structure the business applications marketplace (i.e. products that are used in the daily functioning of all types of organizations worldwide) to support the selection of an academic record management system [Car+04]. Although the idea was well-accepted by the community (the paper [Car+04] was awarded best paper by the 3rd International Conference on COTS-Based Software Systems –ICCBSS’04-), the proposed taxonomy was static, domain-specific, constructed by experience and did not deal with COTS marketplace characteristics. Consequently, the COTS searching problems detailed in Section 1.1 were still remaining. The need of pursuing further research for improving COTS searching gave the origin to the research line led by this thesis.

This Chapter is structured as follows. Section 3.1 discusses the research design, techniques and methods used to drive our research goal. This research has been divided into 2 phases: formative and summative. In this chapter we focus on the formative one. It is called formative because it served as the origin and evolution of the ideas and concepts presented in this thesis; whilst the second phase (presented in Chapter 10) was used to evaluate the whole method developed during the formative stages.

Section 3.2 introduces a summary of the formative case studies involving the evolution of the development of the GOTHIC method simultaneously coupled with validation; whilst Section 3.3 details the action-research stages used to develop the

GOTHIC method. Finally, Section 3.4 summarizes the answers obtained for the research questions and Section 3.5 provides a discussion about the threats of research validity.

3.1 Research Design

The general paradigm employed to reach our research goal was an iterative process based on the action-research approach [Avi+99], [Gla94] jointly with case studies [Yin03]. It was chosen because it permits a flexible design of research questions through an iterative research process, trying several action plans to solve the detected problems in a suitable way.

Action research is also known by many other names, including participatory research, collaborative inquiry, emancipatory research, action learning, and contextual action research. Action research is “learning by doing” - a group of people identify a problem, do something to resolve it, see how successful their efforts were, and if not satisfied, try again. More formally, it consists in an engineering cycle addressed by 5 steps [Gla94]:

- Step 1.* Diagnosis of a problem,
- Step 2.* Examination of options to solve the problem,
- Step 3.* Selection of options and execution,
- Step 4.* Analysis of the results, and
- Step 5.* Repeat until no further improvements are possible.

Due to the general characteristics of the action-research approach, the methods used in our research cycle were variform. The aim was to develop solutions on the basis of the observed industrial problems and needs. Following the action-research steps, the research questions (as introduced in Section 1.4) were iteratively defined and tackled as:

Step 1 (*Diagnosis of a problem*) was addressed by results of the studies related to RQ1.

RQ1: What are the actual challenges of COTS selection processes?

To tackle this research question, the process and risk issues affecting CBSD were firstly identified. This was done through a literature survey study on how COTS selection processes have been addressed both in practice and in theoretical frameworks, as well as our own experiences that also range from academic to industrial settings.

Although there are several COTS selection challenges, we decided to focus our attention on the mechanisms to deal with COTS searching since it was an area that was lacking of suitable support.

RQ1.1: What are the actual challenges of COTS searching processes?

Therefore, an in-depth study was performed for enlarging the previous survey to investigate the COTS searching associated problems both in literature and practice. We also had the opportunity to participate in an empirical study applied to some Norwegian industries (lead by Professor Reidar Conradi) in order to investigate how COTS selection processes are done and the resources that are actually used for searching such components [Ger06], [Ger07].

Results obtained in these studies motivated research question 2 and subsequent action-research steps.

RQ2: How can we support COTS searching challenges?

Step 2 (*Examination of options to solve the problem*) was largely based on literature survey and the initial ideas and shortcomings regarding solutions in other areas. These ideas were processed in several brainstorming sessions, for example, and reviewed with other researchers and experts in the field. Steps 3 (*Selection of options and execution*) and Step 4 (*Analysis of the results*) were implemented iteratively refine and re-assess research questions and the identified solutions as a whole in order to synchronize them and embody them in the proposed GOTHIC method.

Several case studies were performed in order to address the next research questions:

RQ2.1-Can goal-oriented approaches be used to produce useful results for dealing with COTS searching challenges?

RQ2.2-How can we characterize COTS in the marketplace?

RQ2.3-How can relevant information related to COTS be gathered, evaluated and synthesized?

RQ2.4-How can such information be maintained for its reuse in different COTS selection processes?

As a result of this iterative process, the GOTHIC method was tailored. The performed case studies are called “formative” since their main objective was to shape the design of the method.

The analysis and follow-up of progress were based on the results obtained by the formative case studies. Results were analyzed and evaluated by researchers and practitioners. Some interviews, meetings, and also refereed publications were used. Data obtained was used in two ways: in fine-tuning the method solution as a whole and as feedback to method development.

With respect to Step 5 (*Repeat until no further improvements are possible*), the engineering cycle was conducted in three stages or main iterations (further detailed in Section 3.3), each one with well-defined objectives and activities. The results of each stage were analyzed and used to refine the objectives and activities of the succeeding ones. Unfortunately, in the dynamic area of software engineering it will not be conceivable to state that further method enhancements would not be possible. The second phase of our research (presented in Chapter 10) was used to evaluate the whole method developed during the formative stages.

Fig. 3.1 shows the different phases of this research, the studies performed and their relation with the research questions. Results from the first two studies addressing RQ1 and RQ1.1 were further reported as state of the art and state-of-the-practice in Chapter 2. The methodological approach of the action-research process is discussed in this chapter as formative studies and results are reported throughout this thesis dissertation in the corresponding GOTHIC method’s activities Chapters. Finally, the evaluation studies of the whole method (i.e., the summative phase) are reported in Chapter 10.

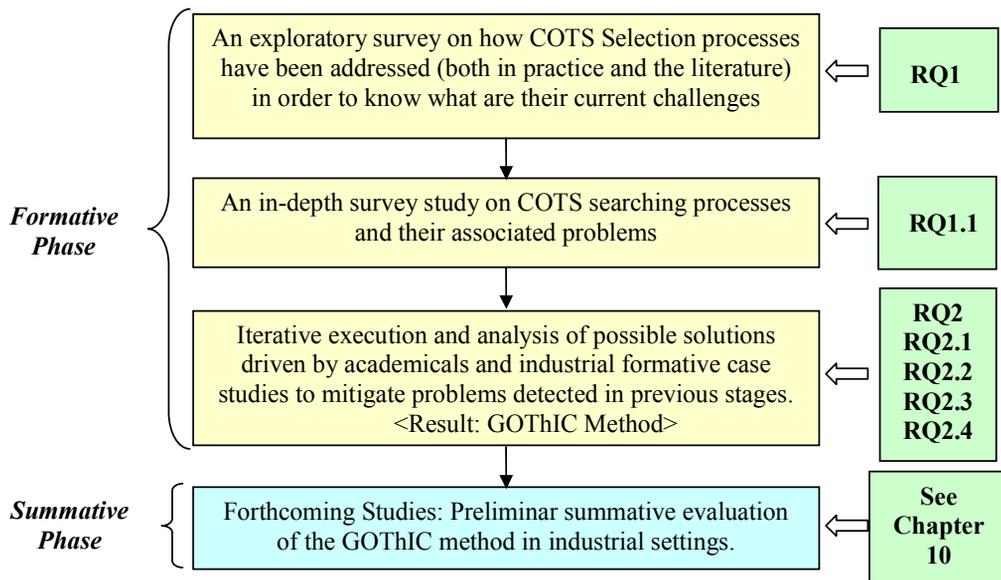


Fig. 3.1 Research phases, research questions and corresponding studies

3.2 Formative Case Studies

The case studies presented in this section enabled the development of the GOTHIC method.

Each of these case studies involves a COTS domain:

- *Business Applications Tools* (BA) case study. It refers to COTS that are used in the daily functioning of all types of organizations worldwide.
- *Software Applications Development tools* (SAD) case study. It means, COTS tools devoted to support activities related in the construction of software applications.
- *Requirements Engineering Support Tools* (REST) case study. It refers to COTS that are used to support Requirements Engineering activities.
- *Real Time Synchronous Communication Tools* (RTSC) case study. It means the COTS technologies used to enable communication and collaboration among people in a “same time-different place” mode.

Table 3.1 summarizes the availability of data for each of the case studies discussed in this chapter.

In all the case studies presented in this chapter, an exhaustive literature study related with the corresponding domain was performed.

Furthermore, an iterative analysis of the application of different methods for classifying software, component reuse, requirements engineering, and goal-oriented research was carried out. The selection of the case studies was mainly addressed by the

thesis author knowledge and expertise, as well as the availability of industrial or academic data to test with.

Table 3.1 Availability of Case Studies Data

Case Study	Data Available
Business Applications Tools	[Aya-Fra05], [Aya+04TR], [Aya-Fra06TR]
Software Applications Development Tools & Requirements Engineering Support Tools	[Aya+05TR], [Aya+04c], [Aya+05a]
Real Time Synchronous Communication Tools	[Aya-Fra06TRa], [Aya-Fra06a], [Aya-Fra06b], [Aya-Fra06c]

3.2.1 Business Applications (BA)

The BA case study was the first case study performed and was addressed to endorse the classifying mechanisms to arrange COTS and the process of using goal-oriented approaches.

In particular we used the GBRAM method for providing methodological approach to the COTS marketplace arrangement. This case study was appealing and well suited for beginning this thesis research for two main reasons. First, from previous experiences in COTS selection projects related with this domain, we had useful knowledge to start working with, and second we also had industrial data from real COTS related selection cases in order to validate our findings.

In this case, we mainly focus on learning, applying and adapting the GBRAM techniques to our purposes. To analyze the GBRAM suitability, diverse information sources were analyzed to elicit BA goals; also information from previous BA COTS selection processes was gathered and assessed.

Some interviews with experts and practitioners were performed, providing an understanding of their expected requirements for BA tools and how these requirements were satisfied by the resulting goals. They served as a source for goal and scenario identification that led us to formalize and validate the process of goal-oriented taxonomies construction.

3.2.2 Software Applications Development (SAD) and Requirements Engineering Support Tools (REST)

Once we obtained preliminary data of the suitability of GBRAM to our purposes by the BA case study, we intertwined the study and construction of several branches of the *Application Development* category. Particularly the REST subcategory was refined in deep.

The main objectives of this process were in the one hand to test the scalability of the intended method (i.e. by applying it to categories of different sizes) meanwhile it was also tuned. On the other hand, this process was also crucial to identify and address issues for making feasible the systematic construction of COTS taxonomies.

The selection of the case studies was determined by the availability of experts knowledge and COTS projects information in the RE domain given the large background of the GESSI group in this context.

3.2.3 Real Time Synchronous Communication Tools (RTSC)

The RTSC case study was intended to iteratively design and validate some improvements to the activities of the GOTHIC method.

It was mainly addressed to design and calibrate the different artifacts produced, and validate some aspects of its usability.

This case study was chosen since it represents a popular COTS domain, greatly used in almost all organizational settings (e.g., instant messaging tools). Therefore, it was easier to explain the benefits of the method to industrial audience and to get feedback of the usability of the outcomes.

3.3 Formative Research Stages Used to Develop the GOTHIC Method

Formative case studies extended over time and GOTHIC evolved as a result of its application to them. The cyclic approach we adopted to develop the method was one in which it was iteratively refined by its application to the formative case studies and the adoption of new techniques intended to overcome the drawbacks emerging from the empirical evidence gathered in each stage (as detailed below). Following this criteria, the main study of the formative research has been conducted in three stages.

The next sections summarize each one of the stages including a brief description of their objectives, the activities performed and the principal findings resulting from them. Publications are also cited because they offered valuable insights from experts as evaluation of the results of each stage.

3.3.1 First Stage: Proposal of an Initial Version of the Method

Objectives

- ◆ To identify problems associated to the searching of COTS in real COTS selection processes.
- ◆ To identify a set of mechanisms and/or techniques that could be helpful to deal with the identified problems.
- ◆ To select the mechanisms and/or techniques that better reach the objective of the research.
- ◆ To propose a method to drive COTS taxonomies construction.

Activities

- ◆ Survey of the state-of-the-art and investigation of the state-of-the-practice on COTS selection methods with special focus on COTS searching issues.

- ◆ Analysis and identification of suitable approaches to provide a systematic rationale for the construction of effective COTS taxonomies that support COTS searching and selection issues.
- ◆ To perform some case studies (BA, SAD and REST) to evaluate the suitability of the identified approaches and improve their drawbacks.
- ◆ Evaluation of the incoming results by practitioners and experts. A lightweight action research method called participatory observation was applied.

Results

- ◆ Goal-Oriented approaches coupled with decision trees structures were useful to deal with COTS marketplace evolvability and interoperability problems.
- ◆ It resulted in a very preliminary version of the method to build COTS taxonomies inspired on GBRAM and the i^* approach as a goal-modelling technique to record interoperability issues. The method was called GBTCM [Aya+05a].
- ◆ GBTCM customized GBRAM to the COTS taxonomies context.
- ◆ Some meetings with practitioners and experts were held in order to get their feedback about the intended method.
- ◆ The introduction of goal-oriented approaches resulted in an increased understanding and management of the taxonomy content; it was perceived as a more reliable COTS selection processes by the practitioners and experts.
- ◆ After case studies evaluation, some GBTCM design flaws were evident: Since GBRAM was conceived to focus on a different in a setting than COTS market goal-oriented taxonomies, more customization and improvement to our purposes was required to reach our objectives (e.g., the i^* approach did not handle explicitly the particularities of COTS selection, it seemed to be addressed more to support the development of bespoke software rather than the construction of systems based on COTS, therefore it also needed to be adapted to our purposes).

Published Results

Several publications resulted from this stage:

- ◆ [Aya+04a]. It explored the applicability goal-oriented approaches, specifically GBRAM and some other approaches suitable to our purposes. It was published in the proceedings of the *8th World Multiconference on Systemics, Cybernetics and Informatics* (SCI 2004).
- ◆ [Aya+05a]. In this paper, a preliminary version of the GOTHIC method called GBTCM (Goal-Based Taxonomy Construction Method) was proposed. Specifically, we showed how GBRAM was customized. We adjusted the inputs and modified the output. We adapted and pruned some activities in order to obtain

the statement goals to be considered for the construction of COTS taxonomies in any area. For clarifying concepts, in this paper we presented some insights of the REST case study. It was presented in the “4th International Conference on COTS-Based Software Systems (ICCBSS05)”.

- ◆ [Aya+04c] and [Aya+05c]. These publications detail the procedure for constructing a COTS taxonomy for the REST case study. The taxonomy and the obtained information reached significant benefits to the selection of systems and tools that aid to Requirements Engineering-related actors to simplify and facilitate their work. It also claims to foment the use of standards and requirements reuse in order to support different process of selection and integration of components. The first publication refers to the “VII Workshop on Requirements Engineering (WER 2004)”. This paper was selected as one of the best papers of the workshop and after a second review process it was included in the “*Journal of Computer Science and Technology. Special Issue on Software Requirements Engineering* Vol. 5, No. 2, 2005”.
- ◆ The preliminary bases for these publications were the technical reports [Aya+04TR] and [Aya+05TR] which fully detail the customization of GBRAM to our purposes and the GBTCM application to some case studies.

3.3.2 Second Stage: Validation and Improvement of the Resulting Method in Academic Cases

Objectives

- ◆ To obtain new empirical evidence of the applicability of the newly proposed method.
- ◆ To identify method problems and drawbacks and to refine it.
- ◆ To explore the use of rigorous approaches to deal with each one of the identified drawbacks.

Activities

- ◆ Finishing the construction and tuning of goal-based taxonomies for additional categories in the SAD and REST case studies following the method proposed on the first stage.
- ◆ Further study of the i^* and its customization to the COTS context.
- ◆ Exploration of diverse techniques to improve the method effectiveness.
- ◆ Identification of different artefacts and processes to improve the taxonomies construction, their management evolution and reuse.
- ◆ Analysis of potential tools for supporting the method activities as GBRAT [Ant+96], Protégé 2000 [Mus+95] and Taxonomy Tool [Gra+04].

- ◆ Performing some case studies activities by using the corresponding potential tools.
- ◆ Evaluation of the incoming results by practitioners and experts.

Results

- ◆ We found some method design flaws, some due to the use of GBRAM in a different context, others due to our method as such. The flaws were:
 - GBRAM is a requirements acquisition method; therefore the sources of information are mainly human beings, which is not the case in the COTS context.
 - GBRAM lacks of proper mechanisms to deal with the huge amount of unstructured information of the COTS marketplace.
 - GBTCM did not give the required importance to the analysis of the domain, which is more difficult than in a non-COTS context because expertise is needed not only on the domain itself but also on how this domain is represented in the marketplace.
 - GBRAM is a one-shot method, with no orientation to knowledge reuse.
 - GBTCM focused on market segments but did not consider the COTS components themselves.
 - GBTCM definition was not oriented to having tool-support.
- ◆ The *i** approach was successfully customized to handle the particularities of COTS selection; both to support the rationale of the taxonomies construction and to represent COTS market segment dependencies.
- ◆ The method flaws were successfully overcome by a more consolidated version of the method [Aya-Fra06a] presented as GOTHIC (Goal-Oriented Taxonomy and reuse Infrastructure Method). It was formally structured and defined into seven activities with their respective artefacts, processes and relationships
- ◆ A conceptual model defining the GOTHIC method was constructed [Aya-Fra06a].
- ◆ Taxonomies constructed in previous stages of our research were further addressed, taking as a base the improved GOTHIC method.
- ◆ The analysis of existing potential tools for supporting the method activities lead to decide the implementation of some modules for the DesCOTS system [Gra+04], a tool already developed in the GESSI group to support several activities of the COTS selection process.
- ◆ We discovered some drawbacks that mainly hampered the reusability purposes of the method. These drawbacks stemmed from different issues, most of them derived

from the fact that the method did not consider a more exhaustive program of information sources analysis and the reuse of knowledge from these sources, as well as an appropriate COTS domain analysis that help to manage validate and customize the Knowledge Base. Therefore, we realized that further investigation was needed to address the 1, 2, 6 and 7 method activities in an optimal way [Aya06].

Published Results

Published results of this stage include:

- ◆ The work performed together with other members of the GESSI group that was the base for customizing the *i** approach to the GOTHIC purposes
 - [Aya+04b] and [Aya+05b]. These publications detail the use of the goal- and agent-oriented models as *i** in other disciplines than requirements engineering and organizational process modelling. They justified our decision of using *i** in diverse COTS selection related processes and presented a comparative study of the three most widespread *i** variants: Eric Yu's seminal proposal, the Goal-oriented Requirement Language (GRL) and the language used in the TROPOS method. We also proposed a generic conceptual model to be used as reference framework of these three variants and we showed its use for generating specific models for the three mentioned variants, as well as for other existing proposals. The first publication was presented in an Iberoamerican forum called "*4as. Jornadas Iberoamericanas de Ingeniería de Software e Ingeniería del Conocimiento, (JIISIC 2004)*" and the second one refers to an improved version of the first one presented in a worldwide forum "*International Workshop on Agent-Oriented Software Development Methodology (AOSDM 2005)*".
 - [Gra+05], [Fra+07]. These papers defined a general methodology for building *i** Strategic Dependency Models, called RiSD for supporting the targets of the different research lines pursued by the GESSI group, included the COTS dependencies modeling pursued by GOTHIC. The first one was presented in the "*17th International Conference on Software Engineering and Knowledge Engineering*" 2005. Subsequently it was selected as one of the best papers of the Conference and after an extensive revision it was recently published in the "*International Journal of Software Engineering and Knowledge Engineering*".
- ◆ [Aya05PT] It refers to the Thesis Project document that satisfactorily fulfilled the requirements to be considered a PhD thesis proposal. Such document contained the results of the research made to this date.
- ◆ [Aya06], It tackled the research method used to develop the method, current results and ideas guiding the next stage of research. It was presented in the *13th Doctoral Consortium* at the *18th Conference on Advanced Information Systems Engineering (CAISE 2006)*.

- ♦ [Aya-Fra06a]. It presents the GOTHIC method, describing its characteristics, activities and their respective artefacts and relationships; also the conceptual model defining the method was stated. It was presented in the *9th International Conference on Software Reuse (ICSR 2006)*.
- ♦ [Aya-Fra06b]. It reports the improvements on the 3rd, 4th and 5th GOTHIC activities, that represent the goal-oriented core of the method by integrating goal-acquisition techniques and our results of customizing the *i** approach to our purposes. It was presented in the CAISE Forum collocated at the *18th Conference on Advanced Information Systems Engineering (CAISE 2006)*.

3.3.3 Third Stage: Improvement of the Method to support suitable reuse and Empirical Evaluation

Objectives

- ♦ To explore the use of rigorous approaches to deal with the identified drawbacks.
- ♦ To propose a mechanism to ameliorate these problems.
- ♦ To empirically validate such mechanisms in industrial contexts.
- ♦ To perform qualitative studies addressed to COTS selection experts and practitioners in order to evaluate and improve some GOTHIC method activities. With this aim, the author of this thesis performed a research stay at the Norwegian University of Science and Technology, IDI department (Department of Computer and Information Science), specifically in the Software Engineering group [SU] led by Professor Reidar Conradi.
- ♦ To encompass the obtained results into the whole GOTHIC framework.

Activities

- ♦ Development of target improvements to the GOTHIC method.
- ♦ Design of empirical studies, questionnaires and artefacts to obtain empirical evidence to shape the method activities.
- ♦ To perform a case study to design and validate the integration of these improvement strategies in the whole method.
 - Design of an exhaustive program of information sources acquisition, their reuse and quality assurance.
 - Design of a suitable domain analysis specifically to address COTS selection problems
 - Design of a suitable process to ensure the correctness and completeness of the constructed taxonomies, and to customize them to specific users needs.

- ♦ Evaluation of the incoming results by practitioners and experts to evaluate the effectiveness of the improvements. Participatory observation was applied.

Results

- ♦ A study aimed to empirically investigate the informational dimensions and data quality aspects that are really expected for performing an informed COTS selection [Aya-Fra08]. Results from this study are reported in [Aya-Fra07] and are the base of the design of our COTS Information Quality management strategy introduced in Chapter 5. The study was divided into two stages: the first one is addressed by a questionnaire aimed to investigate the data quality dimensions of the information. The second part of the study use the data obtained in the first one to design an interview aimed to further explore data quality measurement issues supporting COTS selection. The questionnaire used was based on the seminal Data Quality Survey Questionnaire by Wang and Strong [Wan-Str96].
- ♦ An approach to systematically tackle the COTS informational quality problems by stating a reference model based on the ISO/IEC 9126 tree-like standard, embracing quality indicators that facilitate the collection, storage, retrieval, analysis and reuse of information in a quality assurance environment was designed. It was subsequently implemented in a software tool by Fernando Messegue, as a final degree project at UPC under my advice and supervision of Dr. Xavier Franch
- ♦ The lack of comprehensive mechanisms to record and manage the required information for supporting COTS selection was overcome by designing a strategy based on domain analysis principles. Due to the diversity of the information to capture, we propose different dimensions of interest for COTS selection that were covered by different reusable artifacts. These artifacts were articulated by means of a single framework based on the ISO-IEC 9126 quality standard.
- ♦ We reached the integration of all COTS related issues into a single framework based on the ISO/IEC 9126-1 quality standard that provides a synergic approach among the artefacts produced in all GOTHIC method activities.
- ♦ To ensure trustworthiness and customization of the taxonomies constructed with GOTHIC to the users' needs, we defined a process of taxonomy validation and management based on the repeated application of transformation rules over the nodes of the source hierarchy. We define the syntactic form of the rules and also their applicability conditions as properties on the involved goals.
- ♦ Empirical evidence of COTS usage in industrial settings. During my research stay at NTNU, we performed a qualitative industrial survey as part of course TDT4735 Depth Project in Software Engineering. It was performed as the student work of Marinela Gereá and subsequently as her Master Thesis at NTNU under the supervision of Professor Reidar Conradi and Dr. Carl Fredrik-Sørensen. I actively participated throughout this study by providing my COTS-related experience in all preparation of the study, brainstorming sessions and results' analysis.

- ◆ Based on the empirical evidence obtained, we envisaged new paradigms to improve the Knowledge Base Management activity of the method. The proposed approach was prototyped in a software tool. Such software prototype was implemented by Kristian Aaslund and Simon Larsen, as a Master Thesis project at NTNU co-advised by Dr. Carl Fredrik-Sørensen and me, and under the supervision of Professor Reidar Conradi.

Published Results

Published results of this stage include:

- ◆ [Aya-Fra07]. Based on the empirical evidence obtained, this technical report presents our approach for systematically tackling the quality aspects of the COTS related information to be considered for performing an informed selection. An improved version of this report supported with improved empirical data [Aya-Fra08] is being elaborated to be submitted to a journal.
- ◆ [Mes07]. In this final career project report, the implementation issues of the COTS Information Quality management strategy are discussed.
- ◆ [Aya-Fra06c]. In this paper we presented our domain analysis approach for gathering the information needed to describe COTS market segments as required for effective COTS selection. Due to the diversity of the information to capture, we proposed different dimensions of interest for COTS selection that are covered by different domain models. These models were articulated by means of a single framework based on the widespread software quality standard ISO-9126. The paper was presented in the *25th International Conference on Conceptual Modeling* (ER 2006). An extended version of this paper was also published as a research report [Aya-Fra06TRa].
- ◆ [Aya-Fra05]. This paper tackled the process of goal-oriented taxonomy validation. We defined the syntactic form of the rules and also their applicability conditions as properties on the assigned goals. We applied them to a particular case, a taxonomy for BA. Such paper was presented in the “*16th International Conference and Workshop on Database and Expert Systems Applications*” (DEXA 05). An improved version of this work, including the formal demonstrations of the proposed transformation rules is being elaborated to be submitted to a journal.
- ◆ [Aya+07]. Based on the empirical results obtained in [Ger06] and [Ger07], we envisaged an approach to effectively build, maintain and make available the GOTHIC infrastructure to enable systematic support for COTS selection. It is based on the Open Source collaboration paradigm and social computing interaction. It was presented in the *Third International Conference on Open Source Systems* (OSS 2007)/*IFIP working group 2.13 Open Source Software*.
- ◆ [Aas-Lar07]. This Master Thesis project report details the implementation issues of the prototype of the strategy presented in [Aya+07].

Fig. 3.2 summarizes the contributions and published results in each of the research phases of this thesis. The formative research phase involved the evolution of the research work simultaneously coupled with validation (i.e., its central role was shaping the GOTHIC method by integrating successful results of the different 3 research stages described in this Chapter). Results of such stages and their relation with research questions are related in next subsection.

The summative evaluation was addressed to validate the method developed during the formative evaluation. A preliminary industrial evaluation of GOTHIC was performed during my research stay at NTNU. It was developed within the European ITEA project, Norwegian COSI (Co-development using inner & Open Source in Software Intensive products) [COS]. Results obtained so far are presented in Chapter 10.

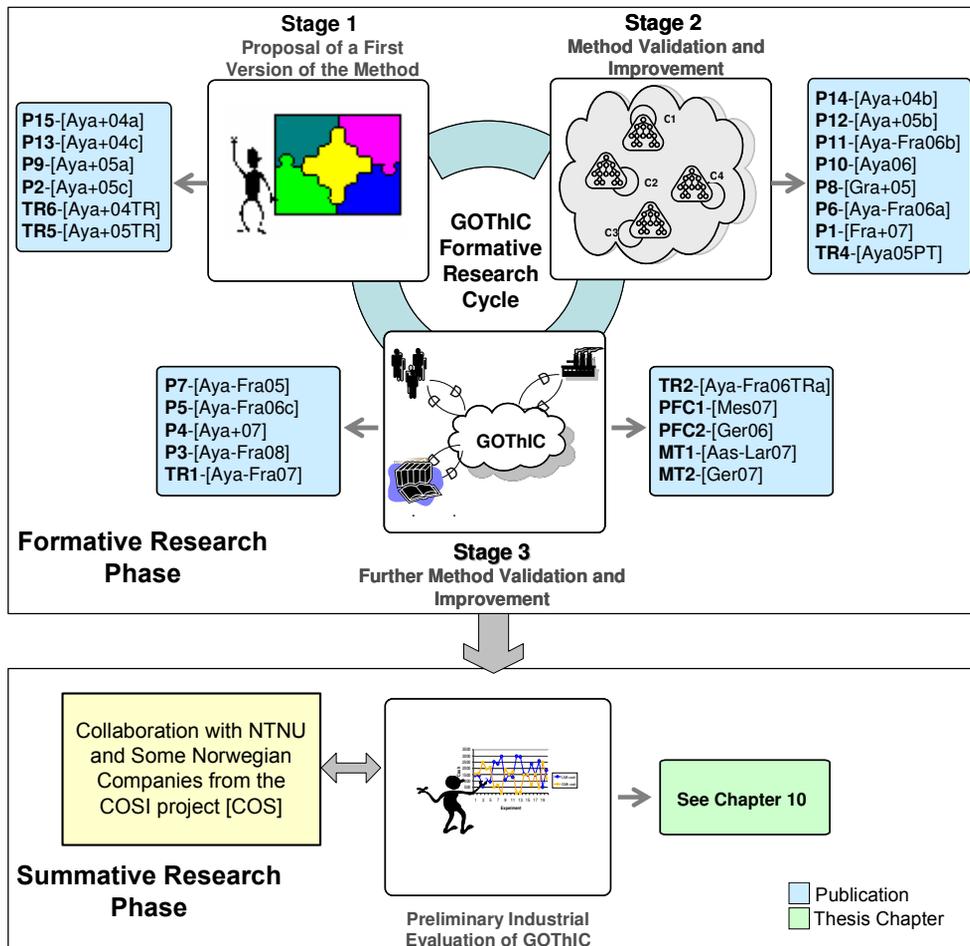


Fig. 3.2 Research phases and their corresponding publications

3.4 Answers to Research Questions

This section intends to describe the answers to the research questions based on the results obtained through our research work. Table 3.2 provides a summary of the results.

Table 3.2 Research Questions Revisited with respect to the obtained results

Research Question	Answer
RQ1: What are the actual challenges of COTS selection process?	These questions were answered by the studies reported in [Ger06], [Ger07] and literature survey. These results are enclosed in Chapter 2 of this thesis.
RQ1.1: What are the actual challenges of COTS searching processes?	
RQ2: How can we support COTS searching challenges?	To answer this question, several sub-questions related below were defined and tackled. The GOTHIC method was conceived as a harmonized integration of these results.
RQ2.1: Can goal-oriented approaches be used to produce useful results for dealing with COTS searching challenges?	Our results showed that the introduction of goal-oriented approaches for structuring COTS taxonomies resulted in an increased understanding and management of the taxonomy content; it was perceived as a more reliable COTS selection processes by the practitioners and experts asked. Further explanation of the suitability of the approaches is given in Section 2.4.1 whilst Chapter 7 describes the goal-oriented core of the resulting method.
RQ2.2: How can we characterize COTS in the marketplace?	Informational dimensions required to perform an informed COTS selection were identified from experts. Results are further described in Chapter 6. To structure such information goal-oriented taxonomies properties were applied and flexible mechanisms were defined to manipulate the taxonomies with respect to the marketplace evolution and user needs (Chapter 8 further describes this process).
RQ2.3: How can relevant information related to COTS be gathered, evaluated and synthesized?	An Information Quality management strategy was envisaged in order to deal with the diversity of COTS related information. Chapter 5 fully describes the strategy and the process that led to its development.
RQ2.4: How can such information be maintained for its reuse in different COTS selection processes?	Several processes were harmonized in order to pursue reuse. In general the GOTHIC method was designed following the Experience Factory (EF) paradigm introduced by Basili [Bas+94b] and the Learning Software Organization (LSO) [Ruh01]. Chapter 6 and 9 further describe the main reuse approaches taken.

3.5 Threats of Validity Discussion

A fundamental discussion concerning results of the main study is how valid they are. The qualitative action-research method selected involves using the researcher as the instrument for the research. A short discussion of the effects of the main researchers involved is therefore appropriate [Rob02].

The author of this dissertation has been working within the CBSD area since 2003 and has been performed research within the area for over four years. In addition, I received a formation in empirical research in software engineering by taking some academic courses as well as advanced courses. I have also worked with experienced people in the empirical research field and COTS selection research and practice that have provided their guidance. In addition, I have the opportunity to participate in some industrial both empirical experiments and qualitative studies related to components selection and usage.

Other researchers greatly involved in this research (i.e., participating and/or advising this research) have a strong background in software engineering, COTS related issues and process improvement, both in industry and academia.

To reduce the threats to validity of this research, we performed different strategies based on the threats to validity and corresponding strategies proposed by Robson [Rob02] and summarized in Table 3.3. The table defines six strategies which address three types of threats to validity, namely reactivity, researcher bias, and participant bias. The strategies may reduce the type of threat, increase it or have no effect, as defined in the table.

Table 3.3 Strategies for dealing with threats to validity [Rob02]

Strategy	Threat to Validity		
	Reactivity	Researcher Bias	Participant Bias
Prolonged Involvement	Reduces Threat	Increases Threat	Reduces Threat
Triangulation	Reduces Threat	Reduces Threat	Reduces Threat
Peer debriefing	No effect	Reduces Threat	No effect
Member Checking	Reduces Threat	Reduces Threat	Reduces Threat
Negative Case Analysis	No effect	Reduces Threat	No effect
Audit Trail	No effect	Reduces Threat	No effect

Reactivity refers to that the researcher may impact on the studied setting. Researcher bias refers to the preconceptions that the researcher may bring into the studied situation; this may impact on how the researcher asks questions or interpret results or answers. Participant bias is based on the participant's attitudes towards the study; it may be suspicion, leading to withhold information, or in the other end of the scale, companionship, leading to attempts to give the answer they think the researcher wants.

The specific strategies we take into account for the validity threats of our research are explained below according to the strategies related in Table 3.3.

Prolonged involvement refers to that researchers have a long-term relationship with the studied object. In our research, we have had a long-term research involvement in formative case studies to try the effectiveness of different techniques for dealing with the detected COTS searching problems. To provide a balance between researcher bias threats we considered necessary the evaluation of the results of the case studies for practitioners and researchers (e.g., meetings, interviews and submitting papers with the results of the case studies as well as holding seminars and talks to explain and discuss our results). This was also assumed to provide a balance between reactivity threats.

Triangulation refers to having multiple sources for the study information. In this research, it was generally attained in three different ways, which further increases the validity. A summary is provided below:

- ◆ Data triangulation. Multiple data sources were used in the study, interviews, archival and informal.
- ◆ Investigator triangulation. In some cases, interviews or surveys were performed by other researchers (commonly students) and their results compared with our own results.
- ◆ Theory triangulation. Case studies results were evaluated by multiple perspectives (i.e., practitioners and researchers). Therefore their feedback was also varied. This approach also provides negative case analysis, i.e., trying to find another explanation to the observed phenomenon.

Peer debriefing means that researchers involve independent peers to review the research. As mentioned above, results from case studies were evaluated by practitioners and researchers on the area.

Member checking refers to involving the interviewing or questioning people in giving feedback to the researchers. This was used continuously, by receiving useful feedback from practitioners and researchers.

An extensive approach to audit trail was used in the research, i.e., keeping a full record of all case studies, from data collection all through the analysis. This data, as well as the useful feedbacks and comments from researchers and practitioners have been recorded by using the NVivo Software [NVi], keeping full traceability back to any study result. Moreover, the analysis of results in each case study was conducted with rigor.

External validity, as stated by Wohlin [Who+00] is concerned with generalizations of results outside the scope the case studies performed in our research. At this respect, although a case study always is coloured by its specific context, we try to perform diverse case studies involving COTS domains of different sizes and criticality. By the feedback from practitioners and researchers, we can say that it seems reasonable to generalize the activities of the GOTHIC method in the way that their prescriptions may be useful and applied to any domain and not only in the performed case studies. Nevertheless, we are aware that some fine-tuning processes must be performed depending on the domain to be dealt with, and; more extensive studies must be applied in industrial practice to demonstrate the method validity (as explained in Chapter 10).

In summary, reasonable countermeasures to validity threats have been implemented in the research. It may therefore be contended that the validity of the research is sound.

Chapter

4

The GOTHIC Method

This thesis addresses the critical nature of the COTS marketplace (i.e., diversity, size, evolvability, and interoperability) and the lack of available and well-suited information to obtain a quality assured search.

Existing COTS selection approaches (as those surveyed in Chapter 2) usually fail to address the process of searching and reusing COTS and information about them from the marketplace [Aya-Fra06a].

Based on several industrial and academic experiences and aimed to contribute to these problems, this chapter introduces the *Goal-Oriented Taxonomy and reuse Infrastructure Construction* (GOTHIC) method. It was conceived as a set of interrelated and synergic strategies to deal with the problems mentioned in Section 1.1.

It provides prescriptive support for the construction and maintenance of flexible goal-oriented taxonomies that describe the contents of the COTS marketplace and, also for organizing and recording COTS related information and knowledge in order to get full potential of suitable software and knowledge reuse.

The approach can be used in the context of an organization or in a generic or global context. The reuse infrastructure obtained by the application of the method may be used in several COTS searching processes for obtaining the appropriate criteria for locating the most appropriate kind of components and reuse information about them.

In contrast to other approaches proposed for classifying COTS, GOTHIC assumes that COTS information has not been previously gathered and provides support for gathering, assessing and recording such information in a suitable and reusable way. The goal-oriented nature of the method and the diverse artifacts produced during its

application provide the rationale for structuring, representing, evolving, and reusing COTS marketplace related information in a repository or knowledge base.

The underlying principle of the reuse infrastructure obtained with GOTHIC relies on the flexible goal-oriented classification schema that helps to categorize all COTS related with a domain in an understandable way and in a coarse-grained level. It also supports the assessment of components by stating their related information in a uniform way.

Next sections of this chapter introduce an overview of the method and the main benefits expected from it whilst in Chapters 5 to 9 the strategies for dealing with these activities are further explained and illustrated.

4.1 GOTHIC: A Method to Build a COTS Domain Reuse Infrastructure Based on Goal-Oriented Taxonomies

The GOTHIC method has been conceived as a process in which six main activities are iterated and/or intertwined to develop an evolvable COTS domain reuse infrastructure. Such reuse infrastructure supports COTS searching processes in such domain and reuses the knowledge gained in each one of these processes.

Fig. 4.1 shows the high-level activities as boxes, they are: Exploration of Information Sources (Activity 1); COTS marketplace domain analysis (Activity 2); Identification, refinement and statement of goals (Activity 3); Establishment of dependencies (Activity 4); Goal-Taxonomy Structuring (Activity 5); Taxonomy validation (Activity 6); and Knowledge base management (Activity 7).

The dotted box denoted as *goal-oriented core* groups the activities related with the identification, manipulation, and representation of goals to construct the evolvable taxonomies. Although all the method activities are illustrated as sequential for clarity, these activities do not need to be performed sequentially; rather, they may be performed concurrently and iteratively with occasional interleaving.

In the next paragraphs, the GOTHIC method activities are briefly introduced to provide an overview of the method. In Chapters 5 to 9 they are further detailed.

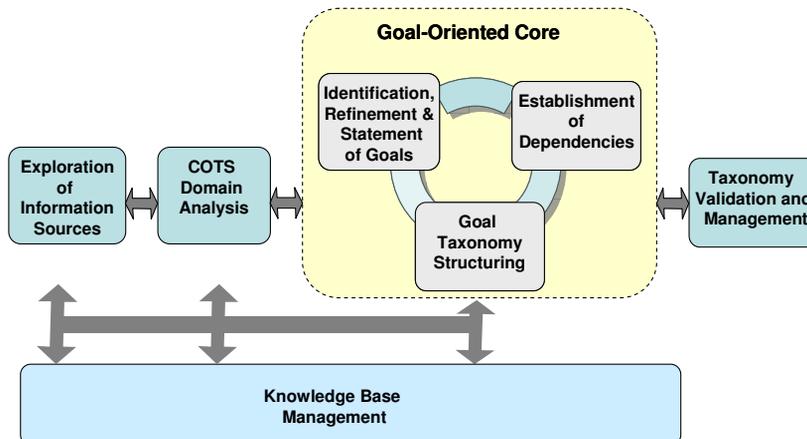


Fig. 4.1 High-level activities of the GOTHIC method

Activity 1: Exploration of information sources

This activity provides support for locating and choosing relevant information by means of a suitable information acquisition program which allows extracting knowledge from the COTS domain by reconciling the characteristics of the available sources (e.g., diversity of its type, supporting media, cost, etc.) with those of the taxonomy construction project.

Details of this process are found in Chapter 5 and [Aya-Fra07].

Activity 2: COTS marketplace domain analysis

In order to facilitate its analysis and structuring, COTS marketplace has to be broken into domains grouping related services. These domains should be described by several characteristics. The goal of this activity is to identify, record and represent the basic elements and relationships of the COTS domain that is being analyzed for reusability purposes. It is done from the information sources acquired by the previous activity (Activity 1). Due to the diversity of the information to capture, we propose different models to record the five dimensions of interest for COTS selection: Fundamental Concepts, Functionality, Quality of Service, Non-Technical Description, and Interoperability. Some of these models are further constructed and detailed as part of other method activities which are run concurrently (e.g. Activities 1, 3, 4 and 5). Finally, as part of our proposed COTS domain analysis strategy, these models are articulated by means of a single framework called *Domain model* that is based on the ISO-9126-1 quality standard.

A further explanation is given in Chapter 6 and [Aya-Fra06c].

Activity 3: Identification, refinement, and statement of goals

The use of goal-oriented approaches helped us to understand and capture the objectives covered by a COTS domain in a natural way and at various levels of abstraction whilst goals remains more stable with respect to changes. The process that conform this activity is based on the GBRAM method. Heuristics are provided to perform these iterative activities which main objectives are: *Identification* aims at extracting goals from available sources obtained from Activity 1 by applying different goal-acquisition techniques. *Refinement* entails the goal refinement considering obstacles, scenarios to uncover hidden goals and mechanisms to discover synonymous or duplicated goals. *Statement* consists on expressing the goals in a systematic way; it also specifies the pre/post conditions for the goals. To summarize all this information we use goal-schemas.

Further information of this process can be found in Chapter 7 and [Aya-Fra06b].

Activity 4: Establishment of dependencies

This activity helps to identify and establish the different dependencies that a COTS category or market segment may have with another by analyzing the goals obtained in the previous activity. Specifically, we have identified four basic types of dependencies: Goal dependency, Task dependency, Resource dependency; and Soft-goal dependency.

To represent these abstract relationships, we use a combination of goal- and agent oriented model called *i** SD models [Yu95].

Support in identifying the appropriate kind of dependencies and their recording is given in [Gra+05] and [Aya-Fra06b]. Chapter 7 also details this activity.

Activity 5: Goal-taxonomy structuring

This activity provides support for organizing the COTS related information in a hierarchical structure. The rationale to organize the goals comes from the analysis of pre- and post-conditions stated for each goal. Subsequently, goals are operationalized in terms of variables which state the semantics to each taxonomy node for being used as a decision tree. The tree is composed of two types of nodes, market segments and categories. Market segments are the leaves of the taxonomy and represent atomic entities covering a significant group of functionality; whilst categories serve to group related market segments and/or subcategories.

Further information of this process can be found in Chapter 7 and [Aya-Fra06b].

Activity 6: Taxonomy validation and managing

In order to be useful for driving COTS search processes, we require some mechanisms to make the taxonomy flexible and at the same time to ensure its trustworthiness. Therefore, in this activity we address the process of taxonomy validation and managing by using decision trees properties. The process was defined as the repeated application of some stated transformation rules over the nodes to manipulate the hierarchy until reaching a stop condition. This process not only ensures the trustworthiness of the resulting taxonomy, but also provides a means to manipulate the taxonomies to represent the abstraction level required by the context of use for which the taxonomy is intended.

This process is detailed in Chapter 8 and published in [Aya-Fra05].

Activity 7: Knowledge base management

The method is based on an Experience Factory (EF) perspective [Bas+94b] in order to enable the Learning Software Organization (LSO) [Ruh01] paradigm. Therefore, all the method's activities and their resulting artifacts are aimed and synchronized to build a repository capable of support reuse, maintenance and evolution of the obtained knowledge gained during the COTS reuse infrastructure process, as well as the knowledge gained in each COTS selection experience. This activity refers to populate the COTS repository or knowledge base and managing its contents. It is important to remark that it is aimed to enable the storage, location and retrieval of COTS information (not necessarily the components themselves).

The strategy defined to effectively populate and maintain the resulting repository or knowledge base is detailed in Chapter 9 and [Aya+07].

A generalized summary of the high-level inputs and outputs of each GOTHIC activity is presented in Table 4.1.

Table 4.1 High-level Inputs and Outputs of GOTHIC Activities

Activity	Inputs	Outputs
Exploration of Information Sources	COTS-related information	Prioritized set of COTS related information sources and their management
COTS Domain Analysis	Prioritized set of COTS related information sources Diverse models completed and refined by other method activities	Domain Model of the COTS market segment
Identification, Refinement and Statement of Goals	Prioritized set of COTS related information sources	Goal Schemas. Domain Model refinement
Establishment of Dependencies	Prioritized set of COTS related information sources Goal Schemas	COTS dependencies relations stated as <i>i*</i> SD models. Domain Model refinement
Goal Taxonomy Structuring	Prioritized set of COTS related information sources Goal Schemas COTS dependencies relations	Variable assignment to goals that represent taxonomy nodes Goal-oriented hierarchy
Taxonomy Validation and Management	Goal-oriented hierarchy	Correct, complete and ad-hoc goal taxonomy
Knowledge Base Management	All inputs and outputs of the previous activities are recorded	A domain reuse infrastructure composed of all the evolvable artifacts produced throughout the method, residing in a domain knowledge base or repository organized on goal-oriented taxonomies aimed to support the storage, location, and retrieval of COTS information (not necessarily the components themselves).

4.2 Characteristics of the Proposal

The UML class diagram in Fig. 4.2 defines the form that the repository or knowledge base aimed by the GOTHIC method exhibits.

At the heart of this model lies the taxonomy composed of two types of nodes, market segments and categories, which are characterized by their goals. From a semantic point of view, market segments stand for the basic types of COTS available in the marketplace (e.g., the market segment of anti-virus tools or spreadsheet applications). As a consequence, COTS are associated with market segments and not with categories (although an indirect relationship exists, because market segments belong to categories). Components may cover more than one market segment.

From the analysis of some information sources which are gathered, analyzed, and prioritized according to several characteristics (as related in Activity 1), the domain analysis activity (Activity 2) is performed. In fact, the domain analysis activity leads the

performance of all GOTHIC method activities where several artifacts are produced. These artifacts are bounded to taxonomy nodes.

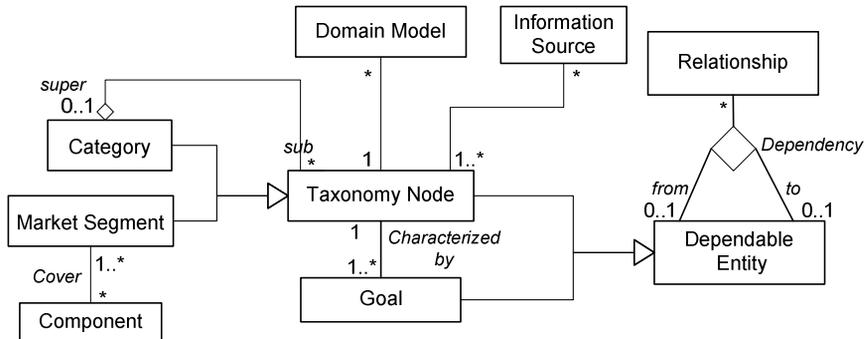


Fig. 4.2 Conceptual model for goal-oriented COTS taxonomies: overview

The identification, refinement and statement of goals (*Activity 3*), the establishment of dependencies (*Activity 4*) and the goal taxonomy structuring (*Activity 5*) enclose the goal-oriented core of the method. Dependencies among nodes provide a comprehensive view of the marketplace. In the case of dependencies among market segments, they stand for interoperability issues (e.g. mail server systems depend on anti-virus tools to support integrity). Concerning categories, more abstract relationships are modeled. In addition to taxonomy nodes, dependencies may involve goals, when the relationship can be established more accurately. The *Dependable Entity* super-class allows modeling this situation comfortably. Note that dependencies are represented by a ternary association, because they involve two elements (depender and dependee) and the relationship itself.

To validate and manipulate the resulting goal-taxonomy, a well-defined process is provided (*Activity 6*). Finally, the ultimate goal of the method is to populate and maintain a domain knowledge base (*Activity 7*) with the deliverables of all other method activities that have been accordingly designed.

In next chapters, the model presented in Fig. 4.2 will be further detailed during the explanation of each method activities from Chapter 5 to 9.

4.3 Intended Improvements

Based on the State-Of-The-Art, State-Of-The-Practice, the problems identified in Section 1.1, and the objectives of this thesis, we expect the GOTHIC method provides the following high-level improvements (which are then further discussed from Chapters 5 to 9, and summarized in Chapter 11):

4.3.1 Improvement on the Effectiveness of COTS Marketplace Organization

To know the kind of COTS that are available in the marketplace and to know which of them could be useful to solve a specific problem, structuring the marketplace is a growing need. It facilitates the searching, comparison, and the decision about the appropriateness of the COTS to the system-to-be. Furthermore, it could be the basis to reuse COTS related knowledge to gain the full potential of large-scale software reuse.

GOTHIC provides methodological support to COTS taxonomies construction and reuse of knowledge and information gained in each selection process

To effectively deal with several COTS searching problems, COTS domains should be described by several characteristics and, several models have to be constructed for reusability purposes. This requires a well-defined methodological guidance and some techniques for the construction of COTS taxonomies that are the backbone of a COTS reuse infrastructure.

The activities and sub-activities of the GOTHIC method are well-defined and several well-founded techniques have been used to overcome COTS marketplace related problems. Several mechanisms from other areas are used as heuristics and rules based on decisions trees properties to support COTS taxonomies construction and organize COTS related information. Such methodological support improves the taxonomies understandability, usability, effectiveness, and reliability. Furthermore, the GOTHIC method does not depend on the extent and characteristics of the addressed domain or taxonomy built (e.g., a small part of the COTS marketplace such as photo processing software, or a huge portion like business applications).

GOTHIC uses an iterative and intertwined approach to the construction of a COTS domain reuse infrastructure

Due to the volatile nature of COTS domains, an iterative approach is a basic requirement. On the one hand, diverse information related with the COTS domain being addressed is identified as knowledge of the domain increases, thus the artifacts produced during the different activities of the method may be extended and/or refined all the way through the process. On the other hand, the level of detail to which the infrastructure has to be constructed, depends on the context of use and the final application for which the infrastructure is intended. Moreover, an iterative approach, in which the construction steps can be intertwined at any point helps to refine and evolve the reuse infrastructure not only to the desired level of detail but also to the arising of new COTS products characteristics.

GOTHIC provides mechanisms to evolve and adequate taxonomies to ad-hoc needs whilst ensuring their trustworthiness

COTS taxonomies may be manipulated with GOTHIC by the application of a set of transformation rules applied to the taxonomy nodes. These rules allow accomplish a desired level of detail whilst ensuring the taxonomy trustworthiness.

The resulting taxonomy provides an external view enclosing all COTS related information that is: well-founded (with a clear rationale of the proposed structure), validated (sound, complete, pair-wise disjoint and balanced) and ready to browse (using the defined classifiers).

4.3.2 Improvement on Managing COTS Marketplace Characteristics

Goals imply stable concepts. Based on the notion of goal, the method aims at building abstract, well-founded, and stable taxonomies, which may evolve as the marketplace does.

GOTHIC uses goal-oriented approaches to deal with COTS marketplace evolvable characteristics

Specifically, some of the problems related in Section 1.1 are dealt in the next way:

- ♦ *Uncontrolled COTS marketplace*: Goal-oriented taxonomies offer a natural way to be understood and used to categorize any COTS product.
- ♦ *Growing size of the COTS marketplace*. Appearance of a new market segment or category is easier to handle than in other approaches, since it requires locating its place in the taxonomy using the defined classifiers, and once there even some useful artifacts are inherited (e.g., quality models and glossaries of the domain). Proliferation and trustworthiness of COTS related information is taken into account by prioritizing information sources in the basis of specific criteria (e.g., time, money, reliability, ...).
- ♦ *Rapid changes in the COTS marketplace*. This fact points out the need to separate conceptually the COTS from the services that they cover. Thus, taxonomy nodes do not stand for types of COTS available but for related groups of functionalities, it makes the taxonomy more robust with respect to the segment barriers movement effect mentioned in Section 1.1.

4.3.3 Improvement on COTS Information Rendering

The GOTHIC method integrates some strategies for gathering the information needed to describe COTS market segments as required for effective COTS selection.

GOTHIC integrates all informational dimensions for selecting COTS in a uniform Domain Model (meta-model)

Due to the huge amount and diversity of the information to capture, the method includes an information acquisition and reuse strategy combined with a domain analysis approach that not only impacts positively on reuse, but also ameliorates some well-known obstacles for COTS selection processes success.

- ♦ *Type of descriptions available for COTS*. The GOTHIC method identifies activities to cope with the diversity, lack of structure, reliability and reuse of information about COTS. It systematically tackles these information quality problems by stating a reference model embracing quality indicators that facilitate the collection, storage, retrieval, analysis and reuse of information in a quality assurance environment (See Activity 1).
- ♦ *Lack of Standards for COTS descriptions*: To avoid the lack of COTS standardization concerns, GOTHIC provides a unified framework based on the well-known ISO-IEC 9126-1 quality standard that integrates all the informational dimensions for selecting COTS in the domain (i.e., the GOTHIC's domain model). This artefact is bound to taxonomy nodes and is reused through them. Thus, once the taxonomy node that covers the required goals of the user is located, this artifact (domain model) helps to elicit and negotiate the requirements, making easier the evaluation of components in a uniform way. For doing so, we can proceed manually, or use tool support ranging from a simple spreadsheet to a more sophisticated tool, e.g. our DesCOTS system [Gra+04], as explained in Chapter 9.

- ♦ *Dependencies among COTS.* We represent explicitly these dependencies with a model built with i^* , a widespread and accepted notation in some other disciplines (e.g., requirements engineering, and agent-oriented development). It allows not only to represent and record these dependencies but also to transfer this knowledge from one experience to another.

4.3.4 Improvement on Managing and Reusing COTS Related Issues

The GOTHIC method deals with the lack of comprehensive mechanisms to record, manage and reuse the required information for supporting COTS selection by identifying several artifacts which contribute to the reuse of knowledge gained in each experience as the “experience base” paradigm introduced by Basili [Bas+94] and the Learning Software Organization (LSO) approach [Ruh01].

GOTHIC allows the construction and maintenance of an evolvable COTS domain reuse infrastructure or knowledge base supported by goal-oriented taxonomies aimed to be gradually improved by its application to several COTS selection processes

The method proposes suitable models to record and reuse the informational dimensions required to select COTS, and provides effective strategies to harmonize them in an ISO/IEC 9126-1 standardized framework. Moreover, GOTHIC provides mechanisms to allow the incremental growth of the knowledge base, containing meta-data information describing COTS.

4.4 Intended Audience

The GOTHIC method has been envisaged as a way to enable efficient COTS information and knowledge reuse. As it embraces a process for building a COTS domain reuse infrastructure, it is mainly addressed to organizations that usually carry out COTS selection processes and find valuable to accumulate experiences from past selection processes in order to improve their practice. Chapter 9 provides an overview of the kind of organizations and schemas where the method could be used.

In a strict sense, given the required investment on building and maintaining the GOTHIC reuse infrastructure, we argue that the approach is mainly oriented to medium and large size companies. However, we have actually envisaged other possible application to up-start, populate, and maintain the reuse infrastructure to make the approach feasible to all kind of organizations in an open and collaborative environment (it is discussed in Chapter 9 and detailed as future work in Chapter 11).

4.5 Applicability of the Proposal

We argue that the extra effort needed for applying GOTHIC to build a COTS domain reuse infrastructure is more helpful and less risky than acting reactively in searching components. So, we suggest the applicability of the method in organizations that deal with big projects and multiple selections occur in which inefficient decisions would have critical consequences.

To be more precise, GOTHIC requires the following characteristics to be applicable:

- The application of the method should be addressed to a domain that is of general interest. This means that a great deal of COTS selection processes is taking place from its related market segments. Some examples are: communication infrastructure, ERP systems, security-related systems, etc. In these contexts, the number of selection processes that take place will be high and then reusability of the models likely to occur.
- The addressed market segments offer COTS of coarse-grained granularity. This makes domain understanding more difficult, time-consuming and cumbersome and therefore domain analysis and taxonomy construction are helpful. Market segments such as CRM and ECM systems are typical examples, whilst time or currency converters are not. In these cases, having knowledge available and classifiers to know when a market segment is of interest is a great help. This last point is especially appealing in those selection contexts in which the organization that is interested in the selection does not have clear requirements about the kind of system needed.

4.6 Summary and Discussion

The method describes seven main activities and several sub-activities or steps, designed to drive the construction of COTS domain related taxonomies that describes the contents of the COTS marketplace (see Fig. 4.3).

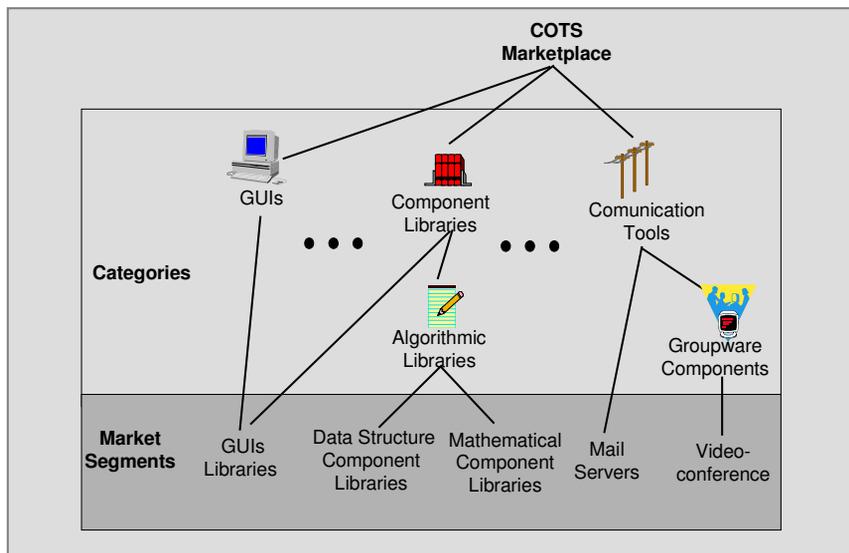


Fig. 4.3 Overview of the COTS marketplace structuring

Such taxonomies construction lead to the creation of domain-knowledge repositories populated with COTS related information –not necessarily the components itself-. Such construction has a well defined rationale which increases the efficiency of the process whilst improving the reliability of the deliverables. The resulting artifacts can be seen as a friendly and flexible taxonomy and knowledge base that can support the COTS selection process.

The method is based on an experience factory [Bas+94] and Learning Software Organization [Ruh01] perspectives, which refers to an infrastructure for reusing and managing life cycle experience, knowledge, processes and products for software development. Experiences are collected from software development projects, and are packaged and stored in an experience base. By packing, we mean to generalize, tailor and formalize experiences so that they are easy to reuse and manage. The different deliverables of the method are summarized in a meta-quality model of the domain (based on the ISO/IEC 9126-1 quality standard) that is flexible and may be instanced to the branches to the taxonomy and the components that belong to them, in such a way that several important information (e.g. functional, interoperability, quality, non-technical) is easy to find and can be reused and compared.

Chapter

5

Activity 1:

Exploration of Information Sources

Nowadays, the amount of information available about COTS is vast and still growing. As it was mentioned previously, to select COTS, decision-makers have to face not only the current diversity of COTS types available in the marketplace, but also the great deal of widespread, heterogeneous, and unstructured information describing each of them [Ber+03], [Tau+04], [Cec+06], [Ast+06], [Ber+06]. The quality of this information largely determines the quality of the decisions made, and ultimately affects the quality of the whole software system and its development [Ber+03], [Ast+06]. Since COTS selectors must rely on the information for their decision-making processes, ensuring Information Quality (IQ) is a critical success factor.

Over time, librarians and other information professionals have developed a set of criteria to be used to evaluate IQ based on careful experts' examination (e.g., authority, format, scope, etc.) [Boo-Smi00]. However, these criteria are too general and do not provide much guidance to the particular problem of COTS selection. Some recent approaches propose the use of automatic or semiautomatic search engines to identify COTS, e.g., [Sea+98], [Yan+06], [Sja-Beu06] (see Chapter 2). But to the best of our knowledge, they have not reached a generalized consensus on their utility for the community and do not address the IQ problems. Therefore, IQ is still a critical open issue from the COTS selection perspective [Ber+03], [Cec+06], [Ank+03], [Sim-Dil06].

The goal of this research is twofold:

- ▶ To develop a comprehensive framework that states those important IQ aspects to perform an informed COTS selection in order to be used as metadata to aid COTS related information searching and analysis. Furthermore, to investigate how these aspects may be feasibly gauged using a hierarchical quality model schema, providing metrics to assess the value of the information.
- ▶ To offer a tool-supported conceptual model to add capabilities for recording, managing and reusing IQ metadata in several COTS domain selection processes.

This chapter reports current results, and is structured as follows. Section 5.1 provides a brief background of previous research and greatly justifies the need of this study. Section 5.2 details the processes, methods and techniques used to capture IQ needs in the context of COTS selection and the associated metrics. Section 5.3 encloses the obtained results related in the previous sections in a conceptual model that is used as a reference for systematically support COTS selectors decisions-making.

5.1 Background

Domain analysis is the basis to build a reuse infrastructure (see Chapter 6). However, the success of domain analysis is directly related to the quality of the information used to perform this activity.

The industrial evaluation of GOTHIC shown the critical difficulty reported by software engineers to collect, process and analyze the vast amount of information sources for performing the process of domain analysis and therefore for reaching a trustworthy decision-making.

While many approaches exist to state quality characteristics and requirements for improving COTS selection processes, and the quality of the COTS documentation has been considered as a crucial quality aspect affecting their usability [Ber+06]; the issue of where and how to get trustworthy information about them in an efficient manner has been left out [Bob+02]. It is causing several over-costing problems or even abandoned projects because of wrong decisions based on untruthful information [Cha02].

