AGREEMENT TECHNOLOGIES

cost Action IC0801

Why shall we do this?

# *Why* shall we do this?

## Organizational awareness as an approach to create dynamic, flexible and context-aware eBusiness applications (the CONTRACT and ALIVE projects)


Javier Vázquez-Salceda

SMA-UPC

Knowledge Engineering and Machine Learning Group

UNIVERSITAT POLITÈCNICA DE CATALUNYA

*https://kemlg.upc.edu*

# Contents

- Introduction
  - Problems in SOA for e-Business applications
- Distinguishing WHAT from HOW
  - Contract-Based Business Process Descriptions
  - Norms to describe (acceptable) behaviour
- Distinguising WHY from WHAT
  - Bring experience from human societies/organisations
  - Organisational modelling
- Conclusions and Future Challenges

Why shall we do this?

Why shall we do this?

# Introduction



Knowledge Engineering and Machine Learning Group
UNIVERSITAT POLITÈCNICA DE CATALUNYA
*https://kemlg.upc.edu*

Why shall we do this?

# Towards distributed business

- Now a days, computing trends move toward distributed solutions
  - computer systems are networked into large distributed systems;
- e-Business technologies are also moving from intra-organization or limited B2B into flexible, multiple inter-organization relations
  - The ability to seamlessly exchange information between companies, business units, customers, and partners is vital for the success of companies
  - Problem: most organizations employ a variety of applications that store/exchange data in dissimilar ways, and cannot “talk” to one another productively.
- It is expected that soon most e-Business applications will require dynamic integration of a large number of complex services.

# Current trend: Service Orientation

- Technical progress in the area of Service-Oriented Architectures (SOAs) has been based on many sources
  - enterprise interoperability, grid computing, software engineering, database and knowledge-base theory, artificial intelligence, object-oriented systems.
- Main areas of progress include:
  - interoperability (SOAP  WSDL  and OGSI );
  - discovery and management (UDDI  and WS-Management)
  - orchestration and choreography (WS-BPEL , XPDL , ebXML  and WS-CDL );
  - association of semantics with Web-services (OWL-S  and WSMO).
- These developments have raised the possibility of
  - deploying large numbers of services
  - in intranets and extranets of (private/public) organizations, and the public Internet,
- All these forms the baseline environment for software applications.

Why shall we do this?

# SOA, e-Business and the 'Future Internet'

- Visions of Service Oriented Business Environments are well established
  - Systems able to communicate and reconfigure at runtime
  - Systems able to adapt to their environment and identify new (business) opportunities
  - Systems able to dynamically combine sets of building block services into new applications

- huge challenges remain, in particular:
  - Greater scale and openness *conflict with* standard assumptions about the behaviour of actors in the world
  - Increased Autonomy / Flexibility *conflict with* our ability to ensure predictable execution
  - Dynamic discovery / late binding *conflict with* the need for Sound Legal Guarantees

- Is current SOA technology prepared for these challenges?

# Problem 1: Services without memory

- One important limitation in (most) current implementations of SOA comes from their initial focus on interoperability requirements, and especially the principle of **stateless services**
  - services as stateless components offering very simple functionalities that composed may bring complex computation.
  - All the required information to operate goes in the invoking message
- Although this *stateless approach* eases interoperability, it makes it difficult (if not impossible) to have services that can dynamically detect and adapt their behavior to contextual changes or opportunities.
- Some patches have been made to have **statefull services**, but the SOA framework has not been adapted properly to manage application states.

# Problem 2: Where is my organisation?

- Existing technologies for the web mostly ignore organizational aspects of the application domain:
  - They provide designs of low abstraction level, based on
    - (static) descriptions of tasks,
    - or even, the actual (remote) method invocations
  - They loose track of the underlying aims and objectives that motivate the interaction among the different peers.

- Current web technologies are not organization-oriented but rather task- or method-centric.

- Some researchers treat *workflows* as *'business logic'*, but these are really static models that give no room for adaptation.
  - Every single exception must be foreseen for the whole distributed system to operate without errors.

# Problem 3: Where is my context?

- Another important limitation of both Web service and Semantic Web service technologies is that they do not fully cover one of the identified requirements to support both the Web 2.0 and the Future Internet: ***context-awareness***.

- If services are to behave flexibly in dynamic, changing environments they should be aware of their context in order to
  - identify new opportunities,
  - detect relevant changes
  - adapt their internal behavior and/or the way they interact with others.