In order to overcome the risks associated to poor domain quality information [Shan+03], we realized the need of integrating an *Exploration of Information Sources* activity (i.e., an IQ management strategy) into our GOTHIC method to perform an efficient and proactive *COTS Domain Analysis* (Activity 2), introduced in Chapter 6.

As stated in Chapter 2, quality is a difficult term to define, and there is no a single definition or standard of quality. Most of the work currently conducted in the area of information quality research has looked at quality from the organisational or information producer perspective [Bur+04]. Therefore, most of IQ definitions that have been defined for use by information providers are not suited to the information consumer.

To perform this research, the information consumer's perspective of quality was followed. The two main characteristics of this approach are:

- ▶ The consumer has no control over the quality of available information.

- ▶ The aim of the consumer is to find information that matches their personal needs, rather than provide information that meets the needs of others.

Thus, we consider COTS selectors as COTS related information consumers. The typical information consumer wants to find the best available information that meets their requirements, at that point in time, in their current domain of interest and project. This may not necessarily be the best possible result as the consumer often has restrictions, such as the time available to spend searching for information. For example, the consumer may need the information quickly so is unable to wait several hours while all possible sources of information are investigated to find the best result across all sources. In this case the consumer will be willing to accept the best possible results obtainable within the given restriction, such as currently available data, data within their price range, or all data that can be obtained within a specified time limit.

Moreover, we use the notion of information product [Wang+03] (i.e., information is treated as a product).

In this context, we aimed to develop a framework embracing quality indicators that facilitate the collection, storage, retrieval, analysis and reuse of COTS domain information in a quality assurance environment [Aya-Fra07]. It captures the aspects of IQ that are important to COTS selection and provides a systematic approach for supporting COTS selectors to decide information sources to use according to their specific quality project needs. The approach is based on the use of heuristics that support the extraction of knowledge about the COTS market segment of interest by reconciling the characteristics of the available sources with those of the taxonomy construction process.

5.2 Capturing Information Quality Dimensions for COTS Selection

The work presented here is based on relevant approaches from the IQ research, e.g., [Boo-Smi00], [Shan+03], [Wan-Str96], [Lee+01], [Wang+03], several industrial experiences and case studies analyzing COTS selection processes (see Chapter 3); interviews run in software companies [Ger06], [Ger07], and is being iteratively improved with empirical data obtained from an ongoing on-line questionnaire [Aya-Fra08].

Under action-research premises, we iteratively identified the IQ problems in the COTS selection setting, trying out suitable IQ research approaches to resolve them, adapting and evaluating how successful such strategies are in practice, until a satisfactory solution comes out. Interviews were used to conduct an explorative survey in some Norwegian software companies. They are fully reported in [Ger06], [Ger07].

These studies consisted of semi-structured interviews conducted to managers, software architects and developers involved in COTS selection projects. Our first goal was to collect information about the problems they face to get COTS related information, followed by inquiring what they mean by IQ to perform an informed selection. As mentioned above, this approach is currently being improved with a more extensive study [Aya-Fra08]. Current results are detailed below.

5.2.1 Identifying COTS Selection Information Problems

To find out which are the most relevant dilemmas that companies face whilst processing information during COTS selection, we asked interviewees about the resources they usually used to locate COTS and/or information about them, as well as the perceived utility of such information for performing the different COTS selection activities.

Table 5.1 COTS Related Information Sources Types

Types of Source	Description	Information Rendering	Examples
Existing Hierarchies/ Taxonomies	COTS categorizations that offer descriptions of diverse COTS with different objectives.	<ul style="list-style-type: none"> • Classifications • Categories; • Glossaries 	ComponentSource.com KnowledgeStorm.com SourceForge.net
Vendor Information	Information provided by the COTS supplier as its characteristics, documentation and comparatives with previous or existent versions	<ul style="list-style-type: none"> • Brochures; • Evaluation forms; • Benchmarks 	Any Commercial Firm
Related Standards	Since there is no specific standardization concern to describe COTS, some industrial organizations group the main vendors of particular domains maintain up-to-date information that sometimes could be considered as a reference.	<ul style="list-style-type: none"> • Descriptions; • Glossaries 	Internet Mail Consortium (IMC), Workflow Management Coalition (WfMC), Enterprise Content Management Association (AIIIM)
Independent Reports	Third party organizations ranging from research to consultant often provide support for selecting COTS	<ul style="list-style-type: none"> • Papers, • Comparative Tables • Descriptions and tips 	Scientific: Specialized Journals, Textbooks Divulagation: Specialized Websites Technical: Gartner, Forrester, ...
Experiences on the Field	Knowledge or practical information usually provided by experts or domain stakeholders that relate own experiences or lessons learned.	<ul style="list-style-type: none"> • Technical reports • Forums • Talks • Seminars and Courses 	ICCBSS panels, SEI courses, Luncheons, interviews, CeBASE repository,
Test of Tools and Systems	Test descriptions of tools which have been really used. They allow envisaging the real behaviour of a COTS in a specific environment	<ul style="list-style-type: none"> • Test results; • User's manuals 	Test results Tucows.com CMSmatrix.org
Others	Provide some specific functionality to find expected COTS functionalities. They can range from specialized searching tools to open source code detection tools	<ul style="list-style-type: none"> • Queries 	Agora Koders Tool Google.com/codesearch

Summarizing the answers, in Table 2.9 (introduced in Chapter 2) we provided a list of some existent resources, their intended objectives, characterization mechanisms, retrieval schema, information rendering, and some additional information they offer. Such table intends to serve as a guide for the identification of potential information sources for the domain of interest using information acquisition techniques (e.g., literature review, web screening, etc.). Since there are a great variety of types of information sources available, we decided to group them for assessing their characteristics and actual IQ problems. Table 5.1 shows this grouping and some representative examples. Based on our findings, also heuristics were designed to support this process (see Section 5.3.1).

Interviewees agreed that extracting COTS information from these resources is a critical process because they do not have control over their availability, accessibility, heterogeneity, impartiality, incompatibilities, inconsistencies, and mistakes that make difficult to guarantee IQ and lead to failures that cost dearly. This could be the reason why there was not consensus of the utility of these resources in the COTS selection community. Furthermore, we found that in order to reach project constraints (mainly in terms of time and resources), actual decision-making processes for finding and/or processing COTS information are rarely documented and usually based on vague factors as own experiences and intuition, even in the cases of organizations that periodically performed COTS selection projects. Also other researchers have coincided with this finding [Li06], [Tor-Mor04]. This fact increases the risks of damaging the whole software development process and continuously loose tacit knowledge when more experienced people are replaced. This justifies further research on this topic.

5.2.2 Determining IQ in the COTS Selection Context

IQ researchers agree on the meaning of *high-quality information as information which is fit for use by consumers* [Lee+01].

We considered crucial to collect empirical information from COTS selectors (both COTS researchers and practitioners) about their IQ needs for performing an informed selection and to assess what they generally mean by IQ.

The analysis of the outcomes was based on the framework presented in [Wan-Str96] and refined in [Lee+01]. It helped to determine a basis for assessing IQ from the information consumers' perspective, and suggests to group IQ needs into 4 high-level IQ dimensions which are described by a set of quality assets that represent a single aspect or construct of IQ, as shown in Table 5.2.

Intrinsic IQ implies that information has quality in its own right. *Contextual IQ* highlights the requirement that IQ must be considered within the context of the task at hand; it must be relevant, timely, complete, and appropriate in terms of amount, so as to add value. *Representational* and *Accessibility IQ* emphasize the importance of computer systems that store and provide access to information; that is, the system must present information in such a way that it is interpretable, easy to understand, easy to manipulate, and is represented concisely and consistently; also, the system must be accessible but secure.

Table 5.2. Basic IQ dimensions to as suggested by [Wan-Str96]

IQ Dimension	Definition	Quality Assets
Intrinsic	Denotes that information has quality in their own right.	Believability, Accuracy, Objectivity and Reputation
Representational	Includes aspects related to the format and meaning of the data.	Concise Representation, Representational Consistency, Interpretability, and Easy of understanding
Accessibility	Emphasizes the importance of the role of systems for providing access to information in a secure setting.	Accessibility, and Access Security
Contextual	Highlights the requirement that IQ must be considered within the context of the task at hand.	Relevancy, Timeliness, Completeness, and Appropriate Amount of Data, Value-added.

Further evidence exist that these approaches provide comprehensive coverage of the multi-dimensional IQ construct in very diverse settings [Shan+03], [Lee+01], [Wang+03].

As a next step, we intended to fit the IQ needs elicited from COTS selectors within the IQ dimensions stated in Table 5.2. Such results are shown in Table 5.3.

Of course, different points of view on IQ were obtained from different interviewees, but such differences were related to their specific projects requirements thus, they were successfully represented by the high-level dimensions. To fit our findings into the stated dimensions, we carefully analyzed the elicited IQ needs of COTS selectors to match them into the enclosed assets of each proposed dimension; as a result, the assets were adapted, i.e. redefined, abstracted, deleted, and carefully reviewed until agreement was reached. In addition, although in most cases decision-making processes were based on vague factors, interviewees recognized some important facts that they take into account to assess IQ.

The most significant issues of this process are summarized in Table 5.3 and detailed below:

- Intrinsic IQ. All the quality assets proposed were included since they were evident to COTS selectors. Several facts with respect to this item were reported.
- Representational IQ. It included all the proposed assets. Although some misunderstanding existed with respect to *Concise Representation* and *Representational Consistency*, they were generally understood as highly structured information and homogeneity respectively.
- Accessibility IQ. COTS selectors understood accessibility assets well. The *Access Security* asset was deleted since it was considered that COTS selectors acted as users of the information and its security aspects were not relevant from their perspective. We also replaced the name of the *Accessibility* asset by *Availability* in order to avoid misunderstandings with the name of the IQ dimension it belongs to.
- Contextual IQ. The need of considering IQ within the context of their domain specific project was clearly recognized by COTS selectors. This was consistent with the information consumer perspective followed.
- IQ Project Issues. Remarkably, a fifth dimension, the project dimension named *IQ Project Issues*, was added. This last relevant change appeared because our findings showed that IQ in the COTS selection context is largely determined by the resources allocated to the software development project and related policies and procedures; therefore we needed to take this into account.

Please note that there is a high synergy among the elicited COTS selection IQ needs. Hence, many intuitive relationships become evident, for instance, some IQ needs are shared by different assets from different perspectives (e.g. the *Value-added* asset is closely related to the facts addressed by *Concise Representation* and *Representational Consistency* to denote the extent of the value added; *Reputation* enhances *Believability*; and *Accuracy* greatly depends on *Timeliness*, ...).

Table 5.3. Excerpt of COTS selection IQ needs & facts elicited from COTS selectors

IQ Dimension/Assets		COTS Selection IQ Need	Some IQ Facts Detected
Intrinsic			
1	Believability	Credible information	Due to the commercial nature of the market, one should be sure of the credibility of the information
2	Accuracy	Precise information	Due to the high volatility of the market, documents become obsolete very quickly
3	Objectivity	Impartial point of view	Commercial and sponsored resources tends to be highly partial
4	Reputation	Coming from good sources	Well-known authors/sources represent a high reliability and trustworthiness. Appropriate references, related resources, and resource-dependencies are good indicators to recognize reputation.
Representational			
1	Concise Representation	Highly Structured	COTS information tends to be unstructured, especially in the cases of quality of service and non-technical information.
2	Representational Consistency	Homogeneity	The lack of standards for documenting COTS results in many kinds of documentation. Such heterogeneity makes difficult to easily compare IQ.
3	Understandability	Easy to be understood	Information addressed to the general public is usually easy to understand, but the information addressed to experts requires a specific background to be understood.
4	Interpretability	Easy to be interpreted	Different kinds of representation exist (e.g., ER models, Natural Language). They should be easy to interpret by the skills and background of the involved people.
Accessibility			
1	Availability	Free availability preferred	Some informational resources or support are available only for a fee.
2	Easy of Operation	Easy way to find and retrieve the information.	COTS information tends to be very difficult to be located. The way of obtaining the data is variform (e.g. direct download, subscription based, etc.). The resources required to process it should be compatible with the resources allocated to the project.
Contextual			
1	Relevancy	Useful, and appropriate to the project needs	Easy for the project team to understood and process the information with the resources they have allocated)
2	Timeliness	Sufficiently current and up-to-date	The date of creation or update is the best indicator for that asset
3	Completeness	Covering all the informational project needs	Information should cover the scope of the project to assure a good understanding of the requirements.
4	Appropriate Amount of Data	Adequate volume of information to be analyzed by the resources available.	Depend on the skills of the people allocated to the project and the size of the source.
5	Value-Added	Add value to the project operations	Heterogeneous and Unstructured information is difficult to be processed. When its structure makes easier the work of the project team in any of their tasks, it is considered a value-added.
IQ Project Issues			Describes the main IQ needs of the COTS selection project
1	IQ Project needs	Describes the IQ needs of the COTS selection project	The high-level project goals drive the IQ COTS Selection processes, they states the criticality of the project and its IQ needs.
2	Allocated Resources	Aspects related to the set of resources allocated to the project for performing the IQ process	The resources allocated to the project play a crucial role in determining product suitability.

5.2.3 Determining a Measurable Framework for Assessing IQ in the COTS Selection Context

Our next goal was to develop a comprehensive framework that stated our IQ findings; and how they could be feasibly gauged. An important aspect to be taken into account for determining this measurable framework was the relevance of domain knowledge. Data can be useless for one purpose but adequate for others, and domain knowledge is necessary to distinguish these situations.

To manage and gauge all these different views and needs on quality, quality models seem the most appropriate type of artefact since they provide a measurable framework which precisely defines and consolidates the different views of quality. Specifically, we propose an ISO/IEC 9126 tree-like structure [ISO-9126] because:

- ISO/IEC 9126 quality standard is one of the most, if not the most, widespread quality standard available in the software engineering community, as introduced in Chapter 2. Therefore, most COTS selectors are familiar with it.
- It is compatible with the *domain model* from GOTHIC, outlining a uniform framework well-suited for the integrated evaluation of all COTS selection related issues.
- It allows considering IQ aspects as requirements from the beginning of the COTS selection process in the same way that we have technical and non-technical requirements.
- It allows optimal reusability of product quality features throughout different COTS selection processes.

The ISO-IEC 9126-1 tree-like structure is based on a hierarchical model that offers quality characteristics to represent the most important quality aspects. These characteristics are further refined into multiple levels of subcharacteristics, which in turn are decomposed into attributes, yielding to a multilevel hierarchy. Intermediate hierarchies of attributes may appear making thus the model highly structured. At the bottom of the hierarchy there are the measurable attributes, whose values are computed by using some metric.

In order to elaborate the quality model from the IQ dimensions presented in Table 5.3, we adopted one of the most widespread approaches, the Goal-Question-Metric (GQM) approach [Bas+94], for analyzing each asset belonging to the stated dimensions. Any assumption about the assets other than they are built on top of the validated framework presented in Table 5.3 was done.

In GQM, goals of the product under measurement are identified, and then some questions are raised to characterize the way the assessment of a specific goal is going to be performed. Last, a set of metrics is associated with every question in order to answer it.

The final result of the GQM approach is a hierarchical structure in graph-like form, since metrics may influence in more than one question, and questions may be related to more than one goal. Goals are composed of four elements: purpose, issue, object and viewpoint. In our framework, these elements take the following form:

- ▶ **Purpose:** Presence of a particular feature or characteristic in the quality model.
- ▶ **Issue:** The GQM recommends identifying quality goals; then, we define one issue for each asset stated in Table 5.2. As a consequence, we have as many goals as assets.
- ▶ **Object:** Always the dimensions to which the asset belongs to.
- ▶ **View Point:** Always IQ needs for COTS selectors.

An excerpt of this process is shown in Table 5.4. To define the metrics, we have used the general theory of software measurement presented in [Fen-Pfl97] as conceptual basis to define the metrics for IQ aspects. Metrics can be objective or subjective and can be as simple as Integer or Boolean values or more complex as Lists, Sets or Functions. An example of a predefined function is the *mean* function used in Table 5.4, which has the meaning of returning the mean of all values of the function it encloses (e.g. *AuthorsBel*, *AuthorBelMarks*, *ProvBelMarks*). Moreover, we have considered very important that external marks can be computed to determine some quality attributes; for instance, to determine the believability of an author by the marks that other people have stated about him/her, and also to determine the believability of these marks by the believability of the markers.

Table 5.4 Excerpt of the GQM approach used to guide the information storage and metrics definition

Purpose: <i>Have an appropriate</i> Issue: <i>Believability</i> Object: <i>Intrinsic IQ</i> View Point: <i>COTS Selectors</i>			
Question	Metric	Value	Kind of metric
Who is/are the author(s) of the product?	Author(s) Name	AName= Set (String)	Objective, it is part of the own product properties
What is the believability of the Author(s)?	For each author, his/her believability	AuthorsBel= Function(String→TScore) TScore: {Very High, High, Low, Very Low}	Subjective, external property of the product
What is the overall believability of all authors?	Average of all authors believability	OveAuthorsBel= Mean(AuthorsBel)	Subjective, external property of the product
How is the author believability obtained?	Average of all marks that he/she has received	IndAuthorBel= Mean(AuthorBelMarks)	Subjective, external property of the product
What are the marks that the author has received?	Marks available for the author (Marker Available)	AuthorBelMarks= Function(String → TScore)	Subjective, external property of the product
What is the believability of those marks	For each Marker, his/her believability	BelMarks= Function(String → TScore)	Subjective, external property of the product
What is the provider Organization?	Organization Name	OrgName=String	Objective, it is part of the own product properties
What is the believability of the Organization?	Organization Believability	OrgBel= Mean(ProvBelMarks)	Subjective, external property of the product
What are the marks that the provider has received?	Marks available for the author (Marker Available)	ProvBelMarks= Function(String → TScore)	Subjective, external property of the product
What is the believability of those marks	For each marker, his/her believability	BelMarksOrg= Function(String → TScore)	Subjective, external property of the product
...			

GQM analysis led us to observe that different kinds of IQ metrics were obtained. Some of them were objective properties inherent to the product whilst others were

subjective or objective but external. In addition some metrics have applicability preconditions (e.g., Marks available for the author, which precondition is the availability of some Marker). The use of external metrics was greatly influenced by the intended reusability we aimed for the quality model.

Subsequently, following some principles stated in [Car-Fra06] for building a good ISO-IEC 9126-1 tree-like quality model, and our results of applying the GQM approach to each asset, we obtained a highly reusable quality model that can be adapted, improved and modified as required.

Table 5.5 presents an excerpt of such obtained model which has been structured according to the following guidelines:

Table 5.5 Excerpt of the ISO/IEC 9126-1 framework for stating COTS IQ¹

Characteristic/Subcharacteristics/Attributes				Metric	Description
1 Intrinsic Characteristic					Own properties of the information source denoting its quality characteristics.
1 Believability					
1 Author-Based Believability					Aspects that describe the believability of the product based on its authors.
	1	Author(s) Name		AName = Set (String) AName ≠ ∅ The names are directly obtained	Describes the name of the author(s) of the product.
	2	Author(s) Believability		Derived Attribute OveAuthorsBel= Mean(AuthorsBel)	Describes the overall authors' believability by the average of the believability of all authors.
		1	Individual Author Believability	AuthorsBel= Function(String→ TScore) TScore: {Very High, High, Low, Very Low} Dom(AuthorsBel) = AName ∀ x ∈ AName: AuthorsBel (x) = Mean (AuthorBelMarks)	Describes the individual believability of the authors by the average of all marks that he/she has received
		1	Opinion Marks about the Author	AuthorBelMarks = Function(String→ TScore) TScore: {Very High, High, Low, Very Low} These marks are directly obtained	Describes the marks that markers have done about the author
2 Provider Based Believability					Aspects that describe the believability of the product based on the organization that provides it
	1	Provider Name		OrgName=String OrgName ≠ ∅	Describes the name of the product provider
	2	Organization Type		OrgType= Function(String→ TOrg) TOrg:{ Academy, Standards, Commercial}	Describes the type of the organization provider
	3	Organization Believability		OrgBel= Function(String → TScore) Dom (OrgBel) = OrgName ∀ p ∈ OrgName: OrgBel (s) = Mean (ProvBelMarks)	Describes the believability of the organization provider
	1	...			

- **Characteristics.** The five highest-level quality factors correspond to the 5 IQ dimensions obtained in the previous section: *Intrinsic IQ*, *Representational IQ*,

¹ A more detailed version of this model is presented in Annex 2. It is being iteratively refined and improved as more empirical data is gathered.

Accessibility IQ, *Contextual IQ* and *IQ Project Issues*. Due to their nature, the first three characteristics group IQ features to describe the product and can be reused in all COTS selection projects, whilst *Contextual IQ* is envisaged to record the extent to which the product features meets the *IQ Project Issues*. As a logical consequence of this fact, please note that *Contextual* characteristics have been not further refined into specific attributes in the model. The reason behind is that its intended attributes tends to be derived from the others attributes belonging to the others characteristics. This reasoning provides a flexible model that can be instantiated to the specific needs of any COTS selection project. Such structure allows an optimal degree of reusability not only of the product but also the knowledge gained in every use of it.

- **First-level subcharacteristics.** The first 5 characteristics have been decomposed into 17 subcharacteristics that correspond to the assets leveraged in Table 5.3.
- **Intermediate subcharacteristics.** More than 30 intermediate subcharacteristics were used since most of the first-level subcharacteristics stated above were still too abstract to be measurable, and more COTS IQ needs were still remaining to completely express the requirements of our interviewees. Thus, whenever it was required an additional level of subcharacteristics for structuring or levelling purposes was added. It was primarily based on the GQM application results. Subcharacteristics are used mainly for classification means. This is the case of the *Intrinsic IQ/Believability* subcharacteristic which has been decomposed into other subcharacteristics: *Author Based Believability*, *Provider Based Believability* and *Marker Based Believability*.
- **Attributes.** Subcharacteristics were further decomposed into over 50 IQ attributes. To decide which attributes were the most suitable for evaluation and reusability purposes, we choose the most representative ones obtained from the application of GQM to the five COTS IQ dimensions' assets.

In general, they are two kinds of attributes: basic attributes which stand for objectively measurable features (e.g., *Author Name* attribute categorized under the *Intrinsic IQ/Believability* subcharacteristic); and derived attributes which require to be additionally decomposed into other attributes (e.g., *Author(s) Believability* which has been decomposed into *Individual Author Believability* and *Opinion Marks about the Author*).

In order to measure the attributes of our quality model, metrics were required, so we attained the metrics previously obtained by the GQM application. For derived attributes, sometimes it is not possible to find an objective metric to derive its value in terms of the attributes in which it is decomposed. In these cases, subjective metrics are required. It is evident from the model (See Table 5.5) that some IQ attributes may influence several other attributes or subcharacteristics, and thus hierarchic overlapping is also supported in the approach by considering the hierarchy as a graph.

It is important to stand out that we try to preserve homogeneity among the metrics of the attributes, mainly because as we mentioned above, it is common that quality features are closely related; and such homogeneity is the basis for a successfully *Contextual IQ* attributes estimation, since it is stated as the combination of the other IQ characteristics. This is also the reason why no metrics are provided for *Contextual* characteristics (since they are derived from others). In addition, although all quality

features are involved in determining the overall IQ for a COTS selection project, elaborated types of relationships among quality features and also intensities of these relationships exist, and may be depicted by means of tabular representations as done in [Chu+00].

The characteristics, subcharacteristics, and attributes used on the proposed model (which current version is presented in Annex 2) have resulted representative of most of the empirically elicited COTS needs to date. Of course, several other subcharacteristics and attributes may be added or deleted as required in order to tune the model to the specific contexts of use.

We are aware that more empirical data is required to completed and further validate the model with the assets that are really used in the COTS selection practice (research in progress is seeking these issues). Current case studies performed to date using this framework have lead promising results. As a result, information sources (i.e. products) are ranked based on the extent to which they fit to the context and COTS selection project characteristics (e.g., criticality of the domain, expected frequency of taxonomy use in future selection processes; resources allocated to the project, especially deadline, money and person/months; current and future knowledge of the domain and technical skills of the conformed team). Such prioritization is reached by computing a consensual result from reconciling the Intrinsic, Representational, and Accessibility attributes with *Contextual* and *IQ COTS Selection Project* attributes. The homogeneity of the metrics used allows these comparisons.

Table 5.6 is an excerpt of some of the information sources attributes considered for performing the prioritization of sources for the RTSC case.

Table 5.6 Excerpt of the Information Sources prioritization in the RTSC case

PName	Type of Source	Organization Provider	Author	Location	Cost	...
Session Initiation Protocol	Standard	Engineering Task Force	Free	...
H.323	Standard	International Telecommunication Union			±80€	
IMTC	Independent Report	International Teleconferencing Consortium			Free	
RTC-Gartner-1	Hierarchy	Gartner			Free	
...						

5.3 A Systematic Approach for Managing and Reusing COTS Information Sources

In the previous sections we have achieved our primary goal, namely to understand what quality means for COTS selectors, and how it can be feasibly recorded, reused, gauged and integrated into our GOTHIC approach. Moreover, we enriched our approach by organizing our further findings as a set of heuristics for supporting the collection, arrangement and decision-making processes (see Section 5.3.1).

Although the obtained IQ attributes resulted satisfactory enough to describe IQ requirements in several COTS industrial and academic selection project case studies,

we realized that supporting decision-making in the face of increasing information volumes and COTS information characteristics, demands a systematic management of IQ to inform COTS selectors about the quality and adequacy of the information to their tasks without having to do a full inspection or regenerating the data anew. Besides, the reuse of the metadata about the information sources and their assessment in diverse selection processes of the same domain, would improve the efficiency and effectiveness of future selection processes.

As a result, we have partially implemented our previous work as a conceptual model for systematically supporting COTS selectors not only to collect, to storage and to assess IQ, but also to (re)use and manage it for improving their decision-making process. The current conceptual model is detailed in Section 5.3.2.

5.3.1 Heuristics to Support IQ Assessment in the COTS Selection Context

Some examples of heuristics driven the information arrangement and decision making processes are:

- “Diverse types of information sources exists, they can be grouped into: Hierarchy, Standard, Vendor Information, Independent Reports (of scientific, divulgation and/or technical nature), Oral Information, Test Of Tools Reports, Experiences, Other”,... Table 5.1 provides descriptions and examples.
- “Information sources available can provide insights into a diverse range of software packages and/or vendor characteristics, but no requirements identified from these sources should be used without careful consideration of their confidence”.
- “Information from experts is good at quickly identifying general principles, offering explanations, validating analyses, and providing pointers that could be cross-validating with Independent Reports”.
- “Information from COTS providers tends to highlight the strengths and hide the weaknesses of licensing packages”.
- “Information from standards related to the field, are the best for identifying COTS high-level goals”.
- “*Test of tools and Systems* are useful for complementing the information from vendors regarded to detailed information on typical interfaces, architectures, dependencies between products for enabling or complementing their functionality”.

Annex 1 further describes the heuristics driving this activity.

5.3.2 A Conceptual Model for Systematically Supporting COTS Selectors Decision-Making

Results obtained in our preliminary study, have been enclosed into a conceptual model. This model is being refined with as more empirical data is gathered. A comprehensive sketch of the current model is shown in Fig 5.1. A more detailed view of the model is reported in [Mes07]. The ultimate goal of this intended model is to incrementally build a catalogue of COTS related information sources, and describe them by means of the quality features defined in the quality model explained in the previous section. This

leads to inform COTS selectors about the quality of the sources to decide if they are adequate to their objectives.

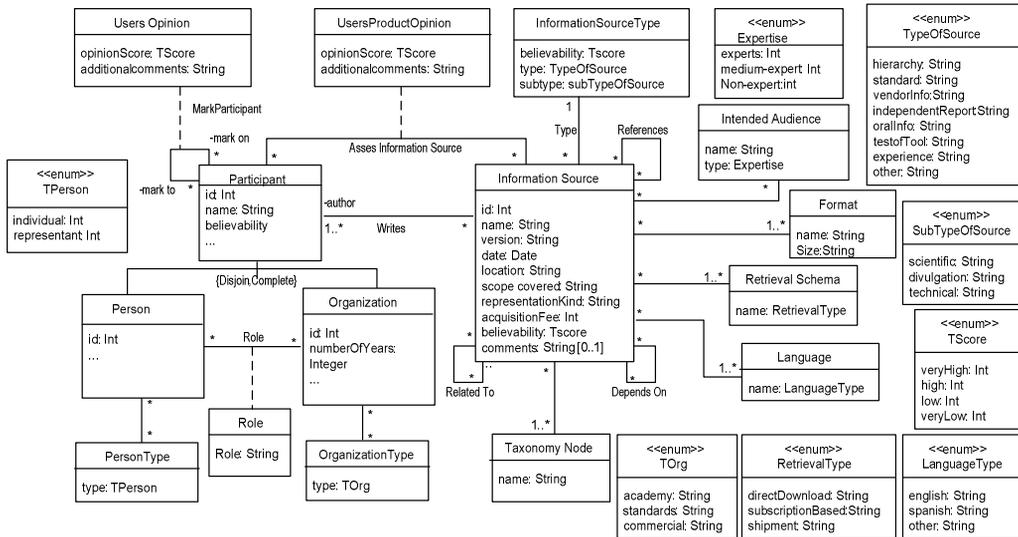


Fig. 5.1 An excerpt of the COTS IQ Reference model

At the heart of this model lies the *Information Source* class described by a set of class attributes or class relationships (e.g., the *believability* class attribute, or the *Retrieval Schema* class). All of them correspond to the IQ attributes identified in the quality model.

Some attributes correspond directly to metrics previously identified; for instance, the *Representation Kind* attribute categorized under the *Understandability* subcharacteristic, is directly stated by the *RepresentationKind* attribute of the *Information Source* class. On the other hand, information sources are characterized by diverse *InformationSourceType* (as detailed in Table 5.1) and their extraction is supported by the application of the heuristics mentioned in Section 5.3.1.

From the reuse and management perspective, we define the *Participant* class to describe the subjects that provides a mark, create information sources, or are members of a producer organization. This class is defined by a set of objective attributes (e.g., name), and a subjective attribute named *believability*. This class can refer to a *Person* or an *Organization*. A person can play several roles in an *Organization*. On the other hand, Participants play the *Author* or *Marker* role. Author refers to the *Information Source* creator. Marker refers to who gives an opinion and/or a mark about the believability of the information source based on his/her own assessment of the source. Such opinion is denoted by the *UserProductOpinion* association class. Hence, an *Information Source* can collect more than one *UserProductOpinion*. In the same way, *Participants* can provide marks and comments about the believability of other participants.

Many other quality relationships extracted from the quality model exist; some of them are easily inferred by the model. For instance Accessibility attributes of the *Information source* as *Format*, *Retrieval*, *Schema*, *Acquisition Cost*; Representational ones as *Language* or *Representation Kind*; and some kind of special relationships

among the sources as *Related To*, *Depends On*, and *Reference* that are Intrinsic attributes that denote, the products related, dependent or referenced by the *Information Source* and have quality implications.

Of course, some integrity restrictions are defined in the model, e.g., Authors cannot issue a mark about their own *Information Sources*, or a *Participant* cannot make a mark about himself/herself.

Additionally, we have integrated this approach into our GOTHIC method. The way to do that is to consider that the *Information Source* and *Taxonomy Node* classes introduced in Fig. 4.2 in Chapter 4 are in fact the same as the stated in Fig.5.1. Hence, as mentioned in Chapter 4, a GOTHIC taxonomy is used to locate the taxonomy node that fulfills the needs of the user in charge of the COTS selection process. Once located, the information sources related to each node can be assessed to obtain high-quality information to adapt the domain model to the specific requirements of the selection project, applying the rules we defined in [Aya-Fra05] and detailed in Chapter 8.

Thus, the approach described here can be used to guide data collection, storage and use, allowing the comparison of various information sources in terms of their quality value for a specific project.

Given the huge amount of information sources available, we considered crucial the implementation of this strategy in a software tool for supporting this activity.

So far we have implemented the conceptual model in a software tool [Mes07]. Meanwhile we have recorded over 150 information sources we have (re)used in several COTS domain-related analysis projects.

Details of the tool that has been developed to support this activity are found in Chapter 9.

5.4 Summary and Discussion

This research develops a framework that captures the aspects of IQ that are important to COTS selectors. It was integrated into the GOTHIC method activities framework for establishing the properties of the information to be used in the *Domain Analysis* activity (see Chapter 6) in order to make it more successful. This framework provides a systematic approach for supporting COTS selectors to decide information sources to use according to their specific quality project needs.

Relevant aspects of our research are:

- The resulting strategy driving the exploration and analysis of information sources was based on relevant approaches from the IQ research, e.g., [Boo-Smi00], [Sha+03], [Wan-Str96], [Lee+01], [Wang+03], several industrial experiences and case studies analyzing COTS selection processes [Aya06], as well as interviews run in software companies [Ger06] and [Ger07].
- Quality dimensions required to perform an informed COTS selection were identified from experts, researchers, and practitioners as well as theoretically.

- The set of attributes that allows gauging, reusing, prioritizing, and managing IQ to support COTS decision-making were defined. These attributes were organized into an ISO/IEC 9126 tree-like structure which outlines a uniform framework for the integrated evaluation of all COTS selection related issues in the GOTHIC method. Hence, IQ requirements can be considered from the beginning of the COTS selection process in the same way we have other kind of requirements (e.g., functional, non-technical, ...). Moreover, we put emphasis on reuse, which allows transferring knowledge from one information quality assessing experience to another. See Annex 2 for a current representation of the model obtained.
- Heuristics are provided to locate and choose information sources suitable to the quality objectives and/or resources allocated to any COTS selection project.
- A conceptual model and a tool support called IQ-Tool (introduced in Chapter 9) to systematically tackle the collection, storage, retrieval, and analysis of information sources in COTS selection processes has been defined [Mes07]. It is being improved by empirical information which is allowing its refinement [Aya-Fra08].
- Finally, it is important to remark that although reaching the proposed IQ assessment process and their related artifacts may seem time-consuming, they are in fact incrementally constructed, improved and (re)used by the iterative application of the method to several selection processes.

Our ongoing and future works with respect to this activity include:

- ▶ The integration of the IQ-Tool into the DesCOTS system [Gra+04] (see Chapter 9).
- ▶ To extend the approach with more empirical data by applying an online questionnaire to a broader population [Aya-Fra08].
- ▶ To develop functions to systematically compute a consensual result from the match among the *IQ Project Issues'* attributes to the *Intrinsic*, *Representational*, and *Accesibility IQ's* attributes. It means to systematically generate *Contextual IQ* attributes results. To do this, we are basing our efforts on the criteria for information quality reasoners defined for Wang et al. [Wang+03]. Our main intention is to provide support and flexibility in dealing with the subjective, decision-analytic nature of IQ judgements.

Chapter

6

Activity 2:

COTS Domain Analysis

The goal of the COTS domain analysis activity of GOTHIC is identifying and recording the most important aspects of a particular COTS domain in the COTS marketplace for reusability and management purposes. It is done from the information sources previously obtained in the *Information Sources Exploration* activity described in the previous chapter.

COTS domain analysis activity runs in parallel to several other GOTHIC activities by guiding the production of artifacts or models throughout all activities of the method.

The strategy to perform *Domain Analysis* includes producing several models, written according to some widespread notations and standards. These models are then integrated and synchronized using the ISO/IEC 9126-1 quality standard as a unifying framework called “*domain model*”. Such *domain model* is the basis of the GOTHIC method to gain knowledge for identifying the correct goals and to build a reuse infrastructure with several kinds of reusable assets of interest for COTS selection processes. This strategy was also published in [Aya-Fra06c].

This Chapter is structured as follows. Section 6.1 provides a brief background of previous research and greatly justifies the need of the study of domain analysis in the COTS selection context and its feasibility. The informational dimensions for evaluating COTS are identified in Section 6.2. Section 6.3 discusses the most appropriate types of models to record these informational dimensions whilst Section 6.4 explains how these models are integrated into a unified one. Section 6.5 outlines the impact of domain analysis on COTS selection and Section 6.6 illustrate the approach by an example

which obtained domain model is further described in Annex 3. Finally, summary and conclusions of the most relevant aspects of the chapter are presented.

6.1 Background

Systematic reuse is based on the observation that quality and productivity can be significantly increased by shifting the focus of software engineering to a domain-centered view by means of building an infrastructure support.

The engineering discipline concerned with building these optimal reusable assets is called *domain engineering* [Pri-Ara91]. *Domain engineering* supports the notion of domain, a set of applications that use common concepts for describing requirements, problems, capabilities and solutions.

Particularly, being part of domain engineering, *domain analysis* has been identified as a major factor in the success of software reusability. *Domain analysis* refers to the process of acquiring and consolidating information about an application domain, so that reusable infrastructure can be designed reliably [Frak+98]. Its purpose is to identify the basic elements of the domain, to organize an understanding of the relationships among these elements, and to represent this understanding in a useful way by means of different types of models [SDA].

The different existing views on domain modelling (e.g., [Corn96], [Nei80], [Pri-Ara91]) share then the same goal: to facilitate quality software development by reusing the knowledge of the addressed domain.

Reuse is not a context-independent activity. The type of artifact to be reused impacts on the reuse models to be adopted and the reuse processes to be undertaken; therefore, it is a fact that the reuse discipline has to evolve as new paradigms and artifacts emerge. This is the case of the CBSD paradigm [McC89].

As mentioned in Chapter 2, although several COTS selection methodologies, processes and techniques have been formulated, they are mainly oriented to individual selection processes and do not explicitly address reusability issues. Even in the cases in which a reuse infrastructure is suggested (e.g., OTSO, CARE, PECA), no real support or precise guidelines are offered. This lack of explicit proposals for dealing with COTS selection has been also recognized from the reuse discipline [Mor06].

To solve this problem, it seemed feasible to use domain analysis for recording and structuring information about COTS in order to enable their efficient reuse. However, as far as we knew, COTS technology issues had not been explicitly addressed in the domain analysis discipline (although of course many concepts of domain analysis apply to this particular case).

Therefore, we developed a particular strategy of domain analysis for supporting COTS selection.

In this strategy we produce several domain models covering different dimensions that capture and represent the most important aspects of a particular COTS segment in the COTS marketplace. All the models are integrated using a unifying framework. We use widespread notations and standards to represent the dimension models. This

strategy was integrated as a fundamental basis for building the COTS segment reuse infrastructure aimed by the GOTHIC method. It was published in [Aya-Fra06c].

Being part of GOTHIC, the domain analysis activity has the mission of producing a *domain model* (i.e., a representation of the most important aspects of a COTS segment) that serves as the basis to gain knowledge for identifying the correct goals and to build the reuse infrastructure with several kinds of reusable assets for COTS selection processes.

6.2 Domain Analysis for Supporting COTS Selection: Dimensions

In the previous section we have justified the convenience of having domain models for describing COTS marketplace segments. In this section we identify several dimensions of interest for describing the COTS information required during COTS selection processes. Each dimension will be described by a model.

The identification of the dimensions was done by analyzing the different informational needs and facts on COTS selection processes that have been reported in the literature (e.g., [Ber+03], [Req+05], [Li06]), our GESSI group experiences in the field (e.g., [Fra-Car03], [Car+04b], [Car+05], [Car06]), as well as some empirical studies in industry we had the opportunity to participate [Ger06], [Ger07].

To search, elicit and process all the information from which domain analysis will be performed, we made use of the framework introduced in Chapter 5. It provides an information quality model supported by heuristics that facilitate the collection, storage, retrieval, analysis and reuse of COTS information in a quality assurance environment. It captures the aspects of information quality that are important to COTS selection and provides a systematic approach for supporting COTS selectors to decide information sources to use according to their specific project needs.

Fundamental concepts

In the COTS context the same concept may be denoted by different names in different products or even worse, the same term may denote different concepts in different products. Currently, it is not usual to find places in the COTS marketplace where fundamental concepts are stated. Most normally, one may find items (products, services, etc., belonging to one or more market segment) whose description uses some terms in a rather obscure way, making those descriptions difficult to use (especially when comparisons among candidates are needed), customize them and make them evolve as the marketplace does [Aya-Fra05]. See Section 2.3.2 for examples of repositories containing COTS descriptions.

On the other hand, every single COTS segment defines lots of concepts that are used over and over, e.g., anti-virus tools have “viruses”, e-mail systems have “messages” and “folders”, etc. These concepts may be related in many ways, e.g. “messages” are “stored” inside “folders”. A poor knowledge of these fundamental concepts and their semantic relationships may interfere with the efficiency and effectiveness of COTS selection processes, especially taking into account some of the risky COTS technology characteristics introduced in Section 1.1 (e.g., COTS marketplace growing size and

rapid changes). Therefore a model for representing all this information is needed. Its purpose is to settle the scope of a particular segment, to define its main concepts (both as a vocabulary and as a semantic model) and the relationships that facilitate the understanding of the domain as a whole. The resulting model can be used as a reference framework for the segment. To build this model, information sources such as standards and textbooks are useful (see Chapter 5). We recommend to choose one of the most trustable sources as starting point, then to synthesize the corresponding dimensions of the domain model, and last to calibrate this dimension with other informational sources. The resulting model can there be used as a reference framework for the segment.

Functionality

COTS have their functionality already built-in. Hence, instead of traditional requirements that specify “must” and “should” needs, requirements for CBS articulate broad categories of needs and possible trade-offs.

Most of the existing COTS categorization proposals are based on COTS functionality attributes for searching COTS. Thus, COTS functionality is a primary source of information in COTS selection processes. Consequently, a model must cover this dimension. But a good balance is needed. On the one hand, the most representative functionalities of a particular segment should be included (e.g., virus repair, automatic resending of messages) and described up to a level of detail that enables efficient survey and evaluation of particular COTS. On the other hand, if too much detail is given, several obstacles mentioned in Section 1.1, remarkably growing size and rapid changes of the COTS marketplace are harder to overcome, since a lot of information would need to be updated continuously. Also, too much detail may commit the description of the functionality to the behaviour of particular components.

Quality of service

Since the quality factors are likely to break the tie when several COTS candidates provide the required functionality, the role of quality information becomes utterly important for driving COTS selection [Car+03]. In particular, quality requirements have been recognized as crucial by the methods and processes proposed so far for driving COTS selection (see Chapter 2). Thus, efforts are required to obtain reliable and comprehensible descriptions of COTS quality of service in an efficient way. We propose then a dimension for stating quality of service.

The resulting model needs to offer a structured description of the COTS segment addressed, organizing the different quality factors hierarchically (e.g., Throughput and Response Time as sub-factors of Time Efficiency) and should also include metrics for the quality factors. This model may serve as a framework in which particular COTS may be evaluated and compared to user requirements during selection processes.

Non-technical description

Despite the fact that the evaluation of candidate COTS from a technical point of view (functionality and quality of service) is necessary, experiences in COTS selection show

that non-technical information¹ must be taken into account and, in fact sometimes it is even more important than the technical information [Car-Fra06]. As a result, we need to record this information.

This new dimension must distinguish several concepts and focus on the commercial nature of COTS, stating information about licensing issues, provider reputation, post-sale supporting services, etc.

One should be aware that part of the information may be difficult to obtain (e.g., provider finance information) and the corresponding factor may not be included in the model for this reason.

Interoperability

The analysis of any COTS market segment shows that some relationships among components exist. We have analyzed the types of dependencies that may exist and we have concluded that a COTS may need another for:

- *Enabling its functionality* (e.g., document management tools need workflow technology to define life cycles).
- *Complementing its functionality with an additional feature, not originally intended to be part of its suitability* (e.g., a web page edition tool can complement a web browser to facilitate web page edition).
- *Enhancing its quality attributes* (e.g., resource utilization can be improved significantly using compression tools).

However, in the COTS selection arena, interoperability has been dealt within a case-by-case basis. Furthermore, most of the COTS selection methods proposed so far just address single component selection, they do not even address the need to select a suite as final solution. Therefore, we propose a new dimension to cover this need, otherwise COTS selection becomes not trustable. It is worth remarking that, since we are describing not a particular COTS but a whole segment, interoperability issues must not be stated in much detail (e.g., data formats, API specificities, etc.); instead the model should include the needs and expectations that one type of component has on others in a very high-level way.

6.3 Domain Analysis for Supporting COTS Selection: Models

Taking into account the informational dimensions required by the COTS technology, in this section we discuss which are the most appropriated types of models for representing them. A first observation is that, due to their diversity, various types of models will be probably required.

In the domain analysis field, a variety of methods and techniques have been proposed as: FODA, DARE, ODM, DSSA and PLUS (see [Fer-Veg99] and [Poh+05] for a

¹ Information that does not refer directly to the intrinsic quality of software, but to its context, including economic, political and managerial issues; e.g., adequacy of the procedures imposed by the COTS with respect to procedures of the organization.

survey) which use a diversity of different types of artefacts and mechanisms to record the knowledge that range from the traditional requirements models (namely models of data, behaviour, and function), as Data Flow diagrams [McM-Pal84], Entity-Relationship (ER) models [Che76], Object Oriented models [Coh-Nor98], UML models [UML] Scenarios [Poh+01], and Feature models [FOD], to UML metamodeling techniques and more elaborated UML extensions and stereotypes for dealing with domain structural elements, relations and variability [Poh+05], [Gom05].

In practice, these proposals vary in their terms, notations, and emphases, but in general they are focused on designing product lines or product families for promoting reusability between software applications by means of an intended reuse plan [Poh+05], [Gom05].

We have studied whether the models proposed by the actual domain analysis practices could be suitable for recording all the COTS informational dimensions. As far as we know, none of these approaches has examined in depth the special kind of relationships and information that the COTS technology requires. We found that although some commonly used models could fit well enough for representing some dimensions, some other dimensions were still lacking of an adequate representation and analysis addressing the COTS information reusability and management.

For instance those relationships that enable interoperability among components, which could be partially fulfilled by establishing “Artifact Dependencies” (a special kind of variability in variability models for Software Product Lines design [Poh+05]), as well as the dimension related with stating non-technical information and quality of service (this last could also be partially addressed by test cases, but generally they are considered to be out of domain analysis). Additionally, although in [Leu-Leu03] the usefulness of having domain models to reuse COTS related information is argued, they do not explain what should be the characteristics of the domain models, neither how they should be constructed. For that reason, it is a fact that actual domain analysis approaches do not address in an optimal way all the fundamental informational dimensions required for assessing COTS in terms of expressiveness and adequateness, structure, and compatibility.

Hence, existent domain analysis strategies have to be somehow adapted and complemented to fully deal with the COTS technology characteristics [Alm+06], [Vit+03a]. In the rest of this section, we propose a set of models for covering all the required COTS informational dimensions using widespread notations and standards.

Table 6.1 Domain analysis practices for representing COTS dimensions

COTS Dimension	Domain Analysis Practices	Our approach
Fundamental Concepts & Standardized Descriptions	ER Models, Feature Models, UML Diagrams, Glossaries, etc.	UML Class Diagrams + LEL
Functionality	Data Flow Diagrams, Scenarios, UML Diagrams, etc.	UML Use Case Diagrams + brief individual descriptions
Quality of Service	None	ISO/IEC 9126-1
Non-Technical Description	None	3 categories of non-technical factors
Interoperability	None	* SD Models

Table 6.1 summarizes our proposal and makes clear the gap for recording non-technical descriptions and interoperability with respect to other domain analysis approaches.

Fundamental Concepts

Two types of artifacts are adequate for representing fundamental concepts:

- a. Conceptual data models or feature-oriented models to express the semantic meaning of the terms in the market segment together with their relationships.
- b. A glossary to set up a vocabulary of the domain with information about synonymous and other lexical relationships.

In particular, we have chosen UML class diagrams for representing the semantic information due to its expressiveness and acceptance in the community.

For the glossary, the Language Extended Lexicon (LEL) [Lei-Fra93] approach provides an adequate level of service since it allows capturing the meaning and fundamental relationships of the particular symbols (words or phrases) of the domain. The glossary includes at least the terms that appear in the rest of the models (e.g., the names of classes, attributes, and associations of the UML class diagram). One could also think of the general concept of ontology [Grub93] for capturing all the information needed.

Functionality

Any approach based on the concept of scenario seems a good option. We remark that the important point is to use the right level of detail.

Specifically, we propose the use of UML use case diagrams for defining the functionalities of the COTS segment and a brief format of use cases [Coc01] for describing them individually.

Details of the use of such models are given in Activity 3 of the GOTHIC method, related in Chapter 7.

Quality of Service

Quality models [Fra-Car03] provide a measurable framework which precisely defines and consolidates the different views of quality (e.g. performance, reliability, integrity, etc.) required for COTS evaluation. Among the different existing proposals, we adopted the ISO/IEC 9126-1 standard [ISO9126] for several reasons discussed in Chapter 2, remarkably: it provides a two-level departing catalogue but at the same time it is highly customizable to each different COTS segment; there are some metrics already defined for this standard; and it is widespread.

Non-Technical Description

Not only in the domain analysis context but in general, it is not usual to find models for representing non-technical information. Usually, some categories are recognized and for each of them, a list of non-technical factors identified. In [Car-Fra06] 3 high-level

factors and 15 second-level sub-factors referring to COTS supplier information (e.g., organizational structure), business information (e.g., licensing schemes) and other non-technical information about the product (e.g., history) are identified and structured in the form of an ISO-IEC 9126 quality model as shown in Table 6.1.

Such catalogue has been further improved and elaborated in several low-level characteristics and their associated metrics in [Car+07a], [Car+07b], and is available on line at <http://www.lsi.upc.es/~gessi/QMTool/CQM/ExtNTISO.html>. We suggest to use at least the 3 high-level factors and 15 second-level sub-factors to capture this dimension.

Table 6.2 High-level characteristics and subcharacteristics describing COTS Non-technical factors

Characteristic/ Subcharacteristic		Description	
1	Supplier	Characteristics of the supplier that can influence the quality of the software product.	
	1	Organizational Structure	Description of the organizational structure of the supplier company.
	2	Positioning and Strength	Description of the position and orientation of the supplier company in the market.
	3	Reputation	Recognition of the capability of the supplier to perform similar projects based on past experiences and certifications.
	4	Services Offered	Description of the services offered by the supplier.
2	5	Support	Description of the support mechanisms offered by the supplier company.
	Business	Characteristics of the contract among the supplier and the client that can influence the quality of the software product.	
	1	Licensing Schema	Description of the COTS licensing options.
	2	Ownership	Description of the aspects in relation to the intellectual property rights.
	3	Guarantees	Detail of the guarantees provided over the product.
	4	Licensing Costs	Description of the costs components and total cost of ownership for the different licensing options available
	5	Platform Costs	Estimation of the cost for the required production platform
6	Implementation Costs	Estimation of implementation costs based on similar past experiences.	
	7	Network Costs	Estimation of additional costs for network operation.
3	Product	Characteristics of the commercial aspects of the software product that can influence its quality.	
	1	History	Evolution of the COTS since it has been offered to the clients.
	2	Deliverables	Detail of the out-of-the-box and expected post-implementation deliverables
	3	Parameterization/ Customization	Description of the initial effort required for the product to operate.

Interoperability

Interoperability of COTS is usually described by means of APIs or data formats. However, as already explained in Section 6.2, we are interested in describing not particular COTS but the general behaviour of all the components belonging to a COTS market segment, therefore we need more abstract descriptions. The combination of goal- and agent-oriented models provided a good response to our needs. Goals allow expressing needs and expectations in a high-level way, whilst agents are an appropriate

way to model COTS segments. Then, one COTS segment may state that depends on another to attain a goal. We have chosen *i** Strategic Dependency (SD) models [Yu95], adapting its semantic to represent COTS segments and their dependencies. Details of the construction of such models are given in Activity 4 of the GOTHIC method, related in Chapter 7.

6.4 A Unified Model for COTS Domains

The models proposed in Section 6.3 cover the informational dimensions that were identified in Section 6.2. However, the primary goal of COTS segments domain analysis is to characterize COTS for their evaluation and selection, so it is clear that having these dimensions structured in separate models hampers domain understanding and model management. For this reason, we realized the need of a unifying model which facilitates this goal.

From the dimension models given, quality models seemed the most appropriate type of artefact. Therefore, if we succeed in putting all the models in an ISO/IEC 9126-1 quality model we will have our goal attained.

6.4.1 Integrating all the COTS domain models into the ISO/IEC 9126-1

In this subsection we aim at integrating the models obtained so far, even considering their different nature, into an ISO/IEC 9126-1 quality model. Fig. 6.1 shows an overview of our proposed framework.

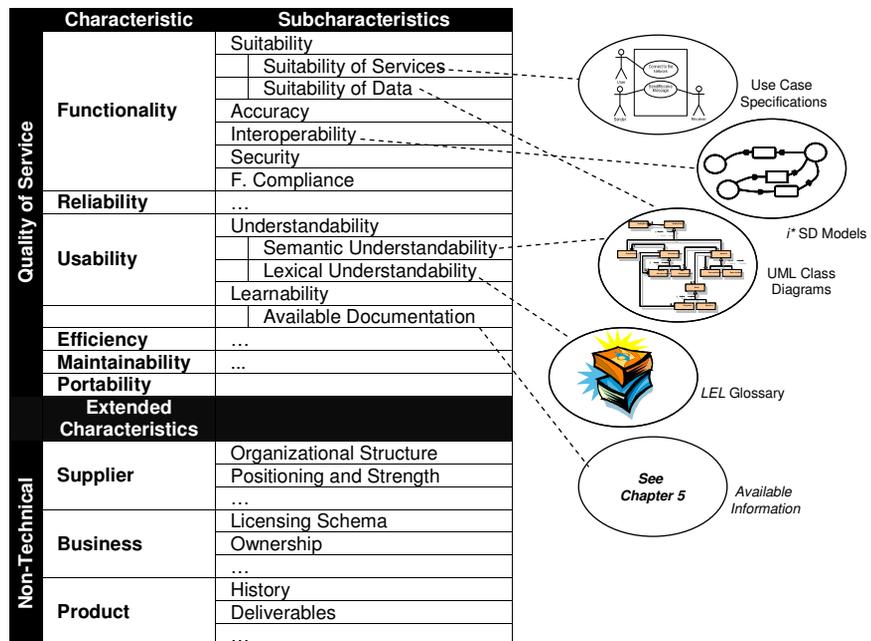


Fig. 6.1 Overview of the ISO/IEC 9126-1-based quality model for COTS

Functionality

Regardless of having the same name, the functionality of a COTS segment does not correspond with the ISO/IEC 9126-1 *Functionality* characteristic. Instead, it corresponds to the *Suitability* concept that is a subcharacteristic of *Functionality*. However, since functionality focuses on the services provided but not the data managed, we create a new subcharacteristic:

- *Suitability of Services*, belonging to *Suitability* (a subcharacteristic of *Functionality*) that contains the UML Use Case diagram and the individual use case descriptions.

Fundamental concepts

The UML class diagram is related to two ISO/IEC 9126-1 subcharacteristics. On the one hand, as the case before, *Suitability*, because some of the classes (and their attributes) and relationships are defining part of the suitability of the COTS segment. On the other hand, *Understandability*, which is a subcharacteristic of *Usability*, because having a UML class diagram provides a reference framework that allows testing how much a particular COTS adheres to it. For the same reason, also the LEL glossary supports *Understandability*. Therefore, we create 3 new subcharacteristics:

- *Suitability of Data*, belonging to *Suitability*, contains the class diagram;
- *Semantic Understandability* belonging to *Understandability* (a subcharacteristic of *Usability*) also contains the class diagram;
- *Lexical Understandability*, belonging to *Understandability*, contains the glossary.

Non-technical description

Since non-technical factors proposed in [Car+07a] are actually in an ISO-9126-1-form (see Table 6.2), in the intended integrated model, we only define the 3 high-level ones as characteristics and the other 15 as subcharacteristics.

Interoperability

Interoperability is also a subcharacteristic of *Suitability* and in this case, we just consider the *i** SD model as the description of *Interoperability*.

Other Considerations

The existence and quality of the documentation available (i.e., information), from which all domain analysis is based on, is crucial to assess several quality aspects of the product affecting its learnability and several other issues. This is mainly related with the concept of the *Learnability* subcharacteristic belonging to *Usability* in the ISO/IEC 9126-1. Thus, we add a subcharacteristic belonging to *Learnability* named *Available Documentation* which contains some information from the previously gathered sources summarized in the Information Quality Model introduced in Chapter 5 and further presented in Annex 2.

We create 2 new subcharacteristics belonging to *Available Documentation*:

- *Provider Documentation*, contains information that is directly provided by the supplier of the component
- *External Documentation* refers to all related documentation that is not provided by the supplier of the component.

6.4.2 Transforming the Models into the ISO/IEC 9126-1 Framework

Although we have achieved our primary goal, namely integrating all the dimension models under the same umbrella, there is still a question left that may be considered as a drawback when using the domain model for COTS evaluation purposes: the fundamental concepts, functionality, and interoperability models are expressed with their own formalisms which are not straightforward to evaluate.

In this subsection we deal with this problem by providing rules that map the constructs in these models into ISO/IEC 9126-1 quality factors. Furthermore, we state how their metrics are defined. These rules are defined in such a way that they could generate the new, final model automatically from the former models. This integrated model is called *Domain Model* in our GOTHIC approach.

Next paragraphs describe the corresponding rules to map the content of the diverse dimension models into the ISO/IEC 9126 framework. Once these rules are applied, evaluation of COTS may be done in a more uniform and comfortable way. But of course, the original models should be preserved since they are easier to understand and evolve.

Functionality

For each use case UC appearing in the Use Case diagram, a quality attribute UC belonging to the *Suitability of Services* subcharacteristic is created. The individual use case specifications are part of the description of these quality attributes.

For each obtained quality attribute, an ordinal metric which can take three values, Satisfactory, Acceptable and Poor, is created. These values express how a particular COTS covers the service represented by the use case.

Fundamental concepts

For each class or association C appearing in the class diagram that represents a concept provided by the COTS in the segment, a quality attribute C belonging to the *Suitability of Data* subcharacteristic is created. The elements of the class diagram are part of the description of these quality attributes.

For each obtained quality attribute, an ordinal metric which can take three values, Satisfactory, Acceptable and Poor, is created. These values express how a particular COTS provides the data represented by the class or association. These values will be obtained during evaluation by using different criteria (e.g., whether all the attributes are provided, whether the instances are permanent or not, etc.).

Each term of the glossary is included as part of the description of the quality attribute(s) it is related to. The same happens with the elements of the class diagram that were not tackled in the previous step.

Last, two numerical metrics are bound to the *Semantic Understandability* and *Lexical Understandability* attributes. The values of these metrics will count the number of semantic and lexical discrepancies of a particular COTS with respect to the reference models.

Non-technical description

No rules are required in this case since non-technical characteristics are already described in the ISO/IEC 9126 format.

Interoperability

For each agent A appearing the i^* SD model, except the agent S that represents the COTS segment we are modeling, a subcharacteristic A belonging to *Interoperability* is created.

For each dependency G among S and A, an attribute G is created. For each obtained quality attribute, we create an ordinal metric whose values depend on the type of the corresponding dependency: if goal, values are Attained and Not Attained; if resource, Provided and Not Provided; if task, Executed or Failed; if softgoal, Satisfactory, Acceptable and Poor.

Other Considerations

For each available documentation provided by the COTS supplier in the Information Quality Model (introduced in Chapter 5), four minimum attributes belonging to *Available Documentation/Provider Documentation* are created to describe the extent of the COTS related information available: *Documentation and User Manuals*; *FAQ and Tips*; *Help Files*; *Online Help Documentation*. The same happen with the *Available Documentation/External Documentation* subcharacteristic, that is also described by *Documentation and User Manuals*; *FAQ and Tips*; *Help Files*; *Online Help Documentation* attributes.

For each one of these attributes, we create a nominal metric which can take 4 values: NotProvided; Basic; Medium; Advanced. These values express the extent of the information provided by the supplier and non-suppliers of the COTS respectively.

In addition, with respect to the other ISO/IEC 9126-1 characteristics and subcharacteristics, we use the approach of Carvallo et al [Car05T], [Car-Fra06], [Car+07a], [Car+07b] for completing the ISO/IEC 9126-1 hierarchy.

6.5 Domain Analysis-Based COTS Selection

This domain analysis strategy has been integrated into the GOTHIC method by considering that the obtained domain model introduced in Fig. 6.1 is in fact the *Domain Model* that appears in Fig. 4.2 in Chapter 4 and Fig. 6.2.

Fig. 6.2 sketches the conceptual model of the COTS domain analysis activity. The *Information Source* class refers to the same class introduced in Fig. 4.2 and Fig. 5.1 that encloses the concept of information quality and its management, detailed in Chapter 5 and from which the domain analysis activity is based on.

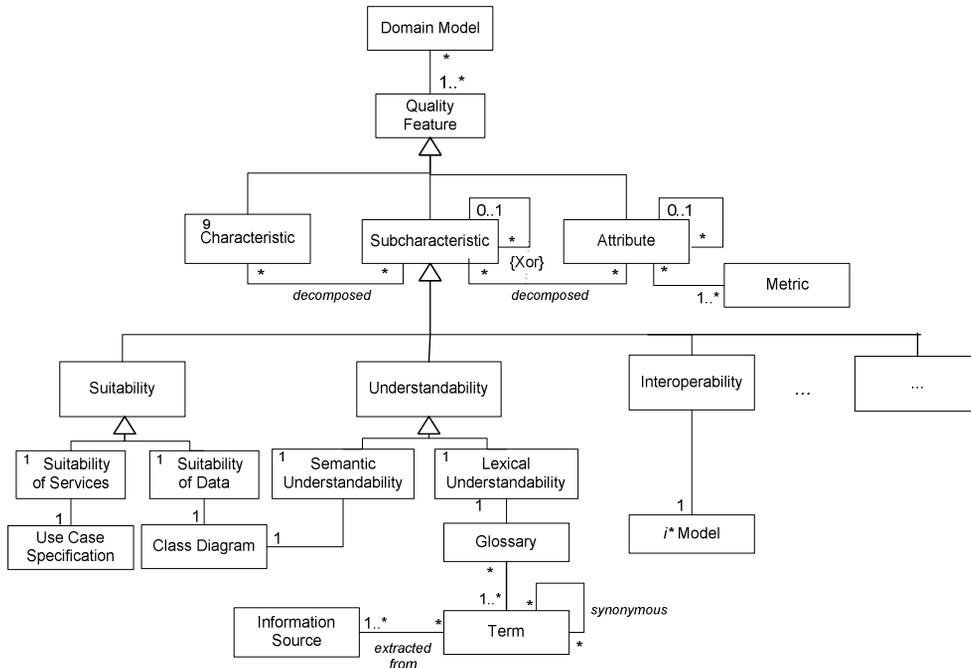


Fig. 6.2 Conceptual model excerpt for the GOTHIC domain model

As stated in Chapter 4, a GOTHIC reuse infrastructure is used to locate the taxonomy node that fulfils the needs of the user in charge of the selection process. Once located the taxonomy node, its domain model may be used to guide the rest of the selection process by refining this model with more specific requirements. The factors stated in the *domain model* help to elicit and negotiate the requirements, making easier the identification of mismatches among components characteristics and the requirements.

Moreover, those factors corresponding to the stated requirements are used to evaluate the capabilities of the candidate components in a uniform way, using the metrics defined in the model. For reaching the COTS domain analysis process, we can proceed manually, or use tool support. These tools range from the IQ Tool (introduced in Chapter 9) for dealing with the information sources from which domain analysis is based on; and to a simple spreadsheet or a more sophisticated tool, e.g. the DesCOTS system [Gra+04] (described in Chapter 9) for processing, editing and maintaining the produced *Domain Model*.

Reusability of the *Domain Model* (i.e., quality model) downwards categories and market segments of the domain hierarchies is a way to support reuse.

We have observed throughout our experiences that some quality features appear over and over, and this repetition is directly connected to the characteristics embedded in the

characterization attributes (i.e. the attributes that allow the partitioning of the nodes, explained in Chapter 7). The recognition of COTS market segments and categories improves reusability: once a new COTS market segment has been identified, its quality model can be constructed by inheriting the features of the *Domain Model* for those COTS categories in the hierarchy which it belongs to. During the process, new categories may be identified, abstracting commonalities of this new domain with others. As a result, a *Domain Model* bound to a category of the taxonomy collects all the quality features common to all its sub-categories and market segments. Since then, any quality model for a particular selection process may reuse the *Domain Model* of the corresponding COTS domain.

6.6 Case Study Example

For illustrating this activity, we present some excerpts of how we proceed to obtain the *Domain model* for the Real-Time Synchronous Communication (RTSC) case study.

As mentioned in Chapter 3, the RTSC case study embraces the various tools and technologies used to enable communication and collaboration among people in synchronous mode. Examples include instant messaging (IM), chat, audio/video conferencing, white-boarding, and application/desktop sharing. Synchronous means “same time – different place” mode. Thus, RTSC tools require to be connected at the same time, for example during Internet video or audio conferencing; and support a variety of media types, ranging from plain text to full multimedia.

It is worth to remark that the resulting artifacts related in this case study are thoroughly performed by other activities of the GOTHIC method. Therefore, most of the examples related in this section refer to or are complemented in other chapters.

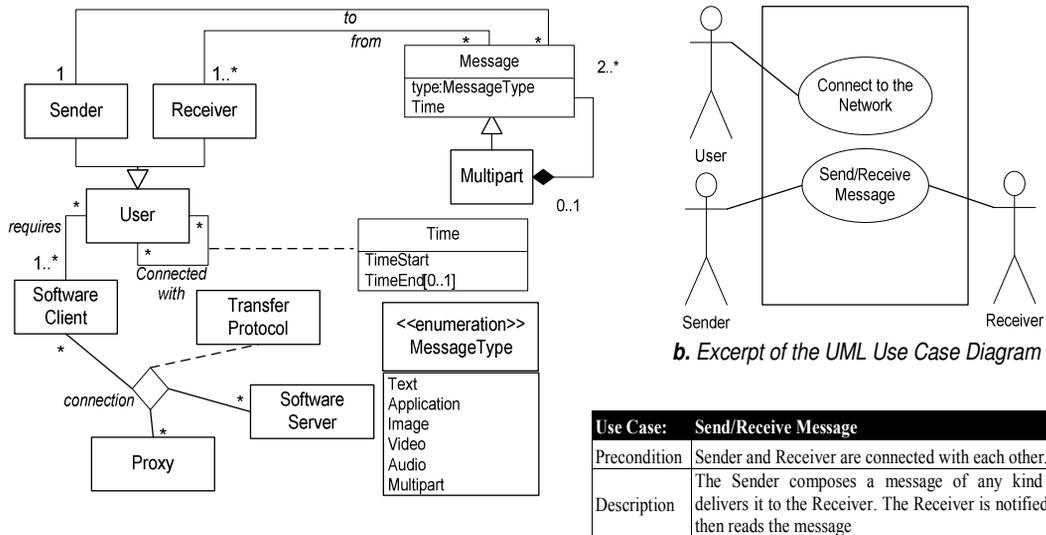


Fig. 6.3 Excerpt of some domain models constructed for the RTSC case

Fundamental concepts

Part of the UML class diagram is presented in Fig. 6.3a. Several key concepts are stated as classes. These concepts are of different nature, e.g. human roles (e.g. *Sender* and *Receiver*), artefacts of any kind (either physical or informational, e.g. *Message*), software and hardware domain-specific components (e.g. *Software Client*, *Software Server* and *Proxy*), etc. Inside these classes, we identify attributes but just those that play a crucial part in the domain, e.g. *Message* that can be of different types. Domain relationships are also of different kinds. Thus, we can see a high-level relationship among the human roles *Sender* and *Receiver* which are generalized into a *User* class. On the other hand, associations may be of very different nature. For instance, we have permanent or at least very stable relationships (e.g., among *User* and *Software Client*) while others are highly dynamic (real-time connections that are created and destroyed dynamically). OCL restrictions may be used to decorate the model appropriately.

Functionality

The use case model for functionality focuses on the most characteristic services offered by packages in this domain. Fig. 6.3b shows some for the RTSC domain, namely *Connect to the Network* and *Send/Receive Message*. Others such as *Send Video Message* or *Connecting Multiuser Session* are not included either because they are not considered general enough but specific of a few COTS, or because they are considered as secondary. In addition, we can also check that the individual use case specification of *Send/Receive Message* presented in Fig 6.3c follows the given recommendation of being very abridged. Further examples of the use case and scenario artifacts are found in Chapter 7, Section 7.2.

Interoperability

As it is the usual case in COTS domains that offer a lot of functionality, we may identify several relationships with other types of COTS.

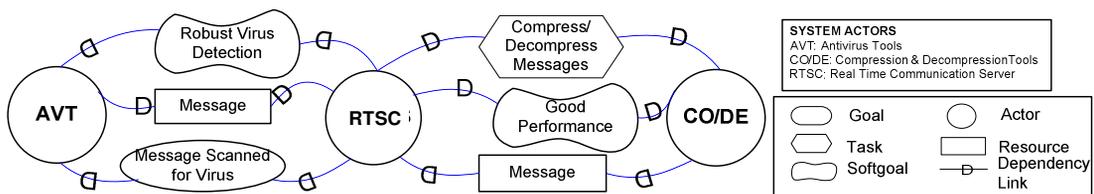


Fig. 6.4 Some dependencies among RTSC Tools and other types of tools

In Fig. 6.4 we introduce as example two COTS market segments related with RTSC, AntiVirus Tools (AVT) and Compression/Decompression Tools (CO/DE), all of them modelled as i^* actors. Among their relationships, we find: a RTSC component relies on an AVT component for detecting viruses (goal dependency, since the AVT decides the best way to do it) and requires this detection to be robust (softgoal dependency, because the concept of “robust” detection is matter of negotiation); a RTSC component depends on a CO/DE one to compress/decompress messages automatically (task dependency, because the RTSC states when and how these automatic activities are done); a RTSC

component may improve its performance using a CO/DE component (softgoal dependency, because the concept of “good” performance is matter of negotiation); and both related components need the message to work with from a RTSC component (resource dependency, because it is an informational entity).

Activity 4 of GOTHIC greatly deals with the construction of this model. So a detailed explanation of such construction is related in Chapter 7, Section 7.3.

Quality of service

In Table 6.3 we decompose a bit the *Understandability* subcharacteristic with the *Adherence to Best Practices* and *Supported Interface Languages* attributes.

Table 6.3 Excerpt of the quality model for the RTSC case

Quality factor		Metric	Description
3	Usability		ISO/IEC 9126-1 Characteristic
	1 Understandability		ISO/IEC 9126-1 Subcharacteristic
	3 Interface Understandability		Effort to recognizing the logical concepts and its applicability by means of interfaces.
	1 Adherence to Best Practices	ADP: 4valueOrder[Ordinal] 4valueOrder = (Optimal, Good, Fair, Poor)	How well events and elements of the interface comply with user interface best practices.
	2 Supported Interface Languages	SIL: Languages = Set(Labels[Nominal]) Labels = (Spanish, Catalan, English, ...)	Languages supported by the interface.

We include specific metrics that help to evaluate and compare user requirements. The first metric illustrate the subjective case, whilst the second one illustrates a metric that is both objective and structured (set of values). The description included in the table is in fact part of the glossary but appears for legibility purposes.

Non-technical description

Table 6.4 shows an excerpt of the refinement of a non-technical factor of a product, its history.

Table 6.4 Excerpt of a non-technical factor decomposition for the RTSC case

Non-technical factor		Metric	Description
3	Product		Non-technical characteristics of a COTS product that may influence COTS selection
	1 History		Evolution of the COTS since it has been offered to the clients
	1 Product in Market	Time: Years; Years: Integer	Number of years the product has been in the marketplace
	2 Versions of the Product	List Of Version; Version: Tuple (NumberVersion, Time); NumberVersion: String; Time: Years, Years: Integer	Versions currently available in the marketplace
	3 Patches per Version	List Of VersionPatches; VersionPatches: Tuple(NumberVersion, Number); NumberVersion: String; Number: Integer	Number of patches of each version

Note the similarity compared to quality of service description, which facilitates further integration. It should be mentioned that non-technical factors are very similar among different COTS segments.

Table 6.5 shows the integration of the presented excerpts in the unifying model using the mapping rules introduced in the Sections 6.4.1 and 6.4.2.

Table 6.5 Excerpt of the unifying model for the RTSC case²

Quality factor		Metric	Description
1	Functionality		See ISO/IEC 9126-1
	1 Suitability		See ISO/IEC 9126-1
	1 Suitability of Services		See Section 6.4.1 and 6.4.2
	1 Connect to Network	CN: 3ValueOrder[Ordinal]; 3ValueOrder = (Satisfactory, Acceptable, Poor)	See Fig.6.3b
	2 Send/Receive Message	SRMsg: 3ValueOrder[Ordinal]	See Fig. 6.3b
	...		
	2 Suitability of Data		See Section 6.4.1 and 6.4.2
	1 Message	Msg: 3ValueOrder[Ordinal]	See Fig. 6.3a
	2 Connected with	Cw: 3ValueOrder[Ordinal]	See Fig. 6.3a
	...		
2	Interoperability		See ISO/IEC 9126-1
	1 Anti-Virus Tools		See Fig. 6.4
	1 Robust Virus Detection	RVD: 3ValueOrder[Ordinal]	See Fig. 6.4
	2 Message Scanned for Virus	MSV: GoalValue[Ordinal]; GoalValue = (Attained, Not Attained)	See Fig. 6.4
	3 Message	Msg: ResourceValue[Ordinal]; ResourceValue = (Provided, NotProvided)	See Fig. 6.4
	2 CO/DE Tools		See Fig. 6.4
	1 God Performance	GP: 3ValueOrder[Ordinal]	See Fig. 6.4
	2 Compress/Decompress Messages	CDMsg:TaskValue[Ordinal]; TaskValue = (Executed, Failed)	See Fig. 6.4
	3 Message	Msg: ResourceValue[Ordinal]	See Fig. 6.4
3	...		
2	...		See ISO/IEC 9126-1
3	Usability		See ISO/IEC 9126-1
	1 Understandability		See ISO/IEC 9126-1
	1 Semantic Understandability	SU: Number[Unit]; Number=Integer	See Section 6.4.1 and 6.4.2
	2 Lexical Understandability	LU: Number[Unit]	See Section 6.4.1
	3 Interface Understandability		See Table 6.3
	1 Adherence to Best Practices	ADP: 4valueOrder[Ordinal]; 4valueOrder = (Optimal, Good, Fair, Poor)	See Table 6.3
	2 Supported Interface Languages	SIL: Languages = Set(Labels[Nominal]) Labels = (Spanish, Catalan, English, ...)	See Table 6.3
	2 Learnability		
	1 Available Documentation		Extraction of information from IQ Model introduced in Chapter 5
	1 Provider Documentation		See Section 6.4.1 and 6.4.2
	1 Documentation and User Manuals	Content: Nominal; Content=(NotProvided, Basic, Medium, Advanced)	See Section 6.4.1 and 6.4.2
	2 ...		

² The complete model is presented in Annex 3

Quality factor		Metric	Description
	2	External Documentenration	See Section 6.4.1 and 6.4.2
	1	Documentation and User Manuals	Content: Nominal; Content=(NotProvided, Basic, Medium, Advanced)
	2	...	
4	...other ISO/IEC characteristics		See ISO/IEC 9126-1
Non-technical factor		Metric	Description
1	Supplier		See [Car+07a] and [Car+07b]
2	Business		See [Car+07a] and [Car+07b]
3	Product		See [Car+07a] and [Car+07b]
	1	History	
	1	Product in Market	Time: Years; Years: Integer See Table 6.4
	2	Versions of the Product	List Of Version; Version: Tuple (NumberVersion, Time); NumberVersion: String; Time: Years, Years: Integer See Table 6.4
	3	Patches per Version	List Of VersionPatches; VersionPatches: Tuple(NumberVersion, Number); NumberVersion: String; Number: Integer See Table 6.4
	2	...	

Our approach was complemented by using (when applicable) several subcharacteristics appearing in several COTS domains, as reported in [Car+07a] and [Car+07b]. The complete obtained model is presented in Annex 3.

6.7 Summary and Discussion

We have detailed the domain analysis approach for building a reuse infrastructure for supporting COTS selection processes enclosed in our GOTHIC method.

This approach is based on the application of domain analysis principles for recording and representing all the required information for evaluating COTS. Our proposal relies on several industrial experiences that have been undertaken under action-research premises, complemented with literature survey and grounded theory.

Concerning domain analysis, we have concluded that existing approaches were not oriented to support reuse in the COTS framework, consequently the need of mechanisms to analyze and create a reuse infrastructure for COTS domains gave the origin to the stated strategy.

With respect to COTS selection:

- We have put the emphasis on reuse, making a concrete proposal based on the domain analysis technique which allows transferring knowledge from one experience to another.
- We have explicitly identified the informational dimensions required for the effective and efficient selection of COTS.

- We have offered guidance for representing these informational dimensions using appropriate types of models.
- Using some mapping rules, we have integrated all these models into a single one, based on a well-known standard called ISO/IEC 9126, which is highly oriented to support the evaluation of the candidate components. This model was complemented by using (when applicable) several subcharacteristics appearing in several COTS domains, as reported in [Car05T].
- Given this representation, we may use some existing tool-support to conduct the evaluation of candidates in the framework of the ISO/IEC 9126-1 standard.
- Domain analysis not only impacts positively on reuse, but also ameliorates some well-known obstacles for COTS selections success (as those mentioned in Section 1.1). Remarkably, using domain analysis principles we avoid those semantic and syntactic discrepancies that are common in the COTS marketplace.

Finally, for understandability of the domain analysis strategy used in this activity, an example on applying this strategy to the RTSC case study was illustrated. Annex 3 presents the whole model obtained by the application of our approach which has been complemented by the framework proposed in [Car07a] and [Car+07b].

Chapter

7

Activity 3, 4, 5:

Goal-Oriented Core of GOThIC

The goal-oriented core of the GOThIC method is defined as an iterative and incremental process of harmonized activities: *Identification, Refinement and Statement of Goals* (Activity 3); *Establishment of Goal Dependencies* (Activity 4); and *Goal Taxonomy Structuring* (Activity 5). Although they are described as different activities, the techniques and models used in each of them are really complemented or improved as the knowledge of the domain increases.

“A goal is an objective that should be achieved and may be formulated at different levels of abstraction, ranging from high-level strategic to low-level technical concerns” [Lam01].

Goal characteristics (e.g., expressiveness, stability and evolvability) as used in the context of requirements engineering (see Chapter 2) make suitable the use of goal-oriented approaches for representing and managing COTS marketplace structuring efforts (a further explanation of this fact is provided in Section 2.4.1 in Chapter 2). In this chapter, a detailed explanation of the goal-oriented strategy followed is presented.

The chapter is organized as follows: Section 7.1 provides a brief background of previous research in the COTS classification and goal-oriented areas (further introduced in Chapter 2). Section 7.1, 7.2 and 7.3 detail the *Identification, Refinement and Statement of Goals* activity; *Establishment of Dependencies* activity; and *Goal Taxonomy Structuring* activity respectively. Examples are given to illustrate them. Finally, Section 7.4 concludes with a brief summary and discussion.

7.1 Background

Systematically sorting software into classes that specify their allowable uses is a complex and central task to achieve efficient software reuse. Although the classification of reusable components has been an active area since several years ago, the COTS classification specific area has recently emerged (see Chapter 2).

Several COTS classification mechanisms have been proposed; but they share various drawbacks that make their use for characterizing COTS difficult (see section 2.2.3 in Chapter 2 for a detailed explanation). In summary, while existing approaches paid more attention to the structure of the classification, they do not deal with the methodological aspects required to support its construction, neither with mechanisms to support its imminent evolution as a consequence of marketplace progress.

The three GOTHIC activities forming part of the goal-oriented core seek to provide a methodological foundation for constructing an efficient and evolvable domain classification schema. These activities were built upon existing approaches as goal-oriented approaches (e.g., [Ant97], [Yu95], [Reg05]) and decision trees [Qui86], but applied in different ways and with different objectives than traditionally. The use and appropriateness of these approaches is discussed in Chapter 2, Section 2.4.

Goal-oriented mechanisms are defined in order to extract and organize COTS domain goals from the *Exploration of Information Sources* activity (Chapter 5). Domain goals semantically represent related groups of functionalities instead of services that COTS belonging to such domain offer. Also goal dependencies are explicitly represented by using i^* models. Subsequently, decision trees properties are used to organize such goal information as taxonomies (i.e. classification schemas). As a result, such taxonomy and their associated information and mechanisms make our proposal more handy and robust with respect to COTS marketplace evolution and representational needs.

Next subsections describe the resulting goal-oriented strategy by its three related GOTHIC activities, the used techniques and produced artifacts.

7.2 Activity 3: Identification, Refinement and Statement of Goals

Inspired by the iterative process of goal identification, refinement and statement used in GBRAM method (see Fig. 2.4) we envisaged a process for the identification, refinement and statement of goals from suited sources previously obtained in Activity 1.

GBRAM offers a wide range of useful heuristics and procedural guidance for identifying and refining goals. Our goal-oriented strategy makes use of some of these GBRAM heuristics and definition mechanisms. Table 7.1 summarizes the kind of heuristics used; some of them have been modified and adapted to fit to the GOTHIC objective.

Moreover, these heuristics were complemented with others derived from our experiences and observations from the case studies related in Chapter 3. The detailed set of heuristics obtained to date is presented in Annex 1.

Table 7.1 Glossary of GBRAM heuristics used in GOTHIC’s Activity 3

Code	Definition
HIG	Heuristic for Identifying Goals
HIS	Heuristic for Identifying Stakeholders
HIA	Heuristic for Identifying Agents
HIC	Heuristic for Identifying Constraints
HRR	Heuristic for Refining Redundancies
HRS	Heuristic for Refining Synonymous
HEO	Heuristic for Elaborating Obstacles
HES	Heuristic for Elaborating Scenarios

The use of heuristics includes some abstraction mechanisms based on the Inquiry Cycle model [Pot+94]. It consists of a series of questions and answers designed to pinpoint where and when the information needs to arise. Some of them are straightforward and generic but others make sense only in conjunction with specific questions about the domain. For instance, an abstraction mechanism that may be employed to extract goals is analyzing the statement by asking: “what goal(s) does this statement/fragment exemplify?” and usually some goals become evident from the description [Ant97].

Next paragraphs describe all elements of Activity 3. To illustrate the concurrent processes of goal identification, refinement, and statement we provide as example the RTSC case study.

7.2.1 Goal Identification and Refinement

The identification and refinement of COTS domain goals assemble an iterative process aimed at extracting and refining goals from the domain related information identified in Activity 1 (see Chapter 5). It is done by applying heuristics (see Annex 1) and diverse goal-acquisition techniques –as those summarized in [Reg05]–, among which the Inquiry Cycle, scenario construction and consideration of obstacles for refining goals play a crucial role.

Some of the mechanisms used throughout this process are:

7.2.1.1 Supporting Mechanisms, Techniques and Models

A mechanisms that resulted helpful throughout the goal-oriented core of GOTHIC for the elicitation, representation and organization of domain goals and their assignment to COTS functionalities at various levels, was the identification of different types of domain actors, as proposed by [Car05T]. Table 7.2 summarizes the different types of actors involved in COTS domains.

An additional resource to consider, regarding to the actors, is their relationship with stakeholders. In [Ant97], a stakeholder is anyone that claims an interest in the system and may be associated to system goals. Their identification is supported by heuristics (see Annex 1) and it is a vehicle for considering multiple viewpoints and potentially affected parties for various goals within the domain. Multiple stakeholders may be associated with a goal. Anton also defines the concept of agent as the responsible for ensuring the completion of a goal at any given time; this concept is the one we use to

define our actors. Thus, it is important to stand out that regarding to terminology; the concept of agent used in [Ant97] is the same as the actor concept used in our approach.

The initial stating of high level goals of the domain is a difficult task endorsed by the amount and diversity of information sources related to the domain.

From our former case studies we learnt that it should be considered as a good practice to base the identification process on the most solid and confident of the sources (as prioritized in Activity 1, see Chapter 5) for extracting the main high level goals in order to assure the consistency of the set of goals, and then extracting and refining goals from the remaining sources.

As a matter of fact, the goal identification usually started by intuitively selecting a set of environmental actors and their root goals, whose decomposition usually leads to the discovery of new actors, stakeholders, and goals.

Table 7.2 Different types of actors related to COTS domains

	Types of Actors	Definition
Environmental Actors	Human Actors	They represent different types of users involved in the COTS domain that is being analyzed; e.g. system administrators and end-users.
	Organization Actors	They represent external organizations with their own goals and/or providing some services to the systems of the domain, e.g. certification authorities required to provide/validate electronic keys.
	Hardware Resource Actors	They represent mechanical devices governed by the systems of the domain or providing data to the domain systems, e.g., scanners, bar-code readers, firewalls, etc.
	Software Actors	They represent software systems, which provide system functionalities to the domain; e.g. the mail clients or web browsers in the case of mailing systems. They could be of two different types: Core system actors and supporting system actors.
System Actors	Core System Actors	This kind of actors provides the core functionality of the domain. Most of the committed and critical dependencies of environmental actors are usually linked to them. Some examples of core system domain actors are the Mail Server in e-Mail domain, and the Document Management tool in the case of Enterprise Content Management Systems.
	Supporting System Actors	They do not provide the core functionality of the domain. Instead they offer services required by the core actors, in order to fulfill some of their dependencies with environmental actors. All supporting actors have dependencies with core actors, but not necessarily among them. They may also have dependencies with environmental actors, but usually not in relation to the core functionality of the domain.

As stated in [Car05T], there is not a clear and unique method to support the identification and refinement process, however in this thesis we mainly used the approaches followed by [Ant97] and [Yu95] which techniques (refinement mechanisms; and actor reasoning respectively) are widely used in the community –although any other goal identification technique can be used-. The use of both techniques resulted greatly complemented because the construction of *i** models was essential not only for recording dependencies among COTS (as it is detailed in Activity 4 explained below); but also to illustrate the identified goals and clarify the understanding the domain. For

avoiding syntactic and semantic discrepancies common in the widespread and unstructured COTS marketplace information, we found useful the use of the Language Extended Lexicon (LEL) [Lei-Fra93] for capturing a glossary of the domain which homogenizes terms of different information sources and helps to detect synonymous or duplicated goals (as introduced in Chapter 6).

Table 7.3 An excerpt of actors and goal identification

Domain: Real Time Synchronous Communication (RTSC).			
Definition: It includes the various tools and technologies that can be used to communicate people in real-time, it means “same time but different place”			
Goal: Provide RTSC			
High Level actors identification			
Actor	Abbreviation	Type	High-level Goal
RTSC-Server	RTSC-S	Software/Core	Provide RTSC infrastructure
RTSC-Client	RTSC-C	Software/Core	Enable RTSC among RTSC-S and RTSC-User
RTSC-User	RTSC-U	Human	Send and Receive messages
RTSC-Administrator	RTSC-A	Human	Put RTSC-S to work accurately and efficiently
Firewall	Fwall	Hardware	Protect from unauthorized access (Filter incoming requests)

Table 7.3 illustrates a list of environmental actors that interact directly with high level goals in the RTSC case study. We identify their type and also give a short description of the actor’s main goal. As it can be noted, the main goal for which the RTSC domain is required has been initially refined into 5 high-level goals in relation to the main services for which the environmental actors depend on the domain. These high-level goals have been further refined to several sub-goals and so on.

The refinement of goals may be addressed by considering several techniques as the scenarios construction, identification of pre- and post-condition of goals, goal obstacles, mechanisms to discover synonymous or duplicated goals and i^* SD models. Please see [Ant97] and [Yu95] for a detailed explanation of these goal-identification techniques and the way they may be used. In [Fra+07] we also provide a methodological approach for supporting the discovery of goals and the construction of i^* models; the most relevant heuristics of the approach have been summarized in Annex 1.

The level of detail in identifying goals will vary greatly depending on the context of use and final application for which the reuse infrastructure aimed by GOTHIC is intended. Table 7.4 and 7.5 are examples of some of the mechanisms that are used for refining goals in the RTSC case.

Table 7.4 A scenario excerpt of the RTSC case study

Action Initiator	Goal	Consumed Resources	Produced Resources	Action Addressed
RTSC-User (Sender)	Message Sent	Message	Message, Receiver address	Requesting to Software Client
RTSC-Client	Sent Request to the Server	Message, Receiver address	Sender address	Requesting to Software Server
RTSC-Server	Messages Routed	Message, Sender and Receiver address	Routed Receiver address	Sending to Software Client (Receiver)
RTSC-Client	Message Delivered	Message, Sender address	Message	Deliver to a Human User (Receiver)
RTSC-User (Receiver)	Message Received	Message, Sender address	Message	Answering

Table 7.5 Example of the identification and refinement process

Goals	Goal Obstacles	Name of the related Use Cases
Provide RTSC infrastructure	-There is no infrastructure available -Users Not Connected at the same time -Firewall not configured adequately	-Users Communicated in Real Time -Session Established ...
Message Sent/Received	-There is no compatibility with RTSC-C -Message is coded -Message is infected by virus -It is not a desired message	-Session Established -No Compatibility among Transfer Protocols -Coded Messages -Undesirable messages ..
...		

As the construction of scenarios and application of heuristics provide us an understanding of the domain, the *i** SD environmental model is iteratively refined. It provides a graphical idea of the domain that is helpful to the application of some requirements engineering techniques as interviews, clarifying and unifying concepts from stakeholders, and a better understanding of what environmental actors expect from the domain. For understandability purposes, all the *i** SD models' characteristics and usage are tackled in Section 7.3.

7.2.2 Statement of Goals

Statement of goals consists on summarizing the goal information by stating it in a systematic way called goal schema. We use a pre/post style for specifying these goals, i.e. stating which conditions are met when others hold, as showed in Table 7.6.

To manage the amount of goal information that is produced during the process, we have used the Taxonomy Tool module of the DesCOTS System (see Chapter 9). It offers a hierarchical structure that allows defining goals, subgoals and recording the goal schema information. Other tools that could be suitable for performing this task is the sophisticated GBRAT tool [Ant+96], and even simple spreadsheets tools.

Table 7.6 An example of a goal-schema

Goal:	Multiuser RTSC Established
Description	Provide RTSC in a Multi-user Environment
Actor/Agent	RTSC-Client
Stakeholder(s)	RTSC-Client, RTSC-Server, RTSC-U
Precondition(s)	1) Provide RTSC infrastructure; 2) Enable interaction among RTSC-S and RTSC-User 3) Number of users >2
Postcondition(s)	Multiuser RTSC Established
Subgoal(s)	1) Multi-user Textual Communication Established; 2) Multi-user Videoconferencing Communication Established ...

7.3 Activity 4: Establishment of Dependencies

For COTS reuse being effective, in Chapter 6 we argued that dependencies among COTS shall be identified and recorded explicitly for their repeated use during different selection processes. These dependencies help organizations involved in a selection

process to find out that some goals that they want to achieve with a COTS will not be satisfied if they do not have or procure other COTS.

Therefore, we envisaged that the goal-domain classification we pursued also provided a great opportunity for including these dependency relationships as an additional element for structuring the COTS marketplace. We found that analyzing the goal information, relationships among COTS categories and market segments can be gradually identified and declared as dependencies using i^* SD models [Yu95].

7.3.1 Use of i^* SD models

One of the strongest points of i^* is its graphic utility and the freedom it provides for defining and using its elements. However, we found that when working over a specific problem or domain, some difficulties arise: there is an overload of variants of the i^* language, a lack of guidelines for producing the models, and tool support is essential. In order to address those aspects, in [Aya+05b] we presented a reference meta-model to analyse the different variants of the i^* language and, in [Fra+07] we proposed a methodology for guiding the i^* SD models construction.

Thus, we state that i^* dependencies may be of four different types:

- ♦ *Goal dependencies* {G}, when an actor depend on another to attain a goal;
- ♦ *Task dependencies* {T}, when an actor requires another to perform an activity in a given way;
- ♦ *Resource dependencies* {R}, when an actor depends on another for the availability of some data; and
- ♦ *Soft goal dependency* {S}, when an actor depends on another to achieve a certain level of quality of service.

In fact, task dependencies are rarely used in our approach because they represent “how” to do something, while we want to represent “why” it has to be done. It means that we do not want to represent an excessive level of detail but the general services that are expected to be provided for COTS.

To state the domain goals in an adequated way using i^* SD models, we follow the lineaments proposed in [Car05T].

7.3.2 Identifying COTS dependencies

Fig. 7.1 provides an excerpt of the initial i^* SD model that help to summarize, refine and understand much of the goal information obtained by other techniques (as those shown in Tables 7.2, 7.3, 7.4 and 7.5). i^* models are iteratively refined as the domain knowledge increases and by the identification of COTS dependencies.

The semantic of the model presented in Fig. 7.1 is as follows:

- The RTSC-C depends on the RTSC-S for the messages to be sent/received, messages to be routed and the storage of information of the RTSC-U. Because the RTSC-C acts as a mediator among the RTSC-S and the RTSC-U, it needs to provide an image of the resources in the RTSC-S which are available to the RTSC-U (e.g.

user information, available list of contacts, incoming request communications, etc). They are modeled as resource dependencies from the RTSC-C to the RTSC-S.

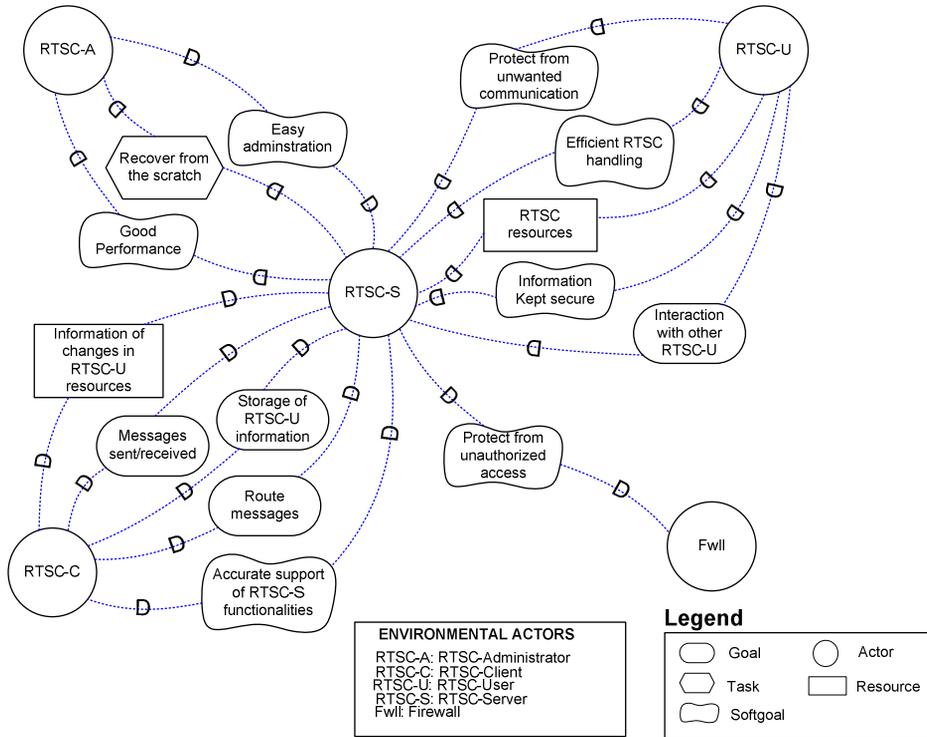


Fig. 7.1 *i** SD model progressively constructed to assess the RTSC domain

- Due to the nature of the RTSC domain, RTSC systems are platforms which endorse groupware cooperation. This fact is reflected by the Interaction with other RTSC-U for which RTSC-U depend on RTSC-S.
- Soft-goals have been identified following the ISO/IEC 9126-1 characteristics and subcharacteristics. It provides us insights of the kind of goals that should be taken into account for describing the functionality expected of the domain. For instance, there exist soft-goals concerning security, efficiency and usability. Some examples are: The RTSC-S depends on the Fwll to protect from unauthorized access; the RTSC-U depends on the RTSC-S to protect from unwanted communication (communication denied) and for its private information to be kept secure; and it also depends on the RTSC-S for the efficient RTSC handling.
- The RTSC-S behavior depends on the environmental actors, as reflected by the two dependencies stemming from RTSC-S to RTSC-A. The RTSC-A depends on the RTSC-S to allow an easy administration (e.g. provide administrative tools, configuration manuals, online help, etc). The RTSC-S on the other hand depends on the RTSC-A for being “fine tuned” to maintain a good performance. These two dependencies have been modeled as soft-goals. And finally the RTSC-S depends on the RTSC-A to perform the recovery from the scratch in case of a system failure.

The refinement of the high-level dependencies appearing in Fig. 7.1 can be observed in Fig. 7.2.

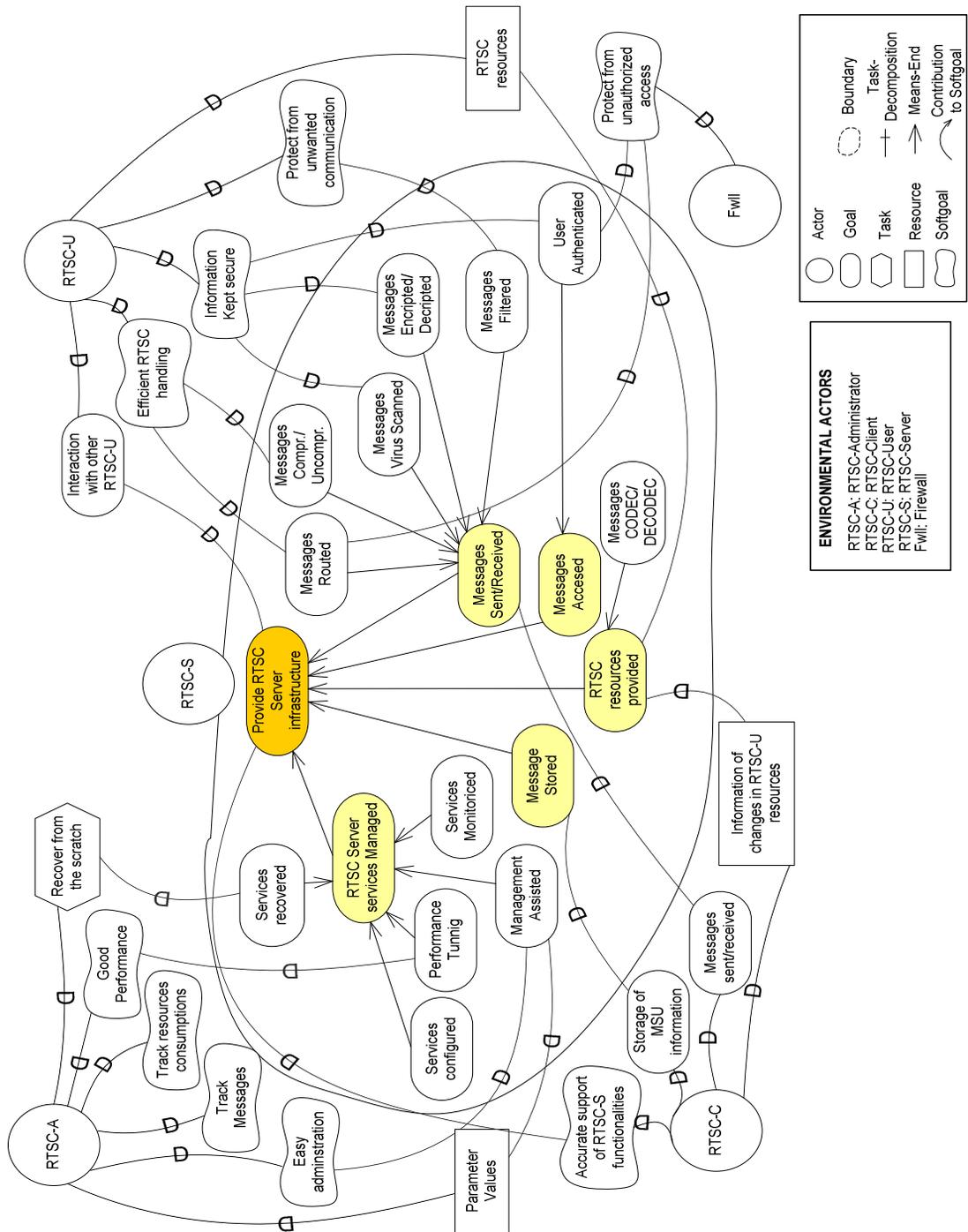


Fig. 7.2 Example of the refinement and establishment of COTS dependencies

The identification of system actors is the basis for the COTS dependencies identification and assessment. By analyzing some of the available COTS functionalities in the marketplace, we can identify system actors covering domain goals because one of their main characteristics is that they may represent COTS available functionalities. The importance of this process is twofold: On the one hand, domain goals may be further refined, as the case of the goal Messages Sent/Received that can be refined into two sub-goals One to one Messages Sent/Received, and Multi-user Message Sent/Received, given by the fact that these two functionalities are relevant to the available tools covering this goal. On the other hand, goals reveal services that are expected to be covered by new system actors.

This process is supported by reviewing COTS related information (see Chapter 5) that provides an overview of the current COTS available in the market and the functionalities they provide. The assessment of available taxonomies (as those cited in section 2.3.2) that although not well-structured and presenting some problems, may be used to identify system actors covering domain goals.

Fig. 7.3 provides an excerpt of the progressively identification of environmental and system actors. System actors are not supposed to be mapped directly onto individual COTS, because there are many cases that could not fit with this assumption given the actual offering of the COTS marketplace and its constant evolution. Therefore it is better to decouple the domain goals to the functionalities that current COTS provide in order to maintain our proposal flexible. Some representative examples of this are the next cases: an actor may be covered by one COTS; a COTS covering the services of more than one actor; an actor covered by two COTS at the same time for survivability reasons; or even some services that can not be covered by any COTS (requiring some bespoke software to be developed).

In the RTSC case study, the RTSC-Server (RTSC-S) has been identified as the core actor of the system. RTSC-S provides the main functionality required to make the RTSC infrastructure available –from the point of view of the user-. They enable the exchange of messages (of many kinds: textual, files, video, voice and other formats) as well as their adequate treatment, storage and access. RTSC-S also provides management facilities which allow for their configuration, monitoring, recovering from failures and tuning for an optimal performance. The configuration activities include the management of typical system resources such as user and group profiles, and access control lists, but also other more specific of the domain such as collaborative work. The configuration of such resources is required in order to make them available to the users and also to perform actions such as the authentication of users of the system.

Table 7.7 summarizes some of the goals that the identified system actors may cover.

Supporting actors are related to the goals of the domain not covered by the RTSC-S. Data Compression Tools (DCT) are required by the RTSC domain, as well as by other systems performing heavy data transmission across local networks and/or through the internet, to optimize the performance and resource consumption the RTSC domain on the other hand relies on Routing Tools (RT) in order to messages be routed to their destination.

RTSC security can be defined with regard to several aspects.

Table 7.7 Excerpt of the identification and coupling of system actors

	System Actor	Abbreviation	Goals	Available Functionalities
Core System Actors	RTSC-Server	RTSC-S	<ul style="list-style-type: none"> ➤ Provide RTSC infrastructure <ul style="list-style-type: none"> ➤ Intra-organizational RTSC infrastructure ➤ Internet based infrastructure ➤ WAN infrastructure 	<ul style="list-style-type: none"> ➤ Intra-organizational RTSC infrastructure ➤ Internet based infrastructure ➤ WAN infrastructure
			<ul style="list-style-type: none"> ➤ Messages Sent/Received <ul style="list-style-type: none"> ➤ Messages Sent/Received to a User ➤ Messages Sent/Receive to more than one user ➤ Messages Stored ➤ Messages Accessed ➤ RTSC Resources provided ➤ RTSC Services Managed ➤ ... 	<ul style="list-style-type: none"> ➤ One to one communication ➤ Multi-user communication ➤ Text, Audio, Video, data messages ➤ Collaborative production of resources (e.g., shared applications in real time) ➤ Sharing resources
	RTSC-Client	RTSC-C	<ul style="list-style-type: none"> ➤ Enable interaction among RTSC-S and RTSC-User <ul style="list-style-type: none"> ➤ Compatibility with RTSC-S <ul style="list-style-type: none"> ➤ Messages Sent/Received ➤ ... 	<ul style="list-style-type: none"> ➤ Intra-organizational RTSC infrastructure ➤ Internet based infrastructure ➤ WAN infrastructure ➤ Text, Audio, Video, data messages ...
Supporting System Actors	Routing Tools	RT	<ul style="list-style-type: none"> ➤ Messages Routed 	<ul style="list-style-type: none"> ➤ Routing messages
	Directory Services	DS	<ul style="list-style-type: none"> ➤ Resources Stored (e.g. RTSC-U information) ➤ Resources Accessed ➤ Resources Assigned ➤ Resources Structured ➤ Permissions Validated 	<ul style="list-style-type: none"> ➤ Management of messages
	Codec/Decoded Tools	CO/DEC	<ul style="list-style-type: none"> ➤ Messages Coded/Decoded 	<ul style="list-style-type: none"> ➤ Coding/Decoding messages
	Data Compression Tools	DCT	<ul style="list-style-type: none"> ➤ Messages Compressed/Decompressed 	<ul style="list-style-type: none"> ➤ Compress/Decompressing messages
	Anti-Spam Tools	AST	<ul style="list-style-type: none"> ➤ Messages Filtered (Protect from spam) 	<ul style="list-style-type: none"> ➤ Filtering messages from spam
	Anti-virus Tools	AVT	<ul style="list-style-type: none"> ➤ Messages Virus Scanned (protect from virus infections) 	<ul style="list-style-type: none"> ➤ Anti-virus protection
	Data Encryption Tools	DET	<ul style="list-style-type: none"> ➤ Messages Encrypted/Decrypted 	<ul style="list-style-type: none"> ➤ Encrypt/Decrypt messages
	Backup and Recovery Tools	BRT	<ul style="list-style-type: none"> ➤ System Data and Messages Backed-up/Restored 	<ul style="list-style-type: none"> ➤ Backup and Recovering systems
	Billing Tools	BT	<ul style="list-style-type: none"> ➤ Resources Usage Tracking 	<ul style="list-style-type: none"> ➤ Track of resources
	Message Tracking Tools	MTT	<ul style="list-style-type: none"> ➤ Messages Tracking 	<ul style="list-style-type: none"> ➤ Track of messages
	Configuration and Administration Tools	CAT	<ul style="list-style-type: none"> ➤ Management Assisted (assist. on Services Configured, Performance Tuning, Services Recovered) 	<ul style="list-style-type: none"> ➤ Management supported by software
Overall Administration Tools	OAT	<ul style="list-style-type: none"> ➤ Centralized and complete control of the RTSC resources 	<ul style="list-style-type: none"> ➤ Overall management of RTSC systems supported by software 	

On the one hand RTSC-U aim to be free from unwanted messages and that his/her information is kept secure from unauthorized access. The first one is achieved by the RTSC-S which requires at least an authorization for contact, in the other case; this is partially achieved by the RTSC-S which requires at least, user's login and password authentication.

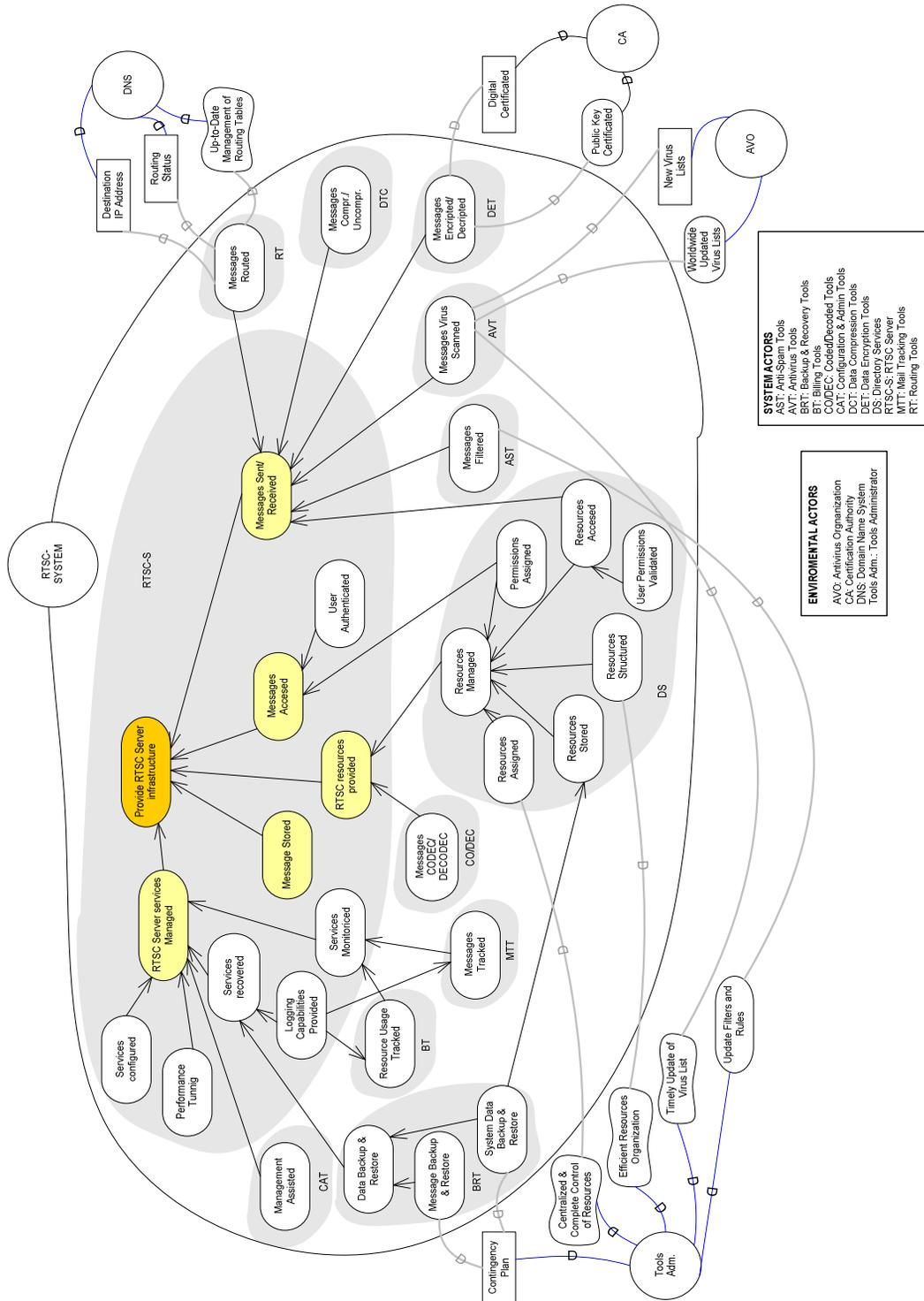


Fig. 7.4 Example of discovering other environmental actors

Additional support actors are required to support other security related sub goals, namely Antivirus Tools whose goal is to scan messages and protect systems from accidental or electronic virus infections, and Data Encryption Tools which provide authentication, data integrity and non-repudiation. RTSC rely on Configuration and Administration Tools to assist on their management. They provide feature rich and friendly interfaces that helps on the installation, configuration, tuning and recovery activities that MS may require.

For instance, we found that DS services provide integrated management of network resources, allowing global access to distributed resources such as system user's information, addresses list, distributed folders, etc. DS store information about objects on the network and make this information easy for administrators and users to find and use. This is also the case of the Message Tracking tools (MTT) and the Billing Tools (BT) system actors, which were identified after reviewing information of administrative aids for RTSC-S components.

Message Tracking Tools (MTT) are used to track and report the status of messages being sent and received by the RTSC-U, whilst Billing Tools (BT) are used to provide an account of how system resources are being used. A similar fact is regarding to the discovery of the Backup and Recovery Tools (BRT) system actor, which is required to safeguard and recover system and user data in case of system failure (supporting the recoverability of the system).

Although domain goals and system actors covering them should not be restrictedly coupled, a great interrelationship exist among them, because system actors provide a guide for identifying which kind of COTS are required by the domain and should be analyzed in the procurement processes; whilst domain goals help to identify what kind of functionalities can be grouped in a single actor. Fig. 7.4 depicts these findings.

As a result of the analysis for identifying COTS covering the domain goals, new soft-goals, goals, and environmental dependencies may be identified. For instance, the RT system actor depends on DNS (Domain Name Server) actor to gather destination IP addresses where messages have to be delivered. Regarding security, RTSC-S depends on certification authorities in order to get public keys certified by means of digital certificates, which are required for their authentication. Also, in relation with system security, the RTSC-S depends on anti-virus organizations to maintain worldwide updated virus list, and to continuously provide plug-ins in the form of new virus list required by the system to be protected from them. In addition, this "refinement" process also requires to review the goal dependencies assignments in order to re-link the corresponding dependencies to the new identified actors.

The i^* SD model presented in Fig. 7.5 illustrates an overall overview of the dependencies we found in the RTSC case study:

- RTSC-A dependencies. The RTSC-S depends on the RTSC-A to perform the recover from scratch {T}, and also to fine-tune it to achieve a good performance {S}. However, in order to provide these services, the RTSC-A depends on several system actors: the BT to track the resource consumption {S} by the users, the MTT to track messages {S} sent/received by the RTSC-U, and the CAT which provides to the RTSC-A an interface for an easy administration {S} of the RTSC-S.

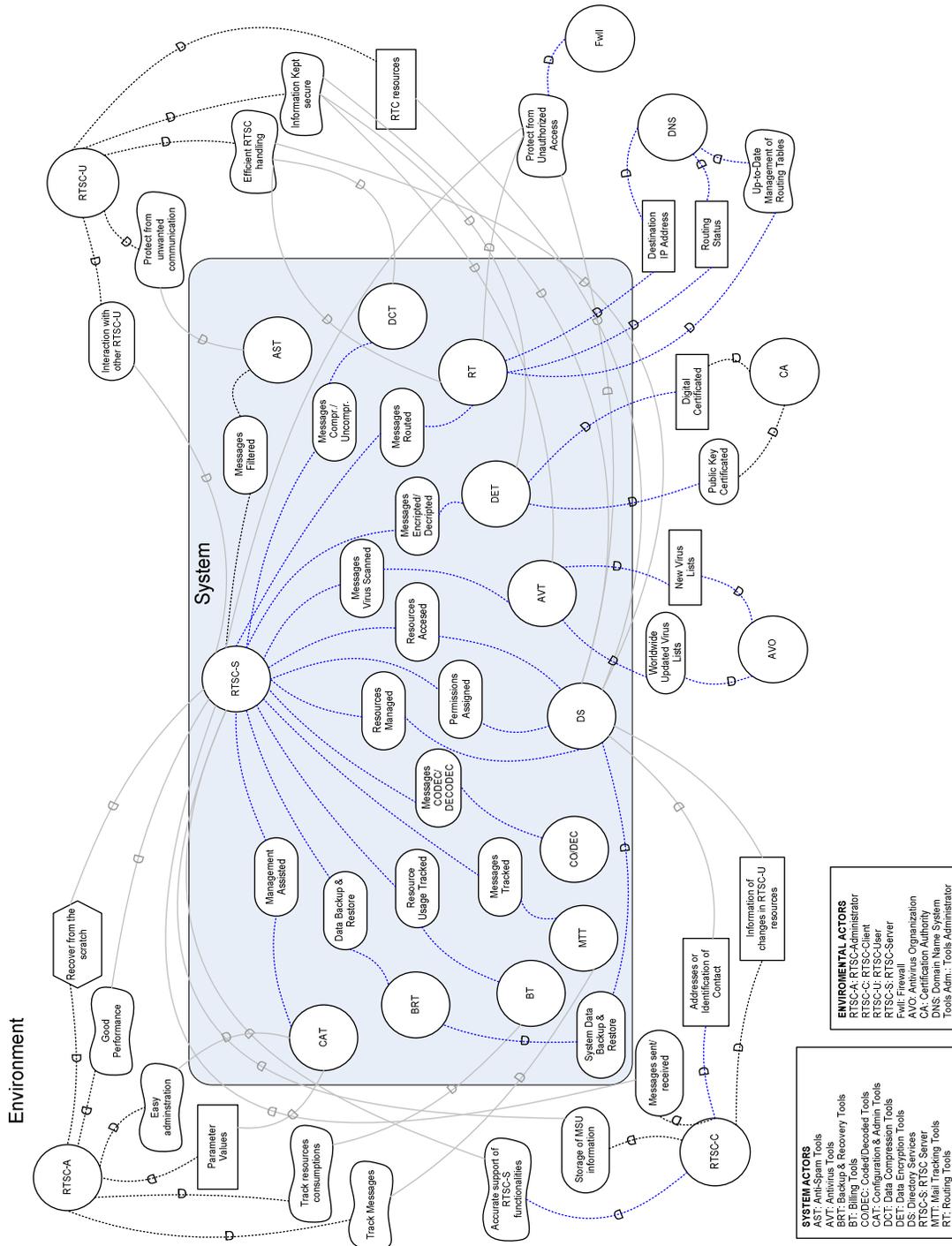


Fig. 7.5 Overview of the *i** SD model obtained¹

¹ the Tools Adm. environmental actor is not included to simplify the view.

- At its turn, the CAT depends on the RTSC-A to input the parameter values {R} required for the good performance {S} of the RTSC-S.
- RTSC-U dependencies. The RTSC-U may depend on the RTSC-S for cooperation with other RTSC-U {G}. This is due to the fact that the RTSC-S provides the core functionality of the RTSC-SYSTEM required for the exchange of messages. The RTSC-U also depends on the AST to Protect From Unwanted Messages {S} and the DS for RTSC Resources {R} to be assigned and accessed.
- RTSC-C dependencies. RTSC-C dependencies are mapped to the RTSC-S, and the DS system actors. The RTSC-S provides the core system functionality required by the RTSC-C, thus, it is responsible for the Messages Sent/Received {G} and the Accurate Support of RTSC-C Functionalities {S}. Because RTSC-Cs act as mediators among the system and RTSC-U, they need to provide an image of the system resources. This image is obtained by means of resource dependencies on the DS, namely the Information of Changes in RTSC Resources {R} and Addresses {R}. The responsibility for Storage of MSU Information {G} is assigned to both the RTSC-S and the DS system actors, the former is responsible for the storage of the messages and the latest for the storage of the remaining related resources. As RTSC-Cs are external to the system and in general not controllable by RTSC-A, its selection and configuration is usually relied to RTSC-U. Therefore, the services that they provide not only depend on the services offered by the system, but on the capabilities of the selected client.
- Fwll dependencies. The system depends on the Fwll environmental actor to Protect it From Unauthorised Access {S} of external users. Once decomposed into system actors it was clear that several of them, namely, RT, RTSC-S and DS depend on it for this security soft-goal to be achieved.
- CA dependencies. The dependencies among the system and the CA environmental actor have been mapped to the DET system actor. DET systems actors depend on CA environmental actors in order to get Public Key Certified {G}, by means of Digital Certificates {R}.
- AVO dependencies. AVO environmental actors are required by AVT system actors to maintain Worldwide Updated Virus Lists {G}, and to periodically get New Virus Lists {R}. Therefore dependency links among the RTSC-System and the AVO environmental actor are mapped to the AVT system actor.
- DNS dependencies. Dependencies among the system and DNS are mapped to the RT system actor. RT depends on the DNS environmental actor to gather Destination IP Addresses {R}, and the Updated Routing Status {R}, but also for the Up-to-Date Management of Routing Tables {S}.
- Overall Tools Adm. dependencies. (They are not included in the model of Fig. 7.2 to simplify the view). There are several supporting system actors which depend on Tools Adm., to be configured and continuously fine-tuned for a Good Performance {S} and also for their Recover from Scratch {T} in case of their failure. Additionally, the BRT depends on it to define a contingency plan {R}, the AVT for the Timely Update of Virus Lists {S}, the AST to keep Updated Filters and Rules {G} and the DS for an Efficient Resource Organization {S}. Also the Tools Adm.

depend on all of them for an Easy Administration {S} and on the DS for a Centralized & Complete Control of Resources {S}.

The *i** SD models constructed provide not only an explicit representation of COTS dependencies but also a graphical schema that is helpful to the application of some requirements engineering techniques as interviews, clarifying and unifying concepts from stakeholders and a better understanding of what domain actors expect. Thus, such models may be used as a basis for the domain documentation, discussion, and refinement.

7.4 Activity 5: Goal Taxonomy Structuring

From the technical perspective of enabling the COTS location and retrieval mechanisms, the process of finding classes that help COTS re-users to find them is a crucial task to make the classification schema understandable.

Putting forward an effective classification schema is a two-step process [Han-Kam01]. In the first step, a classification model is built and tuned describing a predetermined set of data classes or concepts; each object is assumed to belong to a predefined class, as determined by one of the attributes, called the class label attribute. In the second step, the model is used for classification of other objects. Typically, the produced classification schema is placed in the form of classification rules, decision trees, or mathematical formulae.

Classical classification approaches related in Chapter 2 resulted greatly static for addressing the organization of the evolvable objects in the marketplace; therefore, they are not adequate to provide a dynamic mechanism to organize in a flexible way our goal-schema aiming to classify COTS related to the domain. As a result, we made use of a hybrid approach involving more dynamic attribute-value assignment and decision trees properties.

In an attribute-value classification, objects are described by a set of attributes and their values. A decision tree is a flow-chart-like tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and leaf nodes represent classes or class distribution. Using both approaches provides a natural and evolvable schema to manage the classification of COTS domain goals based on several of their goal attributes whilst it is possible to manipulate the taxonomy in a formal way in order to be highly reusable. It is, focusing on goals, instead of specific functionalities, allows a more fluent communication with any type of re-users using the taxonomy (i.e., domain experts, non-experts, etc.), and it can be adapted to the required perspective of the intended users as well as to grow and evolve for classifying future COTS.

Although, there are many algorithms for decision tree induction, among we can mention ID3 [Qui86], C4.5 [Qui93] and CART [Bre+84], the hybrid mechanism we used to classify COTS domain goals can be described as:

- To construct the goal-oriented taxonomy, the main goal of the domain that is being analyzed is designed as the root node. It represents the whole population of the COTS domain that is being classified.

- Analyzing the goal information of core system actors and its related functionalities, a set $X = \{x_k\}_n$ of independent variables that characterize the intended taxonomy is obtained. Among all these values, we select the ones that make the “best” partitioning into more homogeneous and usable partitions. They are called “classifiers”, thus goals would be expressed as a combination of values of these classifiers.

The reason why we only consider core system actors goals is because they represent the high-level goals of the domain whilst supporting system actors imply goals that are considered as desirable but secondary or originally intended as parts of other domains. Moreover, information about dependencies is already stated in the artifacts attained to the intended taxonomy.

- *Goal satisfaction* is defined by means of assignment to the variables, therefore for each assignment $ass = (x_1 \leftarrow v_1, \dots, x_n \leftarrow v_n)$, the expression $sat_{ass}(G)$ yields true if the goal G evaluates to true for this assignment, otherwise false.
- Taxonomy nodes are stated following the goal information (e.g., pre- and post-conditions, dependencies, etc.), variables and values assignment. We repeat such process until classifiers can not be split into more nodes covering a significant set of functionalities in the marketplace.

Table 7.8 Excerpt of system actor’s identification & coupling with domain goals

Actor	Goals	Available Functionalities	Classifier	Values
RTSC-S	Provide RTSC infrastructure	<ul style="list-style-type: none"> ➢ Intra-organizational RTSC infrastructure ➢ Internet based infrastructure ➢ WAN infrastructure 	NetInfrastructure	<ul style="list-style-type: none"> ← Intra-organizational ← Internet based ← WAN infrastructure
			ApplicationType	<ul style="list-style-type: none"> ← RTSC-S ← RTSC-C
		<ul style="list-style-type: none"> ➢ One to one communication ➢ Multi-user communication ➢ Text, Audio, Video, data messages ➢ Collaborative production of resources ➢ Sharing Resources 	Infrastructure	<ul style="list-style-type: none"> ← Intra-organizational ← Internet based ← WAN infrastructure
			ConnectedUsersSupported	<ul style="list-style-type: none"> ← Only Two (one to one) ← More than two (multi-user)
			Purpose	<ul style="list-style-type: none"> ← Collaborative production of resources ← Sharing Resources
			ResourcesSupported	<ul style="list-style-type: none"> ← Text, Audio, Video, Data, Multipart.
...				
RTSC-C	Enable interaction among RTSC-S and RTSC-User	<ul style="list-style-type: none"> ➢ Intra-organizational RTSC infrastructure ➢ Internet based infrastructure ➢ WAN infrastructure ➢ Text, Audio, Video, data messages ➢ ... 	Net-Infrastructure	<ul style="list-style-type: none"> ← Intra-organizational ← Internet based ← WAN infrastructure
			ClientArchitecture	<ul style="list-style-type: none"> ← Web-Based ← Install-Based
		ConnectedUsersSupported	<ul style="list-style-type: none"> ← Only Two (one to one) ← More than two (multi-user) 	
		Purpose	<ul style="list-style-type: none"> ← Collaborative production of resources ← Sharing Resources 	
		ResourcesSupported	<ul style="list-style-type: none"> ← Text, Audio, Video, Data, Multipart. 	
		...		