- In many cases correct, adaptive behavior is (arguably) nearly impossible to guarantee without effective information about context.

# Context in SOA

## Business Process Descriptions

- In order to bring context into a distributed service computation, current approaches are often based on the use of **(static) business process models** as a basic mechanism to support service composition.
- A **business process** specifies, among others:
  - the potential execution order of operations from a collection of Web services,
  - the data shared between these Web services,
  - which partners are involved and how they are involved in the business process,
  - joint exception handling for collections of Web services.
- There are competing initiatives for developing business process definition specifications which aim to define Web services composition: ***orchestration*** and ***choreography***.

# Context in SOA
## Orchestration

- **Orchestration** defines the workflow between services from the "perspective of a single party", specifying the sequence and conditions in which one Web service invokes other Web services.
- Orchestration describes how services can interact at the message level, including the business logic and execution order of the interactions.
- *Standard-de-facto*: **Business Process Execution Language (BPEL)**
  - a layer on top of the Web services Description Language (WSDL)
  - BPEL defining how the operations can be sequenced to support business transactions
- Problem: BPEL specifications only indicate the orderings of different tasks in a centralized and rigid way.

# Context in SOA
## Choreography

- **Choreography** is described from the perspective of all parties (common view) and defines the complementary observable behavior between participants in a business process collaboration.
  - A common view defines the shared state of the interactions between business entities
  - It can be used to determine specific deployment implementation for each individual entity.
  - The choreography tracks the sequence of messages that may involve multiple parties/multiple sources, and each party describes the part they play in the interaction.
- Main approach: **Web Service Choreography Description language (WS-CDL)**, specifies collaboration in terms of *roles* and *work units*
  - A **role** enumerates the observable behavior a party exhibits to collaborate with others
  - **Work units** consist of *activities* (incl. interaction activities) and *ordering structures*

Why shall we do this?

# Problems of Orchestration/Choreography in SOA

- The limitations in current orchestration and choreography approaches to capture context are:
  - They tend to be
    - static,
    - prone to failure (the failure of one service in the chain typically makes the workflow to fail)
    - very difficult to design and debug (the designer needs to foresee all possible execution paths and specify them in the workflow).
  - They model systems at a single level of granularity
    - services offered by individuals, corporations, multinationals, or departments within companies are modeled all with the same abstractions and with the same granularity.
  - They tend to have very little knowledge about the interaction context.
    - For instance, they lack explicit knowledge of regulations in the environment.

# Problems of Orchestration/Choreography in e-Business applications

- There are some additional issues to solve when trying to model and build e-Business applications:
  - How to manage workflows in non-trivial open environments, where not all services are owned by the same organization?
    - we cannot assume that all parties are either benevolent or that they will deliver results unless explicit obligations are defined and enforced.
    - should workflows be agreed upon by all parties before they can be executed?
  - What if critical applications simply cease to function if services provisioned from third parties disappear or malfunction?
  - If e-Business applications (and their business processes) are meant to change/adapt/evolve through time, how are such applications going to be:
    - Designed (without foreseeing all possible interactions),
    - Deployed (with dynamic composition in mind)
    - Managed (with change/adaptation/evolution being natural).

- Therefore, this is not a good approach to tackle new generations of service technologies, able to dynamically adapt and reconfigure in an ever-changing environment.

# Steps towards future e-Business

- Idea: to use advances in Artificial Intelligence, Institutional and Organisational theories to create the next generation of Business technologies

- In my view, first two steps:
  - Provide more flexible ways to specify business interactions, abstracting away from the low-level details.
    - **Distinction between *WHAT* to do and *HOW* to do it**
  - Add ways to better describe context and the interaction between the system activity and the context changes.
    - Our approach: to add motivational drives to these business systems, so they can reconsider their actions and identify new opportunities if context changes
    - **Distinction between *WHY* do things and *WHAT* to do**


# Step 1: Distinguishing between what to do and how.

*(Business interactions guided by high-level contractual specifications)*

Knowledge Engineering and Machine Learning Group
UNIVERSITAT POLITÈCNICA DE CATALUNYA
*https://kemlg.upc.edu*

# SLA's as a (Business) Process Description

- In SOA, there exist more powerful mechanisms to describe processes than workflow-oriented technologies.
- One trend: **Service-Level Agreements (SLA's)**
  - They represent (contractual) agreements between service providers and consumers
  - They may specify the levels of availability, serviceability, performance, operation, or other attributes of the service.
  - Typically encompass
    - the *SLA contract definition* (basic schema with the QoS (quality of service) parameters),
    - *SLA negotiation*,
    - *SLA monitoring*,
    - *SLA enforcement* (according to defined policies).