Table 7.8 shows an excerpt of the variables assignment to goals for the RTSC case as well as its variables assignment.

Table 7.9 Excerpt of a possible goal-oriented taxonomy for the RTSC case

Category	Node Name	Abbrev.	Classifier	Satisfaction Values
ROOT	RTSC	RTSC	ApplicationType	←RTSC-S; ←RTSC-C
1.1	RTSC-Server	RTSC-S	Purpose	←Collab. Production of Resources; ←Sharing Resources
1.1.1	Collab. Communication & Dev.of Groupware Application Server	CDGS	ConnectedUsers Supported	←OnlyTwo; ←MoreThan2
1.1.1.1	Multi-User Collaborative Communication Server	MCCS	NetInfrastructure	←Intra-Organizational; ←Internet; ←WAN;
1.1.1.2	One-to-One Collaborative Communication Server	OCCS	NetInfrastructure	←Intra-Organizational; ←Internet; ←WAN
1.1.1.1.1	Intra-organizational Multi-User Collaborative Communication Server	IOMCCS	Resources Supported	← Text; ← Audio; ←Video; ← Data; ←Multipart
	...			
1.1.1.1.2	WAN Multi-User Collaborative Communication Server	WANMCCS	Resources Supported	← Text; ← Audio; ←Video; ← Data;← Multipart
	...			
1.1.1.1.3	Internet Multi-User Collab. Communication Server	IMCCS	Resources Supported	← Text; ← Audio; ←Video; ← Data;← Multipart
1.1.1.1.3.1	Internet Multi-User Collab.Com. Server supporting Text	IMSTx		
1.1.1.1.3.2	Internet Multi-User Collab. Com. Server supporting Audio	IMSAu		
1.1.1.1.3.3	Internet Multi-User Collab. Com. Server supporting Video	IMSVi		
1.1.1.1.3.4	Internet Multi-User Collab. Com. Server supporting Data	IMSDa		
1.1.1.1.3.5	Internet Multi-User Collab. Com. Server supporting Multipart message	IMSMu		
1.1.2	Sharing Com. Application Server	SCAS	ConnectedUsers Supported	←OnlyTwo; ←MoreThan2
1.1.2.1	One-toOne Sharing Com. Server	OSCS	NetInfrastructure	←Intra-Organizational; ←Internet;←WAN;
	...			
1.2	RTSC-Client	RTSC-C	Purpose	←Collab. Production of Resources; ←Sharing Resources
1.2.1	Sharing Com. Application Client	SCAC	ConnectedUsers Supported	←OnlyTwo; ←MoreThan2
	...			
1.2.2	Collaborative Com. and Development Application Client	CCDC	ConnectedUsers Supported	←OnlyTwo; ←MoreThan2
1.2.2.1	Multi-User Collab. Com. Client	MCCC	NetInfrastructure	←Intra-Organizational; ←Internet; ←WAN;
1.2.2.2	One-to-One Collab. Com. Client	OCCC	NetInfrastructure	←Intra-Organizational; ←Internet; ←WAN
	...			
1.2.2.1.1	Intra-organizational Multi-User Collab. Com Client	IOMCCC	Resources Supported	← Text; ← Audio; ←Video; ← Data; ←Multipart
	...			
1.2.2.1.2	WAN Multi-User Collab. Com Client	WANMCCC	Resources Supported	← Text; ← Audio; ←Video; ← Data; ← Multipart
	...			
1.2.2.1.3	Internet Multi-User Collab. Com Client	IMCCC	Resources Supported	← Text; ← Audio; ←Video; ← Data; ←Multipart
1.2.2.1.3.1	Text	IMCTx		
1.2.2.1.3.2	Audio	IMCAu		
1.2.2.1.3.3	Video	IMCVi		
1.2.2.1.3.4	Data	IMCDa		
1.2.2.1.3.5	Multipart	IMCMu		

Table 7.9 is an example of a taxonomy for the RTSC case study and the variables assignment, considering that all the assignments are inherited downwards the hierarchy.

It is important to stand out that it is not our purpose to construct the “best” taxonomy (if any exist) but to guide the taxonomy designer to build one. The level of detail depends on the particular taste of the taxonomy designer and of course on the information needs of the taxonomy users.

In addition, questions and answers are attained to the hierarchy to guide the users through the taxonomy browsing process. We identified subsequently other characterization attributes, their corresponding values, questions, and answers (see Table 7.10).

Table 7.10 Example questions and answers attained to taxonomy nodes

Classifier	Question (s)	Answer(s)
<i>NetInfrastructure</i>	Which is the infrastructure you will use for enabling real time synchronous communication?	<ul style="list-style-type: none"> ▶ Intranet I(Intra-organizational) infrastructure ▶ Internet Infrastructure ▶ WAN Infrastructure
<i>ApplicationType</i>	Are you requiring Client or Server Technology?	<ul style="list-style-type: none"> ▶ Server ▶ Client
<i>ConnectedUsersSupported</i>	How many users do you want to connect at the same time?	<ul style="list-style-type: none"> ▶ One to One ▶ More than 2 users
<i>Purpose</i>	What is the purpose of your system, to collaborate for developing resources or only sharing resources?	<ul style="list-style-type: none"> ▶ Collaborative communication for developing applications ▶ Sharing Resources
<i>Resources Supported</i>	What Kind of resources you need to share or collaborate on?	<ul style="list-style-type: none"> ▶ Text ▶ Audio ▶ Video ▶ Data ▶ Multipart
...		

The questions are applied at different levels in the taxonomy, and some of them are applied in more than one branch.

Fig. 7.6 shows a partial and graphical view of such goal taxonomy structuring and the questions/answers attained.

From the semantic point of view, we distinguish among two types of nodes: categories and market segments. Market segments are the leaves of the taxonomy whilst categories serve to group related market segments and/or subcategories. In order to classify new COTS, their attribute values are tested against the decision tree. A path is traced from the root to a leaf node that holds the class prediction for that component or its related information.

To make the taxonomy easily reusable trustworthy and flexible, based on decision trees properties we defined some rules that help to ensure some taxonomy’s trustworthiness as well as to manipulate the taxonomy nodes as a result of marketplace

evolution of other required views of the information in the repository. This is detailed in Chapter 8.

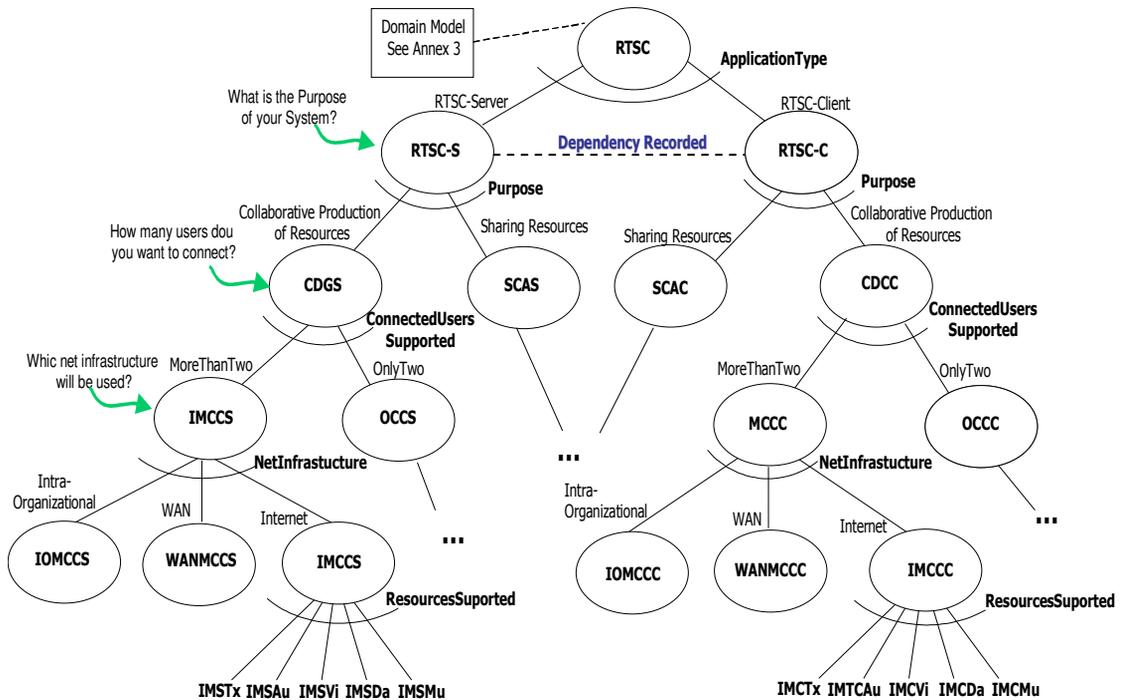


Fig. 7.6 Partial View of the goal-oriented hierarchy for the RTSC case

As it can be observed, the Domain Model introduced in Chapter 6, is attached to the taxonomy. Reusability of such Domain Model downwards categories and market segments of the hierarchies is a way to support this objective. We have observed throughout our experiences that some quality features appear over and over, and this repetition is directly connected to the characteristics embedded in the characterization attributes (i.e. classifiers). The recognition of COTS market segments and categories improves reusability: once a new COTS market segment has been identified, its Domain Model can be constructed by inheriting the features of the models for those COTS categories in the hierarchy which it belongs to. During the process, new categories may be identified, abstracting commonalities of this new domain with others. As a result, a quality model bound to a category of the taxonomy collects all the quality features common to all its sub-categories and market segments. Since then, any quality model for a particular selection process may reuse the Domain Model of the corresponding COTS domain.

7.5 Summary and Discussion

To conclude, it is worth to mention that as the whole goal-oriented core process of GOTHIC is an iterative and incremental process harmonized with all the other method activities. The produced artifacts are in fact most of the artifacts or information used to accomplish the artifacts proposed in Activity 2 (Domain Analysis). For instance, the

scenario models used to identify and refine goals are in fact the artifacts suggested for recording COTS domain functionality. The use of *i** models for understanding the domain are also useful to record COTS dependencies. Also the glossary of terms (that is suggested to be refined throughout the method) is improved by the addition of concepts of the domain coming from the identification of goals and the definition of synonymous.

The most relevant characteristics of our goal-oriented strategy designed for dealing with COTS marketplace evolvability and interoperability are:

- It is based on the application of well known techniques for supporting the COTS search and reuse.
- We have put the emphasis on providing a prescriptive approach for building flexible, abstract, well-founded and stable taxonomies based on goal-oriented approaches.
- We proposed the use of COTS dependency models by adapting the *i** approach to our objectives. Therefore, the semantic of the models obtained provides the rationale for the decisions taken and allows transferring knowledge from one experience to another.
- The use of goal-oriented approaches not only impact positively on dealing with COTS marketplace evolvability and interoperability, but also improves the understanding and manipulation of the taxonomies since goals permit communication among people using a language based on concepts with which they are both comfortable and familiar.

Chapter

8

Activity 6:

Goal Taxonomy Validation and Management

The aim of this activity is to ensure the trustworthiness of the goal-hierarchy obtained in the previous activity (Activity 5), as well as the management of the hierarchy to deal with the marketplace evolution and different representation needs of the information (i.e., different views of the repository by different users). It is done by performing a four-step process of rules application over the goal-hierarchy nodes. To introduce such process, the Chapter has been structured as follows: Section 8.1 introduces the predicates and functions defined to drive the process. Section 8.2 details the definition of the transformation rules and the four-step process used to apply them to goal-hierarchies. Section 8.3 makes clear the goal-oriented GOTHIC approach. Section 8.4 illustrates the approach by means of two examples. Summary and discussions are presented in Section 8.5.

8.1 Predicates and Functions

Our definition of the transformation rules is based on the goal satisfaction described in the previous chapter and stated below:

Given G a boolean predicate defined over variables x_1, \dots, x_n , $G(x_1, \dots, x_n)$, and given ass an assignment of variables defined as $ass = (x_1 \leftarrow v_1, \dots, x_n \leftarrow v_n)$, the expression $sat_{ass}(G)$ is defined as the evaluation of G over ass , $G(v_1, \dots, v_n)$.

Please see Chapter 7 for details of this definition

Whilst this variable assignment is used to provide semantics to the hierarchy nodes, the transformation rules introduced in this chapter are applied over the nodes to ensure a clear and understandable rationale for the classification, as well as the correctness and completeness of the obtained taxonomies.

To introduce the transformation rules we declare that a goal-oriented taxonomy T is a tree over the domain of goals defined over variables x_1, \dots, x_n . As such, we need predicates and functions as shown in Table 8.1

Table 8.1 Predicates and functions over taxonomies

<i>Belongs</i> (T, A): the element A belongs to T
<i>Root</i> (T, A): A is the root of T
<i>Leaf</i> (T, A): A is a leaf in T
<i>Parent</i> (T, A): returns the parent of A in T
<i>Children</i> (T, A): returns the set of children of A in T
<i>Siblings</i> (T, B, C): B and C are siblings in T
<i>Successors</i> (T, A): returns the set of successors of A in T
<i>Ancestors</i> (T, A): returns the set of ancestors of A in T
<i>Goal</i> (A): returns the goal of element A

Throughout the rest of this chapter we use the following predicates on goals with the following meaning and abbreviations:

Table 8.2 Predicates, semantics and abbreviations used over goals

<i>Predicate or function</i>	<i>Semantics</i>	<i>Abbrev</i>
<i>impliesGoal</i> (G, H): the goal G implies the goal H	$\forall \text{ass: } \text{sat}_{\text{ass}}(G) \Rightarrow \text{sat}_{\text{ass}}(H)$	$G \Rightarrow H$
<i>not-impliesGoal</i> (G, H): the goal G does not imply the goal H	$\exists \text{ass: } \neg(\text{sat}_{\text{ass}}(G) \Rightarrow \text{sat}_{\text{ass}}(H))$	$G \not\Rightarrow H$
<i>soft-impliesGoal</i> (G, H): the goal G implies the goal H for some assignment whilst the reverse is not true	$\exists \text{ass: } \text{sat}_{\text{ass}}(G) \Rightarrow \text{sat}_{\text{ass}}(H) \wedge \neg \exists \text{ass: } \text{sat}_{\text{ass}}(H) \Rightarrow \text{sat}_{\text{ass}}(G)$	$G \pm \Rightarrow H$
<i>disjointGoals</i> (G, H): goals G and H are mutually exclusive	$\forall \text{ass: } \neg(\text{sat}_{\text{ass}}(G) \wedge \text{sat}_{\text{ass}}(H))$	$\neg(G \wedge H)$
<i>equivGoals</i> (G, H): goals G and H are equivalent	$\forall \text{ass: } \text{sat}_{\text{ass}}(G) = \text{sat}_{\text{ass}}(H)$	$G \equiv H$
<i>emptyGoal</i> (G): goal G is never satisfied	$\forall \text{ass: } \neg \text{sat}_{\text{ass}}(G)$	$G = \emptyset$
$F = \text{diffGoals}(G, H)$: obtain the difference of goals G and H	$\forall \text{ass: } \text{sat}_{\text{ass}}(G) \wedge \neg \text{sat}_{\text{ass}}(H) \Leftrightarrow \text{sat}_{\text{ass}}(F)$	$F = G - H$
$F = \text{unionGoals}(G, H)$: obtain the union of goals G and H	$\forall \text{ass: } \text{sat}_{\text{ass}}(G) \vee \text{sat}_{\text{ass}}(H) \Leftrightarrow \text{sat}_{\text{ass}}(F)$	$F = G \cup H$
$F = \text{intersectGoals}(G, H)$: obtain the intersection of goals G and H	$\forall \text{ass: } \text{sat}_{\text{ass}}(G) \wedge \text{sat}_{\text{ass}}(H) \Leftrightarrow \text{sat}_{\text{ass}}(F)$	$F = G \cap H$
<i>UnionGoalsExt</i> ($\{G_k\}_n$), <i>intersectGoalsExt</i> ($\{G_k\}_n$) are extensions to a set of goals (used quantified)		

8.1.1 Conditions Over Taxonomies

A goal-oriented taxonomy T is said to be correct and complete if it satisfies the following conditions:

<p>C1. Parent-child correctness. The goal of each node is implied by its parent goal: $\forall X: \text{Belongs}(T, X): [\forall Y: Y \in \text{Children}(T, X): \text{Goal}(X) \Rightarrow \text{Goal}(Y)]$</p> <p>C2. Siblings correctness. The goals of siblings are disjoint: $\forall X, Y: \forall X, Y: \text{Belongs}(T, X) \wedge \text{Belongs}(T, Y) \wedge \text{Siblings}(T, X, Y): \text{Goal}(X) \cap \text{Goal}(Y) = \emptyset$</p> <p>C3. Completeness. The goals of siblings cover altogether the goal of their parent: $\forall X: \text{Belongs}(T, X) \wedge \neg \text{Leaf}(T, X): [\text{Goal}(X) \equiv \cup Y: Y \in \text{Children}(T, X): \text{Goal}(Y)]$</p>
--

C1 ensures that decomposition of software package types is well-formed, which means that satisfaction of the goal of a node is implied by the satisfaction of its parent goal. C2 that the taxonomy provides a unique way for classifying software packages, which means that there is no variable assignment which makes two siblings satisfy their goals simultaneously. C3 that software packages can always be classified using the taxonomy, i.e. that the taxonomy covers all the possible assignment of variables.

Given these correctness and completeness notions, we can define a top-down process for rearranging a goal-oriented taxonomy: first we remove conflicts among parents and children to ensure C1, next we detect and solve conflicts among siblings to ensure C2 and afterwards we complete the taxonomy identifying new nodes that fulfil the parent goal to ensure C3. We add a fourth step, to tailor the result to the particular (and subjective) taste of the designer with respect to level of detail required and organizational concerns. Transformation rules are used in each step to progress towards the result.

8.2 Four-Step Process for Transformation Rules Application

We have designed 11 basic transformation rules (and some variations of them) that are applied into the 4 identified steps. Throughout this process, taxonomy nodes are manipulated in a formal way not only to obtain a correct and complete taxonomy but also to leveraging its nodes to get similar levels of abstraction and/or get ad-hoc representations.

Some of the rules may take slightly different forms, and we have therefore some different variations of the basic idea which are given different names. In addition, some rules may offer intermediate solutions that although could not solve the inconsistency, enable the application of other rule for solving it; this is the concept of combined rule. Combined rules are explicitly outlined from the contract of basic rules, allowing a wide range of action for solving inconsistencies and manipulating taxonomy nodes.

Application of transformation rules is stated by pre- and post-conditions using the predicates and functions on taxonomies and goals introduced in Section 8.1. In postconditions, the expression $x@pre$ stands for the value of x before applying the rule. In each step we assume as invariants the conditions C_i (see Section 8.1.1) that have been ensured in the previous step of the process. In the rules, satisfaction of the invariant C_i on all the nodes of the tree T is denoted by $C_i(T)$. Our style in writing the rules (preconditions, postconditions, and invariants) makes easier the assumption of applicability of the rules and how they preserve the conditions that apply. Last, we get rid of renamings, we assume that every rule has the right to change the name of the involved nodes for legibility.

For reaching the termination process at each step, metrics are defined. Those metrics depend on the step to be applied and are called m_1 , m_2 and m_3 respectively (step 4 does not have a defined termination process). These metrics are related with the number of inconsistencies that violate their corresponding intended state C_i . The stop condition of each step is achieved when the value of its metric remains 0, it means that there are not inconsistencies violating its intended state. Regarding combined rules, it is easy to infer

the effect that they have on their associated metric since they are composed of basic rules. Particular metrics are further discussed in next sections.

Transformation rules application at each step is approached in a top-down way. More than one rule could be applied in some conditions. In cases when more than one rule is relating exactly the same precondition, the use of one or another depends on the postcondition desired. Possible combinations of rules are explicitly defined in terms of pre- and post-conditions that allow the combination. All transformation rules may be intertwined and iterated as required for reaching their corresponding C_i state and their invariants.

At each step, heuristics has been defined to drive the decision-making process of transformation rules application. In general, heuristics at each step are organized to support the analysis of possible solutions for solving taxonomy's inconsistencies violating their corresponding C_i condition by means of transformation rules application. Problem-solving decisions depend on the particular (and subjective) rationale and experience of the designer with respect to the focus of the intended taxonomy and the level of detail required (i.e., the post-condition desired). Heuristics provide insights about the effect of the decisions taken over the structure of the taxonomy being analyzed (i.e., whether the application of a specific rule affects the actual structure of the hierarchy or not) and the kind of rules that may be applied in any case (i.e., basic or combined). The set of basic and derived transformation rules and their rationale are described in next sections.

Fig. 8.1 provides a graphical summary of the consequences that rule application has on the taxonomy.

8.2.1 Transformation Rules in Step 1

For Step 1, the intended state is $C1$:

$C1$. *Parent-child correctness*. The goal of each node is implied by its parent goal:
 $\forall X: \text{Belongs}(T, X): [\forall Y: Y \in \text{Children}(T, X): \text{Goal}(X) \Rightarrow \text{Goal}(Y)]$

$C1$ ensures that decomposition of software package types is well-formed, which means that satisfaction of the goal of a node is implied by the satisfaction of its parent goal.

A node B is violating the $C1$ condition when its goal is not implied by its parent's goal:

$$\exists B: B \in \text{children}(T, A): \text{Goal}(A) \not\Rightarrow \text{Goal}(B)$$

To solve these inconsistencies, some basic and also derived transformation rules may be applied. Two basic possibilities exist with respect to the current structure of the classification schema:

- To change the structure of the current taxonomy by deleting the node B . The *Removal* rule was designed with this aim.

R1. Removal (T, A, B) Removes the node B (and its successors) from the taxonomy T .
 ► *Postcondition*: $\neg \text{Belongs}(T, B) \wedge \forall Y: Y \in \text{Successors}(T, B@pre): \neg \text{Belongs}(T, Y)$

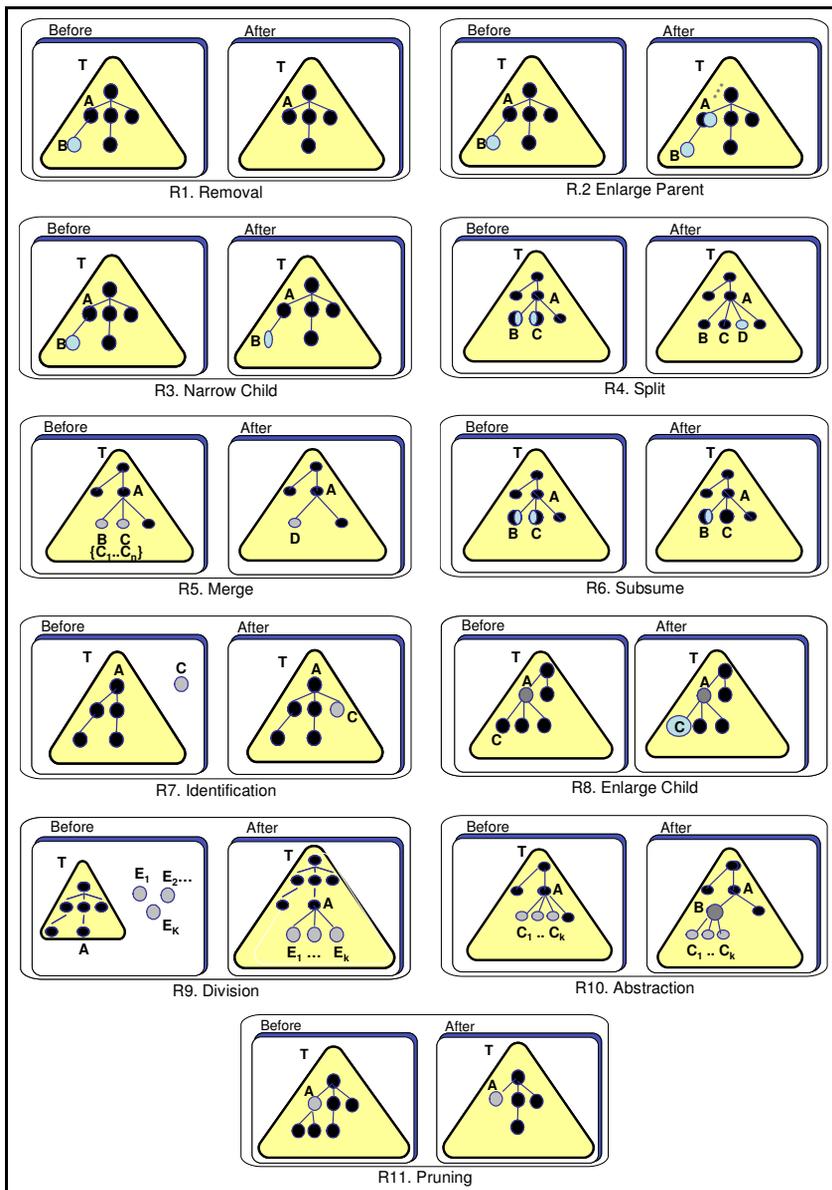


Fig. 8.1 Set of Transformation Rules

In this step we consider a special case of the general Removal rule, called *Hard Removal* (for distinguishing it from another type of removal introduced later) that is applied when the node is not related neither to its parent goal.

R1.1 Hard Removal (T, A, B) Removes the node B (and its successors) from the taxonomy T because its goal is not implied by its current parent goal A.

- ▶ *Precondition:* $\text{Belongs}(T, A) \wedge \text{Belongs}(T, B) \wedge A = \text{Parent}(T, B) \wedge \text{Goal}(A) \not\Rightarrow \text{Goal}(B)$
- ▶ *Postcondition:* $\neg \text{Belongs}(T, B) \wedge \forall Y: Y \in \text{Successors}(T, B@pre) : \neg \text{Belongs}(T, Y)$

- To preserve the current structure of the taxonomy and only rearrange conflicting nodes to avoid violation of C1. So, two different possibilities exist:
 1. To enlarge A's goal to capture the part of B's goal that is not implied by A's. This enlargement must be propagated to all A's ancestors. This enlargement does not cause any conflict violating C1 in higher levels of the tree. The *Enlarge Parent* rule was envisaged with this aim.

R2. Enlarge Parent (T, A, E, B) Enlarges A's goal with the predicate E to capture some part (maybe all) of B's goal that is not implied by A.

► *Precondition:* $\text{Belongs}(T, A) \wedge \text{Belongs}(T, B) \wedge A = \text{Parent}(T, B) \wedge \text{Goal}(A) \not\Rightarrow \text{Goal}(B) \wedge \text{Goal}(A) \Rightarrow E \wedge (E - \text{Goal}(A)) \pm \Rightarrow \text{Goal}(B)$

► *Postcondition:* $\text{Goal}(A) = \text{Goal}(A@pre) \cup E \wedge \forall Y: Y \in \text{Ancestors}(T@pre, A): \text{Goal}(Y) = \text{Goal}(Y@pre) \cup E$

2. If B's goal is partially implied by A's goal, to narrow B's goal to discard some part (maybe all) of its goal that is not implied by A's goal. This narrowing must be propagated to all B's successors.

R3. Narrow Child (T, A, N, B) Narrows B's goal to discard some part N (maybe all) of the goal that is not implied by A's goal.

► *Precondition:* $\text{Belongs}(T, A) \wedge \text{Belongs}(T, B) \wedge A = \text{Parent}(T, B) \wedge \text{Goal}(A) \pm \Rightarrow \text{Goal}(B) \wedge \text{Goal}(B) \Rightarrow N \wedge (\text{Goal}(A) \cap N) = \emptyset$

► *Postcondition:* $\text{Goal}(B) = \text{Goal}(B@pre) - N \wedge \forall Y: Y \in \text{Successors}(T@pre, B): \text{Goal}(Y) = \text{Goal}(Y@pre) - N$

8.2.1.1 Heuristics, Combined Rules and Stop Condition for Step 1

Fig. 8.2 summarizes the heuristics driving the application of transformation rules in Step 1. Application of basic and combined rules is explicitly stated. Decisions about their application depend on the rationale and needs of the taxonomy designer.

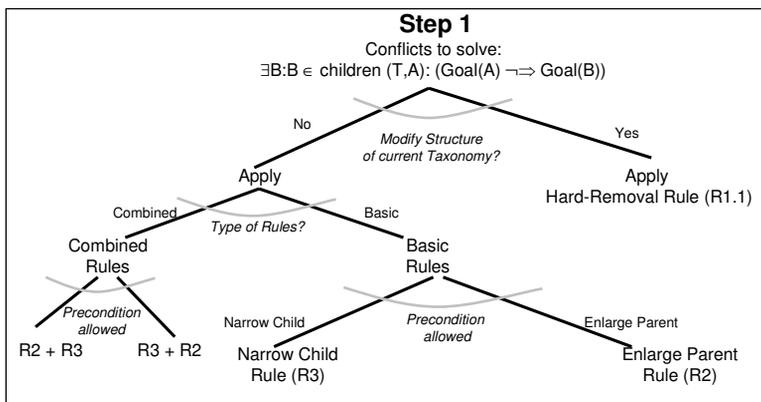


Fig. 8.2 Heuristics driving transformation rules application in Step 1

Please note that using R1.1 rule always implies to solve the intended inconsistency (since the node B that is causing the inconsistency is deleted), whilst R2 and R3 may

allow total or partial solutions depending on the values of the Enlargement (E) and Narrowing (N) respectively. In cases when partial solutions are applied, although the application of one rule may not solve the inconsistency, other rule(s) may be applied as allowed by their pre-conditions to completely solve the inconsistency as stated in Table 8.3. For terminating the Step 1 process, the metric mI counts the number of parent-child pairs that violate the goal implication rule C1. Table 8.3 also denotes the potential results obtained after the application of each basic rule and their effect on mI .

Table 8.3 Effect of transformation rules on the stop condition for Step 1

Rules Applied in Step 1	Completely solves the inconsistency	Effect on the stop condition metric mI	Rule(s) that may be applied to solve the remaining inconsistency	Combination of rules allowed
R1.1 Hard- Removal	Yes	-1	-	
R2. Enlarge Parent	Yes	-1	-	
	No, partially	None	R3	R2 + R3
R3. Narrow Child	Yes	-1	-	
	No, partially	None	R2	R3 + R2

It can be observed that depending on whether the basic rules R2 or R3 completely solve the inconsistency; they decrement by 1 the value of mI . In cases when basic rules do not completely solve the inconsistency but partial solutions are chosen, the value of mI is not modified. Thus, the stop condition for terminating the Step 1 process is achieved when all stored values of mI remain 0; it means that there are no inconsistencies violating C1. It is important to remark then that the metric mI is a non-increasing function.

8.2.2 Transformation Rules in Step 2

Step 2 is intended to reach the C2 condition.

C2. Siblings correctness. The goals of siblings are disjoint:
 $\forall X, Y: \text{Belongs}(T, X) \wedge \text{Belongs}(T, Y) \wedge \text{Siblings}(T, X, Y): \text{Goal}(X) \cap \text{Goal}(Y) = \emptyset$

C2 aims at providing a unique way for classifying software packages, which means that there will not be any variable assignment making two siblings satisfy their goals simultaneously.

C2 is violated when some children B, C of the same parent (i.e., siblings) are not disjoint:

$$\exists B, C: \text{Belongs}(T, B) \wedge \text{Belongs}(T, C) \wedge \text{Siblings}(T, B, C): \text{Goal}(B) \cap \text{Goal}(C) \neq \emptyset$$

In all the cases, since we are in Step 2 and inconsistencies among parents and children have been removed in the previous step, we assume C1(T) as invariant.

To solve C2 inconsistencies, four basic possibilities exist:

- Extracting the common part of the conflicting nodes and making a new node with it. This solution implies a slightly difficult contract, because we must make sure that C1(T) reached in the previous step is not violated after applying the rule. This could happen if for some child of any of the involved nodes, its goal is not totally implied

neither by its current parent after the operation, nor by the new node. In other words, the goal of the child could be partially covered by the old and new nodes.

To reach this contract, the Split rule was formulated:

<p>R4. Split (T, B, C) Extracts the common part of the siblings B and C and creates a new node D with it, which in turn is sibling of B and C. It requires to reallocate B's and C's children to ensure keeping C1(T). To achieve this last, the function <i>reallocateNodes</i> is used.</p> <ul style="list-style-type: none"> ▶ <i>Let:</i> $A = \text{Parent}(T, B); G_D = \text{Goal}(B) \cap \text{Goal}(C)$ ▶ <i>Precondition:</i> $\text{Belongs}(T, B) \wedge \text{Belongs}(T, C) \wedge B \neq C \wedge \text{Siblings}(T, B, C) \wedge \text{Goal}(B) \cap \text{Goal}(C) \neq \emptyset \wedge (\text{Goal}(A) \Rightarrow \text{Goal}(B)) \wedge (\text{Goal}(A) \Rightarrow \text{Goal}(C))$ ▶ <i>Postcondition:</i> $\text{Belongs}(T, D) \wedge \text{Siblings}(T, B, D) \wedge \text{Goal}(D) = G_D \wedge \text{Goal}(B) = \text{Goal}(B@pre) - G_D \wedge \text{Goal}(C) = \text{Goal}(C@pre) - G_D \wedge \forall Z: Z \in \text{Children}(T, B@pre): \text{reallocateNodes}(T, B@pre, Z, D) \wedge \forall Z: Z \in \text{Children}(T, C@pre): \text{reallocateNodes}(T, C@pre, Z, D)$

The function **reallocateNodes**(T, X, Z, D) reallocate the node Z as a child of X or D depending on its goal satisfaction properties in order to ensure keeping C1(T). It is a recursive function that in turns reallocate the children of Z (if they exist) and successively.

The function is defined to perform 3 different actions in the tree different situations that may appear:

- ▶ X implies Z:
 $[\text{Goal}(X) \Rightarrow \text{Goal}(Z@pre)]: \text{Parent}(T, Z) = X \wedge \text{Goal}(Z) = \text{Goal}(Z@pre)$
- ▶ D implies Z:
 $[\text{Goal}(D) \Rightarrow \text{Goal}(Z@pre)]: \text{Parent}(T, Z) = D \wedge \text{Goal}(Z) = \text{Goal}(Z@pre)$
- ▶ Otherwise:
 $\neg \text{Belongs}(T, Z) \wedge \text{Belongs}(T, Z_X) \wedge \text{Belongs}(T, Z_D) \wedge \text{Parent}(T, Z_X) = X \wedge \text{Parent}(T, Z_D) = D \wedge \text{Goal}(Z_X) = \text{Goal}(Z@pre) \cap \text{Goal}(X) \wedge \text{Goal}(Z_D) = \text{Goal}(Z@pre) \cap \text{Goal}(D) \wedge \forall H: H \in \text{Children}(T, Z@pre): \text{reallocateNodes}(T, Z_X, H, Z_D)$

- To remove one of the nodes that is causing the conflict (and its successors). In this case, we apply the *Soft-Removal* rule that is a particular case of the general Removal rule introduced in the previous step with a different precondition to be applied.

<p>R1.2 Soft-Removal (T, B, C) Removes the node B (and its successors) from the taxonomy T.</p> <ul style="list-style-type: none"> ▶ <i>Let:</i> $A = \text{Parent}(T, B)$ ▶ <i>Precondition:</i> $\text{Belongs}(T, B) \wedge \text{Belongs}(T, C) \wedge \text{Siblings}(T, B, C) \wedge \text{Goal}(B) \cap \text{Goal}(C) \neq \emptyset \wedge (\text{Goal}(A) \Rightarrow \text{Goal}(B)) \wedge (\text{Goal}(A) \Rightarrow \text{Goal}(C))$ ▶ <i>Postcondition:</i> $\neg \text{Belongs}(T, B) \wedge \forall Y: Y \in \text{Successors}(T@pre, B): \neg \text{Belongs}(T, Y)$

- To merge the conflicting nodes into a new one, which goal will be the union of their goals, applying a particular case of the Merge rule, called *Inclusive Merge*. Moreover, the children of the merged nodes become children of the new one.

The Merge rule performs this solution.

R5. Merge (T, B, C) Merges siblings nodes B and C into a new one called D. The new node's goal is the union of the merged siblings'. The children of the merged nodes B and C become children of the new node.

- ▶ *Precondition:* $\text{Belongs}(T, B) \wedge \text{Belongs}(T, C) \wedge \text{Siblings}(T, B, C)$
- ▶ *Postcondition:* $\neg \text{Belongs}(T, B) \wedge \neg \text{Belongs}(T, C) \wedge$
 $\text{Belongs}(T, D) \wedge \text{Parent}(T, D) = \text{Parent}(T, B@pre) \wedge$
 $\text{Goal}(D) = \text{Goal}(B@pre) \cup \text{Goal}(C@pre) \wedge$
 $\forall Y: Y \in (\text{Children}(T@pre, B) \cup \text{Children}(T@pre, C)): \text{Parent}(T, Y) = D$

R5.1 Inclusive-Merge (T, B, C) The merged nodes had overlapping goals. It is important to stand out that the children of the merged nodes become siblings and so new conflicts may arise, but they are dealt later in Step 2 due to the top-down analysis of the hierarchy.

- ▶ *Let:* $A = \text{Parent}(T, B)$
- ▶ *Precondition:* $\text{Belongs}(T, B) \wedge \text{Belongs}(T, C) \wedge \text{Siblings}(T, B, C) \wedge$
 $\text{Goal}(B) \cap \text{Goal}(C) \neq \emptyset \wedge$
 $(\text{Goal}(A) \Rightarrow \text{Goal}(B)) \wedge (\text{Goal}(A) \Rightarrow \text{Goal}(C))$
- ▶ *Postcondition:* $\neg \text{Belongs}(T, B) \wedge \neg \text{Belongs}(T, C) \wedge$
 $\text{Belongs}(T, D) \wedge \text{Parent}(T, D) = \text{Parent}(T, B@pre) \wedge$
 $\text{Goal}(D) = \text{Goal}(B@pre) \cup \text{Goal}(C@pre) \wedge$
 $\forall Y: Y \in (\text{Children}(T@pre, B) \cup \text{Children}(T@pre, C)): \text{Parent}(T, Y) = D$

- To subsume the joint part of the nodes into one of them. This solution also implies to make sure that $C1(T)$ is not violated after applying the rule. This could happen if for any child of the narrowed node, its goal is not implied by its parent after the subsume procedure. Thus, the children nodes of the narrowed node that are not implied by the new narrowed goal should be moved totally or partially as children of the enlarged node. If moved just partially, the node is in fact split into two. We may use again the *reallocateNodes* function defined above with this aim. Subsequently, since these reallocated nodes become siblings of the enlarged node's children (if any exist), new conflicts may arise. However, they are also dealt later in the top-down process of Step 2.

The rule driving this solution is called *Subsume*.

R6. Subsume (T, B, C) Subsumes the joint part of B and C into B.

- ▶ *Let:* $A = \text{Parent}(T, B); G_{BC} = \text{Goal}(B) \cap \text{Goal}(C)$
- ▶ *Precondition:* $\text{Belongs}(T, B) \wedge \text{Belongs}(T, C) \wedge \text{Siblings}(T, B, C) \wedge$
 $\text{Goal}(B) \cap \text{Goal}(C) \neq \emptyset \wedge$
 $(\text{Goal}(A) \Rightarrow \text{Goal}(B)) \wedge (\text{Goal}(A) \Rightarrow \text{Goal}(C))$
- ▶ *Postcondition:* $\text{Goal}(B) = \text{Goal}(B@pre) \cup G_{BC} \wedge \text{Goal}(C) = \text{Goal}(C@pre) - G_{BC} \wedge$
 $\forall Z: Z \in \text{Children}(T, C@pre): \text{reallocateNodes}(T, C@pre, Z, B)$

8.2.2.1 Heuristics, Combined Rules and Stop Condition for Step 2

Heuristics driving the application of transformation rules in Step 2 are summarized in Fig. 8.3. To solve each C2 violation, two main options exist the first one implies to preserve the structure of the taxonomy being evaluated by applying the *Subsume* rule; the second one modifies the actual taxonomy structure whilst solving the problem by applying basic or combined rules. The taxonomy designer is in charge of the decision of which rule to apply.

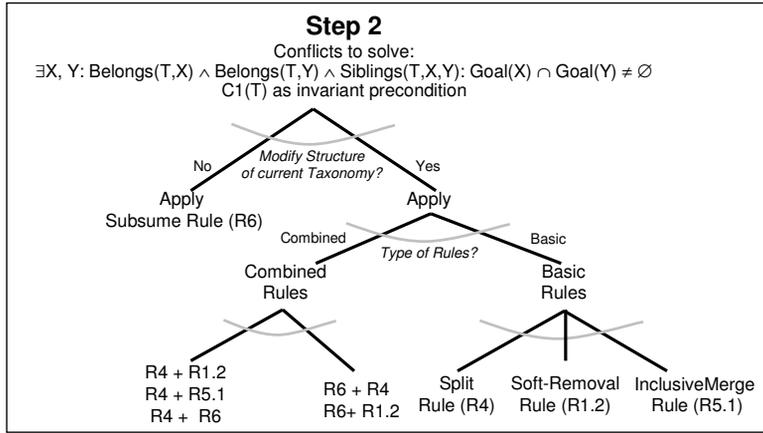


Fig. 8.3 Heuristics driving transformation rules in Step 2

As stated in the rules definition above, sometimes, as a result of their application, more conflicts among children than the one that was being solved may appear. Hence, to reach a stop condition of the step (i.e., C2 is completely satisfied by all taxonomy nodes) we need a more elaborated stop condition metric in which not just the number of inconsistencies but also the position at the tree (more precisely, the level) are taken into account.

Thus, to define the metric $m2$, let N be the arity of the tree defined as the maximum number of siblings that exist in any part of the hierarchy:

$$N = \max X: \text{Belongs}(T, X): \text{sizeOf}(\text{Children}(T, X))$$

$\text{height}(T)$ the height of the tree, and $\text{level}(T, X)$ a function that gives the level that occupies the node X in the tree T .

Then, we define $m2$ as:

$$\sum X, Y: \text{Belongs}(T, X) \wedge \text{Belongs}(T, Y) \wedge \text{Siblings}(T, X, Y): \\ C2\text{-inconsistency}(X, Y) \times N^{2(\text{height}(T) - \text{level}(T, X))}$$

being $C2\text{-inconsistency}$ a function evaluating to 1 if X and Y 's goals are not disjoint, 0 else:

$$C2\text{-inconsistency}(X, Y) = \begin{cases} 1, & \text{Goal}(B) \cap \text{Goal}(C) \neq \emptyset \\ 0, & \text{Goal}(B) \cap \text{Goal}(C) = \emptyset \end{cases}$$

Table 8.4 shows the possibilities that each rule has to solve the inconsistency, as well as the potential combination of rules. The stop condition in this Step is achieved when $m2$ remain 0.

Table 8.4 Effect of transformation rules on the stop condition for Step 2

Rules Applied in Step 2	Completely Solves the Inconsistency	Rule(s) that may be applied to solve the remaining inconsistency	Combination of Rules allowed
R4. Split	Yes	-	-
	No	R1.2 R5.1 R6	R4 + R1.2 R4 + R5.1 R4 + R6
R1.2. Soft-Removal	Yes	-	-
R5.1. Inclusive Merge	Yes	-	-
R6. Subsume	Yes	-	-
	No	R4 R1.2	R6 + R4 R6 + R1.2

8.2.3 Transformation Rules in Step 3

Step 3 is addressed to reach the condition C3.

C3. Completeness. The goals of siblings cover altogether the goal of their parent:
 $\forall X: \text{Belongs}(T, X) \wedge \neg \text{Leaf}(T, X): [\text{Goal}(X) \equiv \cup Y: Y \in \text{Children}(T, X): \text{Goal}(Y)]$

C3 aims at ensuring the completeness of taxonomies, it means that software packages can always be classified using the taxonomy, i.e. that the taxonomy covers all the possible assignment of variables. Therefore, C3 is violated when the decomposition of a node is such that its children do not cover altogether its goal:

$$\exists A: \text{belongs}(T, A) \wedge \neg \text{Leaf}(T, A): [\text{Goal}(A) \neq \cup B: B \in \text{Children}(T, A): \text{Goal}(B)]$$

To solve this kind of inconsistency, 3 possibilities exist:

- To create new node(s) covering the part of goal left. The new node's goal must not violate the states C1 and C2 that have been ensured in the two previous steps.

R7. Identification (T, A, E) Inserts a new node C satisfying goal E as child of an existing one A whose goal is not fully covered by its children's goals. The new node's goal must not violate neither C1 nor C2.

► *Precondition:* $\text{Belongs}(T, A) \wedge \neg \text{Leaf}(T, A) \wedge [\text{Goal}(A) \neq \cup B: B \in \text{Children}(T, A): \text{Goal}(B)]$
 $\wedge \text{Goal}(A) \Rightarrow E \wedge [\forall B: B \in \text{Children}(T, A): \text{Goal}(B) \cap E = \emptyset] \wedge$
 $[\forall B: B \in \text{Children}(T, A): \text{Goal}(A) \Rightarrow \text{Goal}(B)] \wedge$
 $[\forall B_1, B_2: B_1 \in \text{Children}(T, A) \wedge B_2 \in \text{Children}(T, A) \wedge B_1 \neq B_2:$
 $\text{Goal}(B_1) \cap \text{Goal}(B_2) = \emptyset]$

► *Postcondition:* $\text{Belongs}(T, C) \wedge \text{Parent}(T, C) = A \wedge \text{Goal}(C) = \text{Goal}(A) \cup E$

- To enlarge the goal of a child in order to cover the goal of its parent. Therefore new completeness problems may arise in lower levels of the tree, but they are solved later during Step 3.

R8. Enlarge Child (T, A, C, E) Enlarges C’s goal to cover its parent A’s goal when it is not fully covered by its children’s goals. The enlargement is denoted by E and it does not violate neither C1(T) nor C2(T).

► *Precondition:* $\text{Belongs}(T, A) \wedge \neg \text{Leaf}(T, A) \wedge \text{Parent}(T, C) = A \wedge$
 $[\text{Goal}(A) \neq \cup B: B \in \text{Children}(T, A): \text{Goal}(B)] \wedge$
 $\text{Goal}(A) \Rightarrow E \wedge [\forall B: B \in \text{Children}(T, A) \wedge B \neq C: \text{Goal}(B) \cap E = \emptyset] \wedge$
 $[\forall B: B \in \text{Children}(T, A): \text{Goal}(A) \Rightarrow \text{Goal}(B)] \wedge$
 $[\forall B_1, B_2: B_1 \in \text{Children}(T, A) \wedge B_2 \in \text{Children}(T, A) \wedge B_1 \neq B_2:$
 $\text{Goal}(B_1) \cap \text{Goal}(B_2) = \emptyset]$

► *Postcondition:* $\text{Goal}(C) = \text{Goal}(C@pre) \cup E$

- To narrow the goal of the parent in order to discard the part of the goal that is not implied by its children. This case was not considered as a rule, because it implies several violations to C1 that are impractical to be managed in Step 3.

8.2.3.1 Heuristics, Combined Rules and Stop Condition for Step 3

Heuristics driving the application of transformation rules in Step 3 are summarized in Fig. 8.4.

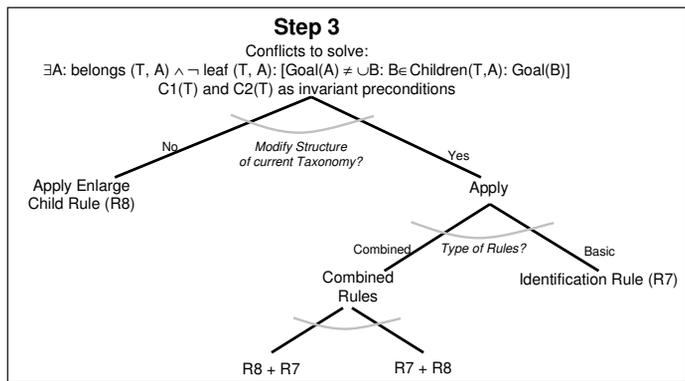


Fig. 8.4 Heuristics driving transformation rules application in Step 3

The metric $m3$ is defined as the number of nodes such that their goal is not entirely cover by their children. To reach the C3 satisfaction state pursued in Step 3, all conflicts violating C3 must be solved, whilst C1 and C2 are ensured as invariants. The stop condition is attained when $m3$ remains 0. Table 8.5 shows the potential postconditions of the rules and their effect on $m3$, possible combinations of rules and their applicability condition are also shown.

Table 8.5 Effect of transformation rules on the stop condition for Step 3

Rules Applied in Step 3	Completely Solves the Inconsistency	Effect on the stop condition metric $m3$	Rule(s) that may be applied to solve the remaining inconsistency	Combination of Rules allowed
R7. Identification	Yes	-1	-	-
	No	None	R8	R7 + R8
R8. Enlarge Child	Yes	-1	-	-
	No	None	R7	R8 + R7

It is obvious that transformation rules at this step can be applied and iterated more than once to solve an inconsistency.

8.2.4 Transformation Rules in Step 4

Step 4 has not a state to be ensured, we only consider as invariants the conditions of previous phases $C1(T)$, $C2(T)$, and $C3(T)$ that must be preserved after any manipulation. In other words, Step 4 just restructures the taxonomy once it has been proven correct and complete with the aim of leveraging the incoming taxonomy.

The rules in this step are oriented to tailor the result to the particular (and subjective) taste of the designer with respect to the level of detail desired. Such that it is easy to infer that the application of any rule on this step only rearranges the hierarchy respecting the previous steps' conditions.

The rationale for changing the form of a correct and complete taxonomy is:

- A leaf A is too abstract (i.e., its attained goal is too coarse-grained, mixing different concepts) and should be decomposed into n nodes satisfying goals E_1, \dots, E_n , to add detail. Thus, the *Division* rule should be applied.

R9. Division ($T, A, \{E_k\}_n$). Breaks a node A into several descendants. Relationships among the new nodes' goals and the divided node's goal shall ensure that $C1$, $C2$ and $C3$ are preserved.

- ▶ *Precondition:* $\text{Belongs}(T, A) \wedge \text{Leaf}(T, A) \wedge \forall k: 1 \leq k \leq n: \text{Goal}(A) \Rightarrow E_k \wedge C1(T) \wedge C2(T) \wedge C3(T)$
- ▶ *Postcondition:* $\forall k: 1 \leq k \leq n: [\text{Belongs}(T, E_k) \wedge \text{Parent}(T, E_k) = A \wedge \text{Goal}(E_k) = E_k]$

- The conceptual gap among a node A and some of its children is too wide, resulting in a too flat hierarchy, and an intermediate node with a new goal must be introduced using the *Abstraction* rule.

R10. Abstraction ($T, A, \{C_k\}_n, B$). Creates a new node B as parent of a set of existing ones $\{C_k\}_n$ that are children of A . The new node B becomes child of A . The new node's goal is the union of the goals of the nodes in the set. The children of the merged nodes become children of the new goal.

- ▶ *Precondition:* $\text{Belongs}(T, A) \wedge \forall k: 1 \leq k \leq n: [\text{Belongs}(T, C_k) \wedge \text{Parent}(T, C_k) = A] \wedge C1(T) \wedge C2(T) \wedge C3(T)$
- ▶ *Postcondition:* $\text{Belongs}(T, B) \wedge \text{Parent}(T, B) = A \wedge \text{Children}(T, B) = \{C_k\}_n \wedge \text{Goal}(B) \equiv \cup_k: 1 \leq k \leq n: \text{Goal}(C_k)$

- The children of a node A do not really add value to the taxonomy, therefore, two choices are valid:

1. To remove all children of A and their successors applying the *Pruning* rule

R11. Pruning (T, A) Eliminates the children (and all its successors) of an intermediate node A .

- ▶ *Precondition:* $\text{Belongs}(T, A) \wedge C1(T) \wedge C2(T) \wedge C3(T)$
- ▶ *Postcondition:* $\neg \text{Belongs}(T, A) \wedge \forall Y: Y \in \text{Successors}(T, A@pre): \neg \text{Belongs}(T, Y)$

2. To merge children into a coarse-grained one. The Merge rule variant used in this Step is called *Non-Inclusive Merge*, a weaker version of the merge of Step 2).

R5.2 Non-Inclusive Merge (T, A, {C_k}_n) Merges several children of an intermediate node A into a new one called D that represents a coarse-grained node.

- ▶ *Precondition:* $\text{Belongs}(T, A) \wedge \neg\text{Leaf}(T, A) \wedge$
 $\forall k: 1 \leq k \leq n: [\text{Belongs}(T, C_k) \wedge \text{Parent}(T, C_k) = A \wedge$
 $C1(T) \wedge C2(T) \wedge C3(T)]$
- ▶ *Postcondition:* $\text{Belongs}(T, D) \wedge \text{Parent}(T, D) = A \wedge \text{Goal}(D) \equiv \cup k: 1 \leq k \leq n:$
 $\text{Goal}(C_k@pre) \wedge$
 $\forall k: 1 \leq k \leq n: \neg\text{Belongs}(T, C_k)$

Step 4 has not an explicit condition to be achieved. The C1(T), C2(T), and C3(T) invariant conditions are ensured by the own rules preconditions. Therefore, Step 4 does not need any termination condition; the step may finish at any moment.

All the rules designed for Step 4 may be used as required according to the applicable preconditions. Heuristics driving the applications of these rules are shown in Fig. 8.5.

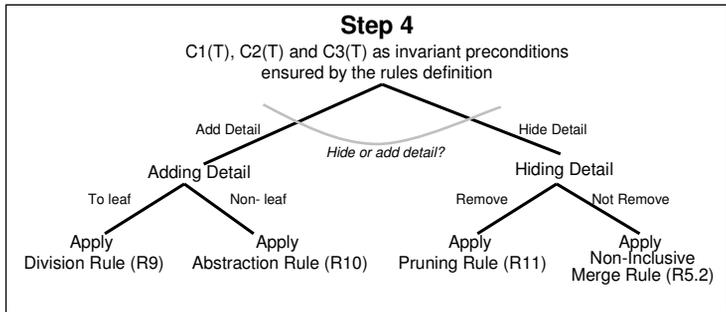


Fig. 8.5 Heuristics driving transformation rules application in Step 4

Table 8.6 presents the transformation rules and their variants according to the step in which they can be used.

Table 8.6 Transformation rules and their applicability to the 4-steps process

Transformation Rule	Step 1	Step 2	Step3	Step 4
R1. Removal	R1.1	R1.2		
R2. Enlarge Parent	✓			
R3. Narrow Child	✓			
R4. Split		✓		
R5. Merge		R5.1		R5.2
R6. Subsume		✓		
R7. Identification			✓	
R8. Enlarge Child			✓	
R9. Division				✓
R10. Abstraction				✓
R11. Pruning				✓

8.3 Goal-Oriented Taxonomies Formulation, Evaluation and Management

In the last two chapters we have presented a proposal for building goal-oriented taxonomies that arrange the existing myriad of COTS into categories and market segments. The process for building goal-oriented taxonomies (presented in Chapter 7) is characterized to be goal-oriented, and domain-independent.

In this chapter, we have precisely defined rules and their associated processes for validating the obtained taxonomies and ensure their manipulation. It endorses the different required views of the information and/or the taxonomy evolution as a result of COTS marketplace change and management.

The goal-oriented approach detailed in Chapter 7 and the four-step process of transformation rules application detailed in this Chapter are complementary. They have been integrated into the GOTHIC method as the main core for validating and manipulating the rationale of classification schemas (i.e., taxonomies) in which the reuse-infrastructure is based on. The GOTHIC metamodel regarding this issue is depicted in Fig. 8.6.

As it is further described in Chapter 7, the *Goal Satisfaction* assignment is defined as the process for giving *Values* to a set of *Variables* that characterize specific *Goals*. Goals at their turn characterize *Taxonomy Nodes*. Also dependencies among goals are recorded by using *i** constructors (i.e., Goal, Softgoal, Task, Resource). Transformation rules introduced in this Chapter may be defined and applied in the described four-step process in terms of these class elements. The conceptual model depicted in Fig. 8.6 is a further refinement of the GOTHIC conceptual model introduced in Fig. 4.2 in Chapter 4.

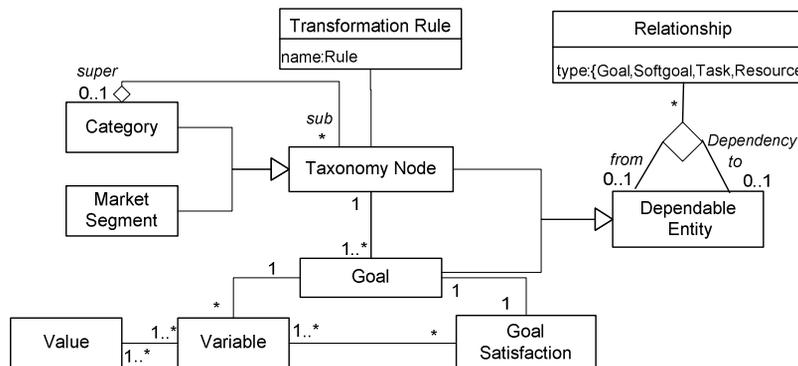


Fig. 8.6 Conceptual model of the goal-oriented core of GOTHIC

8.4 Applying the Goal-Oriented Taxonomy Validation and Management Process

Our proposed goal taxonomy validation and management approach resulted useful to validate and manage a goal-oriented taxonomy fully obtained by applying our GOTHIC approach. However, since reuse is one of the main motivations for this work, we also

consider the extreme case of applying it for restructuring any existing ‘ad-hoc’ classification hierarchy (as those related in Section 2.3.2.).

To demonstrate both facts we present the applicability of the rules to two different examples:

- The first one is related to the goal-oriented hierarchy obtained for the RTSC case that has been developed in the previous chapters.
- The second one refers to an existing ‘ad-hoc’ hierarchy proposed by Gartner [Gar] in the Business Application (BA) domain.

In both cases, we only expose the process of rearranging the taxonomy based on the transformation rules, assuming that the processes of identification, refinement and statement of goals, as well as establishment of dependencies and goal-taxonomy structuring have been previously performed.

8.4.1 Validating and Manipulating the Goal-Oriented Taxonomy Obtained for the RTSC Case Study

In this section, the process of validating and manipulating the RTSC taxonomy obtained in the previous chapters is illustrated.

The RTSC taxonomy introduced in Table 7.9 shows a goal classification schema obtained by following the GOTHIC method activities. However, to ensure its trustworthiness and appropriate evolution with respect to the marketplace changes and different representation needs of the information (i.e., different views of the repository by different users), the four step process described above are applied.

We focus on the branch related to *Collaborative Communication and Development of Groupware Applications (CDGS)*.

The steps followed and their related results are:

- ▶ *Step 1.* Any inconsistency was found. Given the intensive top-down process of goal identification and refinement followed throughout the activities 3, 4 and 5 of GOTHIC, this kind of inconsistencies are not very common.
- ▶ *Step 2.* As the same case of the previous step, no inconsistencies were found.
- ▶ *Step 3.* Some inconsistencies were found, and were solved by the application of fitting transformation rules to complete the parent goal. Domain knowledge is required to drive this process. In this case the incompleteness was given by the fact that a node that covers all existing functionalities was missed.
- ▶ *Step 4.* Some changes were done to the goal hierarchy in order to tailor the result to our particular purposes.

Fig. 8.7 shows the related branch of the taxonomy after each of the 4 steps.

Table 8.7 summarizes the goal-based reasoning and transformation rules application in each step of our example. The relationships among goals that violate some condition are stated in the second column. The other two columns state the rule that is applied at every moment.

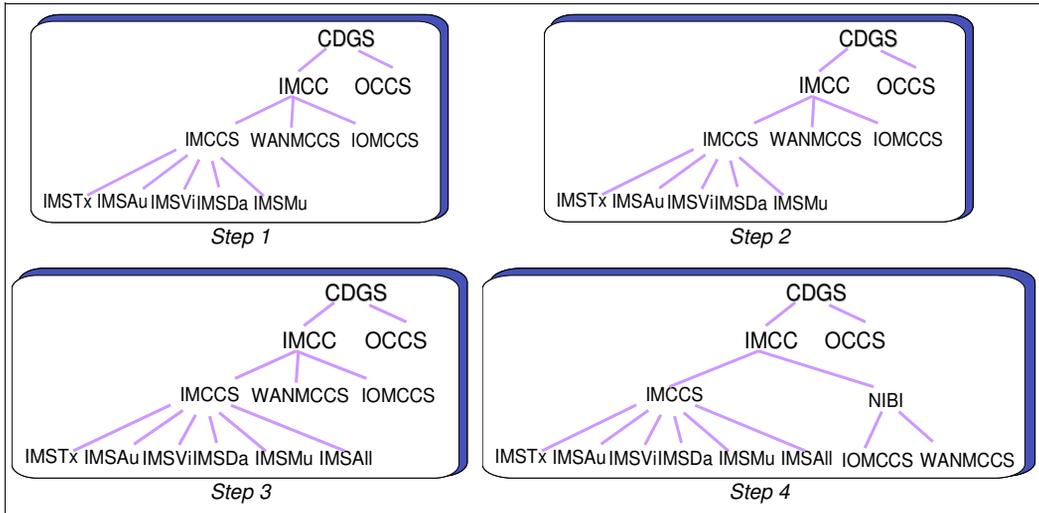


Fig. 8.7 The 4-Step goal-taxonomy validation process for the RTSC case

Table 8.7 Validating the RTSC goal-classification schema

Steps	Goal Reasoning	Rule Applied
Step 1	No inconsistencies	None
Step 2	No inconsistencies	None
Step 3	$\neg (G(\text{IMCCS}) \equiv (G(\text{IMSTx}) \cup G(\text{IMSAu}) \cup G(\text{IMSVi}) \cup G(\text{IMSDa}) \cup G(\text{IMSMu})))$ Left Goal= Support of All Resources $G(\text{IMSAll})$	R7 Identification(RTSC, IMCC, IMAll)
Step 4	Not-Interned Based Infrastructure (NIBI)	R10 Abstraction (RTSC, MCC, $\{(IOMCCS), (WANMCCS)\}$, (NIBI))

Results obtained confirm our supposition that given the top-down process of identification and refinement of goals followed in previous activities of the GOTHIC method, the application of transformation rules in the first steps was not really extensive (in this case, no rules were applied in steps 1 and 2). However, the existence of all these steps helps to ensure the trustworthiness and completeness of the hierarchy (e.g. the existence of a new goal that was not discovered before was identified), whilst step 4 ensure its suitable manipulation.

8.4.2 Validating and Manipulating an Existing COTS Classification Schema

All the aspects of the daily operations of small, medium and large organizations, either private companies or public administrations, heavily depend on the existence of adequate software products to undertake crucial tasks, such as accounting, human resources management, document administration, team work, people communication, business processes monitoring, etc. Therefore, having specific means for discovering which BAs satisfy the needs of an organization is utterly convenient in order to select

the most suitable. Given the relevance and popular use of the BA tools belonging to this market segment, we decided to use it as example.

To choose a BA ‘ad-hoc’ hierarchy, we investigated the way that professional software consultant companies organize the BAs’ services that they offer to their customers. We selected the classification proposed by the Gartner Consulting [Gar]. It is application-oriented and well-suited for our purposes, compared to others whose classification is based on business areas or that include not only software but other assets.

As it is obvious, since such hierarchy it is not goal-oriented, we have to perform a process of discovering goals in the departing classification hierarchy to subsequently apply the described four-step process of for validating and manipulating goal-oriented taxonomies. Such process is summarized in Table 8.8.

Table 8.8 Discovering goals process for an existing ‘ad-hoc’ classification hierarchy

Discovering Goals in the Departing Classification Hierarchy	
Input	The existing classification hierarchy
Output	The existing classification hierarchy with goals attached to its nodes (i.e a goal-oriented hierarchy)
Process	<ul style="list-style-type: none"> – Goal elicitation methods (as those described in the Activity 3 and 4 in Chapter 7) are applied to discover the goals behind each node of the existing taxonomy. – Subsequently, identified goals would be expressed as a combination of the values of their intended classifiers, as described in Activity 5 in Chapter 7).

A first observation of this process is that in most cases we noticed that although some node of the original taxonomy was partitioned into some categories, we couldn’t find an attribute that could be used to support it. It means such taxonomy do not really provide a structured for classifying assets.

To simplify the explanation of our existing taxonomy restructuring approach, we will only focus on a particular part of the original Gartner BA classification, the subtree bound to Supply Chain Management as depicted in Fig. 8.8.

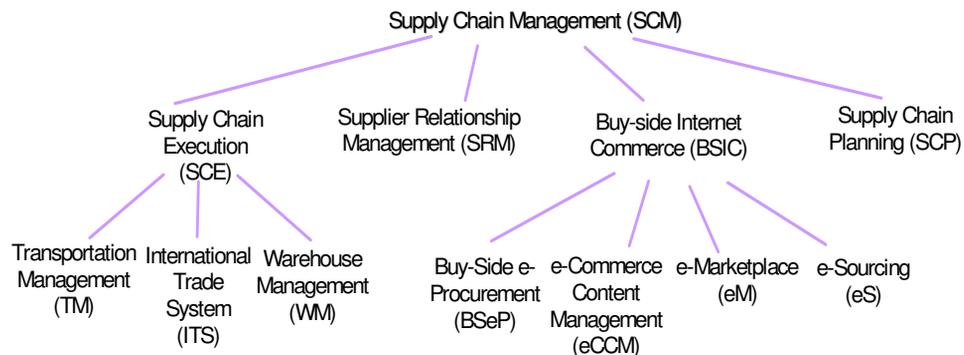


Fig. 8.8 An excerpt of the BA Gartner classification

Furthermore, in this example, we leave one of its nodes (Supply Chain Planning) out of our description.

Table 8.9 shows the elicited goals of the nodes that we are going to manage in the process. Taking into account the form of the tree and the attached goals, Table 8.10 summarizes the goal-based reasoning and transformation rules applied in each step of our example. The relationships among goals that violate some condition are stated in the second column. The other two columns state the rule that is applied at every moment.

Table 8.9 Goals bound to some nodes of the BA taxonomy

Node Names	Goal Attained
Supply Chain Management (SCM)	To encompass the process of creating and fulfilling demands of the market for goods and services, mainly in 2 categories: execution and planning.
Supply Chain Execution (SCE)	To manage relationships with the supplier (sourcing and procurement), and manufacture and logistics aspects.
Transportation Management (TM)	To manage all freight deliver activities across the enterprise.
International Trade Systems (ITS)	To automate the import/export business process.
Warehouse Management (WM)	To manage the operation of a warehouse or distribution center.
Supplier Relationship Management (SRM)	To manage enterprise interactions with the organizations that supply goods and services used.
Buy-Side Internet Commerce (BSIC)	To manage the internet by-side commerce.
eCommerce Content Management (eCCM)	To address how to take unstructured product content and create and manage structured data on internet.

Table 8.10 Transforming the Gartner classification into a goal taxonomy

Steps	Goal Reasoning	Rule Applied	
Step 1	$\neg G(\text{SCM}) \Rightarrow G(\text{BSIC})$	R1.1	Hard Removal(BA, SCM, BSIC)
	$\neg G(\text{SCE}) \Rightarrow G(\text{ITS})$	R1.1	Hard Removal(BA, SCE, ITS)
Step 2	$G(\text{SCE}) \cap G(\text{SRM}) \neq \emptyset$	R5.1	Inclusive Merge(BA, SCE, SRM)
Step 3	$\neg (G(\text{SCE}) \equiv G(\text{WM}) \cup G(\text{TM}))$ Left goals: e-Procurement (eP) eSourcing (eS) Manufacturing Management (MM), Not-eProcurement(NeP) Not-eSource (NeS)	R7	Identification(BA, SCE, eP)
		R7	Identification(BA, SCE, eS)
		R7	Identification(BA, SCE, MM)
		R7	Identification(BA, SCE, NeP)
		R7	Identification(BA, SCE, NeS)
Step 4	Logistics Management (LM) Sourcing (S) Procurement (P) Supplier Interactions(SI)	R10	Abstraction (BA, SCE, {WM, DM}, LM)
		R10	Abstraction(BA, SCE, {eS, NeS}, S)
		R10	Abstraction(BA, SCE, {eP, NeP}, P)
		R10	Abstraction(BA, SCE, {S, P}, SI)

Fig. 8.9 shows the taxonomy after each of the 4 steps.

- ▶ *Step 1.* We found several inconsistencies at highest levels given by the fact that the taxonomy does not have a clear rationale behind. As mentioned above, in several cases it was difficult to find an attribute that can be used for partitioning the nodes. As a result, an appropriate one was defined and the inconsistencies were calculated. Of course, to discern about the relationships among goals, knowledge on the domain was required. Thus, for instance we discover that the goal of SCE is not oriented to automate business processes, such that we can infer that $G(\text{SCE})$ does not imply $G(\text{ITS})$.
- ▶ *Step 2.* Some intersections among goal values were solved.
- ▶ *Step 3.* Several goals were missing (or were belonging to other nodes that were not the adequate ones, so they were removed from them in previous steps). Thus, several missing goals were identified to complete the node's goals.
- ▶ *Step 4.* To tailor the resulting taxonomy to most of the purposes we elicited from experts in the field, we introduced several intermediate categories.

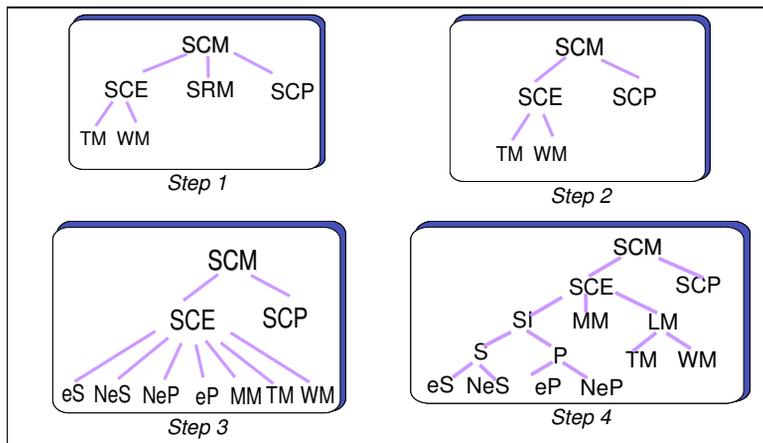


Fig. 8.9 The 4-step goal-taxonomy manipulation process for the BA case

Finally, we also identified questions to elucidate the attribute values of the resulting goal-oriented taxonomy.

As a result of this process we rearranged the whole Gartner BA taxonomy. It originally had 96 nodes, 78 were leafs (i.e., types of software packages) and just 18 intermediate nodes. The longest path from the root to a leaf was of length 6 (just one, and two of length 5), whilst the maximum width was 10 siblings (average width 4,33).

Our process resulted in a taxonomy of 188 nodes, 120 of them leafs. The tree changed its form, with a longest path of length 8 (in fact, 16 paths of this length and 21 of length 7) and 7 siblings at most (average width 3,18).

A group of some COTS selectors related with selecting BA components were asked about the usability of both taxonomies (i.e. the original Gartner BA taxonomy and the goal-oriented obtained).

From these results, we may say that not only the rationale of the original taxonomy was greatly improved, but also it resulted well-suited for most of the purposes of the interviewees, with better defined ways to reach a leaf (i.e., a type of COTS) from the root having fewer alternatives to consider at each step.

8.5 Summary and Discussion

We have identified 11 basic transformation rules specified by contracts that are used and combined along a 4-step process to validate and manipulate goal-oriented taxonomies.

As a result, by the application of the rules:

- We obtain a high-quality taxonomy in which the rationale for the classification is very clear and correctness and completeness are ensured by construction. As far as we know, this rationale distinguishes our approach of other taxonomy proposals as the ones mentioned in Chapter 2.
- Goal-oriented taxonomies may be tailored to the particular (and subjective) taste of the designer and/or their intended use with respect to the level of detail.

Case studies of different size demonstrate the feasibility of the approach. Moreover, it was also demonstrated how the process of goal-oriented COTS taxonomies construction and the associated rules is helpful not only for constructing high-quality COTS domain goal-oriented taxonomies from the scratch but also for transforming existing taxonomies or hierarchies into goal-oriented taxonomies. This last has demonstrated how to profit the knowledge of an existing classification to adapt it to another particular target with a high-quality and high probability of success avoiding efforts.

Some problems observed in most of the existing COTS taxonomies (as those mentioned in Chapter 2) has been addressed. For instance: Categories whose reason-to-be was not clear; categories that were not defined precisely (as a consequence, their granularity was not always adequate); categories that overlapped; the criteria for decomposing categories was never declared and often it was not evident, making hard the use of the hierarchy; levels of abstraction were different at different parts of the hierarchy; and so on.

Finally, it is worth to remark that:

- The proposed transformation rules have been successfully used in our experiences, without needing any additional rule to be defined.
- The goal-oriented taxonomies obtained by GOTHIC are not claimed to be the best (if any exist) but only one solution obtained by the application of transformation rules for tailoring the classification schema to ad-hoc needs and/or COTS marketplace evolution.
- Instead of having a static taxonomy, the proposed process may help to make them dynamic. It means that at any moment, any sound attribute (i.e., set of answers) could be applied to discard some categories and select others. Selection of a category after giving a value to an attribute restricts the set of sound attributes during the browsing process.

- Taxonomies constructed with GOTHIC help to drive a COTS selection process. The selection process will be based on the next three main activities, in addition to other usual ones:
 - ▶ Browsing the taxonomy for locating the COTS market segment of interest.
 - ▶ Analyzing the recorded dependencies for understanding the implications of the selection.
 - ▶ Refining the domain model attained, in order to be used in the evaluation of components.

Chapter

9

Activity 7:

Knowledge Base Management

In the previous chapters, the GOTHIC method has been presented as a method composed of several activities and their deliverables intended to drive the construction of flexible goal-oriented COTS domain taxonomies as a base of a reuse infrastructure from the scratch. This was intentionally done to simplify both the explanation and the understanding of the approach. However, the method has been envisaged following the Experience Factory (EF) paradigm introduced by Basili et al [Ba+94] in order to enable the Learning Software Organization (LSO) approach proposed by Ruhe [Ruh01] in the context of CBSD.

The method aims at improving the return on investment of the taxonomy construction process by building and maintaining a repository of the knowledge obtained during the taxonomy construction process itself as well as the reuse of the experiences and knowledge gained in each selection process where the taxonomy is applied. All the artifacts produced by the method's activities are packaged to support knowledge reuse and evolution, as explained in the Chapter 6 devoted to the *Domain Analysis* activity.

As a result, the flexible COTS domain reuse infrastructure obtained by the method learns from its own development and improves its content later in its usage cycle by reusing the experiences and knowledge gained in each selection process it is applied.

In this chapter, we discuss the GOTHIC method's knowledge base as resembling the Basili's Experience Factory approach and envisaging the Learning Software

Organization appliance. Section 9.1 introduces the EF and LSO paradigms. Section 9.2 relates how the GOTHIC method was conceived to resemble them. Section 9.3 details some contexts of use of the GOTHIC's knowledge base. In Section 9.4 a strategy for populating and maintaining the GOTHIC's knowledge base is discussed. On the other hand, in Section 9.5 the use of software tool support for managing some GOTHIC activities and their deliverables is presented, some of them were developed as a direct consequence of this thesis work. Finally, the chapter concludes with summary and conclusions in Section 9.6.

9.1 The Experience Factory (EF) and Learning Software Organization (LSO) Paradigms

The area of software development is characterized by a particularly rapid technological change. While software is of paramount importance for market success in all high-tech and service domains, software engineering practice does not yet live up to the challenge of effectively reusing software artifacts neither the knowledge gained during their development. Thus, the need for further development of software engineering reuse practices within companies adds to the demand for systematic knowledge management.

According to [Birk+98], software development has several characteristics that distinguish it from "classical" production disciplines:

- Software is developed, not manufactured or produced. Most development techniques cannot be automated. They are human-based, and thus rely on the individual's expert knowledge and skill.
- Each software process and product is different in the terms of goals and contexts. Therefore, the software discipline is inherently experimental. This means that we constantly gain experience from development projects, and strive to provide it for reuse in future, yet different projects. Currently, there is a lack of explicit models and processes, products and other relevant aspects of software projects that can be effectively and efficiently reused.
- Packaging knowledge and experience so as to enhance its reuse potential requires additional resources outside the normal project budget and environments.

In the Software Engineering domain, the idea of reusing software artifacts whilst sharing experiences was initiated about 20 years ago with a more scope and strong emphasis on explicit knowledge. It has gained widespread attention as the so-called *Experience Factory* paradigm [Bas+94b]

The *Experience Factory* paradigm was introduced by Basili et al [Bas+94b]. According to them, *improving the software process and product requires the continual accumulation of evaluated experiences (learning) in a form that can be effectively understood and modified (experience models) into a repository of integrated experience models (experience base) that can be accessed and modified to meet the needs of the current project (reuse)*". The experience base is a logical and/or physical infrastructure aimed at the storage and reuse of all sorts of knowledge (experience and products) resulting from the activities performed in software lifecycle.

The simple *Experience Factory* paradigm concept is that software development projects can improve their performance (in terms of cost, quality, and schedule) by leveraging experience from previous projects.

The concept also takes into account the reality that managing this experience is not trivial and cannot be left to individual projects [Bas-Sea02]. With deadlines, high expectations for quality and productivity, and challenging technical issues, most development projects cannot devote the necessary resources to making their experience available for reuse.

The introduction of the Experience Factory Organization (EFO) solves these problems by separating these responsibilities into two different organizations: the Experience Factory (EF) and the Project Organization (PO). The PO uses packaged experiences and information to deliver software products by calling on the EF's resources for the information, help, and guidance for choosing appropriate processes for the project. On the other hand, the EF extracts information that the PO provides, and installs it into a long-lived experience base. The EF uses this experience base to pass information back from previous projects to the PO and saves the new information for future projects. Thus, the EFO is an organizational unit that supports several software projects. Fig. 9.1 depicts the EF paradigm.

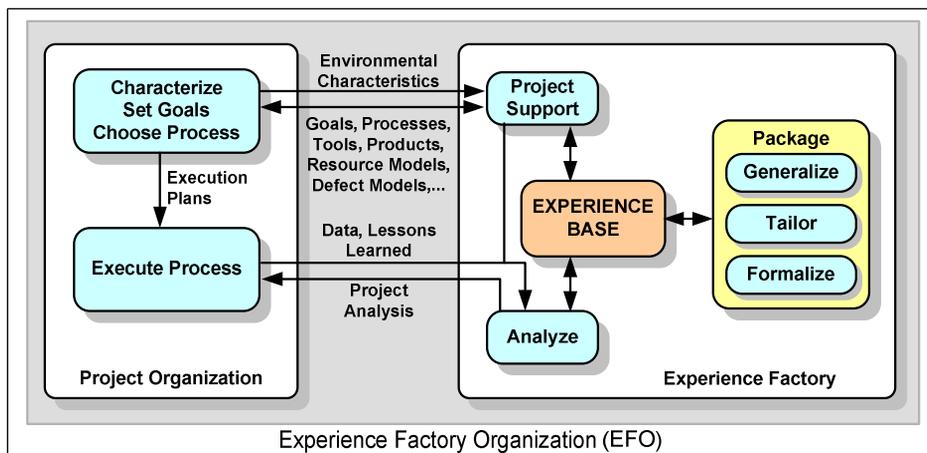


Fig. 9.1 The Experience Factory paradigm

Despite there is a substantial number of Experience Factory implementations, in the form of knowledge repositories, in recent years, the current business needs have made evident the importance of knowledge management into them.

Nowadays, knowledge is recognized as the crucial resource in all complex, human-based (creative) and fast changing business areas. It is the mandatory requisite for gaining and maintaining a competitive advantage and therefore requires continuous, proactive and goal-oriented management. With continuous technological change, globalization, business reorganizations, e-migration, strategic and operational decisions concerning products, processes, technologies or tools and other resources; there is a continuous shortage of the right knowledge at the right place at the right time [Moh+04].

This has made necessary to enhance the knowledge management approach introduced by the Experience Factory paradigm. Therefore, in the last years the *Learning Software Organization* (LSO) paradigm extends the knowledge management approach of the Experience Factory to further deal with knowledge management issues into organizations. The LSO as described by Ruhe [Ruh01] is a continuous endeavour of actively identifying (discovering), evaluating, securing (documenting), disseminating, and systematically deploying knowledge throughout the software development organization. In order to becoming and remain competitive in software development, there is no alternative to become a LSO. Software companies are more and more striving to implement LSO according to the Experience Factory paradigm. In the context of businesses changing ever faster, the challenges are:

- To accelerate the learning processes supported by the LSO
- To extend the scope of knowledge management to all directly and non-directly relevant processes, products, methods and techniques, tools and behaviours in the context of software development, and
- To extend the knowledge management approach of the Experience Factory paradigm to also handle the tacit knowledge within an organization.

9.2 The GOTHIC's Knowledge Base as an EF + LSO

In several works, the crucial need of having a repository for COTS with enough information about them has been recurrently claimed (from the seminal paper of Kontio [Kon96] to more modern proposals as [Mor+00], [Cla+04], [Moh+04], [Wan-Hom06]).

In addition, the need of reusing COTS information within companies demands systematic knowledge and skill management in combination with active usage of this knowledge to support decision-making at all stages of the CBSD lifecycle [Ruh02], [Moh+04].

As a result, the GOTHIC method was envisaged following the EF paradigm and aimed to promote the LSO approach in order to foster the reuse of components information and knowledge through organizations involved in CBSD.

The existence of the repository obtained from the GOTHIC method and the way in which it interacts with the activities of the method itself, largely resembles the Experience Factory paradigm whilst its structure and artifacts are aimed to promote knowledge management at various levels as pursued by the LSO paradigm as depicted in Fig. 9.2.

The reuse infrastructure obtained from the GOTHIC method represents the "*Experience Base*" (EB). This EB base makes the knowledge gathered in past experiences available for the support of new CBSD projects. Each CBSD project needs are supported by existing packaged experiences and information available in the Experience Base. Thus, appropriate plans of action and decisions are taken and executed. Experiences and new information obtained from this executed plans are at they turn analyzed and introduced into the Experience Base.

The Experience Base is composed by the set of deliverables produced by each one of the GOTHIC method's activities. These activities produce the artifacts and feedback mechanisms required to communicate and support the Experience Base evolution and maintenance (performed by the EF), as well as the systematic learning and packaging of reusable experiences from diverse CBSD projects (performed by the PO). Similarly to the EF and LSO paradigms, we do not think about a static and complete Experience Base, but in artifacts that continuously grow to include new reusable elements.

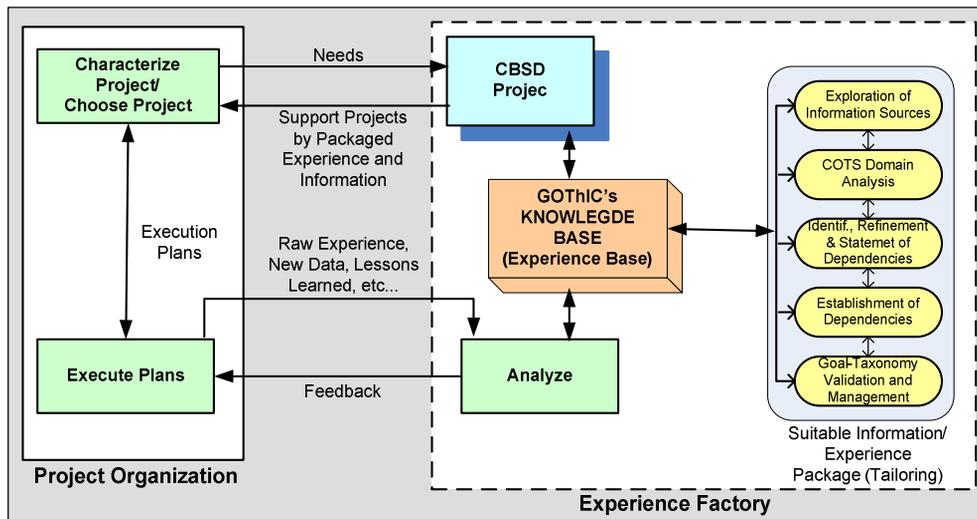


Fig. 9.2 GOTHIC experience base and knowledge reuse artifacts

In GOTHIC's cyclic approach the activities of the method interact with the Experience Base at different levels, either to gather the specific knowledge relevant for them, or to update or extend the knowledge stored with new deliverables or updating the existing ones. The way in which information is packaged (detailed in Chapter 6) enables the potential mechanisms to foster LSO issues in the context of CBSD organizations.

It is important to stand out that the diverse mechanisms required for implementing the LSO paradigm in CDSB organizations are out of the scope of this thesis. It only focuses on encouraging the packaging of knowledge/information required to achieve it.

9.3 The Context of Use of the GOTHIC's Knowledge Base

The use of GOTHIC is mainly addressed to organizations that usually carry out COTS selection processes and find valuable to accumulate experiences from past selection processes in order to improve their practice. This will make future COTS procurement processes for a specific domain much easier, confident, faster, and cheaper.

These companies may range from industrial companies with their own IT department, IT consultant companies offering assessment for business automation to commercial web-based companies or portals selling COTS or offering COTS selection support services.

In general, these organizations may use the method to structure, record, manage, and reuse better their COTS related knowledge. In this context, we can say that the knowledge base obtained by the GOTHIC method can be used at different levels; the next items represent some of them:

- **Intra-organizational level:** It can be constructed and maintained as a centralized database by a certain software organization that intensively performs CBSD in specific domains. This enables the effective communication and reuse of the COTS selection knowledge inside the company (even if they have diverse software development teams around the world). Companies involved in the development of critical software applications may find this schema greatly useful to improve their confidence on their own selection processes.
- **Extra-organizational level:** It can be constructed and used as a centralized database between different organizations involved in the development of software in the same domain having an agreement to submit trustworthy COTS related information to the knowledge base.
- **Marketable level:** It can be constructed and maintained by an independent organization which sells CBSD support through a website. This organization should be responsible for testing and ensuring the accuracy of any information regarding products.

From the intended formative evaluation of GOTHIC, we further studied and assessed the intended contexts of use of GOTHIC repositories in industrial environments (detailed results are described in Chapter 10).

The main concerns found were regarding the heavy upstart cost and difficulty to maintain complete and up-to-date COTS information. All interviewed organizations considered important that knowledge management and reuse initiatives for supporting CBSD in software organizations should be lightweight (i.e. without putting considerable additional burden on developers and end users), and should allow for an incremental adoption (i.e., not requiring large up-front investment before any return on investment is at least visible). This is because companies are rarely willing to invest resources in any kind of experience repository that is in danger of becoming an "experience cementery".

Furthermore, our results show that large-size organizations are more able to afford the creation of centralized organizational units for organizational learning. However, small and medium sized enterprises are not able, in the general case to adopt the use of repositories in intra-organizational levels due to its implementation costs.

In this context, since small and medium enterprises represent a great deal of the European industry, we argue that they must be supported by more lightweight means of creating these knowledge bases with minimal overhead, especially in trendy domains where new technologies constantly emerge. To overcome these industrial issues we envisaged a strategy that is currently being implemented. Next section further details it.

9.4 GOTHIC's Knowledge Base Population and Maintenance

To overcome the issues found for populating and maintaining the GOTHIC knowledge base, we designed a population and maintenance strategy that makes use of the creative

and productive potential of “open-source collaboration” [Aya+07]. In this way, the COTS consumers or (re)users (i.e., individuals, organizations, academic researchers, industrials) can be harnessed to work as a community dedicated to incrementally build and maintain an open reuse infrastructure (i.e., knowledge base) built with GOTHIC. This enables all kind of COTS re-users (i.e., COTS consumers) to store, share, get and re-use COTS related information whilst ensuring smooth start-up and maintenance cost, as well as highly reliable information.

To put forward this strategy we exploit the potential offered by a Wiki-based portal. A Wiki (from the Hawaiian Wikiwiki meaning “fast”) is a collaboratively created and iteratively improved set of web pages [Wag04]. It is considered a powerful knowledge management tool that enables the creation of an incrementally growing system containing the shared knowledge of multiple sources in a centralized infrastructure/repository (i.e. a database server, an application server that runs the Wiki software, and a web server that serves the pages and facilitates the web-based interaction). Thus, exploiting some particular Wiki characteristics (based on the principles described by Wagner [Wag04]) we have designed a proof-of-concept prototype called OTS-Wiki portal [Aas-Lar07]. It is jointly performed with the Norwegian University Of Science and Technology (NTNU).

An important issue to mention is that as we are using the open source collaboration paradigm, we also include Open Source Software (OSS) information into the portal¹. Of course some changes to the domain model produced by GOTHIC –described in Chapter 6- are being tackled in order to embrace all OSS informational dimensions or characteristics into it. However, as it is not part of the main objective of this thesis, these issues will not be described here, we will only focus on the COTS knowledge base issues.

Therefore, regarding COTS, the intended main goals are:

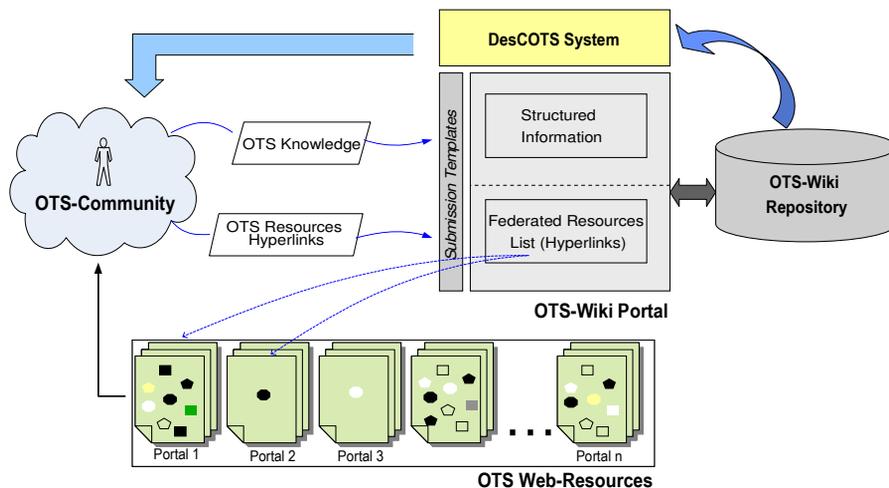


Fig. 9.3 OTS-Wiki Portal Main Interactions

¹ This is the reason why the OTS-Wiki prototype focuses on Off-The-Shelf software components (OTS), that is a term that includes COTS and OSS.

Fostering a COTS Community and Incremental Population of Content. The OTS-Wiki provides the web-based infrastructure for enabling COTS technology (re)users to collaborate as a community in an open-source-like environment, see Fig. 9.3.

Thus, COTS Community users are able, and even encouraged, to share knowledge (e.g., experiences, components information, and vendor comments). Therefore, the incremental population of content in the portal based on the COTS Community participation is expected. We have designed proper templates and guidelines for editing and use in order to share the information in a structured way (as demonstrated in the Wikipedia, an on-line encyclopaedia implemented as a Wiki).

Federating Actual Efforts for Locating and Selecting COTS. In this collaborative environment, COTS Community users are encouraged to add (as hyperlinks) and comment the helpfulness of existing web-resources for locating COTS (as those cited in Table 2.9, called COTS Web-Resources in Fig. 9.3). This is a way of having an up-to date federated list of actual web-resources that the COTS Community users can exploit (artifacts from Activity 1 of the GOTHIC method are improved).

Besides the obvious advantage of using hyperlinks for allowing users to make connections and to drill down into detailed knowledge, hyperlinks are also a potential quality assurance mechanism and relevance indicator. Pages with many links to them indicate a highly useful page. This factor fosters the OTS-Wiki portal to act as a meta-portal for promoting the progressive homogenization of the information contained in different COTS web resources. This is because such resources have an interest of being perceived as highly useful by the COTS Community users.

Enabling Systematic Support for Selecting and Evaluating COTS. Having structured COTS information as claimed by GOTHIC enables systematic support for evaluating and choosing components. We are integrating the DesCOTS system (introduced below) into the *OTS-Wiki*, as stated in Fig. 9.3. It includes a set of tools that interoperate to support the whole COTS selection process. Nevertheless, some other existing or new tools can be developed or designed for using the structured COTS component information from the *OTS-Wiki* portal.

In this scenario, any COTS Community user can use the OTS-Wiki portal as a meta-portal for providing support to:

- f) Searching COTS and information about them supported by a well-defined and dynamic taxonomy.
- g) Recording component information in a structured way supported by the information reuse strategy detailed in Chapter 5 and a common metadata (i.e. domain model from the domain analysis strategy)
- h) Maintaining and reusing such information by the use of suitable and evolvable models capturing all the COTS informational dimensions.
- i) Getting tool support for performing selection processes (e.g., DesCOTS).

Fig. 9.4 provides a i^* SD model that graphically summarizes the main relationships expected by putting forward the strategy.

In [Aya+07] this strategy is detailed and expected usage scenarios are provided in [Aas-Lar07]; some of them are also explained in Section 9.5.2.1.

It is worth to mention that the further exploration of this approach is considered as future work and is intended to be a multidisciplinary European project involving several academy and industrial partners as the Norwegian University of Science and Technology, Politecnico di Torino, Centre de Recherche Public Henri Tudor and Hewlett Packard among others.

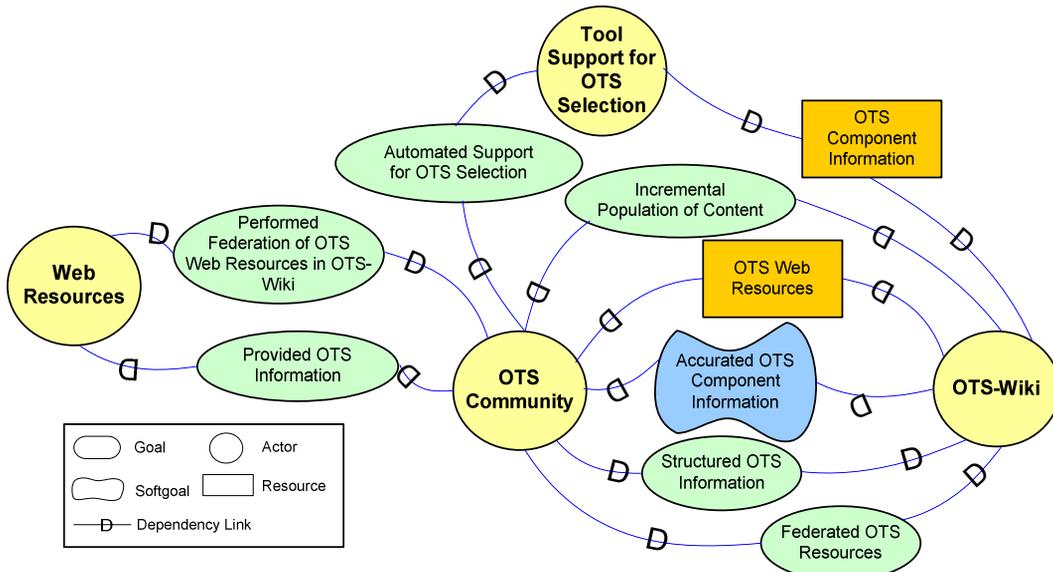


Fig. 9.4 *i** SD model summarizing the GOTHIC population strategy

Our aim is to augment the GOTHIC metamodel with web-intelligence technologies to improve its functionalities. Web intelligence will be applied to analyze user logs of web-search queries, query responses, component choices, and all kinds of solicited comments and reviews – and from these build up, up-date and maintain revised and pragmatic taxonomies (ontologies) in an incremental way among other functionalities.

9.5 Software Tools Supporting Some GOTHIC's Activities and their Deliverables

To perform and manage some of the GOTHIC method activities and their deliverables (i.e., the *knowledge base*) it seems required providing software tool-support.

Although it is not part of the deliverables of this thesis, next sections briefly describe some tools used to support some GOTHIC method activities.

Section 9.5.1 introduces some tools that were previously developed in the context of our GESSI group and were greatly used in throughout our research work since they largely fit into our goals.

Section 9.5.2 relates some tools whose construction was partially guided or bespoke to enclose the concepts presented in this thesis.

9.5.1 Existing Software Tools Supporting GOTHIC's Activities

9.5.1.1 DesCOTS System

Our GESSI research group has previously developed a system called DesCOTS (**D**escription, **e**valuation and **s**election of **COTS** components) [Gra+04]. It was conceived as a set of tools interacting to provide support in the different activities of the process of selecting COTS. This includes both functional and non-functional aspects of the system.

As the research of the GESSI group in the context of COTS selection is progressing, more tools and functionalities have been added to the system. All functionalities are the result of the different needs of our research. In general all our COTS selection related research lines follow a core principle and made use of several of these tools.

The selection process followed builds upon the principle of comparing user requirements with evaluations of COTS, with a focus on quality requirements. Quality Models are then constructed of several quality factors. These factors may differ significantly between the different COTS domains, and therefore it is important to find the quality factors that best suit for the specific description task. Currently, DesCOTS is composed by the following subsystems:

- ▶ **The *Quality Model Tool*** (QM Tool)

The Quality Model Tool [Car04a] provides functionality to define software quality factors to reuse these in different models, to state relationship among the quality factors, to assign metrics for their future evaluations, and to define Requirement Patterns to ease the final stage of the selection process. The Quality Model is not build from scratch, but is provided by the standard ISO/IEC 9126-1.

- ▶ **The *COTS Evaluation Tool***

The Evaluation Tool uses the Quality Model from the suited domain to evaluate the candidate COTS.

- ▶ **The *COTS Selection Tool***

The Selection Tool provides support for two different processes. It first supports the process of defining the selection requirements. Then it analyses these requirements and the evaluations of the COTS to assist the selection of a component. The selection requirements are stated according to the Quality Model for the specific COTS domain, either from the defined Requirement Patters or as new ones. When the requirements list is ready, the selection process tool provides a result of possible components matching these criteria.

- ▶ **The *Taxonomy Tool*** (Taxonomy Wizard):

Many quality factors may be used in many different COTS domains. The Taxonomy Tool is integrated in the Quality Model Tool to provide support for reuse of Quality Models. When introducing a new COTS domain to the tool, it is first placed at the right place in the taxonomy. Then, the tool starts to create the Quality Model by adopting all quality factors belonging to the category. The user may add additional quality factors belonging to that specific domain.

Fig. 9.5 depicts an overview of the DesCOTS subsystems. Some of these already developed modules were useful to perform some of the GOTHIC method activities (some of the modules beared some customization) as the Taxonomy Tool and QM Tool.

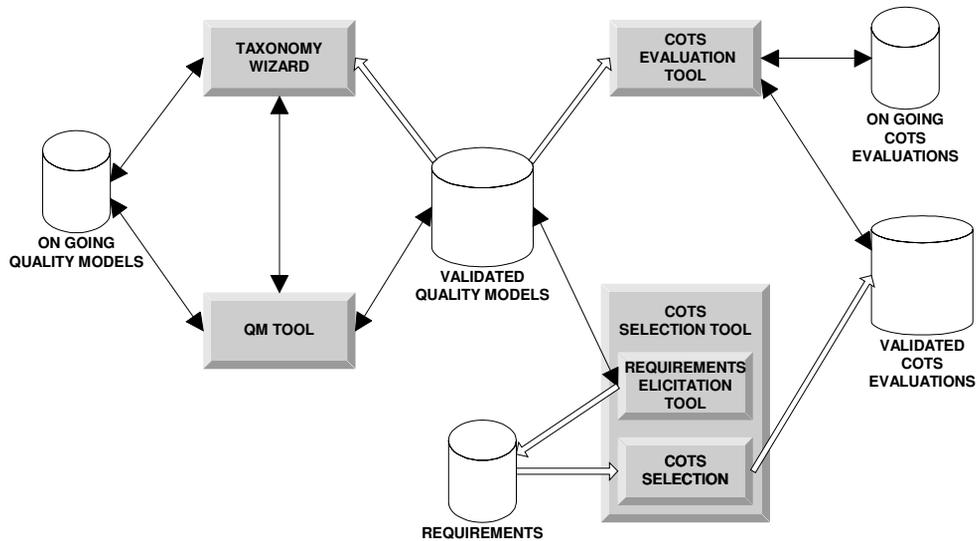


Fig. 9.5 Overview of the DesCOTS System

Use of the Taxonomy Tool in GOTHIC's Activities

The Taxonomy Tool offered support for recording, managing and graphically showing the taxonomy nodes and their dependencies we previously defined with goals by means of the GOTHIC method.

Although the Taxonomy Tool did not provide support for the procedural goal-oriented taxonomies construction, the graphical representation and recording of taxonomy nodes and their dependencies (as goals) were useful for recording and managing goal-taxonomy representations, which then were linked to its specific Domain Model, as explained below. Fig. 9.6 shows a snapshot of the Taxonomy Tool for the BA case study.

Use of QM Tool in GOTHIC's Activities

The use of the QM Tools was helpful to support the construction the Domain Model (introduced in Chapter 6) and the IQ quality model (introduced in Chapter 5), since both are based on the quality standard ISO/IEC 9126-1.

Such support was vital because the construction and managing of quality models is often a time demanding time. However, by using this tool, such process was not performed from the scratch and allowed to reuse several subcharacteristics, attributes and metrics; establish relationships among them; and define new software metrics for their future evaluation.

We saved the quality models in a repository which linked them to their corresponding taxonomy (created by the Taxonomy Tool), in such a way that the users may browse the taxonomy and adopt the quality factors and information that its attained quality models contain.

In addition, exploiting the functionality that allows defining procedural methods for constructing specific quality models, we defined two methods to describe step by step the process used to build correct models (i.e. the Domain Model and the IQ quality model).

Fig. 9.7 shows how we stated one of the strategies (methods) by defining its steps. In addition, QM also supported the glossary construction process related in Chapter 6, by allowing defining the terms used during the quality models construction.

To complement our case studies, we used the COTS Selection and Evaluation Tool for performing specific evaluations of multiple products belonging to a market segment by using the quality models defined previously.

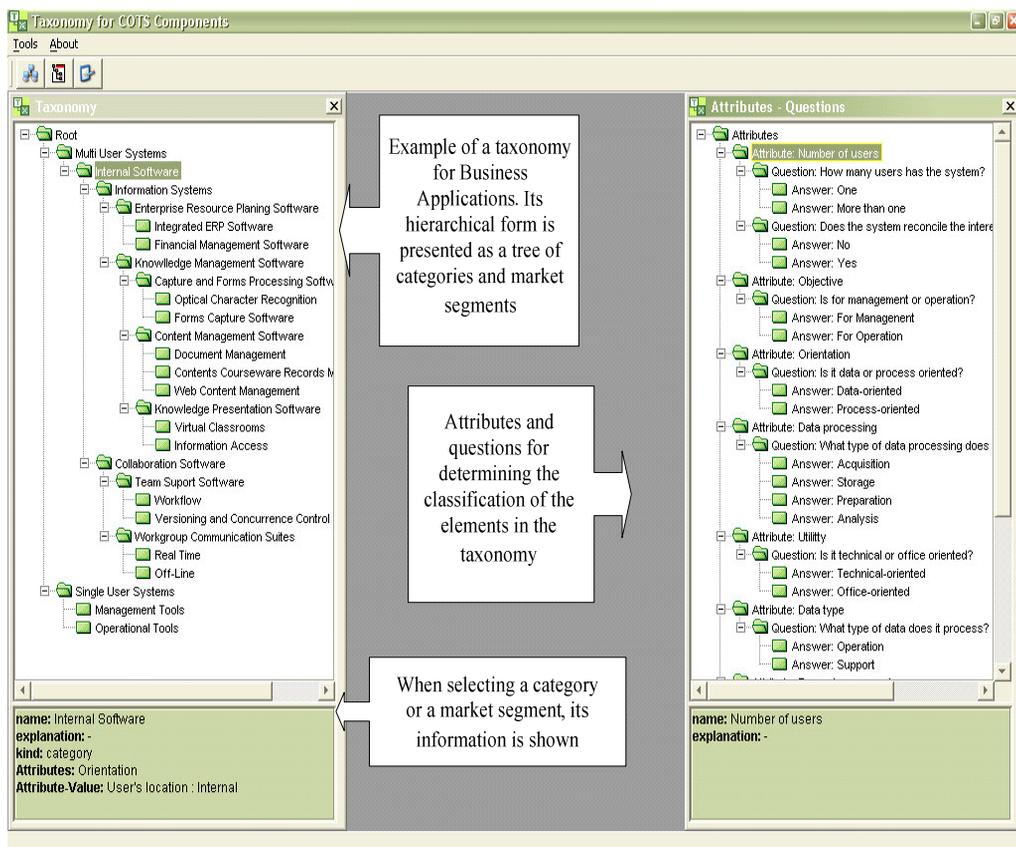


Fig. 9.6 Snapshot of a business application taxonomy with Taxonomy Tool

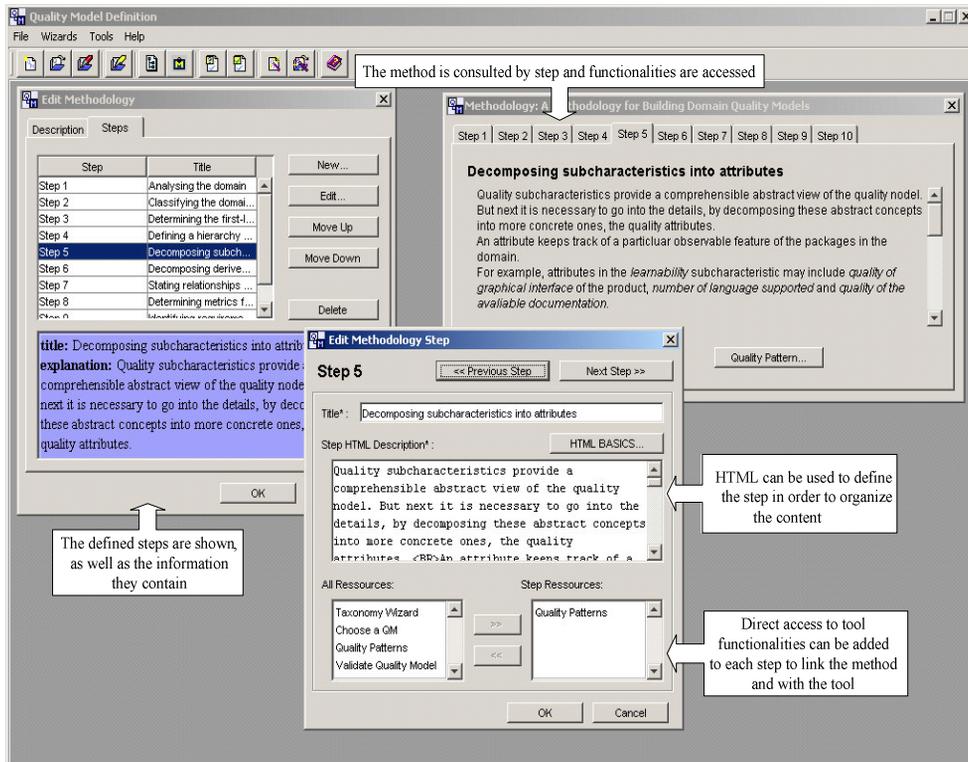


Fig. 9.7 QM snapshot: defining a method

9.5.1.2 REDEPEND-REACT Tool

REDEPEND-REACT is a tool developed by the GESSI group and the City University London for supporting i^* modeling and the analysis of the resulting models [Gra+05b]. In our approach, it was used to support the construction of i^* models.

The activities supported by REDEPEND-REACT involve building the i^* SD model, defining the properties and evaluating the architectures. Also some extra features, such as the construction of a properties catalogue, an actors catalogue and a components catalogue, are provided in order to promote reusability, supporting return on investment.

Basically, the functionality we used was related to the definition of i^* SD and SR models by dragging-and-dropping i^* shapes from the SD and SR Visio stencils provided into the drawing page.

Detailed functionality should be consulted at [RED].

A snapshot of the tool is presented in Fig. 9.8

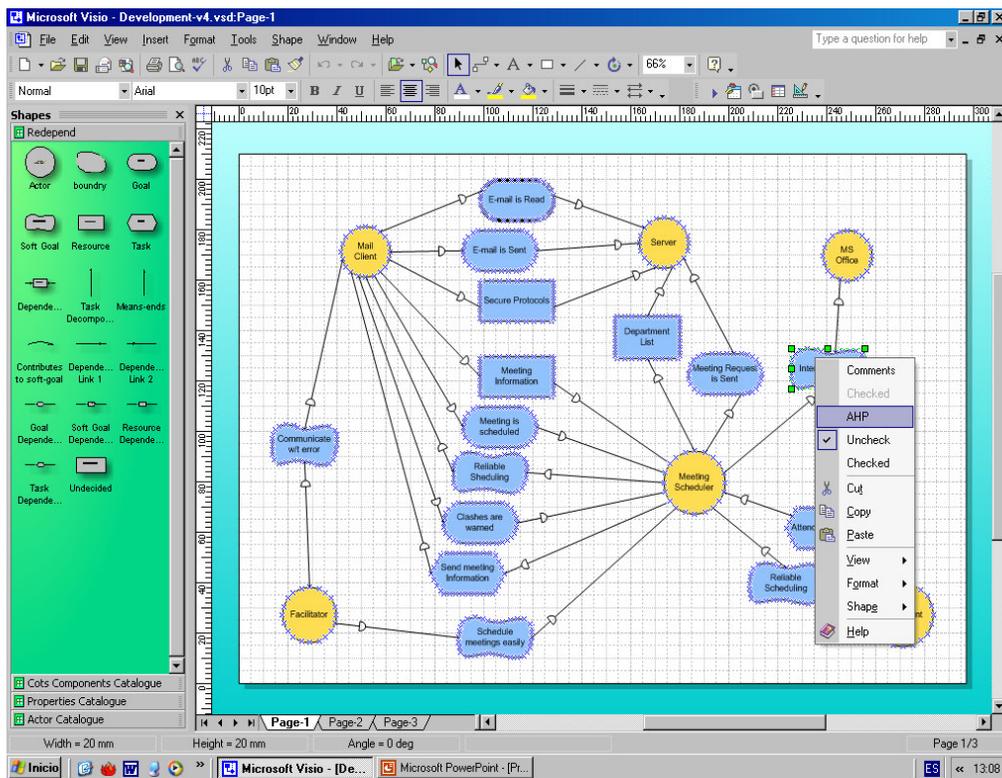


Fig. 9.8 REDEPEND-REACT snapshot: constructing an *i** SD model

9.5.2 New Software Tools Developed for Supporting GOTHIC's Activities

9.5.2.1 OTS-Wiki Prototype

The aim of the OTS-Wiki preliminary prototype is to implement the knowledge base population strategy related in Section 9.4.

The system is currently in a prototype stage and has been developed as a Master Thesis project of two students at NTNU. Detailed documentation is available in [Aas-Lar07].

Fig. 9.9 associates the main scenarios to reach the OTS-Wiki portal high-level goals.

From scenario 9.9a) we realize that the OTS-Wiki portal has been designed as open and freely accessible in order to enable the OTS Community in an open source-like environment. Scenarios 9.9b), 9.9c) and 9.9d) show the refinement of the high-level goals explained in Section 9.4 into other specific sub-goals or functionalities.

The current OTS-Wiki prototype consists on a web-based portal that uses the open wiki principle with accessible and editable data relevant for the OTS community. The most important parts of the OTS-Wiki is how easy it is for the users to find the desired information. Thus, structuring of the information by means of GOTHIC mechanisms

plays an essential role in the navigation efficiency. Implementing the entire vision of the strategy presented in Section 9.4 exceeded the scope of such Master Thesis, but the ideas and plans were prototyped in order to guide further development. Future work, detailed in Chapter 11 is envisaged to pursue a multidisciplinary project to put forward such strategy.

Goal:	Fostering OTS Community
<i>Description</i>	OTS Technology users are encouraged to work as a high performance team for reusing and sharing OTS Components Information in an Open and Freely accessible OTS-Wiki Portal.
<i>Related goal(s)</i>	1.-Incremental Population of Content 2.-Federation of OTS Resources 3.-Enabled Systematic Support for OTS Selection ...
<i>PostCondition(s)</i>	Progressive Foundation of OTS Community
a)	
Goal:	Incremental Population of Content
<i>Description</i>	Users are encouraged to publish and share content they considered helpful to the OTS Community.
<i>Related goal(s)</i>	1.-Submit OTS Component Information 2.-Enabled Active Communication 3.-New Functionality Requested to the Community 4.-Enabled a Glossary Construction 5.-User Profiles to Personalize the Information ...
<i>PostCondition(s)</i>	Incremental growth of the OTS-Wiki portal content
b)	
Goal:	Federation of OTS Resources in OTS-Wiki
<i>Description</i>	Users are encouraged to publish content that they consider may be helpful to the Community.
<i>Related goal(s)</i>	1. - User Introduces a new OTS-Web Resource Hyperlink to the Federated OTS Resources List by means of a template. 1.1. - System Records the hyperlink in the OTS-Wiki Repository. 2. – User Introduces a File 2.1. – Resource is uploaded to the OTS-Wiki Respository. 3. - User Provides a non-web reference 3.1. - Reference is Recorded in the OTS-Wiki Repository
<i>PostCondition(s)</i>	Incremental growth of the federated resources.
c)	
Goal:	Enabled Systematic Support for Selection Process
<i>Description</i>	Tools are provided to support the OTS selection activities automatically, using the standardized data from the repository. It is actually based on the DesCOTS functionality.
<i>Related goal(s)</i>	1.- User Requests automatic support for stating requirements 2.- User Requests automatic support for matching requirements with components. ...
<i>PostCondition(s)</i>	User is Supported to perform and document his or her selection process. System Learns from each selection case (i.e. non-chosen components are recorded for being shown –by analogy- to later searches)
d)	

Fig. 9.9 Goal-based Scenarios designed to reach the OTS-Wiki High-Level goals

The OTS-Wiki prototype has been designed as compatible with DesCOTS in order to be provided with the DesCOTS functionality in future.

Fig. 9.10 shows some of the specific scenarios sub-goals:

Goal:	Enabled Active Communication
<i>Description</i>	Users are encouraged to maintain an active and fruitful communication among them.
<i>Related goal(s)</i>	1-User Creates a Discussion Board 2-User Participates in an existing Discussion Board 3-User Creates a chatting discussion 4-User Participates in an existing Chat ...
<i>PostCondition(s)</i>	Incremental growth of information from the active communication.
a)	
Goal:	Enabled Assisted Search
<i>Description</i>	Users are provided with searching facilities to locate OTS components information.
<i>Related goal(s)</i>	1. - Searching similar terms in the OTS-Wiki Repository. 1.1. - Searching by Keywords 1.2. - Searching by Browsing 2.- Showing Federated <i>OTS Resources List</i> were to find OTS component information
<i>PostCondition(s)</i>	The system shows all the related information found (e.g. actual users of the component, lessons learned, FAQs, forums, related experiences, integration cost, vendor helpfulness). Let non-found components serve as requirements for future/non-registered components.
b)	
Goal:	New Functionality Requested to the Community
<i>Description</i>	Users are provided with a Requesting Board area for requesting information of components functionality that do not already exist in the OTS-Wiki portal (but maybe in other portals) or new component functionalities to the Community.
<i>Related goal(s)</i>	1. - User Makes a Functionality Request 2. - User Answers a Functionality Request 3. - System generates a Request (from scenario 9.10b)
<i>PostCondition(s)</i>	The system manages the status of the requests.
c)	
Goal:	Enabled A Glossary Construction
<i>Description</i>	Users are encouraged to detail the meaning of unknown or confusing terms.
<i>Related goal(s)</i>	1. - User Adds a term to the Glossary 2. - User Associates terms related
<i>PostCondition(s)</i>	Incremental growth of the Glossary.
d)	

Fig. 9.10 Scenario excerpts for enabling OTS-Wiki high-level goals

- Fig. 9.10a). *Enabled Active Communication*: diverse mechanisms (e.g. discussion boards, chat, distribution list, etc.) are provided to enable active communication among community users.
- Fig. 9.10b). *Enabled Assisted Search*: searching in the OTS-Wiki portal may be performed by keyword or by taxonomy navigation. The taxonomy navigation we propose (already implemented in the DesCOTS system [Gra+04]) helps users to analyze their OTS selection problem and finding their suitable market segment by navigating through a hierarchical search tree, ruling out irrelevant nodes through a question-and-answer dialog. If the information requested does not already exist in the OTS-Wiki repository, the system shows the *Federated OTS Resources List* providing hyperlinks to different resources where the information could be found; and generates a *Functionality Request* (Scenario 9.10c).

- Fig. 9.10c). *New Functionality Requested to the Community*: users are able to request and discuss component functionality not found in the OTS-Wiki portal, or with no actual implementation (those for which information was found neither in the OTS-Wiki nor in any other portal). This could result in a competition among OSS communities and COTS providers to make such components, or even encouraging the creation of new OSS communities for supporting such functionality.
- Fig. 9.10d). *Enabled Glossary Construction*: detailing the meaning of unknown or confusing terms is important because it is common in the OTS context that the same concept may be denoted by different names in different products or even worse, the same term may denote different concepts in different products. Therefore, main concepts should be clarified via explanation pages that comprise a Glossary. This glossary also serves to provide semantic relationships among concepts via hyperlinks.

Fig. 9.11 provides a snapshot of the current OTS-Wiki prototype.

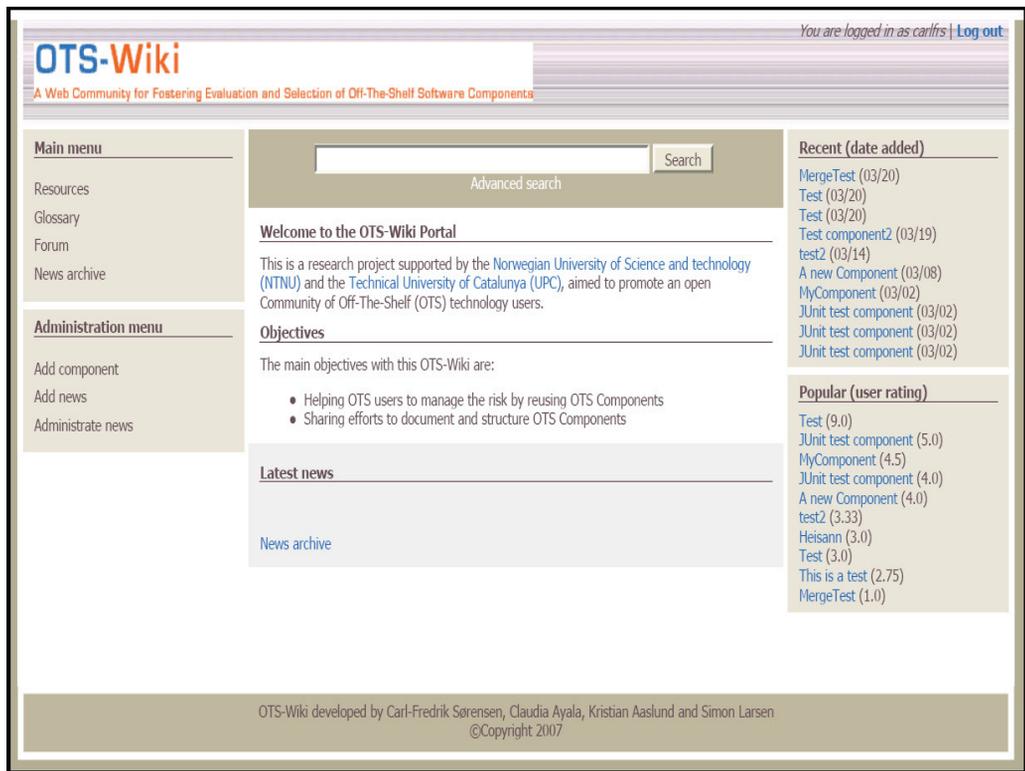


Fig. 9.11 A snapshot of the OTS-Wiki prototype

Some others functionalities are further described in [Aas-Lar07], for instance: to provide user profiles to personalize the information to the different roles needs, and case-base reasoning support for improving the searching processes and selection of multiple components.

9.5.2.2 Information Quality (IQ) Tool

The huge amount and diversity of information resources related with COTS make cumbersome and difficult the tasks that software engineers have to face each time they have to perform a COTS selection process.

In Chapter 5 we define a prescriptive strategy to support software engineers not only to collect and evaluate COTS related information resources, but also to reuse and manage them by reconciling the information characteristics to those of the specific projects. Based on our findings and the conceptual model cited in such chapter, we developed a software tool called IQ Tool as part of an undergraduate student project. Detailed documentation and analysis can be found in [Mes07].

The IQ Tool provides a systematic framework for supporting COTS selectors to reuse, gather, and decide information sources to use according to their specific quality project needs. Its main functionalities are enclosed in the conceptual model presented in Fig. 5.1 in Chapter 5. Among its high-level functionalities we found:

- ▶ To create and maintain a repository of Information Sources categorized by information kinds (e.g., hierarchy, standard,...).
- ▶ The use of the different kinds of information is supported by heuristics to decide its suitability to the user's purposes.
- ▶ Correlate Information Sources to Taxonomy Nodes in order to enable the browsing of Information Sources related.
- ▶ To create and maintain a catalogue of users (i.e. Participants) that provides comments and judgment marks about the Information Sources reliability and usability.
- ▶ To create and maintain a catalogue of information providers and their related judgment marks.
- ▶ Ranking of Information Sources according to several attributes.

The main elements of the system are:

- Participants: Allows the users to maintain a database of people or organizations (i.e., Participants) that are authors of some Information Source and/or perform some evaluation (judgment marks) about other Information Sources or other Participants.
- Information Sources: Allows the users to add, edit, consult and delete the information about the Information Sources in the database. It also allows the users to see the evaluations (judgment marks) that the Information Sources have received.
- Taxonomy Nodes: It refers to Taxonomy Nodes existing in the DesCOTS database. Information Sources are related to them in order to browse the taxonomy and find the related sources.
- Options: Several resources are provided as listing tools for manipulating and screening any element of the database (Information Source and Participants), or ranking tools for prioritize the Information Sources according to specific attributes.

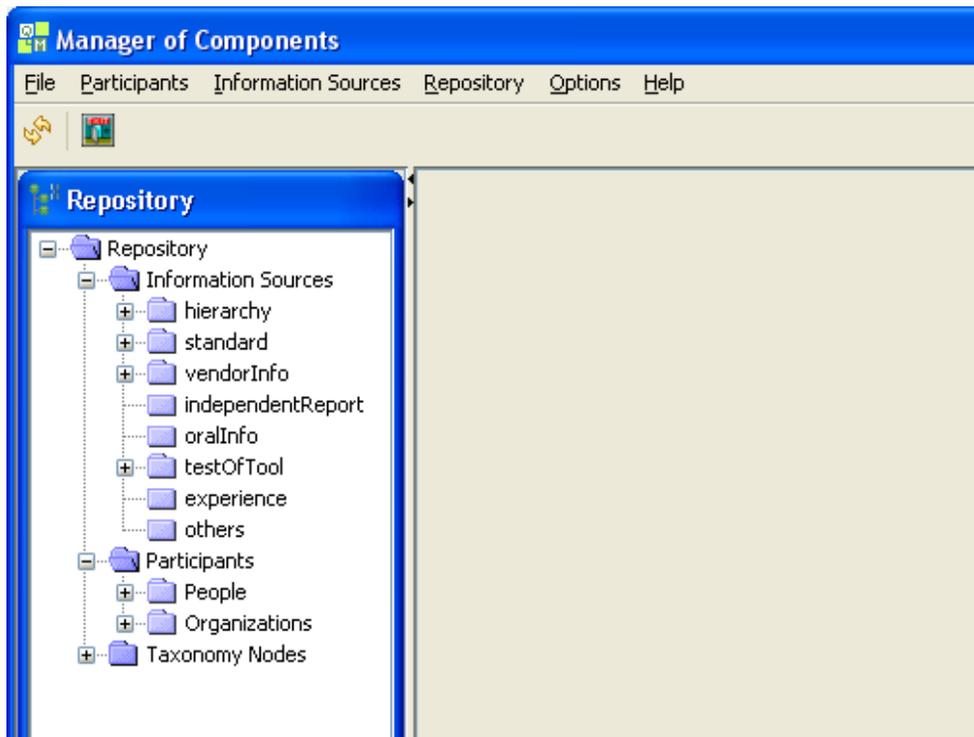


Fig. 9.12 A snapshot of the Information Quality Tool

Research in progress is looking to improve the tool functionality by adding other attributes that are being empirically elicited from COTS selectors. Fig. 9.12 shows a snapshot of the IQ Tool. It is envisaged to be integrated into DesCOTS.

9.6 Summary and Discussion

Having a knowledge base to store, retrieve and manage information during different COTS selection processes involves a lot of advantages. Mainly, it reduces the overall required evaluation time and effort [Moh+04].

To select the right COTS requires a robust decision support model. All the information stored in the GOTHIC knowledge base and its related management and updating mechanisms aim at provide support for such decision support model. It includes stakeholders' experiences and up-to-date relevant knowledge of the domain.

The information in the knowledge base has been structured following the goal-oriented principles which make it very flexible. It does not only impacts on dealing with evolution, but also for supporting the consideration of several scenarios and trade-offs in the requirements engineering process in any COTS selection project.

The strategy presented in this Chapter for populating and maintaining the GOTHIC reuse infrastructure is a feasible and incremental way of dealing with the problems reported with the use of traditional repositories and some drawbacks in COTS selection

processes. It is done by combining the GOTHIC method approach and the “open-source collaboration” approach in a social computing environment:

- It represents smooth start up and maintenance cost for putting forward the GOTHIC reuse infrastructure. It may benefit mainly to small and medium companies that are not able to invest enough money and time to manage a repository themselves.
- Some tools could be used to support the GOTHIC reuse infrastructure build up and maintenance process, as the DesCOTS system and REDEPEND-REACT tool that were previously developed by our research group and were greatly used in our approach; and the IQ Tool and the OTS-Wiki prototype that were developed as a result of this thesis work.
- The OTS-Wiki prototype intends to use the combination of ease and speed of publishing contents, together with the ability of engaging the potential of the community of COTS consumers into the structured knowledge creation process. Although several future work is required to make it operational, it is expected that:
 - It may become a quality platform for very large and up-to-date COTS (and other third party components) knowledge repositories that acts as a Meta-portal for structuring the COTS unstructured information contained in other portals (this is best illustrated by the Wikipedia).
 - It will take advantage of the set of tools developed to support COTS selection (e.g., DesCOTS and IQ Tool)
 - It will foster the (re)use of COTS and promotes communities to address requirements with no actual implementation.

Chapter

10

Method Evaluation

Evaluation is required to assess the usefulness of the method. Software methods, such as GOTHIC, need early validation while under development. The initial case studies discussed in Chapter 3 and presented throughout Chapters 5-9 accomplished this validation. They are best characterized as the formative case studies due to their central role in shaping the method. In contrast, the evaluation discussed in this chapter is best characterized as summative, since its primary role in this thesis is the preliminary evaluation of the method developed during the formative case studies.

As we discussed in Sections 1.3 and 1.4 (Research Goal and Methodological Approach respectively), our main research goal was decomposed into several research questions that were iteratively identified and tackled as we conducted the action-research process. All these research questions helped to face our main research goal.

From the action-research process, additional ideas and objectives that were not originally part of our initial research objectives arose and were also considered as useful results (as stated in Section 1.3.1) leading to the conception of the GOTHIC method as a greatly extended approach. From the research perspective, we can say that the GOTHIC development evolved from the practical needs and results detected in the COTS searching area, which were addressed by the formative case studies discussed in Chapter 3.

In the software engineering area, it is very hard to industrially validate methods such as this. It is because its further summative evaluation necessarily requires the implementation and use of large scale repositories and analysis of COTS that are not feasible to be obtained in a short period of time. Since it would not be reasonable to argue a full industrial validation of the ideas presented in this thesis within the terms of the PhD studies, in this thesis dissertation, for summative evaluation we provide some feasibility and effectiveness insights by means of arguments that extrapolate from

academic cases and post-mortem summaries of industrial cases in order to preliminary answer research questions.

This chapter summarizes and discusses the main summative evaluation approaches currently taken. They provide qualitative empirical data supporting the main research questions of this research, whilst the whole evaluation of the GOTHIC method (including an extended qualitative and quantitative approach) is also discussed as an ongoing and long term project involving several academic and industrial partners (please see Chapter 11).

Section 10.1 justifies the approach taken for preliminary evaluating the GOTHIC method. Section 10.1.1 and 10.1.2 introduce the empirical short/medium term actions performed and qualitative results obtained respectively. Section 10.2 summarizes the main method limitations found. Section 10.3 provides another kind of qualitative evaluation by comparing the GOTHIC method activities with existing alternatives. While theoretically different in research approach taken, the efforts related in each section, seek to evaluate the method.

10.1 Preliminary Industrial Evaluation of GOTHIC

To perform the GOTHIC's method evaluation, a research stay at the Norwegian University of Science and Technology (NTNU), IDI department (Department of Computer and Information Science) was performed; specifically in the Software Engineering group (SU) led by Professor Reidar Conradi. They have a recognized worldwide expertise in empirical research and industrial evaluation issues. Their support has been crucial in the development of this phase of the thesis and envisaging several collaborative research lines from the ideas generated by this research.

The main goal of such stay was the preliminary summative evaluation of the GOTHIC method in industrial settings. To reach this goal, several empirical validation approaches were studied and analyzed.

From our analysis, we concluded that although the correct and complete summative evaluation of the GOTHIC method may imply several perspectives and alternatives (many of them coming from different fields as knowledge-based systems and economic models for software reuse) they generally might imply the usage of the repository over the time and/or the availability of an industrial software team to implement it.

In [Nic+01], the theoretically correct value of an experience-based repository as the one constructed by GOTHIC was defined as the value of all query results minus the cost of all queries, as well as build-up and maintenance activities. The value of a query result is the value of the retrieved cases in terms of money or effort saved, and includes any future use of the retrieved cases that happen without querying the repository.

Thus, we found that to obtain the theoretically correct value for evaluating the entire GOTHIC reuse infrastructure would imply to collect empirical data along several life cycle stages of the repository, from its construction to its evolution and maintenance.

The different stages could be defined as:

- (1) The process of build up the reuse infrastructure with GOTHIC
- (2) The process of populating the component repository
- (3) The process of using and maintaining this repository
- (4) The process of collecting empirical data of the previous stages.

It is obvious that implementing all these processes implies a costing and long-term analysis that is out of the time expected to report this thesis (but not out of our objectives, as explained in Chapter 11). Being aware that software engineering research must be empirical, i.e., methods, proposals and theories must be validated by observations, experiments, surveys, data collection, etc., we have made an effort to formally validate in industry as much as possible some GOTHIC method related issues in a short/medium term period.

10.1.1 Short/Medium Term Empirical Evaluation

We realized that evaluating the method in industrial environments must imply at least a proof-of-concept prototype for supporting the people understanding of the method's scope, its utility and likely cost-effectiveness issues in the respective industrial contexts.

Practical approaches as those discussed in [Jed-Nic03] and [Coo97] take into account the common restrictions related to cost and time for evaluating repository approaches and propose the use of an straightforward monitoring of indicators as the “personal utility” of the delivered information to the user (i.e., utility as perceived by stakeholders) and usability to identify significant trends regarding the value of the repository.

Based on these studies, we addressed the medium/short term evaluation considering several evaluation lines and more important, future research lines were envisaged.

With respect to the short-term evaluation reported in this thesis, different studies were tackled in order to gather valuable qualitative information supporting our aim in industrial settings and well as to understand improvement issues. They were mainly performed in the context of the European ITEA project, Norwegian COSI (Co-development using inner & Open Source in Software Intensive products) [COS] which aims to enable the Norwegian IT sector to fully exploit the benefits and advantages of COTS and Open Source Software (OSS) components.

At that time, these studies can be summarized as:

- An academic seminar to introduce the method to NTNU researchers and getting feedback about the method and the correct way to put forward and evaluate the approach in industrial settings. This seminar resulted in several ideas driving the subsequent studies.
- An industrial explorative survey to further inquiry several important facts of how components are selected in industry and gain a better understanding of how the GOTHIC method can be better harmonized into the industrial practice.

- An industrial seminar to preliminary analyze the perceived easy of use and cost/effectiveness of the method.
- A proof-of-concept prototype (introduced in Chapter 9) was also envisaged to mainly evaluate the usefulness of the resulting products of the method (i.e. the flexible reuse infrastructure and its deliverables). Despite this approach involves a long-term project out of this thesis report, in this Chapter we will discuss its primary goals, which are further explained as future work in Chapter 11.

All these short/medium term studies pursue the GOTHIC evaluation process initiated by this Thesis.

In general, they are qualitative and subjective approaches to gain understanding of how well the method supports the research questions. Our main aims are: on the one hand to gain feedback about the cost-effectiveness and easy of use of the method perceived by COTS selectors; on the other hand to initiate a long-term project to qualitatively and quantitatively evaluate the usefulness of the reuse infrastructure and deliverables obtained by the method among other important issues discussed below.

10.1.1.1 Academic Seminar

A seminar addressed to NTNU researchers was held at NTNU in May 2006. 11 researchers from the IDI department attended.

Participants are actively involved in component selection research and industrial practice, and share a strong common core curriculum in empirical research methods [SU].

We first introduced the GOTHIC method and reported our results in several COTS selection processes. Secondly, our aim was to inquiry discussions about the possible advantages and disadvantages of the method in academic and industrial settings, as well as getting feedback about the perception of the utility and usability of GOTHIC from academics and industrial research experts. We therefore had time for discussion and feedbacks.

During our talk, participants were able to interrupt and ask questions. Participatory observation technique was used to gather data and annotations, as well as to process the data obtained from the seminar and subsequent discussions.

Table 10.1 summarizes most of the comments obtained from participants and observations recorded.

Discussions and feedbacks were significantly useful to realize and improve several technical approaches forming part of the method. They were also crucial for planning subsequent strategies to reach the short-term and long-term method evaluation as reported below.

Table 10.1 Issues regarding the use of GOTHIC as perceived by researchers

Topic	Participant's Comments	Observation
Suitability of produced models	Most of the participants never had heard about <i>i*</i> models.	After a brief explanation of the use of the <i>i*</i> methodology and their advantages for recording COTS dependencies, most of the participants realized its utility in the GOTHIC context. However, the issue of models exploitation was still a concern
Perceived Advantages of the method	Some participants said: "Our and most others' concrete industrial experience with reuse repositories and related classification systems (taxonomies/ontologies) has been disappointing – they are costly to develop and maintain, annotation costly and almost never done, and hard to use (e.g. troublesome dependencies are often not dealt with)".	It was explained that the GOTHIC approach aims at dealing with some problems related with most classical repositories approaches. Several examples were explained and several ideas to put forward the GOTHIC repository were commented. The need of having a repository like this in the context of CBSD was recognized as crucial
	The OSS schema could be a case study that may be benefit from this repository approach.	Some adoption schemas may be studied. Mainly in the context of OSS, because people participation is greatly expected.
Perceived Disadvantages of the method	Hardly industrial validation and adoption.	Industrial evaluation requires a long term project. A lightweight population schema must be designed.

10.1.1.2 Explorative Survey

From the previous seminar, we realized the needs of further analyze and get empirical evidence to support the method taking into account industrial needs. Two of the main concerns were: how COTS selection processes are dealt in industry in order to better synchronize the GOTHIC method activities; and to understand the existing industrial practices for improving the GOTHIC method processes.

Thus, taking advantage of the availability of students taking the course TDT4735 Software engineering at NTNU we asked two of them to perform the study under our guidance. In the design stage of the study, we realized the need to focus not only on COTS, but also to include OSS (i.e., to focus our study on OTS components). It was because; there exists a lot of evidence that a huge percentage of software development industrial processes are actively using both paradigms [Sim-Dil06].

The goal of the study was to discover the actual processes used in industry when it comes to selection and evaluation of OTS components, mainly: Where and how OTS

components are found? How are they evaluated? How are they learnt? How industries take care of knowledge about the chosen components?

Table 10.2 Excerpt of explorative survey results from some Norwegian companies

Issues		Descriptive Findings
When in the development process are OTS components selected?		Large components are usually selected at an early stage during the development process, while small components can be selected anytime during the development process.
Motivations to use OTS components		Higher quality, shorter time to market, and cost are the principal motivations to integrate OTS components into a system or application.
Selection Process		Most of the companies do not use any formal process for the selection of OTS components.
Searching Candidate OTS components		Companies usually find OTS components by searching repositories containing components used before, by Web search (Google), specialized web sites, and OSS communities. An assessment of the problems reported by the use of these resources is made on Chapter 2.
	<i>Evaluating Candidate OTS components</i>	Matching functionality and standards compliance are the most important technical issues when evaluating an OTS component for integrating it into a system or application.
	<i>Deciding OTS components</i>	The final decision is mostly taken by the person who initiates and performs the work related to components harvesting or a leader of the project. A significant difference was observed between company sizes. As indicated by a respondent from a large company: “the project team makes a recommendation and is the Chief Executive Manager who has the final word”. In small and medium companies “it is mostly the software developer who takes the final decision. Such decision is largely influenced by components they have already used, written in programming languages they know, or have been recommended by colleagues.
	<i>Documenting the Decision</i>	Large companies document (but usually not adequately) the rationale behind the choice of the selected component. Small and medium companies generally do not document their rationale. Large companies usually have a person as responsible of the OTS components knowledge in order to reuse this information later in subsequent projects, even if this person does not always have a formal role as “knowledge keeper”. Reported problems by using this kind of “knowledge keeper” are reported in Chapter 2, Section 2.3.

A pre-study was performed and several semi-structured interviews were applied to software engineers in some companies as Keymind Computing, SINTEF ICT, FAST ASA, Statoil ASA, eZ Systems, SUN, and Linpro. It was reported in [Ger06]. Such pre-

study was the basis for performing an improving a qualitative descriptive study reported in [Ger07]. It includes enhanced semi-structured interviews to software engineers in the following Norwegian companies: Visma, Sirius IT, TietoEnator, WebOn, Ageo, DKDigital, Commitment, Grieg Multimedia, and Rivity. Main results from these studies are summarized in Table 10.2

10.1.1.3 Industrial Seminar

The industrial seminar was held in the context of a COSI project meeting at NTNU in October 2006. 12 industrial participants from the COSI project attended.

The session was structured in the next way:

- To introduce the GOTHIC method. Its artifacts and deliverables were emphasized (i.e., the reuse infrastructure and its related artifacts).
- To inquiry the perceived usefulness of the method deliverables, the perceived effort to produce them and use the different models recommended by the method (UML models, *i** SD models, quality models, etc.).
- To discuss the possible perceived advantages and disadvantages of the method in industrial settings.
- To explain the idea of our proof-of-concept-prototype [Aas-Lar07] involving open source collaboration for putting forward the reuse infrastructure aimed by the GOTHIC method.

During our talk, participants were able to interrupt and ask questions.

We had time for discussion and feedbacks. After the seminar, some individual meetings with some interested participants were held. They were able to provide us some of their own COTS selection projects information to perform further post-mortem studies.

Participatory observation technique was used to gather data and annotations, as well as to process the data obtained from the seminar and subsequent discussions.

Table 10.3 summarizes the main observations and feedbacks from the participants with respect to the use of GOTHIC for building a reuse infrastructure into their organizations from a technical perspective. It is important to remark that it is not argued that the method can be applied into industrial organizations, because this implies many other issues (i.e., organizational, strategic, budget, etc.) but that the skills and expertise of software teams may be able to perform the method activities.

After the discussion summarized in Table 10.3, we described the idea to put forward a proof-of-concept prototype implying a globally available OTS-Wiki portal (introduced in Chapter 9). Most of the participants were interested on the idea and help to define several scenarios defining the main expected functionality of the portal: a) Support for locating OTS and information about them; b) Recording component information in a

structured way; c) Maintaining and reusing such information; d) Getting tool support for performing components selection processes. These high-level scenarios are related in Chapter 9 and further explained in [Aas-Lar07]. The industrial background of the participants helped us to mature our industrial conception of the method usability.

Table 10.3 Issues regarding the use of GOTHIC in industrial organizations

Topic	Participant's Comments	Observation
Suitability of produced models	UML class diagrams and use cases are mostly used in their processes	Software teams are familiarized in producing these models
	Quality models are rarely used but seem suitable to foster reuse. Most participants are familiarized with quality models.	It seems that to produce quality models is not a problem from a technical perspective since most industrial teams are familiarized with them.
	<i>i*</i> models are never used, only one of the participants had heard about them.	Studies must be done to provide statistics of the time that industrial users require to learn and produce this kind of models. Complexity of the models grows as the complexity of the domain increases.
	Goal-oriented taxonomies are understandable and easy to use.	Search and retrieval mechanisms seem feasible to be implemented.
	Glossaries are frequently used but not formally. It could be useful to systematize the capture of crucial concepts as proposed in the GOTHIC method.	It seems that to produce glossaries is not a problem from a technical perspective.
Perceived Advantages of the method	It seems an approach that synergically integrates useful strategies to deal with COTS selection processes.	All method activities are proved to be useful in specific case studies, but return on investment analysis must be performed in industry to provide data to evaluate the industrial use of the method.
	Tool support seems a great advantage in performing the method	Some tools are proposed to support the method activities.
Perceived Disadvantages of the method	It seems a cumbersome approach that requires a large investment.	The extra effort needed for applying GOTHIC could be more helpful and less risky than acting reactively, as we intend to corroborate as one of the main hypothesis of our long-term study.
	Small and medium companies are not able to invest in the construction and maintenance of this kind of repositories.	The use of the approach in large companies and the marketing contexts (See Section 9.3) seem the most feasible ones.

10.1.2 Short/Medium Term Qualitative Results

The experiences reported in this thesis dissertation and its short/medium term empirical evaluation demonstrate some facts related to the stated research questions that are summarized in Table 10.4.

Table 10.4 Research Questions Revisited

Research Question	Short/Medium Term Qualitative Results
RQ1: What are the actual challenges of COTS selection process?	Our studies confirm that the detected problems addressed in this thesis are really sound in the industrial context [Ger06], [Ger07]. All participants in the seminars agreed.
RQ1.1: What are the actual challenges of COTS searching processes?	
RQ2: How can we support COTS searching challenges?	To answer this question, the sub-questions related below were qualitatively evaluated.
RQ2.1: Can goal-oriented approaches be used to produce useful results for dealing with COTS searching challenges?	Our formative results showed that the introduction of goal-oriented approaches for structuring COTS taxonomies resulted in an increased understanding and management of the taxonomy content; it was perceived as a more reliable COTS selection processes by the practitioners and experts asked. In the seminars, examples of goal-oriented taxonomies and existing taxonomies were provided. A better understanding was obtained by the use of goal-oriented taxonomies.
RQ2.2: How can we characterize COTS in the marketplace?	The informational dimensions we used to characterize COTS were considered sound and complete by industrial users. Industrial users highlighted the need to further investigate the required OSS informational dimensions in order to address them in the GOTHIC method metamodel.
RQ2.3: How can relevant information related to COTS be gathered, evaluated and synthesized?	The Information Quality management strategy presented was considered sound, but more empirical evaluation should be done in order to better support the practice. Therefore, a further empirical evaluation is being performed, some details may be found in [Aya-Fra08].
RQ2.4: How can such information be maintained for its reuse in different COTS selection processes?	The proof-of-concept repository prototype, called OTS-Wiki has been presented and well accepted by the community. It was presented at the 3rd International Conference on Open Source Systems (see [Aya+07]) and the general feedback from the conference attendees was positive and they evidenced to be interested in using the portal and following its progress.

Furthermore, so far we have gathered qualitative information that helps to drive on-going and future studies, as well as further GOTHIC method improvements based on industrial needs.

10.2 Limitations of the GOTHIC Method

So far, the main limitations of the GOTHIC method in industrial environments can be summarized as:

- ◆ Further studies are needed to provide the industry with effective data to perform a decision making process for adopting the method.
- ◆ The method provides informal, as opposed to formal, semantics for identifying and stating goals and thus it does not support formal and systematic reasoning. The domain expertise of the people involved in performing the method plays a crucial role.
- ◆ While this dissertation presents some useful heuristics which were validated from GBRAM into the GOTHIC context, as well as several lessons learned in formative case studies, the method does not offer a complete and systematic catalogue of heuristics, many of them greatly depend on the analyst rationale.
- ◆ Evolution and manipulation of taxonomies is not a systematic process. It requires knowledge of the domain to discern the decision to take.
- ◆ While well suited for COTS, the method has not been adequately proven and tested for other kind of components (e.g. OSS); therefore further studies should be addressed to include other kind of components into the GOTHIC metamodel.
- ◆ As with all repository approaches, the method requires a considerable initial investment. More empirical evidence is required to argue that although the GOTHIC reuse infrastructure requires an extra effort, it could be more helpful and less risky than acting reactively.
- ◆ The issue of how to put forward the reuse infrastructure in organizations to reach the LSO paradigm is out of the scope of this thesis.

10.3 The comparative of GOTHIC with similar approaches

Another kind of qualitative evaluation is the comparison of GOTHIC with other existing alternatives. Due to the application scope and the set of activities that the GOTHIC method deals with, it is difficult to identify a single approach against which the GOTHIC method can be compared as a whole. However, several aspects or parts of the method, as well as the models resulting from the process, can be compared with some existing approaches. To make the comparison easier, we have addressed each of the activities of GOTHIC separately. The next paragraphs summarize some of the facts resulting from this analysis.

Activity 1: Information Sources Exploration

Although this activity makes use of pre-existing Information Quality (IQ) assessment techniques to deal with the vast amounts of widespread, heterogeneous, and unstructured information describing COTS (as related in Chapter 5); the objectives for which they are used in our approach are very dissimilar to their traditional applications. To the best of our knowledge it is the only attempt to systematically manage and reuse COTS information in a quality assurance setting.

As far as we know, several works have recognized the criticism of the COTS information characteristics, as related in the introduction of Chapter 5. Some search engines have attempted to use information retrieval-based mechanisms supported by intelligent agents, ranking algorithms and cluster analysis to support the finding of COTS information; and many useful studies of different nature have been performed to discover how much of the required information to perform an informed selection is actually available [Ber+03], [Berg06]. However, none of them provide a prescriptive and systematic approach for searching, managing, and reusing COTS information in a quality assurance framework.

Activity 2: COTS Domain Analysis

We adapted and complemented domain analysis techniques to fully deal with the COTS technology characteristics. Table 6.1 summarizes our COTS domain analysis proposal and makes clear the gap for recording non-technical descriptions and interoperability with respect to other domain analysis approaches. Furthermore, using some mapping rules, we integrated all required models to represent COTS informational dimensions into a single one, based on the well-known standard ISO/IEC 9126-1, which is highly oriented to support the evaluation of the candidate components. Therefore, it serves as COTS information metadata. This makes our domain analysis approach unique.

Activities 3, 4 and 5: Goal-Oriented Core of GOTHIC

Although there is a wide range on goal-oriented research: goal modelling, goal specification, and goal-based reasoning for multiple purposes, such as requirements elaboration, verification or conflict management, and under multiple forms, from informal qualitative to formal; in our approach we have used many of these research works, but with objectives very dissimilar to those they had been attempted before. A clear example is the customization of GBRAM and the *i** framework to identify, record and refine COTS related goals and their interdependencies.

In the COTS selection context, some approaches are also making use of goal-oriented approaches [Chu-Coo04], [Alv03] to guide requirements and support decision making processes. However, our approach is not comparable with these approaches mainly because of their different objectives. We can say that our approach may complement these approaches to find the COTS that best fit the stated requirements.

Activity 6: Taxonomy Manipulation and Management

As far as we know, there are not other proposals in the literature addressing the manipulation and validation of goal-oriented COTS taxonomies. In the same way, we

are not aware of any other proposals defining the set of properties that taxonomies must comply to be considered complete and correct. Because of this, activity 6 can not be directly compared to other approaches. Advantages of using this approach are related in Chapter 8.

Activity 7: Knowledge Base Management

Traditional approaches for building, maintaining, and browsing software reuse repositories have suffered from lack of domain-specific components and a heavy “fill-up” investment upfront as well as the incomplete, unreliable, and too static characterization of components as surveyed in [Par-Con07] and already detected in early proposals as [Pou95] and [Mor+02b].

On the other hand, strategies for incremental population and maintenance of repository content and community building have also been weak, ignoring important social factors such as trustful re-user participation to record and edit all kinds of experience information. This situation also arises for other types of repositories, e.g. company-internal experience bases for software improvement [Din-Con02]. Trying to deal with all these issues altogether, our approach may be considered unique.

Chapter

11

Conclusions & Future Work

This document summarizes a PhD thesis in the area of COTS selection, more precisely in relation to the construction COTS marketplace taxonomies as backbone of a reuse infrastructure for supporting some COTS searching open issues.

This chapter reviews the main contributions of our research as well as some future lines of investigation which have emerged along our research work. Specifically, Section 11.1 summarizes the main contributions of the approach whilst Section 11.2 relates the envisaged future work.

11.1 Contributions of the Approach

This thesis intends to contribute in several aspects to the field of COTS searching. Some general departing aspects which make our approach unique with respect to previous proposals are:

GOTHIC provides COTS information recording and searching support

The resulting COTS reuse infrastructure constructed with the GOTHIC method proposed in this thesis, is mainly oriented to support the searching and reuse of COTS related information.

Existing COTS selection methods do not deal with COTS searching in the marketplace. Furthermore, existing COTS taxonomies have not been constructed with a well defined method to endorse them, they lack of rationale, mechanisms to deal with reuse and COTS marketplace characteristics.

In this context the GOTHIC method provides methodological guidance for assessing COTS marketplace domains and building a reuse infrastructure based on goal-oriented COTS taxonomies that help to structuring all COTS related information in a reusable and standardized framework.

GOTHIC deals with COTS Marketplace Characteristics and Improves COTS information Reuse

The GOTHIC method proposes a set of interrelated activities using several techniques aimed to deal with the COTS marketplace characteristics. Specifically, some of the problems related in Section 1.1 are dealt in next way:

- ▶ *Uncontrolled COTS marketplace*: Goal-oriented taxonomies offer a natural way of managing such mess by improving flexibility, understanding and managing to the current COTS marketplace representations.
- ▶ *Growing size of the COTS marketplace*. By using goal-oriented COTS taxonomies, appearance of a new market segment is easier to handle than in other approaches, since it requires to locate its place in the taxonomy using the defined classifiers, and once there even some useful artifacts are inherited (e.g., quality models and glossaries). Proliferation of COTS information is taken into account by prioritizing information sources in the basis of specific criteria (e.g., time, money, reliability, ...).
- ▶ *Rapid changes in the COTS marketplace*. This fact points out the need to separate conceptually the COTS from the services that they cover. Thus, goal-oriented taxonomy nodes do not stand for types of COTS available but for related groups of functionalities, it makes the taxonomy more robust with respect to the segment barriers movement effect mentioned in Section 1.1.

The existence of such goal-oriented taxonomies and their associated knowledge base obtained with GOTHIC would be a basis for the current methodologies for searching components “to be evaluated” into the right segment of COTS marketplace with a high probability of success. It means a significant reduction on time required in the evaluation process and the implicit reuse of the knowledge.

Besides these contributions, several other contributions to the field exist, mainly in relation to the aspects of reliability and effectiveness of resulting taxonomies and artifacts for dealing with COTS marketplace characteristics; and the reusability of the knowledge gained in each experience.

Next subsections present the main contribution of this work regarding these issues.

11.1.1 Reliability and Effectiveness related Contributions

GOTHIC improves COTS taxonomies usability and therefore their effectiveness by adopting and/or adapting several well-known techniques into a method, with well-defined activities and steps, each with its own objectives and rationale. Therefore, it provides a solid rationale which endorses the resulting taxonomies, which are not obtained by common sense, but also by reliable frameworks built from a formal and unambiguous base, applying a well defined rationale.

The main contributions in relation to this issue are:

- GOTHIC incorporates goal-oriented approaches to define a formal basis to support the evolvability and interoperability in the marketplace and gives a natural and understandable rationale to the COTS marketplace structuring, making easier to the users comprehend the COTS available in the marketplace.
- A taxonomy validation and management process has been defined to ensure the correctness and completeness of the taxonomies, whilst it allows the tailoring of the taxonomies to the marketplace evolution and particular needs of the users.

11.1.2 Reusability related Contributions

Another important contribution of the GOTHIC method is the definition of a suitable infrastructure to uphold knowledge reuse and evolution of the marketplace. The main contributions of the proposal regarding reusability are:

- Identification and guidance for constructing artefacts to uphold reuse. The informational dimensions required for the effective and efficient selection of COTS components have been identified and represented using appropriate types of artefacts. Moreover, most software engineers are familiar with the type of artefacts proposed.
- Guidance to build an integrated framework of COTS information reuse: Using some mapping rules, we have integrated all the produced artefacts into a single one, based on the well-known ISO/IEC 9126-1 standard (*Domain Model*), highly oriented to support the reusable evaluation of the candidate components. Given this representation, we may use some existing tool-support to conduct the evaluation of candidates in a uniform way.
- Inheriting produced artefacts downwards the taxonomy. Taxonomy nodes are bound to a domain model enclosing COTS related information. New market segments can be integrated to the taxonomy and their domain model can be constructed by inheriting from and/or tailoring their ancestor's domain models or parts of them.
- The population and maintenance of the Knowledge Base (repository) strategy impacts positively on ameliorating some well-known obstacles for successful repositories whilst overcome the problems addressed by the GOTHIC activities.

Intending to highlight the GOTHIC benefits to the actual roles involved in COTS selection (introduced in Table 2.3 in Chapter 2), Table 11.1 briefly summarizes some high-level intended benefits of this approach to them.

Section 10.3 in Chapter 10 compares GOTHIC activities with existing proposals, and also relates the contributions of each activity with respect to the existing work.

Table 11.1 Intended benefits to the different COTS selection roles

Activity	Intended benefits of GOTHIC to the COTS Selection roles
Market Watcher (MW)	Marketplace exploration is easier and understandable since MW only has to screen the market segment(s) that matches the established requirements.
Quality Engineer (QE)	Quality issues are easier to reach and record by using the heuristics and resources provided by GOTHIC.
Selector (S)	To take the final decision based on the evaluation of the candidates is more reliable since all the information required is in the same umbrella and therefore their comparison is better handled and less risky.
Knowledge Keeper (KK)	Storing and documenting the decision is naturally performed by the produced artifacts for their future use in forthcoming selection processes. And with the performed strategy for open collaboration for populating and maintaining the repository allows small and medium enterprises to reuse their knowledge and others' knowledge to improve their selection processes.

11.1.3 Contributions in collaboration with other members of the GESSI group

There is a great deal of ongoing work within the GESSI research group which is closely bound to the work presented in this thesis. Some contributions have resulted from the combined effort with several members of the group. Most of these contributions resulted from the fact that a better understanding of the i^* framework was needed because it was used in several of our works.

Similarly to other approaches in software and knowledge engineering areas, the GOTHIC method also uses the i^* approach to represent goals, as goals present several characteristics that make them attractive (e.g., expressiveness, stability and evolvability), as claimed in this thesis. We have further endorsed this approach by performing some actions to adapt and clarify the i^* semantics to our intended objectives and make it more understandable and usable.

Some contributions resulting from the combined efforts with other members of the GESSI group are:

- To extend the i^* approach for recording COTS interoperability, and in general for supporting COTS selection processes [Fra+07].
- A reference model of the i^* approach that provides a precise meaning of the concepts therein [Aya+05b].
- The RiSD methodology, aimed at prescribing by heuristics the construction of Reduced i^* SD models for modeling organizational goals and software systems [Gra+05].

Moreover, we are currently working on the extension of the DesCOTS tool [Gra+04] (already implemented by some members of the GESSI group) with some tools resulted

from this thesis as the IQ Tool system [Mes07] and the OTS-Wiki collaborative portal [Aas-Lar07] introduced in Chapter 9.

11.2 Future Work

The work presented in this thesis addresses some of the fundamental problems with COTS searching and reuse. However, much work remains to be done and several research lines remain open for future investigation to improve and extend the research results obtained from this thesis.

Major future research lines are described in Section 11.2.1; whilst Section 11.2.2 relates an intended financed project to put forward some of these envisaged lines.

11.2.1 Major Future Research Lines

Some of the envisaged as future research lines may be described as:

To extend the GOTHIC metamodel to include other kind of reusable components

From the industrial evaluation of GOTHIC we realized the need of industrial users to select not only COTS but also OSS. Therefore, we have recognized the need to focus our ongoing and future research not only on COTS, but also OSS (i.e., OTS).

Additionally, over the past 2-3 year, emerging technologies have radically changed the manner in which consumer and business applications can be accessed over the Internet, leading to the rise of dynamic and rapidly growing new segments in the provision of services, named Internet Web Services industry, which currently is also suffering of procurement-related problems.

Consequently, based on the research performed in this thesis, some further studies are envisaged in order to carefully analyze and adapt the GOTHIC metamodel firstly to embrace OSS component issues into the integrated *Domain Model* obtained with GOTHIC, and secondly to analyze the incorporation of Web services as they are demanded by the industry.

Supporting this claim we found that roughly half of all European software projects make use of OSS, and this amount is increasing rapidly [Li+05]. Evans Data Corporation [Eva05] states that 56% of developers use open source products in 2005, while only 38% used it in 2003. On the other hand, market growth estimations for Web Services are notable. The analyst firm Radicati Group expects that worldwide sales of web services will reach USD \$6.2 billions by 2008 [Rad].

To collect empirical industrial data to evaluate and improve the GOTHIC approach

We realize the critical need to reinforce our GOTHIC approach with empirical research to drive the intended improvements in an optimal way. Although gathering industrial data is commonly a very difficult task, we envisage relying on the strong empirical background and collaboration of some research groups and industrial contacts.

- *Empirical studies in industry are planned in order to gather evidence of the usefulness and effectiveness of GOTHIC in different context and schemas of use.*

Several empirical studies are envisaged in a long term, they range from further investigate the success and usability of the diverse artifacts produced by the GOTHIC method, as well as diverse aspects that range from the success of the open-source-like collaboration concept for dealing with OTS selection challenges (i.e., the proposed population and maintenance strategy detailed in Chapter 9) and the problems reported with the use of repositories.

Some intended metrics are the ability of the intended portal to enable a community of re-users around specific components, ability to promote homogenization, promotion of OSS communities, information reuse, etc. They will be preliminary assessed by the support perceived by the users of the portal, effort saved by using the portal for supporting COTS selection processes. (i.e., how much time is invested in performing the COTS searching activities with and without it); and the reuse percentage in diverse COTS selection processes among similar projects.

- *To improve each GOTHIC activity based on empirically obtained feedback.* For instance, to continue with the research described in [Aya-Fra08] for assessing the quality of the information that is really used in the practice of components selection processes.

At this respect, we want to enhance our approach with more empirical data to make it more efficient. This could be done by automating the evaluation of information source objective properties, i.e., extracting properties such as authors' names and organizations, references, etc., and populating automatically the database with these properties. We envisage the application of technologies such as intelligent agents, ranking algorithms, cluster analysis, web mining/data mining, personalization, recommendation, and collaborative filtering techniques to improve the construction of knowledge over these raw data [Gils+04].

To explore the use of other techniques to support the evolution and management of the classification schemas

We consider that the exploration of novel techniques to support the collaborative evolution and management of the classification schemas (e.g., as the so-called folksonomies) could be interesting for designing suitable mechanisms to collaboratively evolve the proposed GOTHIC taxonomies and associated artifacts. It may be done by extracting and assessing actionable meaning from both structured and unstructured data in the web.

11.2.2 A Multidisciplinary Intended Project

An intended result from this thesis is a multidisciplinary long term project involving academy and industrial partners to further explore and improve several of the concepts originated here. It is mainly addressed to exploit several of the major research lines envisaged in the previous section, and others interesting lines with greatly industrial and social effects.

Some of the intended partners of the project consortium are MicroArt S.L., Norwegian University of Science and Technology (NTNU), Consejo Superior de Investigaciones Científicas (CSIC-IIIa), Politecnico di Torino (Polito), Centre de Recherche Public Henri Tudor (CRPHT), Beijing University of Technology (BJUT), Hewlett Packard (HP), Actimage, and Computas.

Project Goals

Taking advantage of the expertise of the partners, this project aims at a structured and incremental way to federate and reuse the existing efforts for managing components. The idea is to further exploit, extend and improve our proof-of-concept OTS-Wiki prototype [Aas-Lar07] which is based on “open source collaboration idea” with semantic wiki and web intelligence technologies.

The main goals of the project are:

G1. Build a high quality meta-data OTS repository in order to:

- ▶ Support software *customers* (here *integrators*) with efficient means for software development and maintenance by reusing available OSS/COTS.
- ▶ Support software *providers* of OSS/COTS with means to “market” their products.

G2. Encourage the European Society to add value to the software applications developed and used, so that it benefits from more reliable software as well as a faster and cheaper software development process.

- ▶ Incrementally build a cooperative *component community* around the shared components.
- ▶ Incrementally build up pragmatic *classification system* for component annotation.
- ▶ Provide web intelligence inference services to learn from the repository.
- ▶ Provide mechanisms to rank the quality of the information stored.

Web-intelligence techniques will be used to evolve GOTHIC classification schemas and its related artifacts by analyzing and extracting actionable meaning from both structured and unstructured data in the web (mostly textual), e.g., from user-provided dialogs, comments and ratings, experience and test reports, queries and query responses, actual component choices etc. This must be combined with a usage-based growth of available components, together with respective metadata annotations and recorded experience, linked to related user communities. This leads us to recent web technologies, like Wiki for facilitating cooperative web-communities, web portals, and web-intelligence technologies to synthesize and evolve component annotations.

A technology such that may be viewed as a kind of digital library in the sense that it is an online collection of information made accessible to a community of users. As a result, we can incrementally establish:

- A1.** A *federation of existing, domain-specific web repositories* where the components and their descriptions primarily are stored in such repositories,

- A2.** Support for additional *component annotations* (common metadata) and searching according to some “standard” classification system (free text search initially), and
- A3.** *Experience sharing* among a dynamic community of component customers and providers (called users).
- A4.** *Advance of web intelligence techniques* for information extraction and information reputation (or quality ranking) techniques.

On the other hand, by the expected massive use of the OTS-Wiki portal for annotating components and sharing experiences about them, we may obtain a semantically-rich meta-information repository that can be used to facilitate performing: 1) data mining; 2) classification; c) Experimentation and research projects about OTS components and CBSE.

Consortium Roles

To reach the project goals by the consortium, we have envisaged the following roles (not necessarily disjoint):

- Classification schema definition. Role(s) for defining the classification mechanisms that will arrange the contents of the intended digital library. The GOTHIC method results play a crucial role here.
- Authoring. Role(s) for creating new content or capturing existing contents, following the classification mechanisms and metadata definition.
- Workflow. Role(s) for defining processes for maintaining the OTS-wiki digital library, and for exploiting it adequately.
- Semantic knowledge acquisition (web intelligence). Role(s) for synthesising useful information from the daily use of the OTS-wiki digital library facilitating automation of activities and customization to user profiles, as well as quality-based information ranking mechanisms.
- Validation. Role(s) for defining experiments to validate the adequacy of the models, techniques, and methods proposed around the OTS-wiki digital library.
- Exploitation. Role(s) for making an industrial use of the OTS-wiki digital library and providing feedback to improve its structure and/or contents.

Moreover, since our project aims at providing an open and collaborative web community, we are aware that embracing coherent ethical guidelines is essential for building inclusive knowledge societies and raising awareness about the ethical aspects and principles in order to upholding the fundamental values of freedom, equality, solidarity, tolerance and shared responsibility.

List of Abbreviations

BA	Business Applications Case Study
CeBASE	Center for Empirically Based Software Engineering
CBS	COTS-Based Systems
CBSD	COTS-Based Systems Development
CBSE	Component-Based Software Engineering
COSI	Co-development using inner & Open Source in Software Intensive products. Research project.
COTS	Commercial-Off-The-Shelf software components
DALI	Methodologies and tools for the D evelopment, A cquisition, e vaLuation and I ntegration of software components. Research Project
DesCOTS	D escription, e valuation and s election of COTS components System
EF	Experience Factory
EFO	Experience Factory Organization
ERP	Enterprise Resource Planning
GESSI	Software Engineering for Information System Research Group
GBTM	Goal-Based Taxonomy Construction Method (previous to GOTHIC)
GOTHIC	Goal-Oriented Taxonomy and Reuse Infrastructure Construction Method
GBRAM	Goal-Based Requirements Analysis Method
GQM	Goal-Question-Metric approach
IQ	Information Quality
ITTG	Instituto Tecnológico de Tuxtla Gutiérrez
LEL	Language Extended Lexicon
LSO	Learning Software Organization
NTNU	Norwegian University of Science and Technology
OSD	Open Source Definition
OSI	Open Source Initiative
OSS	Open Source Software
OTS	Off-The-Shelf Software
QM	Quality Model
RE	Requirements Engineering
REST	Requirements Engineering Support Tools. Case Study
RTSC	Real-Time-Synchronous Communication
RiSD	Reduced <i>i</i> * SD models. Methodology
SAD	Software Application Development. Case Study
SEI	Software Engineering Institute at the Carnegie Mellon University
SU	Software Engineering group from NTNU

UPC	Technical University of Catalunya
UPIC	towards a Unified approach to the Procurement and Implementation of information system. Research Project
WWW	World Wide Web

Glossary

- ◆ Component-Based Development refers to the processes that lead to the development of a COTS-based software system
- ◆ Component-Based Software Engineering refers to a software system that is built mainly as the composition of COTS components, or as the customization of one single COTS component.
- ◆ Commercial-Off-The-Shelf components refer to third party components that acquired from the marketplace by a fee.
- ◆ COTS Marketplace: The COTS marketplace is characterized by a vast array of COTS components and products claims, extreme quality and capability differences between components, and many components incompatibilities, even when they purport to adhere to the same standards.
- ◆ COTS Domain: It refers to a field that defines a set of common requirements, terminology, and functionality for any COTS constructed to solve a problem in that field.
- ◆ COTS Selection: Refers to the processes of searching COTS candidates to fulfill the system requirements from the marketplace, evaluating them with respect to the system requirements for taking the final decision.
- ◆ COTS Searching: Exploring the marketplace to find COTS candidates and their related information.
- ◆ Goal: Goals are targets for achievement. They are high level objectives of the business, organizations or software components. They express the rationale for intended systems and guide decision at various levels.
- ◆ Goal-Oriented: Diverse methods and techniques are actually based on the notion of goals to reach several objectives.
- ◆ Open Source Software refers to components freely provided by Open Source Communities with some licensing obligations. See formal definition in [OSI].
- ◆ Off-The-Shelf Software refers to a software product that is publicly available at some cost or with some licensing obligations and other software projects can reuse and integrate it into their own products. The term includes COTS and OSS components.
- ◆ COTS consumers: The term refers to all subjects that use COTS to develop software systems. They range from single developers to IT organizations of any size.

- ♦ COTS market segments: In the COTS taxonomies constructed by GOTHIC, they refer to basic types of COTS available in the marketplace. Market segments can be considered as atomic entities covering a significant group of functionality.
- ♦ COTS categories: They serve to group too fine-grained functionalities of market segments.

References

- [Aas-Lar07] Aaslund, K., Larsen S.: “OTS-Wiki: A Web Community for Fostering Evaluation and Selection of Off-The-Shelf Software Components” Master Thesis. Department of Computer and Information Science, Norwegian University of Science and Technology (NTNU). Spring 2007. <http://www.idi.ntnu.no/grupper/su/su-diploma-2007/dipl07-larsen-aaslund.pdf>
- [Ack+02] Ackerman, J., Brinkop, F., Conrad, S., Fettke, P., Frick, A., Glistau E., Jaekel H., Kolar O., Loos, P., Merc., H., Orther, E., Raape, Overhage, S., Sahm, S., Schmietendorf, A., Teschke, T., and Turowski, K.: “Standardized Specification of Business Components”. Memorandum of the GI working group 5.10.3, 2002.
- [Ader03] Ader. M.: Workflow Comparative Study. 2003 Edition, available from www.waria.com
- [Agu05] Aguirre, J.: “IPSComp – Intelligent Portal for Searching Components”. Master Thesis. Vrije Universiteit Brussel-Belgium in collaboration with Ecole des Mines de Nantes-France. 2005.
- [Alm+06] Almeida, E.S., Cordeiro, J.C., Carvahlo, A.P., Alvaro, A., Cardoso Garcia, V., Romero de Lemos Meira, S., Lucrédio, D.: “The Domain Analysis Concept Revisited: A Practical Approach”. International Conference on Software Reuse, ICSR 2006: 43-57.
- [Alv03] Alves, C.: “COTS-Based Requirements Engineering” A. Cechich et al. (Eds.): Component-Based Software Quality, LNCS 2693, pp. 21–39, 2003. Springer-Verlag Berlin Heidelberg 2003
- [Alv-Cas01] Alves, C., Castro, J. “CRE: A Systematic Method for COTS Selection” Proceedings XV Brazilian Simposium on Software Engineering, 2001.
- [And04] Anderson, W.B.: “COTS Selection and Adoption in a Small Business Environment: How Do You Downsize the Process?” International Conference on COTS-Based Software Systems (ICCBSS 2004), p 216.
- [Ant97] Antón, A. I.: “Goal Identification and Refinement in the Specification of Software-Based Information Systems”. PhD. thesis, Georgia Institute of Technology, June 1997.
- [Ant+96] Anton, A.I. Liang, E. Rodenstein, R.A.: “A Web-based requirements analysis tool” In Proceedings of the 5th Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, 1996.
- [Ank+03] Ankolekar, A., Herbsleb, J., Sycara, K. “Addressing Challenges to Open Source Collaboration with Semantic Web”. In proceedings of 3rd Workshop on Open Source Software Engineering, the 25th International Conference on Software Engineering (ICSE). 2003. Portland, Oregon, USA, pp 9-14.
- [Aoy+98] Aoyama, M., Yamashita, T., Kobori, S.: “An Architecture of Software Commerce Broker Over The Internet”. In Proceedings of Worldwide Computing and Its Applications (WWCA'98). Springer Verlag, Volume 1368/1998, pp 97-107, 1998.

- [Ast+06] Astudillo, H., Pereira, J., López, C.: "Evaluating Alternative COTS Assemblies from Imperfect Component Information". *QoSA 2006*: 27-42
- [Ast+06b] Astudillo, H., Pereira, J., López, C.: "Identifying Interesting Component Assemblies for NFRs Using Imperfect Information". *EWSA 2006*: 204-211
- [Avi+99] Avison, D., Lau, F., Myers, M., Nielse, P.A.: "Action Research". *CACM* 42(1), 1999.
- [Aya-Fra08] Ayala, C., Franch, X.: "Assessing What Information Quality Means in OTS Selection Processes". International Conference on Composite-Based Software Systems (ICCBSS 2008). To appear. IEEE Society Press
- [Aya+07] Ayala, C., Sørensen, C.F., Conradi, R., Franch, X., Li, J.: "Open Source Collaboration for Fostering Off-The-Shelf Components Selection". In IFIP International Federation for Information Processing, Volume 234, Open Source Development, Adoption and Innovation. (OSS 2007). June 2007, pp. 17-30.
- [Aya-Fra07] Ayala, C., Franch X.: "A Systematic Approach to Manage Information Quality for Supporting Software Package Selection" Research Report. LSI-07-28-R. Universitat Politècnica de Catalunya, Departamento de Llenguajes y Sistemas Informàtics.
- [Aya06] Ayala, C.: "Systematic Construction of Goal-Oriented COTS Taxonomies" In Proceedings of the 3rd Doctoral Consortium at the 18th Conference on Advanced Information Systems Engineering (CAISE 2006). 5-9 June 2006, Luxembourg.
- [Aya-Fra06a] Ayala, C., Franch, X.: "A Goal-Oriented Strategy for Supporting Commercial Off-The-Shelf Components Selection" In Proceedings of the 9th International Conference on Software Reuse (ICSR). Torino, Italy. Lecture Notes in Computer Science. Volumen: 4039-2006. pp. 1-15. June 2006.
- [Aya-Fra06b] Ayala, C., Franch, X.: "Overcoming COTS Marketplace Evolvability and Interoperability". In Proceedings of CAISE Forum at 18th Conference on Advanced Information Systems Engineering (CAISE 2006) June 2006, Luxembourg.
- [Aya-Fra06c] Ayala, C.; Franch, X.: "Domain Analysis for Supporting Commercial Off-The-Shelf Components Selection". In Proceedings of the 25th International Conference on Conceptual Modelling (ER 2006). Tucson, Arizona, USA. Lecture Notes in Computer Science. Volumen: 4215/2006. Pages: 354-370.
- [Aya-Fra06TRa] Ayala, C., Franch, X.: "Domain Analysis for Supporting Commercial Off-The-Shelf Components Selection" (Extended Version) Research Report LSI-06-16-R. Universitat Politècnica de Catalunya, Departamento de Llenguajes y Sistemas Informàtics.
http://www.lsi.upc.edu/dept/techreps/l1istat_detallat.php?id=916
- [Aya-Fra06tr] Ayala, C., Franch, X.: "A Process for Building Goal-Oriented COTS Taxonomies" Research Report LSI-06-7-R. Universitat Politècnica de Catalunya, Departamento de Llenguajes y Sistemas Informàtics.
http://www.lsi.upc.edu/dept/techreps/l1istat_detallat.php?id=907
- [Aya05PT] Ayala, C. "Systematic Construction of Goal-Oriented COTS Taxonomies".

Thesis Project presented to fulfill the requirements of "Diploma de Estudios Avanzados" and PhD Thesis Evaluation. July 2005. Available at: <http://www.lsi.upc.edu/~cayala/Papers/AyalaThesisProject05.pdf>

- [Aya-Fra05] Ayala, C.; Franch, X. "Transforming Software Package Classification Hierarchies into Goal-Based Taxonomies" In Proceedings of the 16th International Conference on Database and Expert Systems Applications (DEXA 2005). Copenhagen, Denmark. 22-26 August 2005. Lecture Notes in Computer Science. Volumen: 3588/2005. pp. 665- 675.
- [Aya+05a] Ayala, C.P., Botella, P., Franch, X.: "On Goal-Oriented COTS Taxonomies Construction" In Proceedings of the 4th International Conference on COTS-Based Software Systems (ICCBSS 2005). Bilbao, Spain. Lecture Notes in Computer Science. Volumen: 3412/2005. pp. 90-100.
- [Aya+05b] Ayala, C.; Cares, C.; Carvallo, J.P.; Grau, G.; Haya, M.; Salazar, G.; Franch, X.; Mayol, E.; Quer, C.: "A Comparative Análisis of i^* -Based Goal-Oriented Modelling Languajes" In Proceedings of the International Workshop on Agent-Oriented Software Development Methodology (AOSDM 2005), at the SEKE Conference. July 2005. Taipei, Taiwán; China. pp.43-50.
- [Aya+05c] Ayala, C., Botella, P., Franch, X.: "Construction of a Taxonomy for Requirements Engineering Comercial-Off-The-Shelf Components" *Journal of Computer Science and Technology, Special Issue on Software Requirements Engineering* Vol. 5, No. 2, August 2005.
- [Aya+05TR] Ayala, C.P., Botella, P., Franch, X. "Goal-Based Reasoning in the Construction of Taxonomies for COTS Components" Technical Report LSI-05-58-R. Universitat Politècnica de Catalunya, Departamento de Lenguajes y Sistemas Informáticos.
- [Aya+04a] Ayala, C.; Botella, P.; Franch, X. "Goal-Based Reasoned Construction of Taxonomies for the Selection of COTS Products" In Proceedings of the 8th World Multiconference on Systemics, Cybernetics and Informatics (SCI 2004). July 18-21, Orlando, Florida, USA.
- [Aya+04b] Ayala, C.; Cares, C.; Carvallo, J.P.; Grau, G.; Haya, M.; Salazar, G.; Franch, X.; Mayol, E.; Quer, C.: "Análisis Comparativo de Lenguajes de Modelado Orientados a Objetivos basados en i^* " In Proceedings of the Jornadas Iberoamericanas de Ingeniería del Software e Ingeniería del Conocimiento (JIISIC'04). Madrid, Spain. 2004. Pages: 527-540.
- [Aya+04c] Ayala, C.; Botella, P.; Franch, X.: "Construccion de una Taxonomía de Componentes COTS Orientados a la Gestión de Requisitos" In Proceedings of the VII Workshop on Requirements Engineering. Tandil, Argentina. December 2004. ISBN 950-658-147-9. pp. 214-225.
- [Aya+04TR] Ayala, C. P., Botella, P., Franch, X.: "Towards the Definition of a Taxonomy for the COTS Product's Market" Technical Report LSI-04-3-R. Universitat Politècnica de Catalunya, Departamento de Lenguajes y Sistemas Informáticos. 2004.
- [Bai94] Bailey, K-D.: "Typologies and Taxonomies: An Introduction to Classification Techniques". Sage, Thousand Oaks, CA (1994).

- [Ban06] Bandor, M.S.: “Quantitative Methods for Software Selection and Evaluation“ CMU/SEI-2006-TN-026. September 2006.
- [Ban-Da02] Bansiya, J., Davis, G.: A Hierarchical Model for Object-Oriented Design Quality Assessment. *IEEE Transactions on Software Engineering*. 28(1): 4-17 (2002).
- [Bar+05] Barthelet, R. G., D. C. Brogan, P. F. Reynolds, Jr.: ” The Computational Complexity of Component Selection in Simulation Reuse”. Proceedings of the 37th conference on Winter simulation 2005. pages 2472 - 2481 .
- [Bas+04] Basili, V.R., Boehm, B., Davis, A., Humphrey, W.S., Leveson, N., Mead, N.R., Musa, J.D., Parnas, D.L., Pfleger, S.L., Weyuker, E. : “New Year’s Resolution for Software Quality”, Quality Time, J. Hayes Eds. *IEEE Software* 21, 1 (January/February 2004) pp. 12-13.
- [Bas+00] Bass, L., Buhman, C., Comella-Dorda, S., Long, F., Robert, J., Seacord, R., and Wallnau, C. Volume I - Market assessment of Component-Based Software Engineering. Technical report, CMU/SEI - Carnegie Mellon University/Software Engineering Institute, 2000.
- [Bas92] Basili; V.R.: “Software Modeling and Measurement. The Goal-Question-Metric Paradigm”. Computer Science Technical Report Series NR: UMIACS-TR92-96, 1992.
- [Bas+94] Basili, V., Caldiera, G., Rombach, D. Goal/question/metric paradigm. In *Encyclopedia of Software Engineering*. Vol. 1, 1994. J. C. Marciniak, Ed. John Wiley and Sons, New York.
- [Bas+94b] V. R. Basili, G. Caldiera, and H. D. Rombach. Experience Factory. In J. J. Marciniak (ed.), *Encyclopedia of Software Engineering*, Volume 1, John Wiley & Sons, 1994.
- [Bas-Boe01] Basili, V.R., Boehm, B.: "COTS-Based Systems Top 10 List", *IEEE Computer*, Vol. 34, No. 5, May 2001.
- [Berg06] Berg, M.A. “Attitudes to formal Quality Management Systems An Empirical Study in Norwegian Software Industry” Master Thesis. Norwegian University of Science and Technology (NTNU). 2006.
- [Beu-Bøe03] Beus-Dukic, L., Bøegh, J.: COTS Software Quality Evaluation. In Proceedings of the International Conference on COTS-Based Software Systems ICCBSS 2003, LNCS Volume 2580/2003 pp. 72-80
- [Ber+03] Bertoa, M.F., Troya, J.M., Vallecillo, A. “A Survey on the Quality Information Provided by Software Component Vendors”. In Proceedings of the 7th ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering (QAOOSE), 2003, 25-30.
- [Ber+06] Bertoa M., Troya, J.M, Vallecillo, A.: Measuring the Usability of Software Components. *The Journal of Systems and Software* 79 (2006) 427–439.
- [Bhu+07] Bhuta, J., Mattmann, C., Medvidovic, N., Boehm, B.: "A Framework for the Assessment and Selection of Software Components and Connectors in COTS-Based Architectures," p. 6. Sixth Working IEEE/IFIP Conference on Software Architecture (WICSA'07), 2007.

- [Bhu-Boe07] Bhuta, J., Boehm, B.: "Attribute-Based COTS Product Interoperability Assessment" International Conference on COTS-Based Software Systems (ICCBSS) 2007, Alberta, Canada, February/March 2007.
- [Bhu-Boe05] Bhuta, J., Boehm, B.: "A Method for Compatible COTS Selection". Proceedings of the International Conference on COTS-Based Systems ICCBSS 2005. pp. 132-143.
- [Bia+03] Bianchi, A., Caivano, D., Conradi, R., Jaccheri, M.L., Torchiano, M., Visaggio, G.: "COTS Products Characterization: Proposal and Empirical Assessment". ESERNET 2003: 233-255.
- [Birk+98] Birk, A., Kempkens, R., Rombach, D., Ruhe, G.: "Systematic Improvement of Software Engineering Processes". In Proceedings of WI'98, Hamburg, Germany 1998.
- [Bla-Mar85] Blair, D.C., Maron, M.E.: "An Evaluation of Retrieval Effectiveness for a Full-Text Document Retrieval System" *Communications of the ACM*, 28 (4), pp.289-299. 1985.
- [Bob+03] Bobrovsky, M., Marré, M., Yankelevich: "A Software Engineering View of Data Quality", 2003
- [Boe-Abt99] Boehm, B., Abts, C. "COTS Integration: Plug and Pray?" *IEEE Computer*, Vol. 32, No. 1; January 1999 pp. 135-138.
- [Boe+03a] Boehm, B., Port, D., Yang, Y., Bhuta, J., Abts, C.: "Composable Process Elements for Developing COTS-Based Applications", In Proceedings of the International Symposium on Empirical Software Engineering (ISESE 2003). IEEE Computer Society Press.
- [Boe+03b] Boehm, B., Port, D., Yang, Y.: "WinWin Spiral Approach to Developing COTS-Based Applications" EDSET-5, Oregon 2003.
- [Boe+78] Boehm, B.W., Brown, J.R., Kaspar, H., Lipow, M., McLeod, G.J., Merrit, M.J.: "Characteristics of Software Quality", *North Holland Publishing Company*, 1978.
- [Bøe+99] Bøegh, J., Depanfilis, S., Kitchemham, B., Pasquini, A.: A Method for Software Quality Planning, Control and Evaluation. *IEEE Software*, Vol. 23. March 1999.
- [BOO93] BOOTSTRAP team: BOOTSTRAP: Europe's Assessment Method. *IEEE Software*. May 1993.
- [Boo-Smi00] Boop, R.E., Smith, L.: "Reference and Information Services: An Introduction". *Libraries Unlimited*, 2000.
- [Bra+91] Brachman, R.J., McGuinness, D.L., Patel-Schneider, P.F., Resnik, L.A., Borgida, A.: "Living with CLASSIC: When and How to Use a KL-ONE-Like Language" In *Principles of Semantic Networks: Exploration in the Representation of Knowledge*, J.F. Sowa, Ed. *Morgan Kaufmann*, San Mateo, CA, 401-456.
- [Bre+84] Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J., Classification and Regression Trees. *Belmont, CA: Wadsworth International Group*, 1984.

- [Bro+00] Brownsword, L., Oberndorf, T., Sledge C.A.: "Developing New Processes for COTS-Based Systems" *IEEE Software*, Vol. 17, No. 4; July-August 2000. pp. 48-55.
- [Bro+98] Brownsword, L., Carney, D., Oberndorf, T.: "The Opportunities and Complexities of Applying Commercial-Off-the-Shelf Components" CrossTalk SEI series. April 1998. Available at <http://www.sei.cmu.edu/news-at-sei/features/1998/jun/background.pdf>
- [Bug-Abr99] Buglione, L., Abran, A.: "A Quality Factor for Software". Qualta-Congress: quality and reliability. Paris, France. March 25-26, 199.
- [Bur+07] Burgess, M.S.E., Gray, W.A., Fiddian, N.J.: Using Quality Criteria to Assist in Information Searching. *International Journal in Information Quality*. Vol 1, No. 1, 2007.
- [Bur+04] Burgess, M.S.E., Gray, W.A., Fiddian, N.J.: "Quality Measures and the Information Consumer" Proceedings of the 9th International Conference on Information Quality (ICIQ-04)
- [Bur+02] Burgués, X., Estay, C., Franch, X., Pastor, J.A., Quer, C.: "Combined Selection of COTS Components". In Proceedings of the 1st International Conference on COTS-Based Software Systems (ICBSS 2002). Orlando, Florida. LNCS 2255, 2002.
- [Car-Lon00] Carney D., Long F.: What Do You Mean by COTS? Finally a Useful Answer. *IEEE Software*, 17 (2), March/April 2000.
- [Carn+03] Carney, D., Place, P.R.H., Oberndorf, P.: "A Basis for an Assembly Process for COTS-Based Systems (APCS)". SEI *Technical Report*. CMU/SEI-2003-TR-010.
- [Car+03] Carvallo, J.P., Franch, X., Quer, C.: Defining a Quality Model for Mail Servers. In Proceedings of the 2nd International Conference on COTS-Based Software Systems ICBSS 2003. pp. 51-61.
- [Car+04] Carvallo, J.P., Franch, X., Quer, C., Torchiano, M. "Characterization of a Taxonomy for Business Applications and the Relationships Among Them" In Proceedings of the 3rd International Conference on COTS-Based Software Systems (ICBSS), LNCS 2959, 2004.
- [Car+04a] Carvallo, J.P., Franch, X., Grau, G., Quer, C.: "QM: A Tool for Building Software Quality Models". In Proceedings of the 12th IEEE Requirements Engineering International Conference RE 2004. Kyoto, Japan. IEEE Computer Society. 2004. pp. 358-359.
- [Car+04b] Carvallo, J.P., Franch, X., Quer, C., Rodríguez, N. "A Framework for Selecting Workflow Tools in the Context of Composite Information Systems" In Proceedings of the 15th Database and Expert Systems Applications Conference (DEXA), LNCS 3180, 2004.
- [Car+04c] Carvallo, J.P., Franch, X., Grau, G., Quer, C.: "COSTUME: A Method for Building Quality Models for Composite COTS-Based Software Systems" In Proceedings of the 4th International Conference on Quality Software (QSIC 2004). Braunschweig, Germany. IEEE Computer Society. September 2004.

- [Car05T] Carvallo, J.P.: "Systematic Construction of Quality Models for COTS-Based Systems". PhD Thesis. Technical University of Catalunya (UPC). GESSI Group. 2005
- [Car+05] Carvallo, J.P., Franch, X., Quer, C. "A Quality Model for Requirements Management Tools". Book chapter in Requirements Engineering for Sociotechnical Systems, Idea Group, 2005.
- [Car06] Carvallo, J.P. "Supporting Organizational Induction and Goals Alignment for COTS Components Selection by means of i*". In Proceedings of the 5th International Conference on COTS-Based Systems (ICCBSS), IEEE Computer Society, 2006.
- [Car-Fra06] Carvallo, J.P., Franch, X. "Extending the ISO/IEC 9126-1 Quality Model with Non-Technical Factors for COTS Components Selection". In Proceedings of the 4th ICSE Workshop of Software Quality (WoSQ), ACM Digital Library, 2006.
- [Car+07a] Carvallo, J.P., Franch, X., Quer, C.: "Towards a Unified Catalogue of Non-Technical Quality Attributes to Support COTS-Based Systems Lifecycle Activities" Sixth International Conference on Commercial-off-the-Shelf (COTS)-Based Software Systems, 2007. IEEE Computer Society. 2007.
- [Car+07b] Carvallo, J.P., Franch, X., Quer, C.: Determining Criteria for Selecting Software Components: Lessons Learned. *IEEE Software*. March/April 2007.
- [Cec+06] Cechich, A., Réquilé-Romanczuk, A., et al. "Trends on COTS Component Identification and Retrieval" In Proceedings of the International Conference on COTS-Based Software Systems (ICCBSS 2006). IEEE Computer Society, 2006.
- [Cha02] CHAOS Virtual BEACON. "The Cost of ERP". Standish Group, 2002.
- [Che76] Chen, P.: The Entity-Relationship Model –Towards a Unified View of Data. *ACM Transactions on Database Systems*, 1(1), March 1976.
- [Chu-Coo04] Chung, L., Cooper, K.: Defining Goals in a COTS-Aware Requirements Engineering Approach. *System Engineering*, Vol. 7, No.1, 2004
- [Chu+00] Chung, L., Nixon, B.A., Yu, E., Mylopoulos, J.: Non-Functional Requirements in Software Engineering. *Kuwer Academic Publishers*, 2000
- [ClaSoc] Classification Society of North America <http://www.classification-society.org/csna/csna.html>
- [Cla+04] Clark, J., Clarke, C., De Panfilis, S., Granatella, G., Predonzani, P., Sillitti, A., Succi, G., Vernazza, T.: "Selecting Components in Large COTS Repositories" *The Journal of Systems and Software* 73 (2004) 323–331.
- [CMM93] Paulk, M.C., Curtis, B., Chrissis, M.B., Weber, C.V. The Capability Maturity Model for Software. Technical Report CMU/SEI-93-TR-024. Software Engineering Institute, Carnegie Mellon University. February 1993.
- [Coc01] Cockburn, A. Writing Effective Use Cases. Addison-Wesley, 2001.

- [Coh-Nor98] Cohen, S., Northrop, L. "Object-Oriented Technology and Domain Analysis". In Proceedings of the 5th International Conference on Software Reuse (ICSR), 1998.
- [Coo+05] Cooper, K., Cangussu, J.W., Lin, R., Sankaranarayanan, G., Soundararadjane, R., Wong, E.: "An Empirical Study on the Specification and Selection of Components Using Fuzzy Logic" G.T. Heineman et al. (Eds.): CBSE 2005, LNCS 3489, pp. 155-170, 2005.
- [Coo97] Cooper, W.S.: On selecting a measure of retrieval effectiveness. In K. Jones and P. Willet (eds.), *Readings in Information Retrieval*, pages 191–204. Morgan Kaufmann Publishers, 1997.
- [Cor+02] Comella-Dorda, S., Dean, J.C., Morris, E., and Oberndorf, P. "A process for COTS Software Product Evaluation", Proceedings of ICCBSS, February 2002, Orlando, Florida USA, pp 86-92.
- [Corn96] Cornwell, P.C.: HP Domain Analysis: Producing Useful Models for Reusable Software. *Hewlett-Packard Journal*, August 1996.
- [Crn+06] Crnkovic, I., Chaudron, M.R.V., Larsson, S.: Component-Based Development Process and Component Lifecycle. International Conference on Software Engineering Advances (ICSEA'06), pp. 44. 2006
- [Crn+05] Crnkovic, I., Larsson, S., Chaudron, M.: Component-Based Development Process and Component Lyfecycle. *Journal of Computing and Information Technology- CIT* 13, 2005, 4, pp. 321-327.
- [Crn-Lar02] Crnkovic, I., Larsson, S.: Challenges of Component-Based Development. *Journal of Software System*. April 2002. Elsevier Science Home.
- [COS] COSI: Co-development using inner & Open Source in Software Intensive products. European ITEA project.
- [Cur89] Curtis, B. "Cognitive Issues in Reusing Software Artifacts" *Software Reusability: Volume II: Applications and Experience*, T.J. Biggerstaff and A.J. Perlis, Eds., Addison-Wesley, Reading, M. 269-287.
- [DAL] DALI: Methodologies and tools for the Development, Acquisition, evaLuation and Integration of software components Research Project. CICYT Program (MEC), project TIC2001-2165. GESSI Group <http://www.lsi.upc.es/~webgessi/index.html>
- [Dev+91] Devanbu, P., Brachman, R.J., Selfridge, P.G., Ballard, B.W.: "LaSSIE: A Knowledge-Based Software Information System" *Communications of the ACM*, 34(5), pp. 34-49.
- [Din-Con02] Dingsøy, T. and Conradi, R.: "A Survey of Case Studies of the Use of Knowledge Management in Software Engineering", *Journal of Software and Knowledge Engineering*, 12(4):391-414, 2002.
- [Dub-Fra04] Dubois, E., Franch, X.: "Models and Processes for the Evaluation of COTS Components". 26th International Conference on Software Engineering (ICSE) 2004: 759-760.
- [Don+05] Donzelli, P., Zekowitz, M., Basili, V., Allar, D., Meyer, K.N.: Evaluating

- COTS Components Dependability in Context. *IEEE Software* July/August 2005.
- [Dro96] Dromey, R.G.: Cornering the Chimera. *IEEE Software*, Vol. 20, January 1996.
- [Dro95] Dromey, R.G.: A Model for Software Product Quality. *IEEE Transactions on Software Engineering*, 21:146-162, 1995.
- [El+98] El Emam, K., Drouin, J.N., Melo, W. (Eds.): "Spice: The Theory and Practice of Software Process Improvement and Capability Determination". *IEEE Computer Society*, 1998.
- [Ero-Fer03] Erol, I., Ferrell-Jr., W.G.: A Methodology for Selection Problems with Multiple, Conflicting Objectives and Both Qualitative and Quantitative Criteria. *International Journal of Production Economics* vol. 86 (3), Dec 2003. pp. 187-199
- [Ero-Gia06] Erofeev, S., DiGiacomo, P. "Usage of Dynamic Decision Models as an Agile Approach to COTS Taxonomies Construction". In Proceedings of the ICCBSS 2006. IEEE Society Press 2006.
- [Eva05] Evans Data Corporation, "Open Source/Linux Development Survey", Spring 2005.
- [Fen-Pfl97] Fenton, N.E., Pfleeger, S.L. *Software Metrics: A Rigorous and Practical Approach*. 2nd Edition, 1997.
- [Fer-Veg99] Ferré, X., Vegas, S. "An Evaluation of Domain Analysis Methods". In Proceedings 4th CAiSE Workshop on Exploring Modelling Methods for Systems Analysis and Design (EMMSAD), 1999.
- [Fin+96] Finkelstein, A., Spanoudakis, G., Ryan, M. "Software Package Requirements & Procurement" Proc. International Workshop on Software Specification and Design (IWSSD), IEEE Computer Society Press, 1996, pp. 141-145
- [Fir03] Firesmith, D.G.: Using Quality Models to Engineer Quality requirements. *Journal of Object Technology (JOT)*. Vol. 2, No. 5 (September/October 2003).
- [FOD] SEI <http://www.sei.cmu.edu/domain-engineering/FODA.html>
- [Fra+07] Franch, X.; Grau, G.; Mayol, E.; Quer, C.; Ayala, C.; Cares, C.; Navarrete, F.; Haya, M.; Botella, P.: "Systematic Construction of i* Strategic Dependency Models for Socio-Technical Systems" *International Journal of Software Engineering and Knowledge Engineering (IJSEKE)*. Volume 17, No. 1, February 2007.
- [Fra05] Franch, X.: "On the Lightweight Use of Goal-Oriented Models for Software Package Selection" In Proceedings of the 17th Conference on Advanced Information Systems Engineering (CAISE 2005). Lecture Notes in Computer Science. Volume 3520/2005 pp. 551-566
- [Fra-Tor05] Franch, X., Torchiano, M.: "Towards a Reference Framework for COTS-Based Development: A Proposal". Proceedings of MPEC'05. St. Louis Missouri, USA. ACM.

- [Fra-Car03] Franch, X., Carvallo, J.P. "Using Quality Models in Software Package Selection". *IEEE Software*, 20(1), 2003.
- [Fra-Mai03] Franch, X., Maiden, N.: "Modelling Component Dependencies to Inform their Selection". In Proceedings of the 2nd International Conference on COTS-Based Software Systems (ICCBSS 2003). Lecture Notes in Computer Science. Volume 2580/2003 pp. 81-91. February 2003.
- [Frak05] Frakes, W.B.: "A case study of a reusable component collection in the information retrieval domain". *Journal of Systems and Software* Vol. 72, Issue 2, July 2004, Pages 265-270.
- [Fra-Gan90] Frakes, W., Gandel, P.: Representing Reusable Software. *Information Software Technology* vol. 32, pp. 47-54, 1990.
- [Fra-Kan05] Frakes, W.B., Kang, K.: Software Reuse Research: Status and Future. *IEEE Transactions on Software Engineering*. July 2005 Vol. 31, No. 5. pp. 529-536.
- [Frak+98] Frakes, W., Prieto-Díaz, R., Fox, C. "DARE: Domain Analysis and Reuse Environment". *Annals of Software Engineering*, 5, pp. 125-141, 1998.
- [Fra-Fox95] Frakes, W.B., Fox, C.J.: Sixteen Questions About Software Reuse. *Communications of the ACM*. Volume 38, Number 6 pp 75-87, 1995.
- [Fra-Pol94] Frakes, W.B., Pole, T.P.: An Empirical Study of Representation Methods for Reusable Software Components. *IEEE Transactions on Software Engineering*, Vol. 20, No. 8, pp. 617-630. August 1994.
- [Gar] Garner Group <http://www3.gartner.com>
- [Ger06] Gereaa, M.: "Selection and Evaluation of Open Source Components", 15th Dec. 2006, 81 p. part of course TDT4735 Depth Project in Software Engineering, Department of Computer and Information Science, Norwegian University of Science and Technology (NTNU). <http://www.idi.ntnu.no/grupper/su/fordypningsprosjekt2006/gereafordyp06.pdf>
- [Ger07] Gereaa, M.: "Selection of Open Source Components: A Qualitative Survey in Norwegian IT Industry". Master Thesis. Department of Computer and Information Science, Norwegian University of Science and Technology (NTNU). Spring 2007. <http://www.idi.ntnu.no/grupper/su/su-diploma-2007/dipl07-gereaa.pdf>
- [Gils+04] Gils, B. v., Proper, H., Bommel, P. v.: A Conceptual Model of Information Supply. *Journal of Data Knowledge Engineering*, Vol 51, Issue 2, 2004. pp 189-222
- [Gil97] Gillies, A.C.: Software Quality, Theory and Management. *International Thompson Computer Press*, 1997.
- [Gil88] Gilb, T.: Principles of Software Engineering Management. *Addison Wesley*, reading MA, 1998.
- [Gla-Ves95] Glass, R.L., Vessey, I. "Contemporary Application-Domain Taxonomies" *IEEE Software*. July 1995.

- [Gla94] Glass, R.L.: The Software Research Crisis. *IEEE Software*, November 1994, pp. 42-47.
- [Gom05] Gomaa, H. Designing Software Product Lines with UML: From Use Cases to Pattern-Based Software Architectures. *Addison-Wesley*, 2005.
- [Gra+04] Grau, G., Carvallo, J.P., Franch, X., Quer, C. "DesCOTS: A Software System for Selecting COTS Components". In Proceedings of the 30th EUROMICRO Conference, IEEE Computer Society, 2004. The current version of the tool is available at <http://www.lsi.upc.es/~gessi/QMTool/QMTool.html>
- [Gra+05] Grau, G.; Franch, X.; Mayol, E.; Ayala, C.; Cares, C.; Carvallo, J.P.; Haya, M.; Navarrete, F.; Botella, P.; Quer, C.: "RiSD: A Methodology for Building i* Strategic Dependency Models" In Proceedings of the 17th International Conference on Software Engineering and Knowledge Engineering (SEKE'05). 14-16 July, 2005. Taipei, Taiwan, Republic of China. pp. 259-266.
- [Gra+05b] Grau, G., Franch, X., Maiden, N.A.M.: "REDEPEND-REACT: an Architecture Analysis Tool". In Proceedings of the 13th IEEE Requirements Engineering International Conference, RE 2005. Pages: 455 - 456.
- [Gre+02] Gregor, S., Hutson, J., Oresky, C.: "Storyboard Process to Assist in Requirements Verification and Adaptation to Capabilities Inherent in COTS". In Proceedings of the International Conference on COTS-Based Software Systems 2002. LNCS Volume 2255/2002.
- [Gree94] Green, S. "Goal-Driven Approaches to Requirements Engineering" Technical Report DoC TR-93-42 1994, Imperial College of Science, Technology and Medicine, Department of Computing Technical Report, London, UK, 1994.
- [Gri02] Grid, M.: The ROI on COTS. *Journal of the Global Grid Community*. Vol. 1, No.12, September 2, 2002.
- [Grub93] Gruber, T.R. "Toward Principles for the Design of Ontologies Used for Knowledge Sharing", KSL-93-04, Knowledge Systems Laboratory, Stanford University.
- [Han-Kam01] Han, J., Kamber, M.: Data Mining: Concepts and Techniques. *Academic Press, Morgan Kaufmann Publishers*, 2001.
- [Hau02] d'Haultfoeuille, M.: "Projet ERP quelle structure choisir". JDNet Solutions (Benchmark Group), 2002.
- [Hen+05] Henderson-Sellers, B., González-Pérez, C., Serour, K., Firesmith, D.G.: "Method Engineering and COTS Evaluation". In Proceedings of the ACM SIGSOFT 2005.
- [Hen97] Henninger, S.: "An Evolutionary Approach to Constructing Effective Software Reuse Repositories". *ACM Transactions on Software Engineering and Methodology (TOSEM)* Volume 6 , Issue 2 (April 1997) pp.111 – 140. 1997
- [How-Lig06] Howcroft, D., Light, B.: Reflections on Issues of Power in Packaged Software Selection. *Information Systems Journal* 16 (3), 215–235.

- [Hya-Ros96] Hyatt, L.E., Rosenberg, L.H.: "A Software Quality Model and Metrics for Software Quality Assurance", from Project Control for Software Quality. Kusters, R., Cowderoy, A., Heemstra, F., van Veenendaal, E. (Eds.). Shaker Publishing, 1999.
- [IDC] IDC. Available at <http://www.idc.com/>
- [IEEE1661-98] IEEE Standard for Software Quality Metrics Methodology, 1998.
- [INC] INCOSE "International Council of Systems Engineering", Available at <http://www.incose.org/rwg/>
- [ISO12207] ISO/IEC International Standard 12207, Information Technology - Software Lifecycle Processes. (1995).
- [ISO9126] ISO/IEC International Standard 9126-1. Software Engineering-Product Quality-Part 1: Quality Model, 2001.
- [Jac-Tor02] Jaccheri, L., Torchiano, M. "Classifying COTS Products" Proceedings 7th European Conference of Software Quality (ECSQ), Helsinki, Finland, 2002, pp. 246-255
- [Jed-Nic03] Jedlitschka, A., Nick, M.: "Software Engineering Knowledge Repositories" Experiences from ESERNET 2003. LNCS Volume 2765/2003 pp. 55-80.
- [Jen03] Jensen, R.W.: Lessons Learned From Another Failed Software Contract. *CrossTalk, The Journal of Defense Software Engineering*, September 2003.
- [Jin+05] Jinfang, S., Songqiao, C., Bin, W.: COTS Evaluation and Selection Based on Requirements Decomposition. *Chinese Journal of Electronics*, pp 62-67, 2005.
- [Jør-Mol06] Jørgensen, M., Moløkken-Østfold, K.: How Large Are Software Cost Overruns? A review of the 1994 CHAOS report. *Information and Software Technology*, 48(4):297-301, April 2006.
- [Kau05] Kaur, N.: "Retrieving Best Components From Reusable Repository". Master Thesis. Computer Science and Engineering Department. Thapar Institute of Engineering and Technology (Deemed University). June 2005.
- [Kav-Lou05] Kavakli, E., Loucopoulos, P.: "Goal Modeling in Requirements Engineering: Analysis and Critique of Current Methods". *Information Modeling Methods and Methodologies 2005*: 102-124
- [Kei-Tiw05] Keil, M., Tiwana, A.: Beyond Cost: The Drivers of COTS Application Value. *IEEE Software*, May/June 2005 pp. 64-69.
- [Kel95] Kelly, G.A. "The Psychology of Personal Constructs" New York, W.W. Norton.
- [Kel+90] Keller, S., Kahn, L., Panara, R.: "Specifying Software Quality Requirements with Metrics". *Systemas and Software Requirements Engineering – IEEE Computer Society Press –Tutorial (1990)*, pp. 145-163.
- [Kelk+07] Kelkar, M., Perry, P., Gamble, R., Walkevar, A. "The Impact of Certification Criteria on Integrated COTS-Based Systems". *International Conference on COTS-Based Software Systems (ICCBSS) 2007*.

- [Kes07] Kessler, E.: "Assessing COTS Software in a Certifiable Safety Critical Domain". *Informarion Systems Journal* (in press).
- [Kit-Pfl96] Kitchenham, B., Pfleeger, S.L.: Software Quality: The Elusive Target. *IEEE Software*, Vol. 13, No. 1. January 1996, pp 12-21.
- [Kit89] Kitchenham, B.: Software Metrics. In *Software Reliability Handbook*, Elsevier, 1989.
- [Kle-Kaz99] Klein, M., Kazman, R.: "Attribute-Based Architectural Styles". Software Engineering Institute. Carnegie Mellon University. CMU/SEI-99-TR-022, October 1999.
- [Kaz-Kle00] Kazman, R., Klein, M., et al. "ATAM: Method for Architecture Evaluation". Pittsburg (USA). Software Engineering Institute. Carnegie Mellon University, 2000.
- [Kon-Hut07] Kotonya, G., Hutchinson, J.: "A Service-Oriented Approach for Specifying Component -Based Systems" In Proceedings of the 6th International Conference on COTS-Based Software Systems (ICCBSS) 2007. pp. 150-162.
- [Kon96] Kontio, J. "A Case Study in Applying a Systematic Method for COTS Selection". Proceedings 18th International Conference on Software Engineering, IEEE Computer Society Press.
- [Kon+96] Kontio, J., Caldiera, G., Basili, V.R.: "Defining factors, goals and criteria for reusable component evaluation" In CASCON'96. Toronto, Ontario, Canada: IBM Press, 1996.
- [Kon95] Kontio, J. "OTSO: A Systematic Process for Reusable Software Component Selection", University of Maryland, Maryland CS-TR-3478, December 1995.
- [Kry+03] Krystkowiak, M., Bucciarelli, B., Dubois, E.: "COTS Selection for SMEs: A Report on a Case Study and on a Supporting Tool". Proceedings of the 1st RECOTS Workshop, September 2003.
- [Kun03] Kunda, D.: "STACE: Social Technical Approach to COTS Software Evaluation" ". In Proceedings of Component-Based Software Quality - Methods and Techniques LNCS Volume 2693/2003 pp. 64-84
- [Kun-Bro99] Kunda, D., Brooks, L.: "Applying Socio-Technical Approach for COTS Selection" In Proceedings of 4th UKAIS Conference, University of York, April 1999. Mc Graw Hill. pp. 552-565.
- [Lam01] Lamsweerde, A.V. "Goal-oriented requirements engineering: A Guided Tour" Proceedings International Symp. On Requirements Engineering (RE'01). Toronto, Canada, 2001, pp. 249-263.
- [Lau-Ped05] Lausen, S., Vium, J.P.: Communication Gaps in a Tender Process. *Requirements Engineering Journal*. September 2005, pp. 247-261.
- [Lee+01] Lee, Y.W., Strong, D.M., Kahn, B.K., Wang, R.Y. "AIMQ: A Methodology for Information Quality Assessment" Elsevier, Information and Management. 2001.
- [Lei-Fra93] Leite, J.C.S.P., Franco, A.P.M.: "A Strategy for Conceptual Model

- Acquisition” Proceedings of the First IEEE International Symposium on Requirements Engineering. San Diego Ca., IEEE Computer Society Press, 1993, pp. 243-246.
- [Lei89] Leite, J.C.S.P. “Application Languages: A Product of Requirements Analysis”. Informatics Department PUC-/RJ (1989).
- [Leu-Leu03] Leung, H. K. N. and Leung, K. R. P. H.: “Domain-Based COTS-Product Selection Method”. In Proceedings of Component-Based-Software Quality, LNCS 2693, pp 40-63, 2003
- [Li06] Li, J.: “Process Improvement and Risk Management in Off-the-Shelf Component-based Development”. PhD Thesis Sept. 2006. Norwegian University of Science and Technology (NTNU), ISBN 82-471-7920-2, 289 pages,
<http://www.idi.ntnu.no/grupper/su/publ/phd/liphdthesis-22jun06.pdf>
- [Li+06] Li, J., Conradi, R., Slyngstad, O.P.N., Bunse, C., Torchiano, M., Morisio, M.: "An Empirical Study on the Decision Making Process in Off-The-Shelf Component Based Development", In Leon J. Osterweil, H. Dieter Rombach, and Mary Lou Soffa (Eds.): Proc. Emerging Results track at the 28th International Conference on Software Engineering (ICSE 2006), 20-28 May 2006, Shanghai, P.R. China, pp. 897-900.
- [Li+05] Li, J., Conradi, R., Slyngstad, O.P.N., Torchiano, M., Morisio, M., and Bunse C.: "An Empirical Study on Off-the-Shelf Component Usage in Industrial Projects" In Proceedings of the 6th International Conference on Product Focused Software Process Improvement, Oulu, Finland, June 13-15, 2005. Springer Verlag LNCS, Vol. 3547, Jun 2005, pp. 54 – 68.
- [Li+05a] Li, J., Conradi, R., Slyngstad, O.P.N., Bunse, C., Khan, U., Torchiano, M., Morisio, M.: “Validation of New Theses on Off-The-Shelf Component Based Development” In Proceedings of the 11th IEEE International Software Metrics Symposium (METRICS2005), 19-22 September, 2005, Como, Italy.
- [Li+04] Li, J., Bjørnson, F.O., Conradi, R.: "Empirical Study on COTS Component Classification", Proc. International Workshop on COTS Terminology and Concepts, Co-located with International Conference on Component-Based Software Systems (ICCBSS'04).
- [Li+04a] Li, J., Bjørnson, F.O., Conradi, R., and Kampenes B.V.: "An Empirical Study of COTS Component Selection Processes in Norwegian IT companies" In Proceedings of the Int'l Workshop on Models and Processes for the Evaluation of COTS Components (MPEC'04 Arranged in co-location with ICSE'04), May 25, 2004. , Edinburgh, Scotland. IEEEPress, ISBN 0-86341-422-2, pp. 27-30
- [Lin+07] Lin, H., Lai, A., Ullrich, R., Kuca, M., McClelland, K., Shaffer-Gant, J., Pacheco, S., Dalton, K.: “COTS Software Selection Process”. 6th International Conference on COTS-Based Software Systems (ICCBSS) 2007. pp. 114-120.
- [Lin+97] Lichota, R.W., Vesprini, R.L., Swanson, B.: "PRISM: Product Examination Process for Component Based Development," SAST '97, 1997, pp. 61-69.
- [Lop+06] López, C., Astudillo, H.: “Multidimensional Catalogs for Systematic

- Exploration of Component-Based Design Spaces”. IFIP Workshop on Advanced Software Engineering 2006: 32-46
- [Los+03] Losavio, F., Chirinos, L., Levy, N., Ramdane-Cherif, A.: Quality Characteristics for Software Architecture. *Journal of Object Technology* 2(2): 2003, pp. 133-150.
- [Lou-Kav95] Loucopoulos, P., Kavakli, E.: Enterprise Modelling and the Teleological Approach to Requirements Engineering. *International Journal on Cooperative Information Systems*. 4(1): 45-79 (1995)
- [Luc+07] Lucrédio, D.; Brito, K. S.; Alvaro, A.; Garcia, V. C.; Almeida, E.S.; Fortes, R. P. M.; Meira, S. R. L. Software Reuse: The Brazilian Industry Scenario. *Journal of Systems and Software* (JSS), Elsevier, 2008.
- [Luc+04] Lucredio, D., do Prado, A.F., Santana de Almeida, E. “A Survey on Software Components Search and Retrieval”. In Proceedings of 30th EUROMICRO Conference. IEEE Computer Society, 2004.
- [Lun+99] Lundberg, L., Bosch, J., Häggander, D., Bengtsson, P.O.: “Quality Attributes in Software Architecture Design” In Proceedings of the 3th International Conference on Software Engineering and Applications (IASTED) 1999, pp. 353-362.
- [Mai+02] Maiden, N., Kim, H., Ncube, C. “Rethinking process guidance for Software Component Selection”. LNCS 2255, J.C. Dean and A. Gravel (Editors), Springer-Verlang, New York, 2002, pp. 151-164.
- [Mai-Ncu98] Maiden, Ncube, C. “Acquiring Requirements for COTS Selection”. *IEEE Software* Vol. 15, No. 2, March/April 1998.
- [Man-And05] Mancebo, E., Andrews, A.: “A Strategy for Selecting Multiple Components” ACM Symposium of Applied Computing 2005, pp. 1505-1510.
- [Man+07] Mansoor, A., Seema, A., Al-Zobaidie: A Study of the Contracting and Procurement Process for Cots Software Projects. *Journal of Computer Science* 3(3), 2007. pp. 180-185.
- [McC89] McClure, C.: CASE IS Software Automation. Prentice Hall, Englewood Cliffs, ISBN 0-13-119330-9
- [McC+77] MacCall, J.A., Richards, P.K., Walters, G.F.: Factors in Software Quality. RADC TR-77-369, Vols, I, II, III, US Rome Air Development Center Reports NTIS AD/A-049 014, 015, 055, 1977.
- [McM-Pal84] McMenamin, S.M., Palmer, J.F. Essential Systems Analysis. Yourdon Press, 1984.
- [Mer06] Merola, L.: “The COTS Software Obsolescence Threat”. In Proceedings of the 5th International Conference on COTS-Based Software Systems (ICCBSS) 2006, pp. 127-133.
- [Mes07] Messegue, F. “Eina de suport per al anàlisi de dominis”. <http://www.lsi.upc.edu/~cayala/Papers/IQToolDocumentation.pdf> In catalan. Jan 2007.
- [Mey-Obe02] Meyers, B.C., Oberndorf, P.: “Managing Software Acquisition”. SEI Series

- in Software Engineering, 2002
- [Mic+05] Michel, R., Roose, P., Barbier, F.: "Information System for Evaluation of COTS". In Proceedings of ACIS, pp. 64-69.
- [Min04] Minkiewicz, A.F.: "Are COTS Solutions an Affordable Alternative". In Proceedings of the Aerospace Conference, 2004, pp. 4073-4082.
- [Moh+07] Mohamed, A., Ruhe, G., Eberlein, A.: "COTS Selection: Past, Present and Future" Proceedings of the 14th Annual IEEE International Conference and Workshops on the Engineering of Computer-Based Systems (ECBS'07). 2007.
- [Moh+07b] Mohamed, A., Ruhe, G., Eberlein, A.: "Decision Support for Handling Mismatches between COTS Products and System Requirements. In Proceedings of the International Conference on COTS-Based Software Systems (ICCBSS 2007), Banff, Canada, 2007.
- [Moh+04] Mohamed, A., Wanyama, T., Ruhe, G., Eberlein, A., Far, B.: "COTS Evaluation Supported by Knowledge Bases," Lecture Notes in Computer Science, vol. 3096 2004/09 2004. pp. 43-54.
- [Mora+07] Moraes R., Durães, J., Martins, E., Madeira, H.: "Component-Based Software Certification Based on Experimental Risk Assessment". A. Bondavalli, F. Brasileiro, and S. Rajsbaum (Eds.): LADC 2007, LNCS 4746, pp. 179–197.
- [Mor06] Morisio M.: "Reuse of Off-The-Shelf Components". Proceedings of the 9th International Conference on Software Reuse, ICSR2006, Turin, Italy, June 2006. 442 pp. Lecture Note in Computer Science 4039.
- [Mor+02] Morisio, M., Seaman, C.B, Basili, V.R., Parra, A.T., Kraft, S.E., Condon, S.E.: "COTS-based Software Development: Processes and Open Issues". *Journal of Systems and Software* 61(3): 189-199 (2002).
- [Mor+02b] Morisio, M., Ezran, M., Tully, C.: Success and Failure Factors in Software Reuse. *IEEE Trans. Software Eng.*, Vol. 28, No. 4, 2002. pp. 340-357.
- [Mor-Tor02] Morisio, M., Torchiano, M. "Definition and Classification of COTS: A Proposal". In Proceedings of the 1st Conference on COTS-Based Software Systems (ICCBSS), Orlando, Florida; 2002.
- [Mor+00] Morisio, M., Seaman, C.B, Parra, A.T., Basili, V.R., Kraft, S.E., Condon, S.E.: "Investigating and Improving a COTS-Based Software Development Process". In Proceedings of the International Conference on Software Engineering (ICSE'00). Limerick, Ireland 2000.
- [Morr00] Morris, A.T.: "COTS Score: An Acceptance Methodology for COTS Software". In proceedings of the 19th Digital Avionics Systems Conferences (DASC) 2000. Volume 1, Issue 2000.
- [Mus+95] Muse, M.A., Gennari, J.H., Eriksson, H., Tu, S.W., Puerta, A.R.: "PROTÉGÉ-II Computer Support for Development of Intelligent System from Libraries of Components". In Proceedings of the 8th World Congress on Medical Informatics, pp. 766-770. Vancouver, 1995.
The current version of the tool is available at <http://protege.stanford.edu>

- [Nei80] Neighbors, J. Software Construction Using Components. PhD. Thesis, University of California, Irvine, 1980.
- [Neu-Stu07] Neubauer, T., Stummer, C.: "Interactive Decision Support for Multi-Objective COTS Selection". In Proceedings of the 40th Hawaii International Conference on System Sciences (HICSS) 2007, p. 283b.
- [Nic+01] Nick, M., Althoff, K.D., Tautz, C.: "Systematic Maintenance of Corporate Experience Repositories" *Computational Intelligence* 17(2): 364-386 (2001)
- [NPL] National Product Line Asset Center (NPLACE). "Component Test Criteria", available at <http://www.nplace.net/nplaceneu/criteri.html>
- [Nvi] Nvivo. NUD*IST Vivo For Qualitative Research. Distributes by Scolari, QSR (Qualitative Solutions & Research)
- [Obe-Bro97] Oberndorf, P., Brownsword, L.: "Are You Ready for COTS?" Software Institute Engineering. August 1997.
- [Och+01] Ochs, M.A., Pfahl, D., Chrobok-Diening, G., Nothhelfer-Kolb, B. "A Method for Efficient Measurement-based COTS Assessment and Selection- Method Description and Evaluation Results" Proceedings IEEE 7th International Software Metrics Symposium, London, England, 2001, pp. 285-296.
- [Ols99] Olisna, L.: "Website Quality Evaluation Method- A Case Study on Museums". In proceedings of the ICSE 99- 2nd Workshop on Software Engineering over the Internet.
- [OSI] Open Source Initiative, available at <http://www.opensource.org>
- [Par-Con07a] Mohagheghi, P. and Conradi, R.: "Quality, Productivity and Economic Benefits of Software Reuse: A Review of Industrial Studies", *Journal of Empirical Software Engineering*, 55 p.
- [Pat+84] Patel-Schneider, P.F., Branchman, R.J., Levesque, H.J.: "ARGON: Knowledge Representation Meets Information Retrieval" Proceedings of the 1st Conference on Artificial Intelligence Applications (CAIA '84), pp. 280-286.
- [Phi-Pol02] Phillips, B.C., Polen, S.M.: "Add Decision Analysis to Your COTS Selection Process", Software Technology Support Center Crosstalk, April 2002.
- [Poh+05] Pohl, K., Böckle, G., van der Linden, F.J. Software Product Line Engineering. Springer-Verlag, 2005
- [Poh+01] Pohl, K., Brandenburg, M., Glich, A. "Scenario-Based Change Integration in Product Family Development". In Proceedings of the 2nd Workshop on Software Product Lines, 2001.
- [Pou95] Poulin, J.S.: "Populating Software Repositories: Incentives and Domain-Specific Software", *Journal of Systems and Software*, Vol. 30 (1995), pp 187-199.
- [Pou-Ygl93] Poulin, J.S., Yglesias, K.P.: "Experiences with a Faceted Classification Scheme in a Large Reusable Software Library (RSL)". In Proceedings of the 17th International Conference on Computer Software and Applications

- (COMPSAC 1993), pp. 90-99. IEEE, 1993.
- [Por-Che04] Port, D., Chen, S.: “Assessing COTS Assessment: How Much Is Enough?” In Proceedings of the 3rd International Conference on COTS-Based Software Systems, ICCBSS 2004, Redondo Beach, CA, USA Lecture Notes in Computer Science Volume 2959/2004 pp.183-198.
- [Pot+94] Potts, C., Takanashi, K., Antón, A. “Inquiry-Based Requirements Analysis”, *IEEE Software*, 11 (2), March 1994.
- [Pri91] Prieto-Díaz, R, Implementing Faceted Classification for Software Reuse, *Communications of the ACM*, 34(5), 89-97, May 1991
- [Pri-Ara91] Prieto-Díaz, R., Arango, G. Domain Analysis and Software Systems Modelling. *IEEE Computer Society Press*, 1991.
- [Pri87] Prieto-Díaz, R.: “Faceted Classification and Reuse Across Domains”. Proceedings of the Workshop on Software Reuse, Rocky Mountain Institute of Software Engineering, L.Williams- Editor(Boulder, CO), 14 Oct 1987.
- [Pri-Fre87] Prieto-Díaz, R., Freeman, P.: Classifying Software for Reusability. *IEEE Software*. January 1987 pp. 6-16.
- [Pri85] Prieto-Díaz R. “A Software Classification Schema” PhD Dissertation. Department of Information and Computer Science. University of California, Irvine 1985.
- [PRI] PRISMA: Academic Record Management System. Technological Transfer project. GESSI group. Universitat Politècnica de Catalunya. Further details at <http://www.lsi.upc.es/~webgessi/index.html>.
- [Qui86] Quinlan J.R.: Induction of Decision Trees. *Kuwer Academic Publishers* ISSN 0885-6125, March 1986.
- [Quin93] Quinlan, J.R. “C4.5: Programs for Machine Learning” Morgan Kauffman, 1993.
- [Rad] Web Services Market 2004-2008. September 2004, available at <http://www.radicati.com>
- [Rav-Rot03] Ravichandran, T. and Rothenberger, M.A.: Software Reuse Strategies and Component markets. *Communications of the ACM*, 46(8):109–114, 2003.
- [Raw-Mat06] Rawashdeh, A., Matakah, B.: A New Software Quality Model for Evaluating COTS Components. *Journal of Computer Science*, 2(4), 2006, pp. 373-381.
- [RED] REDEPEND-REACT web page:
<http://www.lsi.upc.edu/~ggrau/REDEPEND-REACT/index.html>
- [Reg05] Regev, G. “Where do Goals Come from: the Underlying Principles of Goal-Oriented Requirements Engineering”. 13th IEEE Requirements Engineering Conference 2005.
- [Rei+03] Reifer, D.J., Basili, B.R., Boehm, B.W., Clark, B.: Eight Lessons Learned during COTS-Based Systems Maintenance. *IEEE Software*, September-October 2003, pp. 94-96.

- [Req+05] Réquillé-Romanczuk, A., Cechich, A., Dourgnon-Hanoune, A., Mielnik, J.C.: "Towards a Knowledge-based Framework for COTS components Identification" ICSE-MPEC05, ACM Press, 2005; pp 1-4.
- [Rob02] Robson, C.: Real World Research. Blackwell (2nd Edition) 2002.
- [Rom-Ken07] Romsaiyud, W., Keneto, S.: "Challenges in Selecting COTS Components Guidelines" In Proceedings of the 31st Annual International Computer Software and Applications Conference (COMPSAC 2007), pp. 661-663.
- [Ruh03] Ruhe, G.: "Intelligent Support for Selection of COTS Products". Lecture Notes in Computer Science, Springer, Vol. 2593, pp. 34-45. 2003.
- [Ruh02] Ruhe, G.: "Software Engineering Decision Support – A New Paradigm for Learning Software Organizations". S. Henninger and F. Maurer (Eds.): LSO 2002, LNCS 2640, pp. 104–113, 2003.
- [Ruh01] Ruhe, G.: "Learning Software Organizations". In Handbook of Software Engineering and Knowledge Engineering (S.K. Chang Eds.) Vol. 1. World Scientific Publishing (2001) 663-678.
- [RUP] Jacobson, I., Booch, G., Rumbaugh, J.: The Unified Software Development Process. Addison Wesley Longman, Reading, MA, 1999.
- [Saa90] Saaty, T.L.: Multicriteria Decision Making: The Analytic Hierarchy Process. RWS Publications, Pittsburgh, PA. 1990.
- [Sai+04] Sai, V., Franch, X., Maiden, N.: "Driving Component Selection Through Actor-Oriented Models and Use Cases" In Proceedings of the 3rd International Conference on COTS-Based Software Systems (ICCBSS) 2004, pp. 63-73.
- [Sal-McG83] Salton, G., McGill, M.J.: *Introduction to Modern Information Retrieval*. McGraw Hill, New York, 1983.
- [Sas+06] Sassi, S.B., Jilani, L.L., Ghezala, H.H.B.: "Towards a COTS-Based Development Environment" In Proceedings of the 5th International Conference on COTS-Based Software Systems (ICCBSS) 2006, p 10.
- [Sas+03] Sassi, S.B., Jilani, L.L., and Ghezala, B.H.: "COTS Characterization Model in a COTS-Based Development Environment", 10th Asia Pacific Software Engineering Conference Apec'03, Chiang Mai, Thailand, December 2003. pp 352-361.
- [SDA] Software Engineering Institute (SEI). <http://www.sei.cmu.edu/domain-engineering/>, 2002.
- [Sea99] Seacord, R.C., Nwosu, K.C.: "Component-Based Software Engineering Processes," In Proceedings of the 30th International Conference on Technology of Object-Oriented Languages and Systems (TOOLS) 1999. p. 532
- [Sea+98] Seacord, C., Hissam, A., Wallnau, K.: "Agora: A Search Engine for Software Components" *Internet Computing*, Vol. 2 No. 6, December 1998, pp. 62-70
- [Sed+03] Sedigh-Ali, S., Ghafoor, A., Paul, R.A.: "A Metrics-Guided Framework

- for Cost and Quality Management of Component-Based Software” A. Cechich et al. (Eds.): *Component-Based Software Quality*, LNCS 2693, 2003, pp. 374–402.
- [SEI] Software Engineering Institute. Carnegie Mellon University. Available at: <http://www.sei.cmu.edu/>
- [Sha03] Shaw, M.: “Writing Good Software Engineering Research Paper” In *Proceedings of the the 25th International Conference on Software Engineering (ICSE) 2003*. IEEE Computer Society, pp. 726-736.
- [Sha01] Shaw, M. The coming-of-age of software architecture research. In *Proceedings of the 23rd International Conference on Software Engineering (ICSE 2001)*, pp.657–664, May12–19 2001.
- [Shan+03] Shankaranarayan, G., Ziad, M., Wang, R.Y.: *Managing Data Quality in Dynamic Decision Environments. Journal of Database Management*, 14(4), 14-32, Oct-Dec 2003.
- [Shy-Shi06] Shyur, H.J., Shih, H.S.: A Hybrid MCDM Model for Strategic Vendor Selection. *Journal of Mathematical and Computer Modelling*, Elsevier. Vol. 44, Issues 7-8, October 2006, pp. 749-761.
- [Sim-Bel03] Simão, R.P.S., Belchior, A.D.: “Quality Characteristics for Software Components: Hyerarchy and Quality Guides” *Component-Based Software Quality: Methods and Techniques*. A Cechich, Piattini, M. Vallecillo, A. (Eds.). Springer-verlag. LNCS 2693, 2003.
- [Sim-Dil06] Simmons, G.L., Dillon, T.S.: “Towards an Ontology for Open Source Software Development”. In *IFIP International Federation for Information Processing, Volume 203, Open Source Systems*, eds. Damiani, E., Fitzgerald, B., Scacchi, W., Scotto, M., Succi, G., (Boston:Springer), pp 65-75.
- [Sja-Beu06] Sjachyn, M., Beus-Dukic, L.: “Semantic Component Selection–SemaCS”. *International Conference on COTS-Based Software Systems*, IEEE Society 2006.
- [Somm06] Sommerseth, M.: “Component based system development in the Norwegian software industry”, masters thesis, IDI, NTNU, June 2006, 141 p., available at: <http://www.idi.ntnu.no/grupper/su/index.php3?file=publ/INT-PUBL.php3>
- [Sta-Lub06] Staaden, v.P., Lubbe, S.: A Case Study on the Selection and Evaluation of Software for an Internet Organisation. *The Electronic Journal of Business Research Methods*, Vol. 4 Issue 1, pp 57-66, available online at www.ejbrm.com
- [Sur-Abr03] Suryan, W.; Abran A.: "ISO/IEC SQuaRE: The Second Generation of Standards for Software Product Quality," 7th IASTED International Conference on Software Engineering and Applications, California, USA. 2003.
- [SQU] ISO/IEC FCD 25000, *Software Engineering – Software Product Quality Requirements and Evaluation (SQuaRE) - Guide to SQuaRE*, Geneva: International Organization for Standardization, 2004.

- [SU] Software Engineering Group of the Norwegian University of Science and Technology (NTNU), available at: <http://www.idi.ntnu.no/grupper/su/>
- [Tau+04] Taulavuori, A., Niemela, E., Kallio, P.: Component Documentation- A Key Issue in Software Product Lines. *Journal on Information and Software Technology. Elsevier*, 46 (2004) 535–546.
- [TEC] Technology Evaluation.com, The ERP Evaluation Center. “ERP Decision Hierarchy”, available at <http://www.erpevaluation.com/>
- [Tor-Mor04] Torchiano, M., Morisio, M. Overlooked Aspects of COTS-Based Development. *IEEE Software*, March/April 2004, pp 88-93.
- [Tor+02] Torchiano, M., Jaccheri, L., Sørensen, C.F., Wang, A.I.: ”COTS Products Characterization” In Proceedings of the 14th international conference on Software Engineering and Knowledge Engineering. . SEKE 2002 pp.335-338
- [Tra-Liu97] Tran, V., Liu, D., Hummel, B: “Component-based Systems Development: Challenges and Lessons Learned”. In Proc. of 8th International workshop on Software Technology and Engineering Practice, July 1997, pp. 452-462.
- [Tren-Pon03] Trendowicz, A., Punter, T.: “Quality Modeling for Software Product Lines” In Proceedings of the 7th ECOOP Workshop on Qualitative Approaches in Object-Oriented Software Engineering. Germany, July 2003.
- [Ulk-Sep04] Ulkuniemi, P., Seppänen, V. COTS Component Acquisition in an Emerging Market. *IEEE Software*. November/December 2004. pp 76-82
- [Ulk03] Ulkuniemi, P.: “Purchasing Software Components at the Dawn of Market” Thesis Dissertation. Department of Marketing. University of Oulu, Finland. 2003.
- [UML] UML Specifications. <http://www.uml.org/>
- [UPI] UPIC: towards a Unified approach to the Procurement and Implementation of information system Components. Research Project. Financed by the Spanish Ministerio de Educación y Ciencia. GESSI Group
- [Vaf+06] Vafaie, H., Brown, N.F., Truong, L.: "Methodology for the Selection of Intelligence Analysis Tools," Proceedings of the 18th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'06), 2006, pp. 55-62.
- [Vand+94] VanderVet, P.E., Speel, P.H., Mars, N.J.I. “The Plinius Ontology of Ceramic Materials” Proceedings 11th European Conference on Artificial Intelligence (ECAI 94), Workshop on Comparison and Implemented Ontologies, Amsterdam, The Netherlands, 1994.
- [Vig01] Vigder, M.: Maintaining Component-Based Systems. *Component-Based Software Engineering: Putting the Pieces Together*. Heineman, G., Council, B. (Editors). Addison-Wesley. 2001. NRC 44208.
- [Vig-Dea97] Vidger, M., Dean, J.: “An Architectural Approach to Building Systems from COTS Software Components” Proceedings of the 1997 Center for Advanced Studies Conference (CASCON'97), Toronto, Ontario, Canada 10-13 November 1997. Available at:

<http://seg.iit.nrc.ca/English/abstracts/NRC40221labs.html>

- [Vit+03a] Vitharana, P., Zahedi, F., Jain, H.: “Knowledge-Based Repository Scheme for Storing and Retrieving Business Components: A Theoretical Design and Empirical Analysis”. *IEEE Transactions on Software Engineering*, Vol. 29(7), 2003, pp 649-664.
- [Vit+03b] Vitharana, P., et al. “Design, Retrieval, and Assembly in Component-Based Software Development”. *Communications of ACM*, November 2003/Vol.46, No. 11.
- [Voa04] Voas, J.: Software’s Secrets Sauce: The “-ilities [Software Quality]”. *IEEE Software* 21(6):14, 2004.
- [Voa98] Voas, J.: The Software Quality Certification Triangle. *CrossTalk*, Nov. 1998, pp. 12-14.
- [Voa98b] Voas, J.M.: The Challenges of Using COTS Software in Component-Based Development. *IEEE Computer*, Vol. 31, Issue 6. June 1998, pp. 44-45.
- [Wag04] Wagner, C.: “Wiki: A Technology for Conversational Knowledge Management and Group Collaboration”. *Communications of the Association for Information Systems*, Vol. 13, article 19, pp. 256-289.
- [Wang+03] Wang, R., Allen, T., Harris, W., Madnick, S. “An Information Product Approach for Total Information Awareness” *IEEE*, 2003.
- [Wan-Hom06] Wanyama, T., Far, B.H.: “Repositories for COTS Selection” In Proceedings of the Canadian Conference on Electrical and Computer Engineering (CCECE '06), 2006, pp. 2416-2419.
- [Wan-Hom05] Wanyama, T., Far, B.H.: “Towards Providing Decision Support for COTS Selection” In Proceedings of the Canadian Conference on Electrical and Computer Engineering (CCECE '05.), 2005, pp. 908-911.
- [Wan-Str96] Wang, R.Y., Strong, D.M.: Beyond Accuracy: What Data Quality Means to Data Consumers. *Journal of Management of Information Systems*. Vol. 12, No.4, pp5-34. 1996.
- [War-Far05] Wanyama, T., Far, B.H.: “A Multi-Agent Framework for Conflict Analysis and Negotiation: Case of COTS Selection” *Transactions of the Institute of Electronics, Information and Communication Engineers: Special Issue on Software Agent and its Applications – Vol. E88-D, No.9, September, 2005*, pp. 2047-2058.
- [Win+87] Winston, M.E., Chaffin, R., Hermann, D.J. “A Taxonomy of Part-Whole Relations” *Cognitive Science* 11:417-444.
- [Woh+00] C. Wohlin, P. Runeson, M. Host, M.C. Ohlsson, B. Regnell, and A. Wesslen: *Experimentation in Software Engineering - An Introduction*. Kluwer Academic Publishers, 2000.
- [Yac+00] Yacoub, S., Mili, A., Kaveri, C., Dehlin, M.: “A Model for Certifying COTS Components for Product Lines” In Proceedings of the Workshop on Continuing Collaborations for Successful COTS Development, in conjunction with the 22nd International Conference on Software Engineering

- (ICSE) 2000. Limerick, June 2000.
- [Yan+06] Yanes, N., Sassi, S.B., Jilani, L.: “MoReCOTS: a Specialized Search Engine for COTS Components on the Web”. International Conference on COTS-Based Software Systems, IEEE Society 2006.
- [Yan+05] Yang, Y., Bhuta, J., Boehm, B., Port, D.N.: Value-Based Processes for COTS-Based Applications. *IEEE Software*, Vol. 22, Issue 4, July-Aug. 2005, pp.54-62.
- [Ye-Kel04] Ye, F., Kelly, T.: “COTS Products Selection for Safety-Critical Systems”. In Proceedings of the International Conference on COTS-Based Software Systems (ICCBSS) 2004, pp. 53-62.
- [Yeo-Mil04] Yeo, H.C., Miller, J.: “COTS Acquisition Process: Incorporating Business Factors in COTS Vendor Evaluation Taxonomy” In Proceedings of the 10th International Symposium on Software Metrics 2004, pp. 84-95.
- [Ye-Lo01] Ye, H., Lo, B.W.N.: “Towards a Self-Structuring Software Library” *IEEE Proc.-Softw.*, vol. 148 No. 2, pp. 45-55, April 2001.
- [Yin03] Yin, R.K.: Case Study Research. Design and Methods. (3rd Edition) London (Sage) 2003.
- [Yu95] Yu, E. “Modelling Strategic Relationships for Process Reengineering” PhD Thesis, University of Toronto, 1995.
- [Zhe+06] Zheng, J., Robinson, B., Williams, L., Smiley, K.: “A Lightweight Process for Change Identification and Regression Test Selection In Using COTS” In Proceedings of the 5th International Conference on COTS-Based Software Systems (ICCBSS) 2006, p. 7.
- [Zus97] Zuse, H.: A Framework of Software Measurement. Walter de Gruyter & Co., 1997.

Annex 1. Heuristics Supporting GOTHIC Activities

Heuristics are rules used to provide prescriptive guidance for performing some GOTHIC activities achieving a high probability of success while avoiding wasted efforts. Their use in GOTHIC was inspired by GBRAM. Many of the heuristics proposed in GBRAM were mapped directly or with some adjustment to GOTHIC (mostly those related to identification, refinement, and management of goals, and they are stated with the * symbol), whilst many other heuristics emerge as lessons learned in the different case studies performed and were designed to guide the specific GOTHIC activities. The design and even the application of some heuristics follow the Inquiry Cycle approach [Pot+94] instantiated for goal-based analysis as originally proposed in GBRAM (see [Ant97], page 137).

The objective of this annex is to describe typical heuristics applied when using the GOTHIC method. As discussed in Chapter 3, the lessons learned in the initial case studies served as the origin of the ideas which formulated the GOTHIC method. The heuristics in this annex were derived from these experiences and observations. The utilization of heuristics depends upon the particular GOTHIC activity; thus, we have classified the heuristics in diverse sets applying to specific activities. Tabla A1.1 shows a summary of the heuristics available, and Table A1.2 describes the existing heuristics.

Table A1.1 Glossary of heuristics and their codes in GOTHIC

GOTHIC Activity	Code	Definition	Observation
Exploration of Information Sources	HGIS	Heuristics for gathering Information Sources	Assist to the GOTHIC user in gathering and identifying suitable information sources to analyze COTS informational dimensions.
	HIIS	Heuristics for identifying Information Sources	
Domain Analysis	HPIS	Heuristics for Prioritizing Information Sources	Provide useful insights to prioritize information sources according to their potential trustworthiness.
Identification, refinement & statement of goals	HIG	Heuristics for Identifying Goals	Assist in identifying goals, stakeholders, agents, and constraints from multiple sources
	HIS	Heuristics for Identifying Stakeholders	
	HIA	Heuristics for Identifying Agents	
	HIC	Heuristics for Identifying Constraints	Refinement heuristics employ a series of question and techniques to reduce the size of the goal set
	HRR	Heuristics for Refining Redundancies	
	HRS	Heuristics for Refining Synonymous	
	HEO	Heuristics for Elaborating Obstacles	
HES	Heuristics for Elaborating Scenarios		
Establishment of dependencies	HDD	Heuristics for Determination of dependencies	Assist in determining the type of relationships to be modeled with i^*
Goal Taxonomy Structuring	HSGT	Heuristics for Structuring goal-taxonomies	Based on the trustworthiness of the information source where goals are identified and applying the IC these heuristics help to decide the structure of the intended taxonomy
Taxonomy Validation	-	Not apply	Not apply
Knowledge Base Management	HKBM	Heuristics for Managing the Knowledge Base Repository	It refers to heuristics and guidelines to assist in managing and re(using) the repository.

Table A1.2 Heuristics that Support GOTHIC Activities

Code	Heuristic
HGIS	Diverse types of information sources exists, they can be grouped into: Hierarchy, Standard, Vendor Information, Independent Reports (of scientific, divulgation and/or technical nature), Oral Information, Test Of Tools Reports, Experiences, Other”,... (Descriptions and examples are provided in Chapter 5).
HIIS	Information sources available can provide insights into a diverse range of software packages and/or vendor characteristics, but no requirements identified from these sources should be used without careful consideration of their confidence
HIIS	Information from experts is good at quickly identifying general principles, offering explanations, validating analyses, and providing pointers that could be cross-validating with Independent Reports.
HPIS	Information from standards related to the field, are the best for identifying COTS domain high-level goals.
HIG	Identification of diverse types of actors (See Chapter 7) helps to discover high-level goals and their subsequent decomposition and refinement.
HIG*	Abstraction mechanism may be employed to extract goals from available documentation by asking: What goal(s) does this statement exemplify?, What goal(s) does this statement block or obstruct?
HIG*	Action words that point to some state that is or can be achieved once the action is completed are candidates for goals. They are identified by considering each statement in the available information by asking: Does this behavior or action denote a state that has been achieved, or a desired state to be achieved? If the answer is yes, then express the answer to these questions as goals which represent a state that is desired or achieved.
HIG*	An effective way to uncover hidden goals is to consider each action word and every description of behavior and persistently ask “Why?” until all the goals have been ‘treated’ and you are confident that the rationale for each action is understood and expressed as a goal.
HIG*	Key action words such as: track, monitor, provide, supply, find out, know, avoid, ensure, keep, satisfy, complete, allocate, increase, speedup, improve, make, and achieve are useful for pointing to candidate goals
HIG*	If a statement seems to guide design decisions at various levels, express it as a goal.
HIG*	Goals may be uncovered by examining the information available to identify avoidance goals. Avoidance goals are found by identifying bad states that should be avoided within the system.
HIG*	Goals can be uncovered or discovered by considering the goal dependencies for the previously specified goals by asking: What are the preconditions of this goal? And What are the postconditions of this goal? Since preconditions and postconditions are expressed as goals, it is possible to identify new goals that had not been previously considered or identified by considering each goal’s dependencies.
HIG*	Stakeholders tend to express their requirements in terms of operations and actions

Code	Heuristic
	rather than goals. Thus, when given an interview transcript, it is beneficial to apply the action word strategy to extract goals from stakeholders' descriptions.
HIG*	One should first seek to understand the application domain and goals.
HIG*	Goals are also identified by considering the possible goal obstacles for previously specified goals.
HIG*	Goals may be identified by considering possible scenarios. Given each goal obstacle, one should determine whether or not the occurrence of the goal obstacle would initiate system failures, these obstacles are key candidates for scenario construction and analysis.
HIG*	Goals may be identified by considering constraints.
HIG*	Goals may be identified from process diagrams or standards in the area by searching action words and behaviors, as well as by consistently applying the Inquiry Cycle to clarify the goals and requirements.
HIS*	Any representative affected by the completion or prevention of a goal is a stakeholder. Stakeholders are thus identified by asking: Who or what claims a stake in this goal? Who or what stands to gain or lose by the completion or prevention of this goal? Who will use the system or component?
HIS*	Multiple stakeholders may be associated with one goal. If different stakeholders are associated with a goal, but their associations occur at different times, we should comment these variances to ensure that the role of stakeholders throughout the lifetime of a goal is well understood.
HIA*	Responsible agents may be identified by considering each goal and asking: Who or what agent is, could be, or should be responsible for this goal?
HIA*	At least one agent must be responsible for the completion of each goal. If we are unable to allocate responsibility for a goal to any agent, then we can assume that the goal lies outside the scope of the domain being analyzed. If we believe there is a responsible agent, but doesn't know who or what, then the Inquiry Cycle should be applied.
HIA*	Different agents can be responsible for the completion of the same goal at different times.
HIA*	Agents may be either systems, components, organizations or human agents.
HIC*	Constraints can be identified by considering each statement and asking: Does this fragment impose some constraint on the goal(s)? Does this fragment specify some requirement that must be met?
HIC*	Constraints can be identified by searching by temporal connectives (i.e., during, before, after, etc.). Restate statements that describe when some condition is true or when a goal can be completed as a constraint.
HIC*	Constraints can be identified by searching statements which place limits on the completion of a goal.

Code	Heuristic
HIC*	Since constraints may place a condition on the achievement of a goal, they should be restated as goal obstacles to allow for subsequent elaboration of the obstacle using scenarios.
HRR*	If the same goal appears more than once AND the same agent is responsible for the goal on each occurrence, then all but one of the goals may be eliminated.
HRR*	If the same goal appears more than once BUT two or more different agents are responsible for the same goal at different times, then the goal is left as it is.
HRS*	If two goals are synonymous, reconcile the duplication by eliminating the goal which can be semantically subsumed by the other.
HRS*	Consolidate and refine goals by merging synonymous goals.
HRS*	Ordering goals according to their precedence relations facilitates the identification of synonymous goals.
HEO*	Obstacles can be identified by asking “What other goal or condition does this goal depend on?”, “Can the agent responsible for a goal fail to achieve the goal?”, “If this goal is blocked, what are the consequences?”, “Can the failure of another goal to be completed cause this goal to be blocked?”
HEO*	There is at least one goal obstacle for every goal. This is informally referred to as the trivial obstacle and formally referred to as the normal first case goal obstacles. These obstacles are worded by negating the verb in the goal name.
HEO*	A prerequisite failure obstacle occurs when a goal having a precedence relation is obstructed because the precedence goal fails. Prerequisite failures are identified by considering each goal and asking: What other goal(s) does this goal depend on?
HEO*	An agent failure obstacle occurs when a goal fails because the responsible agent fails to achieve the goal. Agent failures are identified by considering each goal and asking: Can the failure of an agent to fulfill their responsibilities cause this goal to fail?
HES*	An effective way to identify candidate scenarios for construction is to consider each goal and goal obstacles previously identified to determine the reasons why and the circumstances under which a goal may be completed or can fail. By asking “Why?” and “What happens if this goal isn’t achieved?”
HES*	Scenarios construction leads to the identification of new goals of the domain.
HDD	For identifying dependencies, we first identify diverse types of actors (see Section 7.2.1 in Chapter 7). The actors are required to have a clear strategic value for the modeled system; it is useful to use a metaphor to think about the system. In most cases we use a client-server metaphor: an actor (e.g., the client) provides and consumes a resource (e.g., the information) that is under the control of an organization (e.g., the server). This and other metaphors could be organized in the form of a catalogue of i^* organizational patterns
HDD	Identifying dependencies among actors serves as a good goal-refinement mechanism. We provide some heuristics based on the use of i^* models. Thereafter,

Code	Heuristic
	<p>the HDD heuristics provided are accomplishing such assumption:</p> <p>By default, we depict and classify dependencies among actors as goal dependencies, which are the most common type due to their strategic value.</p> <p>The crucial point of this activity is to identify just those dependencies that are really needed. This criteria is obviously fuzzy and therefore the number of dependencies that will arise in this step is inevitably subjective, which in fact is a characteristic of goal-oriented modelling. However, when using a catalogue of <i>i*</i> organizational patterns, the dependencies already proposed in the patterns can be adapted to the system we are modeling.</p>
HDD	<p>To name and classify dependencies into a valid type of <i>i*</i> we propose a set of questions to be answered following a predefined ordering as shown in the following graph:</p> <p>In nodes 1 to 4 a question must be answered; in nodes 5 to 8 a specific type of dependency has been identified; in nodes 9 to 11 some additional softgoal dependencies may be added to the model. In the graph, each type of dependum is identified by a capital letter: Resource, Task, Goal and SoftGoal. Starting at node 1, questions to answer at each node to classify the dependency D, from A to B are:</p> <ol style="list-style-type: none"> 1. Does the depender depend on the depedee to achieve an entity or to attain a certain state? If entity, go to 3; else, go to 2. 2. Is the depender interested in attaining the state following a particular process? If so, classify D as task dependency and go to 5; else, go to 4. 3. Is the depender interested in obtaining the entity following a particular process? If so, classify D as task dependency and go to 5; else, classify D as resource dependency and go to 6. 4. Is there a clear cut criteria to determine the achievement of the state? If so, confirm the dependency D as goal dependency and go to 7; else, classify D as softgoal dependency. 5. Is there some additional restrictions on how to execute the task? If so, for each restriction, establish a new softgoal dependency from A to B. 6. Is there some additional properties that the resource must met to be acceptable? If so, for each property, establish a new softgoal dependency from A to B.

Code	Heuristic															
	<p>7. Is there some extra conditions that the achievement of the goal must satisfy? If so, for each condition, establish a new softgoal dependency from A to B.</p> <p>(Please see [Gra+05] and Fra+07] for details of the process)</p>															
HDD	<table border="1"> <thead> <tr> <th>Dependum</th> <th>Syntax</th> <th>Example</th> </tr> </thead> <tbody> <tr> <td>Task</td> <td>Verb + (Object) +(Complement)</td> <td>Answer doubts by e-mail</td> </tr> <tr> <td>Resource</td> <td>(Adjective) + Object</td> <td>Virus List</td> </tr> <tr> <td>Goal</td> <td>Object +Passive_Verb</td> <td>Information kept preserved</td> </tr> <tr> <td>Softgoal</td> <td> – Goal syntax + Complement – (Object) + Complement ([Dependum]) </td> <td> – Information checked in a transparent manner – Timely[Virus List] </td> </tr> </tbody> </table> <p>To improve the understandability of the i^* dependencies, the names assigned to their dependums shall be kept short and precise and be consistent throughout the model. The table above summarizes the conventions we suggest to use (parenthesis stand for optionality).</p> <p>Longer descriptions can be added to the documentation, especially if using tool support such as REDEPEND (See Chapter 9). We remark the case of softgoal dependencies, in which we distinguish among dependencies that stand alone (node 8 in the graph of the figure above), whose pattern is Goal-Syntax + Complement; and dependencies that qualify another dependum of the model (nodes 9, 10 and 11), in which the qualifier is a Complement and (optionally) the dependum between brackets. Note that using these syntactical patterns we will use short names that are specific to the semantics of the dependum, increasing in this way the comprehension of the model.</p>	Dependum	Syntax	Example	Task	Verb + (Object) +(Complement)	Answer doubts by e-mail	Resource	(Adjective) + Object	Virus List	Goal	Object +Passive_Verb	Information kept preserved	Softgoal	– Goal syntax + Complement – (Object) + Complement ([Dependum])	– Information checked in a transparent manner – Timely[Virus List]
Dependum	Syntax	Example														
Task	Verb + (Object) +(Complement)	Answer doubts by e-mail														
Resource	(Adjective) + Object	Virus List														
Goal	Object +Passive_Verb	Information kept preserved														
Softgoal	– Goal syntax + Complement – (Object) + Complement ([Dependum])	– Information checked in a transparent manner – Timely[Virus List]														
HSGT	Applying the Inquiry Cycle [Pot+94], questions, and answers can be attained to each goal statement.															
HKBM	As new market segments arise, they can be included to the taxonomy by identifying its goal and locate its place in the taxonomy using the defined classifiers, and once there even some useful artifacts are inherited to be refined to explicitly cover the new market segment.															
HKBM	Transformation Rules could be applied to validate and manage the GOTHIC Knowledge Base. See Chapter 8 for rules definition and explicit heuristics of their application.															

Annex 2. IQ COTS Reference Model

The model presented here, is currently being iteratively refined and improved as more empirical data from COTS selectors is gathered.

Table A2.1 COTS IQ Reference Model

Characteristic/Subcharacteristics/Attributes			Metric	Description
1 Intrinsic Characteristic			Own properties of the information source denoting its quality characteristics.	
1 Believability				
1 Author-Based Believability			Aspects that describe the believability of the product based on its authors.	
1 Author(s) Name			AName = Set (String) AName ≠ ∅ The names are directly obtained	Describes the name of the author(s) of the product.
2 Author(s) Believability			Derived Attribute OveAuthorsBel= Mean(AuthorsBel)	Describes the overall authors believability by the average of the believability of all authors.
1 Individual Author Believability			AuthorsBel= Function(String → TScore) TScore: {Very High, High, Low, Very Low} Dom(AuthorsBel) = AName ∀ x ∈ AName: AuthorsBel (x) = Mean (AuthorBelMarks)	Describes the individual believability of the authors by the average of all marks that he/she has received
1 Opinion Marks about the Author			AuthorBelMarks = Function(String → TScore) TScore: {Very High, High, Low, Very Low} These marks are directly obtained	Describes the marks that markers have done about the author
2 Provider Based Believability			Aspects that describe the believability of the product based on the organization that provides it	
1 Provider Name			OrgName=String OrgName ≠ ∅	Describes the name of the product provider
2 Organization Type			OrgType= Function(String → TOrg) TOrg: { Academy, Standards, Commercial}	Describes the type of the organization provider
3 Organization Believability			OrgBel= Function(String → TScore) Dom (OrgBel) = OrgName ∀ p ∈ OrgName: OrgBel (s) = Mean (ProvBelMarks)	Describes the believability of the organization provider
1 Opinion Marks about the Organization			ProvBelMarks = Function(String → TScore) These marks are directly obtained	Describes the marks that markers have done about the organization
3 Marker Based Believability			Aspects that describe the believability of the product based on its related marks	

Characteristic/Subcharacteristics/Attributes			Metric	Description
	1	Marker(s) Name	MName = Set (String) MName $\neq \emptyset$	Describes the name of the markers that have made any mark for the product
	2	Marker(s) Believability	OveMarkersBel=Mean(MarkerBel)	Describes the overall markers believability by the average of the believability of all the markers that have provided some mark for the product.
	1	Individual Marker Believability	MarkerBel= Function(String \rightarrow TScore) TScore: {Very High, High, Low, Very Low} Dom(MarkerBel) = MName $\forall x \in MName: \text{MarkerBel}(x) = \text{Mean}(\text{MarkerBelMarks})$	Describes the individual believability of the markers by the average of all marks that he/she has received
	1	Opinion Marks about the markers	MarkerBelMarks = Function(String \rightarrow TScore) TScore: {Very High, High, Low, Very Low} These marks are directly obtained	Describes the marks that markers have received
2	Accuracy			Aspects that describe the accuracy of the information source.
	1	Verifiableness		Provision of the resources to allow the tracking and verification of the content of the product
	1	History and Versioning		Capability of the product to provide a history of its changes and versions
	1	History Files Information		Information provided by the history files
	1	Fields	Fields= Set(Labels: Nominal); Labels=(Date, Time, ChangePerformed, ...)	List of fields recorded in the history fields
	2	Events	Events= Set (Labels: Nominal); Labels=(Save, Replace, Delete, ...)	List of the events which record information on the history files
3	Objectivity			Aspects that describes to which extent the information source offers an impartial point of view
	1	Source Type		Describes the kind of source the product is
	1	Type	TypeOfSource= Set (Labels: Nominal); Labels=(Hierarchy, Standard, VendorInfo, ...)	Describes the type of the product
	2	SubType	SubType= Set (Labels: Nominal); Labels=(Scientific, Divulgation, Technical)	Describes the subtype of the product
	2	Sponsored Product	Sponsored: Nominal; Sponsored=(True, False)	Describes if the product is supported by some organization
	1	Sponsor organization	SponOrganization= Set(String)	Describes the sponsor organizations of the product

Characteristic/Subcharacteristics/ Attributes		Metric	Description
4	Reputation		Recognition of the reputation and relevance of the information source
	1 Product Based Reputation		Aspects that describe the reputation of the product
	1 Product Name	PName = Set (String) PName ≠ ∅ The names are directly obtained	Describes the name or title of the product
	2 Product Reputation	Derived Attribute OveProductRep (x) = Mean (ProductRepMarks)	Describes the overall product reputation by the average of all reputation marks received.
	1 Opinion Marks about reputation	ProductRepMarks= Function(String→ TScore) TScore: {Very High, High, Low, Very Low} Dom(ProductRepMarks) = PName These marks are directly obtained	Describes the reputation marks that the product has received
2	Representational Characteristics		IQ properties related to the information source rendering
1	Concise Representation		Recognition of the information source structure and format
	1 Use of models	3ValueOrder[Ordinal]; 3ValueOrder = (Satisfactory, Acceptable, Poor)	Describes to which extend models are used to summarize and group information
	2 Kind of Models used	KModel=Set(Labels:Nominal); Labels=(ER, UC, NL, ...)	List of kind of models used in the product
	3 Storing Format	Format=Set(Labels:Nominal); Labels=(doc, pdf, html, ...)	Describes the format(s) the product is available
	4 Size of the product	Size: Set(Float, Label); Label=(pages, mgbyte, kbyte, Gbyte, ...)	Describes the size of the product (storing formats available)
2	Representational Consistency		Aspects that describe the degree of uniformity among the represented elements in the information source.
	1 Models Congruency	3ValueOrder[Ordinal]; 3ValueOrder = (Satisfactory, Acceptable, Poor)	Describes to which extend the models used provide an homogeneous view of the information
	2 Adhered to Standards for Representing Information	AdStd: Nominal; AdStd=(True, False)	Describes if the product is adhered to standard(s) for representing the information
	1 Own Standards	OwnStd: Nominal; OwnStd=(True, False)	Describes if the product is adhered to some own standard for structuring the information

Characteristic/Subcharacteristics/Attributes			Metric	Description
	1	Own Standard Name	OwnStdName=Set(Labels:Nominal); Labels=(OwnStd1, OwnStd2,...)	List of names of own standards used
	2	Public standards		Describes if the product is adhered to some recognized standard for structuring the information (e.g., IEEE Std15501, etc.)
	1	Standard Name	List of names of own standards used	List of names of standards used
3	Understandability			The capability of the product to enable the user to understand whether it is suitable, and how it can be used for particular tasks and conditions of use.
	1	Interface Understandability		Effort for recognizing the logical concepts introduced by the product by means of its interface
	1	Supported Interface Languages	Language=Set(Labels: Nominal); Labels= (Spanish, English, Catalan, ...)	Languages supported by the interface
	2	Global Structure		Effort for recognizing the logical concepts introduced by the product by means of its global structure
	1	Logical structure	3ValueOrder[Ordinal]; 3ValueOrder = (Satisfactory, Acceptable, Poor)	How recognizable and differentiable are the concepts introduced by the product.
4	Interpretability			The capability of the product to enable the user to correctly interpret the information
	1	Required Background		Describes the required background to understand the product
	1	Expert Background	ExpBack: Nominal; ExpBack=(Expert, Non-Expert)	Describes if expert background is required
	1	Level of Expertise	LevExp: Ordinal; LevExp=(High, Medium, Low)	Describes the level of expertise required
3	Accessibility Characteristics			Describes the extent to which the product is available or obtainable.
1	Availability			Describes aspects that affect availability
	1	Required Fee	Fee: Nominal; Fee=(True, False)	Describes if the product availability implies a fee
	1	Price	Price: Float (dollars)	Describes the price to paid for getting the product
	1	Availability Schema	Schema= Set (Labels: Nominal); Labels=(AnnualSubscription,Puntualpayment, ...)	Describes the availability schema
2	Easy of Operation			Describes to which extent it is easy to retrieve and manipulate the information.

Characteristic/Subcharacteristics/ Attributes		Metric	Description
1	Retrievability		Aspects that describe retrievability
1	Location	Loc= Set(String)	Describes the physical location(s) of the product
2	Retrieval Effort	RetEff: Ordinal; RetEff=(High, Medium, Low)	Describes the effort required to retrieve the product
3	Retrieval Type	RetType=Set(Labels: Nominal); Labels=(directDownload, SubscriptionBased, ...)	Describes the process required to get the product
4	Contextual Characteristics		Describes the extent to which the information source is applicable (pertinent) to the specific COTS selection project and its associated resources
1	Relevancy		Aspects that describe if the product is applicable to the project
1	Appropriateness	3ValueOrder[Ordinal]; 3ValueOrder = (Satisfactory, Acceptable, Poor)	Aspects that describe the appropriateness of the product to the COTS selection project
2	Timeliness		Aspects that describe if the timeliness is adequate for the COTS selection project
1	Required timeliness	3ValueOrder[Ordinal]; 3ValueOrder = (Satisfactory, Acceptable, Poor)	Aspects that describe how the information covers the required timeliness of the COTS selection project
3	Completeness		Aspects that describe to which extent the information source covers the informational needs
1	Suitability of the Scope	3ValueOrder[Ordinal]; 3ValueOrder = (Satisfactory, Acceptable, Poor)	Aspects that describe the suitability of the scope of the product to the COTS selection project
4	Appropriate Amount of Data		Aspects that describe if the size of the information source is adequate
1	Adequacy to the Processing capacity	3ValueOrder[Ordinal]; 3ValueOrder = (Satisfactory, Acceptable, Poor)	Aspects that describes the processing capacity of the COTS selection
5	Value-Added		Describes if the information source Add value to the COTS selection project operations
1	Value gained		Describes the value-added that provides the product to the COTS selection project
5	IQ Selection Project Characteristics		Describes the main IQ needs of the COTS selection project. (Such characteristics are currently being empirically obtained. Therefore this part of the model will be refined.
1	IQ Project Needs		Describes the IQ needs of the project

Characteristic/Subcharacteristics/Attributes		Metric	Description
1	Criticality of the Domain		Effort to recognize the criticality of the domain the COTS selection project belongs to
2	Project Changes Predictability		Effort to recognize the expected volatility of the domain
2	Allocated Resources		Aspects related to the set of resources allocated to the project for performing the COTS searching process
1	Human Resources		Describes the aspects related to human resources allocated to the project
1	1 Technical Skills		Describes the technical skills of the conformed team
	2 Person/Month		Describes the person/month assignment to the project
2	Non-Human Resources		Describes aspects related to Not-Human resources as subscription agreements, software resources, etc.
3	Deadline		Aspects that describe associated deadlines of the project
4	Monetary Budget		Aspects that describe the monetary budget associated to the project.

Annex 3. Domain Model for the RTSC Case Study

This Annex includes the complete Domain Model constructed for the RTSC Case Study.

Quality factor		Metric	Description
1 Functionality			ISO/IEC 9126-1
1	Suitability		ISO/IEC 9126 -1
	1 Suitability of Services		Effort to recognize how a particular COTS covers the main services expected from the domain
	1 Connect to Network Suitability		Describes how well the COTS covers the Connect to the Network service by an specific Transfer Protocol
	1	Connect to an intra-organizational network	3ValueOrder[Ordinal]; 3ValueOrder = (Satisfactory, Acceptable, Poor) Describes how well the COTS covers the Connect to the Network service in an intra-organizational network
	2	Connect to internet-based network	3ValueOrder[Ordinal]; 3ValueOrder = (Satisfactory, Acceptable, Poor) Describes how well the COTS covers the Connect to the Network service in an internet-based network
	3	Connect to a WAN network	3ValueOrder[Ordinal]; 3ValueOrder = (Satisfactory, Acceptable, Poor) Describes how well the COTS covers the Connect to the Network service in a WAN
	2 Infrastructure Suitability		Description of RTSC infrastructure elements
	1	Software Server Suitability	3ValueOrder[Ordinal]; 3ValueOrder = (Satisfactory, Acceptable, Poor) Describes how well the COTS covers the Software Server Service to provide the basic infrastructure to establish RTSC
	2	Software Client Suitability	3ValueOrder[Ordinal]; 3ValueOrder = (Satisfactory, Acceptable, Poor) Describes how well the COTS covers the Software Client Service to provide the basic infrastructure to establish RTSC
	3 Sessions Suitability		Description of users supported by the application
	1	User to User Session	3ValueOrder[Ordinal]; 3ValueOrder = (Satisfactory, Acceptable, Poor) Describes how well the COTS covers the User to User Session Service to connect a user with another user in RTSC
	2	Multi-user Session	3ValueOrder[Ordinal]; 3ValueOrder = (Satisfactory, Acceptable, Poor) Describes how well the COTS covers the Multi-User Session service to connect multiple users in RTSC
	4 Applications Suitability		Description of the kind of applications performed by the RTSC infrastructure
	1	Collaborative content creation	3ValueOrder[Ordinal]; 3ValueOrder = (Satisfactory, Acceptable, Poor) Capability of the system to create collaborative content

Quality factor		Metric	Description
2	Sharing resources	3ValueOrder[Ordinal]; 3ValueOrder = (Satisfactory, Acceptable, Poor)	Capability of the system to share resources
5	Messaging Suitability		Attributes related to the messaging suitability
	1 Message Types		Support to the management of messages
1	Text	3ValueOrder[Ordinal]; 3ValueOrder = (Satisfactory, Acceptable, Poor)	Possibility to send/deliver text messages
2	Audio	3ValueOrder[Ordinal]; 3ValueOrder = (Satisfactory, Acceptable, Poor)	Possibility to send/deliver audio messages
3	Video	3ValueOrder[Ordinal]; 3ValueOrder = (Satisfactory, Acceptable, Poor)	Possibility to send/deliver video messages
4	Data	3ValueOrder[Ordinal]; 3ValueOrder = (Satisfactory, Acceptable, Poor)	Possibility to send/deliver data messages
5	Multipart	3ValueOrder[Ordinal]; 3ValueOrder = (Satisfactory, Acceptable, Poor)	Possibility to send/deliver mixed (multipart) messages
2	Message Handling actions		Supported actions that can be performed over messages
1	Send/Receive Message	3ValueOrder[Ordinal]; 3ValueOrder = (Satisfactory, Acceptable, Poor)	Describes how well the COTS covers the Send/Receive Message service by means of a Software Client to enable RTSC among a RTSC-Server and human users.
2	Reply to messages	3ValueOrder[Ordinal]; 3ValueOrder = (Satisfactory, Acceptable, Poor)	Describes how well the COTS replies to message sender directly from received messages
3	Send messages to contact lists	3ValueOrder[Ordinal]; 3ValueOrder = (Satisfactory, Acceptable, Poor)	Describes how well the COTS supports mechanisms to store contact lists for sending them messages
4	Send and receive authenticated messages	3ValueOrder[Ordinal]; 3ValueOrder = (Satisfactory, Acceptable, Poor)	Describes how well the COTS supports mechanisms to authenticate messages originators
5	Send and receive encrypted messages	3ValueOrder[Ordinal]; 3ValueOrder = (Satisfactory, Acceptable, Poor)	Describes how well the COTS supports encryption algorithms to ensure message confidentiality
6	Coding/Decoding messages	3ValueOrder[Ordinal]; 3ValueOrder = (Satisfactory, Acceptable, Poor)	Describes how well the COTS supports coding and decoding mechanisms for sending and receiving messages
7	Send and receive free-virus messages	3ValueOrder[Ordinal]; 3ValueOrder = (Satisfactory, Acceptable, Poor)	Describes how well the COTS supports anti-virus mechanisms for sending and receiving messages

Quality factor		Metric	Description	
	8	Rules and Filters for incoming messages	3ValueOrder[Ordinal]; 3ValueOrder = (Satisfactory, Acceptable, Poor)	Possibility to apply rules and filters to incoming messages; e.g.: -To store incoming messages in specific folders depending on the sender -To avoid messages with specific addresses or content -To deny exchange of messages larger than a predefined size, etc.
	9	Message Tracking	3ValueOrder[Ordinal]; 3ValueOrder = (Satisfactory, Acceptable, Poor)	Describes how well the COTS supports mechanisms for tracking messages
6	Administration Services		Describes how well the COTS covers the Administrator Service to establish and manage accurately and efficiently the RTSC resources.	
	1	Contact List Management	Attributes related to the management of contact lists	
	1	Local Personal Contact list management	3ValueOrder[Ordinal]; 3ValueOrder = (Satisfactory, Acceptable, Poor)	Possibility to manage local personal address lists
	2	Local Common contact list management	3ValueOrder[Ordinal]; 3ValueOrder = (Satisfactory, Acceptable, Poor)	Possibility to manage local shared address lists
	3	Remote Personal Contact list management	3ValueOrder[Ordinal]; 3ValueOrder = (Satisfactory, Acceptable, Poor)	Possibility to manage remote personal address lists
	4	Remote Common contact list management	3ValueOrder[Ordinal]; 3ValueOrder = (Satisfactory, Acceptable, Poor)	Possibility to manage remote shared address lists
	5	List Categories management	3ValueOrder[Ordinal]; 3ValueOrder = (Satisfactory, Acceptable, Poor)	Possibility to assign and manage categories to the contacts
	6	Sorting Contact lists	3ValueOrder[Ordinal]; 3ValueOrder = (Satisfactory, Acceptable, Poor)	Possibility to sort contacts by the categories they belong to
	2	Configuration Services	Attributes related to the configuration services	
	1	Assisted Configuration of Software Server	3ValueOrder[Ordinal]; 3ValueOrder = (Satisfactory, Acceptable, Poor)	Possibility to be assisted to configure RTSC- server
	2	Assisted Configuration of Software Client	3ValueOrder[Ordinal]; 3ValueOrder = (Satisfactory, Acceptable, Poor)	Possibility to be assisted to configure RTSC- client
	3	Overall Configuration Management	3ValueOrder[Ordinal]; 3ValueOrder = (Satisfactory, Acceptable, Poor)	Possibility to be assisted in the overall configuration of the RTSC system
2	Suitability of Data		Effort to recognize how a particular COTS provides the	

Quality factor		Metric	Description
			data represented by the general class model describing the domain. See Fig. 6.3a for an excerpt view of the conceptual model of the domain
1	Message	3ValueOrder[Ordinal]; 3ValueOrder = (Satisfactory, Acceptable, Poor)	Describes how well the COTS covers the concept and its definition.
2	Connected with	3ValueOrder[Ordinal]; 3ValueOrder = (Satisfactory, Acceptable, Poor)	Describes how well the COTS covers the concept and its definition.
3	User	3ValueOrder[Ordinal]; 3ValueOrder = (Satisfactory, Acceptable, Poor)	Describes how well the COTS covers the concept and its definition.
4	User to User Session	3ValueOrder[Ordinal]; 3ValueOrder = (Satisfactory, Acceptable, Poor)	Describes how well the COTS covers the concept and its definition.
5	Multi-user Session	3ValueOrder[Ordinal]; 3ValueOrder = (Satisfactory, Acceptable, Poor)	Describes how well the COTS covers the concept and its definition.
6	Transfer Protocol	3ValueOrder[Ordinal]; 3ValueOrder = (Satisfactory, Acceptable, Poor)	Describes how well the COTS covers the concept and its definition.
7	Logging mechanisms	3ValueOrder[Ordinal]; 3ValueOrder = (Satisfactory, Acceptable, Poor)	Describes how well the COTS covers the concept and its definition.
8	One way communication	3ValueOrder[Ordinal]; 3ValueOrder = (Satisfactory, Acceptable, Poor)	Describes how well the COTS covers the concept and its definition: Also called treated applications are essentially one way flows of information. Typical examples would include information services such as stock prices, or traffic information, and broadcast or on-demand video and audio services.
9	Interactive Communication	3ValueOrder[Ordinal]; 3ValueOrder = (Satisfactory, Acceptable, Poor)	Describes how well the COTS covers the concept and its definition: Also called conversational applications are primarily interactive and will usually have humans present at each end. Typical applications include Internet telephony, audio or video or data conferencing, application sharing, text-based chat, networked games, shared virtual worlds or distributed simulations.
10	Room or channel	3ValueOrder[Ordinal]; 3ValueOrder = (Satisfactory, Acceptable, Poor)	Describes how well the COTS covers the concept and its definition : A virtual venue where a number of people can meet to talk.
11	...		

Quality factor		Metric	Description
2	Accuracy		
	1	Verifiableness	Provision of resources to allow the tracking and verification of the right or agreed results or effects
	1	History and Versioning	3ValueOrder[Ordinal]; 3ValueOrder = (Satisfactory, Acceptable, Poor) Describes how well the COTS provides a history of the changes on the data managed
	2	Logging Capabilities	3ValueOrder[Ordinal]; 3ValueOrder = (Satisfactory, Acceptable, Poor) Describes how well the COTS provides logging mechanisms
	2	Effectiveness	
	1	Self-test results	3ValueOrder[Ordinal]; 3ValueOrder = (Satisfactory, Acceptable, Poor) Describes how well the COTS provides mechanisms to perform direct tests of the right or agreed results or effects over the system
	2	Published tests results	3ValueOrder[Ordinal]; 3ValueOrder = (Satisfactory, Acceptable, Poor) Describes how third party reports state the effects of the system in similar environments
3	Interoperability		ISO/IEC 9126-1 Capability of the software product to interact with one or more specified systems
	1	Direct Interoperability	Capability of the system to directly interact with specified systems
	1	By means of Protocols	Capability to directly interact with other systems by means of supported protocols
	1	Real time transfer protocols	Protocols:Set(Label:Nominal); Label=(H323, SIP, IRC, ...) Supported protocols to send and relay RTSC messages
	2	Real time access protocols	Protocols:Set(Label:Nominal); Label=(RFC2810, ...) Supported protocols used by RTSC-clients to access messages in the server
	3	Network protocols	Protocols:Set(Label:Nominal); Label=(HTTP, ...) Supported Network applications protocols
	4	Wireless protocols	Protocols:Set(Label:Nominal); Label=(WAP, ...) Supported wireless protocols
	2	By means of APIs (connectors)	Capability to directly interact with other systems
	1	To Anti-Virus Tools	Describes the RTSC-System interoperability with Anti-virus tools
	1	Robust Virus Detection	3ValueOrder[Ordinal]; Describes how the COTS accomplish the Robust Virus

Quality factor		Metric	Description
		3ValueOrder = (Satisfactory, Acceptable, Poor)	Detection softgoal.
	2	Message Scanned for Virus GoalValue[Ordinal]; GoalValue = (Attained, Not Attained)	Describes if the COTS accomplish the Message Scanned for Virus goal
	3	Message ResourceValue[Ordinal]; ResourceValue = (Provided, NotProvided)	Describes if the COTS provides the Message resource (i.e., if they are compatible)
	2	Configuration and Administration Tools	Describes the RTSC-System interoperability with Configuration and Administration tools
	1	Easy Administration 3ValueOrder[Ordinal]; 3ValueOrder = (Satisfactory, Acceptable, Poor)	Describes how the COTS accomplish the Easy Administration softgoal
	2	Parameter Values ResourceValue[Ordinal]; ResourceValue = (Provided, NotProvided)	Describes if the COTS provides the Parameter Values resource
	3	Management Assisted GoalValue[Ordinal]; GoalValue = (Attained, Not Attained)	Describes if the COTS accomplish the Management Assisted goal
	3	Backup and Recovery tools	Describes the RTSC-System interoperability with Backup and Recovery tools
	1	Data Backup & Restore GoalValue[Ordinal]; GoalValue = (Attained, Not Attained)	Describes if the COTS accomplish the Data Backup & Restore goal
	2	System Backup & Restore GoalValue[Ordinal]; GoalValue = (Attained, Not Attained)	Describes if the COTS accomplish the System Backup & Restore goal
	4	Billing Tools (BT)	Describes the RTSC-System interoperability with Billing Tools
	1	Resource Usage Tracked GoalValue[Ordinal]; GoalValue = (Attained, Not Attained)	Describes if the COTS accomplish the Resource Usage Tracked goal
	2	Track resources consumptions 3ValueOrder[Ordinal]; 3ValueOrder = (Satisfactory, Acceptable, Poor)	Describes how the COTS satisfies the Track Resources consumptions
	5	Message Tracking Tools (MTT)	Describes the RTSC-System interoperability with Message Tracking Tools
	1	Messages Tracked GoalValue[Ordinal]; GoalValue = (Attained, Not Attained)	Describes if the COTS accomplish the Messages Tracked goal
	2	Track Messages 3ValueOrder[Ordinal];	Describes how the COTS satisfies the Tracking of

Quality factor		Metric	Description
6	Configuration and Administration Tools (CAT)	3ValueOrder = (Satisfactory, Acceptable, Poor)	Messages Describes the RTSC-System interoperability with Configuration and Administration Tools
	1 Management Assisted	GoalValue[Ordinal]; GoalValue = (Attained, Not Attained)	Describes if CAT tools assists on goals as services configured, performance tuning, services recovered, etc.
7	Codec/Decoded Tools	3ValueOrder[Ordinal]; 3ValueOrder = (Satisfactory, Acceptable, Poor)	Describes how CAT Tools satisfy the Good Performance requirement. Describes the RTSC-System interoperability with Coded and Decoded Tools
	2 Good Performance		
8	Directory Services Tools		Describes how Codec and Decoded Tools satisfy the Good Performance requirement. Describes the RTSC-System interoperability with Configuration and Administration tools
	1 Good Performance	3ValueOrder[Ordinal]; 3ValueOrder = (Satisfactory, Acceptable, Poor)	
	2 Compress/Decompress Messages	TaskValue[Ordinal]; TaskValue = (Executed, Failed)	Describes if the Coded/Decoded Tool accomplish the compress/decompress message task
9	Data Encryption Tools (DET)	ResourceValue[Ordinal]; ResourceValue = (Provided, NotProvided)	Describes if the COTS provides the Message resource (i.e. if they are compatible)
	3 Message		
	4 Resources Accessed	3ValueOrder[Ordinal]; 3ValueOrder = (Satisfactory, Acceptable, Poor)	Describes how Directory Services Tools satisfy the Access to resources requirements
	5 Permissions Assigned	3ValueOrder[Ordinal]; 3ValueOrder = (Satisfactory, Acceptable, Poor)	Describes how Directory Services Tools satisfy the Assignment of permissions requirements
	1 Network Resources Managed	3ValueOrder[Ordinal]; 3ValueOrder = (Satisfactory, Acceptable, Poor)	Describes how Directory Services Tools satisfy the Network Resource Management requirements
2 Resources Stored	3ValueOrder[Ordinal]; 3ValueOrder = (Satisfactory, Acceptable, Poor)	Describes how Directory Services Tools satisfy the storing of resources requirements	
3 Resources Assigned	3ValueOrder[Ordinal]; 3ValueOrder = (Satisfactory, Acceptable, Poor)	Describes how Directory Services Tools satisfy the Assignment of resources requirements	

Quality factor		Metric	Description	
	1	Messages Encrypted/Decrypted	GoalValue[Ordinal]; GoalValue = (Attained, Not Attained)	Describes if the Data Encryption Tools accomplish the Messages Encrypted/Decrypted goal
		Good Performance	3ValueOrder[Ordinal]; 3ValueOrder = (Satisfactory, Acceptable, Poor)	Describes how the Data Encryption Tools satisfy the Good performance requirement
	1 0	Routing Tools (RT)		Describes the RTSC-System interoperability with Routing Tools
	1	Messages Routed	GoalValue[Ordinal]; GoalValue = (Attained, Not Attained)	Describes if the Routing Tools accomplish the Messages Routed goal
		Good Performance	3ValueOrder[Ordinal]; 3ValueOrder = (Satisfactory, Acceptable, Poor)	Describes how the Routing Tools satisfy the Good performance requirement
	1 1	Data Compression Tools (DCT)		Describes the RTSC-System interoperability with Data Compression Tools
	1	Messages Compressed/Decompressed	GoalValue[Ordinal]; GoalValue = (Attained, Not Attained)	Describes if Data Compression Tools accomplish the Messages compressed/decompressed goal
		Good Performance	3ValueOrder[Ordinal]; 3ValueOrder = (Satisfactory, Acceptable, Poor)	Describes how the Data Compression tools satisfy the Good performance requirement
	1 2	Anti-Spam Tools (AST)		Describes the RTSC-System interoperability with Anti-Spam Tools
	1	Messages Filtered of Spam	GoalValue[Ordinal]; GoalValue = (Attained, Not Attained)	Describes if Anti-Spam Tools accomplish the Messages filtered from spam goal
		Good Performance	3ValueOrder[Ordinal]; 3ValueOrder = (Satisfactory, Acceptable, Poor)	Describes how the Anti-Spam tools satisfy the Good performance requirement
		Protect From Unauthorized communication	3ValueOrder[Ordinal]; 3ValueOrder = (Satisfactory, Acceptable, Poor)	Describes how the Anti-Spam tools satisfy the Protect from Unauthorized communication requirements
2		Indirect Interoperability		Capability to interact with other systems by means if indirect mechanisms
	1	Anti-virus Organizations		Describes the COTS indirect interoperability with Anti-Virus organizations
	1	Worldwide Updated Virus List	GoalValue[Ordinal]; GoalValue = (Attained, Not Attained)	Describes if the COTS accomplish the Worldwide update virus list goal

Quality factor		Metric	Description
2	New Virus List	ResourceValue[Ordinal], ResourceValue = (Provided, NotProvided)	Describes if the COTS provides New virus list automatically
2	Certification Authorities		Describes the COTS indirect interoperability with Certification Authorities
1	Public Key Certificated	GoalValue[Ordinal], GoalValue = (Attained, Not Attained)	Describes if the COTS accomplish the Public Key certificated goal
2	Digital Certificated	ResourceValue[Ordinal], ResourceValue = (Provided, NotProvided)	Describes if the COTS provides digital certification
3	Domain Name Server		Describes the COTS indirect interoperability with Domain Name Servers
1	Up-to date Management of Routing tables	3ValueOrder[Ordinal]; 3ValueOrder = (Satisfactory, Acceptable, Poor)	Describes how the COTS satisfy the up-dating management of routing tables requirements
2	Destination IP address	ResourceValue[Ordinal]; ResourceValue = (Provided, NotProvided)	Describes if the COTS provides destination IP Address
3	Routing status	ResourceValue[Ordinal], ResourceValue = (Provided, NotProvided)	Describes if the COTS provides status of the routing
4	Firewall		Describes the COTS indirect interoperability with Domain Name Servers
1	Protect from Unauthorized access	3ValueOrder[Ordinal]; 3ValueOrder = (Satisfactory, Acceptable, Poor)	Describes how the COTS satisfy the protect from unauthorized access requirements
4	Security		ISO/IEC 9126-1
1	Application Security		Mechanisms to prevent the accidental or deliberated unauthorized access system functionality
1	Provided by the Application		Mechanisms provided by the system itself
1	Login and Password	GoalValue[Ordinal], GoalValue = (Attained, Not Attained)	Describes if the COTS provides mechanisms of login control with user names and password authentication
2	Execution Control Lists (ECL)	GoalValue[Ordinal], GoalValue = (Attained, Not Attained)	Describes if the COTS provides mechanisms of listing executable files allowed to run on server, specially useful to protect against virus executables

Quality factor			Metric	Description
	3	Access Control Lists	GoalValue[Ordinal], GoalValue = (Attained, Not Attained)	Describes if the COTS provides mechanisms of listing of access privileges to files. They can be defined at local user, group or rest of the world levels
	4	Trust Relationships	GoalValue[Ordinal], GoalValue = (Attained, Not Attained)	Describes if the COTS provides mechanisms of inter-domain level privileges, for interconnection and sharing of resources between different domain users
	2	Provided by Third Parties		Mechanisms provided by the system with the aid of third party organization
	1	Certification System	GoalValue[Ordinal], GoalValue = (Attained, Not Attained)	Describes if the COTS provides supported certification mechanisms
	2	Data Security		Mechanisms to prevent the accidental or deliberated unauthorized access to the data managed by the system
	1	Stored Data		Mechanisms to prevent the unauthorized access to the data stored by the system
	1	Login and Password	GoalValue[Ordinal]; GoalValue = (Attained, Not Attained)	Describes if the COTS provides mechanisms of login control with user names and password authentication
	2	Execution Control Lists (ECL)	GoalValue[Ordinal]; GoalValue = (Attained, Not Attained)	Describes if the COTS provides mechanisms of listing executable files allowed to run on server, specially useful to protect against virus executables
	3	Access Control Lists (ACL)	GoalValue[Ordinal]; GoalValue = (Attained, Not Attained)	Describes if the COTS provides mechanisms of listing of access privileges to files. They can be defined at local user, group or rest of the world levels
	2	Transmitted Data		Mechanisms to prevent the unauthorized access to the data transmitted by the system
	1	Secure transfer protocols	Protocols:Set(Labels:Nominal); Labels=(SSL, ...)	Describes the secure transfer Protocols supported by the COTS
	2	Secure Web transfer protocols	Protocols:Set(Labels:Nominal); Labels=(S-HTML, ...)	Describes the secure web transfer Protocols supported by the COTS
	3	Secure MIME support	Supported:Set(Labels:Nominal); Labels=(True,False)	Describes if the COTS support the MIME standard protocol
5		Functionality Compliance		ISO/IEC 9126-1
	1	Supported RFC's		Attributes describing the satisfaction of standards RFC

Quality factor		Metric	Description
1	RFC3261	3ValueOrder[Ordinal]; 3ValueOrder = (Satisfactory, Acceptable, Poor)	Describes how well the COTS adheres to the RFC3261
2	RFC2810	3ValueOrder[Ordinal]; 3ValueOrder = (Satisfactory, Acceptable, Poor)	Describes how well the COTS adheres to the RFC2810
3	...		
2 Reliability			ISO/IEC 9126-1
1	Maturity		The capability of the software product to avoid failure as a result of faults in the software.
1	Product History		Historic data of the system which leading to the provision of more mature versions over the time
1	Time of Product on Market	Period: Ratio; Period = Float[Year]	Time that the product has on the market
2	Product Versions	Versions: Set(<Label: Ordinal, TimeOnMarket: Ratio>); Label=(unknown), TimeOnMarket = Float(Year)	List of versions available on the market
2	Robustness		Mechanisms to maintain a history if system faults affecting system operation
1	Preoperational Robustness		Mechanisms to maintain a history of system faults affecting system operation before the system is made available to the users
1	Mean Time Between Failure	Period: Ratio; Period = Float[Hours]	Average time between a failure on the system is detected
2	Published Test Results	Test:Set(<Author:Nominal, Date:Absolute>); Author=(unknown), Date=[mm/dd/aaaa]	Third party published benchmarks and test
2	Operation Robustness		
1	Mean Time Between Failure	Period: Ratio; Period = Float[Hours]	Average time between a failure on the system is detected
2	Mean Time to Repair	Period: Ratio; Period = Float[Hours]	Average time required to restore the system operation
2 Fault Tolerance			ISO/IEC 9126-1
1	Transparency		Capacity of the system to keep up its operation without making users aware of its faults

Quality factor		Metric	Description
	1	Automatic Delivery Retries	Supported: Nominal; Supported=(True, False) Support for the automatic relaying of messages in case of delivery failure
3	Recoverability		ISO/IEC 9126-1
	1	System Recoverability	
	1	Automatic Recover from the scratch	Supported: Nominal; Supported=(True, False) Possibility to automatically recover operation in case of system failures
4	Reliability Compliance		ISO/IEC 9126-1
3	Usability		ISO/IEC 9126 -1
	1	Understandability	ISO/IEC 9126 -1
	1	Semantic Understandability	Number[Unit]; Number=Integer Describes the number of semantic discrepancies of the particular component with respect to the reference domain models
	2	Lexical Understandability	Number[Unit]; ; Number=Integer Describes the number of lexical discrepancies of the particular component with respect to the reference domain models
	3	Interface Understandability	Effort to recognize the logical concepts and its applicability by means of interfaces
	1	Adherence to Best Practices	ADP: 4valueOrder[Ordinal]; 4valueOrder = (Optimal, Good, Fair, Poor) Describes how well events and elements of the interface comply with best practices recognized for user interfaces
	2	Supported Interface Languages	SIL: Languages = Set(Labels[Nominal]); Labels = (Spanish, Catalan, English, ...) Languages supported by the interface
2	Learnability		
	1	Training	Training mechanisms provided to learn the software application
	1	Vendors Provided Training	Training:Ordinal; Training=(Not Provided, Basic, Medium, Advanced) Training provided by the supplier of the component
	2	Third Party Provided training	Training: Set (Source: Nominal, Level: Ordinal); Source(Unknown), Level(Basic, Medium, Advanced) Note: Source refers to the individual/organization providing the training Training provided by organizations or individual other than the vendor of the component

Quality factor		Metric	Description
3	Tutorials	Tutorials:Nominal; Tutorials=(Available, Partially Available, Not Available)	Are there multimedia courses provided with software package or available online
2	Documentation		Documentation than can be used to learn the software application
1	Provided Documentation		
1	Documentation and User Manuals	Content: Nominal; Content=(Not Provided, Basic, Medium, Advanced)	Are user and installation manuals as well as other documentation provided with the component?
2	FAQs and Tips	Content: Nominal; Content=(Not Provided, Basic, Medium, Advanced)	Are frequently asked questions and user tips documents provided?
3	Help Files	Content: Nominal; Content=(Not Provided, Basic, Medium, Advanced)	Are help files provided?
4	Online help	Content: Nominal; Content=(Not Provided, Basic, Medium, Advanced)	Is there and internet online help available?
2	External Documentation		Documentation available from sources external to the software application or its provider
1	Vendors Customers support	Support: (Provided: Nominal, Quality: Ordinal); Provided=(Not Provided, Partial, Provided); Quality : (Poor, Fare, good, Excellent)	Does the provider company of the component or its representatives have a customer support department? If they do, how well prepared in the use of the application are the technicians? Do they provide support for the installation/configuration?
2	Online help	Content: Nominal; Content=(Not Provided, Basic, Medium, Advanced)	Is there an external internet online help available?
3	Published Documentation	Docum: Nominal; Docum =(Available, Partially Available, Not Available).	Are there Information sources e.g. books, white papers, etc (other than the provided by the supplier) available for its review?
3	Operability		ISO/IEC 9126-1
1	System Tailorability		Mechanisms of the system to be configured to operate in certain way
1	Global System Tailorability		Mechanisms of the system to be configured to operate in certain way by its administrator

Quality factor		Metric	Description
1	Accounts Administration		Attributes related to the management of users and users accounts
1	Individual Users Management	Configurable: Nominal; Configurable=(True, False)	Support to the definition/management of individual users of the component
2	Users Groups Management	Configurable: Nominal; Configurable=(True, False)	Support to the definition/management of users groups
3	Private and Public Accounts	Configurable: Nominal; Configurable=(True, False)	Support to the definition/management of public and private accounts
4	Users Profiles	Configurable: Nominal; Configurable=(True, False)	Can standards profiles be defined and assigned to individual users or groups?
2	Resources Administration		Attributes related to the management of system resources
1	Web Based Administration	Supported: Nominal; Supported=(True, False)	Authorized administrators can perform tasks such as users and groups management and messages monitoring, from anywhere using a web browser?
2	Administrative Tools and Wizards	Tool: Set(Labels: Nominal); Labels=(Message tracking, Billing Services, ...)	Set of utilities designed to automate configuration and some commonly performed tasks
4	Attractiveness		ISO-IEC 9126-1
5	Usability Compliance		ISO-IEC 9126-1
4	Efficiency		ISO-IEC 9126-1
1	Time Behaviour		ISO-IEC 9126-1
1	Message Throughput	Function: (Platform: Nominal) x (NumberOfUsersConected: Absolute) x (MessageSize: Absolute)	Amount of time required to send a message
2	Multiprocess Support	Supported: Nominal; Supported=(True, False)	Possibility to support administrative tasks such as message store, defragmentation and space recovery, without stopping services
2	Resource Utilization		ISO-IEC 9126-1
1	Deployment		Resources required by the system during its deployment
1	Hardware Resources Required	Resources: Set(<Name:Nominal, Requirement:Nominal>; Name=(RAM, Processor, HD, ...), Requirement=Label[ResourceUnit])	Hardware resources required to deploy the component

Quality factor		Metric	Description		
	2	Software Resources Required	Resources: Set(Labels: Nominal); Labels=(OS, ...)	Software resources required to deploy the component	
3	Efficiency Compliance			ISO-IEC 9126-1	
5	Maintainability			ISO-IEC 9126-1	
	1	Analyzability		ISO-IEC 9126-1	
	1	Analyzable Data		Available data to perform analysis of the system	
	1	History and Versioning		Capability of the system to provide a history of the changes on the data managed	
		1	History Files Information	Information provided by the history files	
		1	Fields	Fields:Set(Labels: Nominal); Labels=(Date, Time, ChangePerformed, ...)	List of fields recorded in the history files
		2	Events	Events:Set(Labels: Nominal); Labels=(Save, Replace, Delete, ...)	List of the events which record information on the history files
	2	Build in Testing Capabilities		Built in testing capabilities implemented into the system	
		1	Message Delivery Notifications	Supported: Nominal; Supported=(True, False)	Information automatically provided by the server if delivery problems are found
		2	Message Reception Notifications	Supported: Nominal; Supported=(True, False)	Information automatically provided by the server when new messages arise
		3	Message Tracking and Monitoring	Supported: Nominal; Supported=(True, False)	Tracking of messages across network domains. Users can check the status of their sent messages.
	2	Build In Analysis Capabilities		Mechanisms provided by the system to generate/store versions of the system data	
		1	Message Tracking and Monitoring	BuildIn: Nominal; BuildIn=(True, False)	Tracking of messages across network domains. Users can check the status of their sent messages.
		2	Automated RTSC Server Usage reporting	Supported: Nominal; Supported=(True, False)	Manage the messaging environment via direct statistical analysis of servers' performance and connectivity. For example: track the number of web client users versus non-web based clients to a server
		3	Expert Analysis Tools	Tools:Set(Labels: Nominal);	Analyze server functions over time for performance tuning,

Quality factor		Metric	Description
		Labels=(unknown)	capacity planning and trend prediction. Set and track service level agreements, correlate performance statistics and more
	4	Billing Services	BuildIn: Nominal; BuildIn=(True, False) Track, report and analyze system usage for billing, charge-back and capacity planning purposes
2	Changeability		
	1	Development Documentation	
	1	Documentation and User Manuals	Content: Nominal; Content=(Not Provided, Basic, Medium, Advanced) Are user and installation manuals as well as other documentation provided with the component?
	2	FAQs and Tips	Content: Nominal; Content=(Not Provided, Basic, Medium, Advanced) Are frequently asked questions and user tips documents provided?
	3	Help Files	Content: Nominal; Content=(Not Provided, Basic, Medium, Advanced) Are help files provided?
	4	Online help	Content: Nominal; Content=(Not Provided, Basic, Medium, Advanced) Is there and internet online help available?
	5	Vendors Customers support	Support: (Provided: Nominal, Quality: Ordinal); Provided=(Not Provided, Partial, Provided); Quality : (Poor, Fare, good, Excellent) Does the provider company of the component or its representatives have a customer support department? If they do, how well prepared in the use of the application are the technicians? Do they provide support for the installation/configuration?
	6	External Online help	Content: Nominal; Content=(Not Provided, Basic, Medium, Advanced) Is there an external internet online help available?
	7	Published Documentation	Docum: Nominal; Docum =(Available, Partially Available, Not Available). Are there Information sources e.g. books, white papers, etc (other than the provided by the supplier) available for its review?
3	Stability		
	1	Operational Stability	
	1	Updates Frequency Rate	UpdRate: ratio; UpdRate=Integer[Times/Year] Average time among updates (patches) of the component
4	Testability		
			ISO/IEC 9126-1

Quality factor		Metric	Description
1	Message Delivery Notifications	Supported: Nominal; Supported=(True, False)	Information automatically provided by the server if delivery problems are found
2	Message Reception Notifications	Supported: Nominal; Supported=(True, False)	Information automatically provided by the server when new messages arise
3	Message Tracking and Monitoring	Supported: Nominal; Supported=(True, False)	Tracking of messages across network domains. Users can check the status of their sent messages.
4	Expert Analysis Tools	Tools:Set(Labels: Nominal); Labels=(unknown)	Analyze server functions over time for performance tuning, capacity planning and trend prediction. Set and track service level agreements, correlate performance statistics and more
5 Maintainability Compliance			ISO/IEC 9126-1
6 Portability			ISO/IEC 9126-1
1 Adaptability			ISO/IEC 9126-1
1	Supported Operating Systems	OS:Set(Labels: Nominal); Labels=(Windows, Unix, Linux, ...)	Choice of operating systems over which RTSC servers may be installed and run
2	Supported Hardware Platforms and Architectures	Platform: Set(Labels: Nominal); Labels=(Intel X-86, IBM AS/400, SunSparc, DEC Alpha, ...)	Choice of hardware architectures over which RTSC servers may be installed and run
3 Choice of RTSC-Clients			Different kinds of clients supported by the RTSC server
1	RTSC program Clients	Client: Set(Labels: Nominal); Labels=(IM clients, ...)	Users that connect to the RTSC server using non-web based application clients
2	Web-Based Clients	Client: Set(Labels: Nominal); Labels=(IM clients, ...)	Users that connect to the RTSC server using web based client applications
3	Mobile Devices Clients	Client: Set(Labels: Nominal); Labels=(PDA, Celphones, ...)	Users that connect to the RTSC server using mostly proprietary pieces of software.
2 Installability			ISO/IEC 9126-1
1 Built In Installation Facilities			Built in capabilities to assist on system installation
1	Administrative Tools and Wizards	Tools: Nominal; Tools=(Installation wizards, configuration tools, ...)	Set of utilities designed to automate configuration and some commonly performed tasks

Quality factor		Metric	Description
2	Installability Support		
1	Documentation and User Manuals	Content: Nominal; Content=(Not Provided, Basic, Medium, Advanced)	Are user and installation manuals as well as other documentation provided with the component?
2	FAQs and Tips	Content: Nominal; Content=(Not Provided, Basic, Medium, Advanced)	Are frequently asked questions and user tips documents provided?
3	Help Files	Content: Nominal; Content=(Not Provided, Basic, Medium, Advanced)	Are help files provided?
4	Online help	Content: Nominal; Content=(Not Provided, Basic, Medium, Advanced)	Is there and internet online help available?
5	Vendors Customers support	Support: (Provided: Nominal, Quality: Ordinal); Provided=(Not Provided, Partial, Provided); Quality : (Poor, Fare, good, Excellent)	Does the provider company of the component or its representatives have a customer support department? If they do, how well prepared in the use of the application are the technicians? Do they provide support for the installation/configuration?
6	External Online help	Content: Nominal; Content=(Not Provided, Basic, Medium, Advanced)	Is there an external internet online help available?
7	Published Documentation	Docum: Nominal; Docum =(Available, Partially Available, Not Available).	Are there Information sources e.g. books, white papers, etc (other than the provided by the supplier) available for its review?
3	Platform Compatibility		Capability of the system to be installed in a specific platform
1	Supported Operating Systems	OS:Set(Labels: Nominal); Labels=(Windows, Unix, Linux, ...)	Choice of operating systems over which RTSC servers may be installed and run
2	Supported Hardware Platforms and Architectures	Platform: Set(Labels: Nominal); Labels=(Intel X-86, IBM AS/400, SunSparc, DEC Alpha, ...)	Choice of hardware architectures over which RTSC servers may be installed and run
3	Coexistence		ISO/IEC 9126-1
1	By means of Protocols		Capability to directly interact with other systems by means of supported protocols
1	Real time transfer protocols	Protocols:Set(Label:Nominal); Label=(H323, SIP, IRC, ...)	Supported protocols to send and relay RTSC messages
2	Real time access protocols	Protocols:Set(Label:Nominal); Label=(RFC2810, ...)	Supported protocols used by RTSC-clients to access messages in the server

Quality factor		Metric	Description	
	3	Network protocols	Protocols:Set(Label:Nominal); Label=(HTTP, ...)	Supported Network applications protocols
	4	Wireless protocols	Protocols:Set(Label:Nominal); Label=(WAP, ...)	Supported wireless protocols
	2	By means of APIs (connectors)		Capability to directly interact with other systems
	...			
4	Replaceability			ISO/IEC 9126-1
	1	Build In Migration Tools		Migration tools built into the system
	1	To/From Other RTSC Servers	Tools: Set(Labels: Nominal); Labels= (unknown)	Tools to migrate systems and user data to/from other RTSC servers
	2	To/From Other OS	Tools: Set(Labels: Nominal); Labels= (unknown)	Tools to migrate system and user data to/from other RTSC server of the same brand in a different operating system
5	Portability Compliance			ISO/IEC 9126-1
Non-technical Factor		Metric	Description	
1	Supplier		Characteristics of the supplier that can influence the quality of the software product. See [Car+07b]	
	1	Organizational Structure	Description of the organizational structure of the supplier of the component	
	2	Positioning and Strength	Description of the position and orientation of the supplier company in the market	
	3	Reputation	Recognition of the capability of the supplier to perform similar projects based on past experiences and certifications	
	1	Supplier Company Existence	NumberOfYears: Integer	Years of the supplier company in the market from its foundation
	2	Quality Process Certification		Certifications of the quality of the process followed by the supplier company given by recognized certification authorities
	1	CMM Level	CMM Level: Integer (1..5)	Capability Maturity Model Level granted to the supplier company

Quality factor		Metric	Description
2	ISO 9000	ISO9000: Boolean	ISO 9000 certificate granted to the supplier company
3	Other Certificates	List Of (Certificate, Level); Certificate: (Spice, SixSigma, ...); Level: String	Other quality process certificates
3	Client Recommendations	List of (Client, Comments); Client:String; Comments:List of String	References and recommendations of the supplier company that other clients have given
4	Services Offered		Description of the services offered by the supplier
1	Organizational Analysis and Process Reengineering	Tuple(ServiceOffered, Description); ServiceOffered: Boolean; Description: String	The supplier company offers services to analyze the current business process of the client and services to restructure these processes in order to align them with the offered system
2	Organizational Change Management	Tuple(ServiceOffered, Description); ServiceOffered: Boolean; Description: String	The supplier company offers services to manage the change from the new system, foreseen the possible risks and providing attenuating measures
3	Parameterization and Adaptation of the Offered Systems	Tuple(ServiceOffered, Description); ServiceOffered: Boolean; Description: String	The supplier company offers services to adapt and parameterized the offered systems on the production framework of the client
4	Installation of the Offered Systems	Tuple(ServiceOffered, Description); ServiceOffered: Boolean; Description: String	The supplier company offers services of installation of the offered systems in the production framework of the client
5	Integration of the Offered Systems	Tuple(ServiceOffered, Description); ServiceOffered: Boolean; Description: String	The supplier company offers services of integration of the offered systems in the production framework of the client
6	Training and Teaching the Offered Systems	Tuple(ServiceOffered, Description); ServiceOffered: Boolean; Description: String	The supplier company offers services of introduction to the new systems that are sold to the client
7	Other Services Offered	List of OtherService; OtherService: Tuple(ServiceOffered, Description)	List of other services offered and possible comments about them
5	Support		Description of the support mechanisms offered by the supplier company
1	Support Channels		Support channels among the supplier company and its clients
1	Direct Support	Tuple(ChannelOffered, Description); ChannelOffered: Boolean; Description: String	The supplier company is who gives support to its clients
2	Indirect Support	Tuple(ChannelOffered, Companies, Description); ChannelOffered: Boolean; Companies: List of String; Description: String	Third party companies are who give support to the clients

Quality factor		Metric	Description
3	Mixed Support	Tuple(ChannelOffered, Companies, Description); ChannelOffered: Boolean; Companies: List of String; Description: String	A combination of the supplier company and third party companies are who give support to the clients
4	Other Supports	List Of Channels; Channels: Tuple(ChannelName, Description); ChannelName: String; Description: String	Other support channels are provided
2	Support Types		Description of the support methods offered by the supplier company
1	Help desk	Tuple (ServiceOffered, Description); ServiceOffered: Boolean; Description: String	The supplier company provides on-demand remote support to the clients
2	Presence Support	Tuple (ServiceOffered, Description); ServiceOffered: Boolean; Description: String	The supplier company provides direct support in presence to the clients
3	Incidence Repository	Tuple (ServiceOffered, Description); ServiceOffered: Boolean; Description: String	The supplier company provides a repository of problem lists and how to solve them to help the clients
4	Other Support Types	List Of SupportType; SupportType: Tuple (ServiceOffered, Description); ServiceOffered: Boolean; Description: String	Other support types are provided
3	Territory Covered		Describes the territory covered by the support channel
2	Business		See [Car+07b]
1	Licensing Schema		Description of the COTS components licensing options
1	Licensing Types	List of LicensingTypes; LicensingTypes: (PerUserGroups, PerServer, PerClients, ...)	List of common licensing options provided
2	Other Types	List of OtherLicensingTypes; OtherLicensingTypes: Tuple(LicensingType, Description); LicensingType: String; Description: String	Other licensing options offered
2	Ownership		Description of the aspects in relation to the intellectual property rights
1	Own Made Product	Manufacturer: Boolean	The supplier company has developed the product
2	Third Party Product	Tuple (Manufacturer, Relationship, ProductReference); Manufacturer: String; Relationship:(Distributor, Partner, ...);	The supplier company has not developed the product

Quality factor		Metric	Description
		ProductReference: String	
3	Ownership Rights		Ownership of the product once installed into the production framework of the client
	1 Copyright	Maintained: Boolean	The supplier company maintains the ownership once the product is installed into the production framework of the client
	2 Copyleft	WithChangesAllowed: Boolean	The supplier company maintains the ownership, but the client can make changes in its copy of the product
	3 Ownership Transferred	ClientOwnership: Boolean	The supplier company gives the ownership to the client
	4 Open Versions	Open: Boolean	There is not an owner of the product
	5 Source Code	SourceCode: Boolean	The supplier company gives the source code to the client
3	Guarantees		Detail of guarantees provided over the product
	1 Guarantees of Fulfillment of the Installation dates	List Of Guarantee; Guarantee: Tuple(Name, Description); Name: String; Description: String	Description of the guarantees that gives the supplier/consultant company of fulfillment of the dates of implementation
	2 Guarantees of Non-Stop Running of the Product	List Of Guarantee; Guarantee: Tuple(Name, Description); Name: String; Description: String	Description of the guarantees that gives the supplier company that the product will be running in a non-stop way
	3 Guarantees about Error Correction	List Of Guarantee; Guarantee: Tuple(Name, Description); Name: String; Description: String	Description of the guarantees that gives the supplier company that they will correct the errors found in the product
	4 Guarantees of Future Support	List Of Guarantee; Guarantee: Tuple(Name, Description); Name: String; Description: String	Description of the guarantees that gives the supplier company that they will give support to the client in the future with respect to the product
4	Licensing Costs		Description of the COTS components and total cost of ownership for the different licensing options available
	1 Per User Group	List Of Interval; Interval: Record(Rang, Cost); Rang: String; Cost: Float(dollars)	Estimate price for each group of users that have access to the product
	2 Per Server	List Of Interval; Interval: Record(Rang, Cost); Rang: String; Cost: Float(dollars)	Estimate price for each server in which the product is installed
	3 Per Client	Cost: Float (dollars)	Estimate cost for a client company
	4 Other Licensing Costs	List Of CostLicensingOptions; CostLicensingOptions: Tuple(LicensingType, Rang, Cost); LicensignType: String; Rang: Description; Cost: Float(dollars)	Estimate prices for other licensing options provided by the supplier company

Quality factor		Metric	Description	
5	Platform Costs		Estimation of the cost for the required production platform	
	1 Hardware Platforms		Estimates the cost of the required hardware platform	
	1	Server Characteristics	List Of ServerChar; ServerChar: Tuple(Characteristic, Value); Characteristic: String; Value: String	List of characteristics of the required servers
	2	Server Costs	Cost: Float(dollars)	Estimated cost of the required servers
	3	Maintenance Server Costs	Tuple (CostPerServer, NumberOfMonths); CostPerServer: Float (dollars); NumberOfMonths: Integer	Estimated recurrent cost for the maintenance of the required servers
	4	Other Hardware Characteristics Required	List Of OtherHardware; OtherHardware: Tuple(Description, Hard); Description: String; Hard:List Of HardChar; HardChar: Tuple(Characteristic, Value); Characteristic: String; Value: String	List of other hardware required and its characteristics
	5	Other Hardware Costs	List Of OtherHardwareCost; OtherHardwareCost: Tuple(Description, Cost); Description: String; Cost: Float(dollars)	Estimated cost for each other hardware required
	2 Software Platforms		Estimated cost of the required software platform	
	1	Operative System	OperativeSystem: String	Operative system required
	2	Operative System Cost	CostPerOperativeSystem: Float(dollars)	Estimated cost of the operative system required
	3	Other Software Required	List Of OtherSoftware; OtherSoftware: String	List of other software products required
	4	Other Software Cost	List Of OtherSoftwareCost; OtherSoftwareCost: Tuple(Other Software, Cost); OtherSoftware: String; Cost: Float(dollars)	Estimated cost for each other software required
	6 Implementation Costs		Estimation of the implementation cost based on similar past experiences	
	7 Network Costs		Estimation of additional cost for network operation	
3 Product			Characteristics of the commercial aspects of the software product that can influence its quality	
	1 History		Evolution of the COTS since it has been offered to the clients	

Quality factor		Metric	Description
1	Product in Market	Time: Years; Years: Integer	Time Of product in the market
2	Versions of the Product	List Of Version; Version: Tuple(NumberVersion, Time); NumberVersion: String; Time: Years; Years: Integer	Versions of the product currently in the market
3	Patches per Version	List Of VersionPatches; VersionPatches: Tuple(NumberVersion, Number) NumberVersion: String; Number: Integer	Number of patches for each version
4	Errors per version	List Of VersionErrors; VersionErrors: Tuple(NumberVersion, Number) NumberVersion: String; Number: Integer	Approximate number of errors identified in each version
5	Compatibility Among Versions	Scheme: (Tools for the migration, ...)	Compatibility schema among versions
6	Compatibility Guarantees	List Of CompatibilityGuarantees; CompatibilityGuarantees: Tuple(Type, Description) Type: String; Description: String	List of guarantees that assure the compatibility among versions
2	Deliverables		Detail of the out-of-the-box and expected post-implementation deliverables
1	Initial Deliverable	List Of Deliverable; Deliverable: (SourceCode, UserManuals, Installation Manuals, RunningCode, ...)	Contents of the first deliverable that the supplier company gives to the client
2	After Installation Deliverable	List Of DocumentationDeliverable; DocumentationDeliverable: (UMLdiagrams, ParameterizationManual, SourceCode, UserManuals, InstallationManual, RunningCode, ...)	Contents of the deliverable that the supplier company give to the client after the installation of the product
3	Parameterization/Customization		Description of the initial effort required for the product to operate
1	Working Team	List Of PossibleComposition; PossibleComposition: (ExternalConsultants, ExternalConsultantsWithExternalSupport, MixedTeam, InternalTeamWithExternalSupport, ...)	Structure of the working team that participates in the parameterization and customization of the product