# Existing contracting/agreement approaches (I)

- **WS-Agreement**
  - Agreements and templates, agreement lifecycle processes
  - No third parties, no multiparty contracts, penalties miss ‘finalizing process’ and do not allow extension, no formal semantics, lack of expresiveness for full contracts
- **Web service Level Agreement (WSLA)**
  - Service-Level agreements and objectives, third parties, extensible language
  - No agreement handling mechanisms, no formal semantics, no notion of interaction context
- **Web services Conversation Language (WSCL)**
  - Used in electronic commerce to agree on how services will communicate
  - Only covers message structure and protocols for execution, no definition of what to do if something goes wrong

Why shall we do this?

# Existing contracting/agreement approaches (II)

- **Rule-Based Service Level Agreement (RBSLA)**
  - Logic-based, includes the use of deontic notions
  - No support from industry (PhD), limited semantics based on events, actions and goals, lack of expressiveness for full contracts
- **OASIS eContracts**
  - Computational representation of human contracts, very expressive
  - Too complex for computational monitoring and verification

- We need more flexible ways to specify the expected behaviour in multi-party business setups, including the expectations of the different parties.
  - Obligations, prohibitions…
- There is also a need for mechanisms that ease the engineering of applications in Cross Organisational Service Oriented Computing environments

Why shall we do this?

# Contract-based SOA Governance

- *Contracts* are the explicit, tangible **representation of service interdependencies**

- Contract-based approaches promise two clear med/long term benefits in Service Oriented Business environments:
  - Closer linkage between technical implementation and responsibilities / obligations
  - Abstraction away from internal execution details in order to support formal verification of distributed enterprise systems

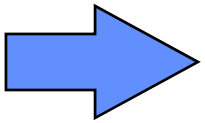
- **Idea: formal verification over *contracts*, *obligations etc.*** rather than over internal code is the way to build sound distributed applications in service oriented environments.

# How to express contractual obligations?

- **Norms** are a flexible way to specify the boundaries of acceptable (legal) behaviour
  - They specify WHAT is acceptable and WHAT is not, but not HOW
  - Agents have autonomy to reach their goals as far as they "move" within the acceptable boundaries.
- Norms ease agent interaction:
  - reduce uncertainty of other agents' behaviour
  - reduce misunderstanding in interaction
  - allows agents to foresee the outcome of an interaction
  - simplify the decision-making (reduce the possible actions)
- To ensure acceptable behaviour, a safe environment is needed: **Electronic Institutions**
  - Safe agent interaction environments
  - They include definition of norms and enforcement mechanisms

# But, how to connect agent abstractions with services?

- *Service Oriented Architectures framework*
  - Broad definition of service as component that takes some inputs and produces some outputs.
  - Services are brought together to solve a given problem typically via a workflow definition that specifies their composition.
- Every application is made up of *actors*
- Every change that happens is an action by an actor
- Actors communicate by sending *messages*
- Every action is triggered by a message
- The outputs of (messages sent by) an actor are *caused* by the inputs to (messages received by) the actor



*Direct mapping to multiagent systems*

Why shall we do this?

# Idea: Intelligent Contractual Environments

- **Contracts**:
  - Make explicit the obligations of each of the parties in the transactions
  - Make explicit what each system can expect from another
- Bind together:
  - The electronic interaction (web services) with
  - The business obligation with
  - Prediction as to whether the system will function to get the job done
- A **contract instantiation** creates a contracting environment
  - Monitors contractual clauses (Deontic statements → **norms!**)
  - This is, in fact, an **electronic institution!**

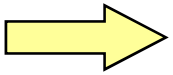
# Norm Representation (I)

- Formal representation of norms needed
- Which logic?
  - Norms permit, oblige or prohibit
  - Norms may be conditional
  - Norms may have temporal aspe
  - Norms are relativized to roles

**OBLIGED, PERMITTED, FORBIDDEN**

**IF** $C$

**BEFORE** $D$**, AFTER** $D$



variant of Deontic Logic

- The representation should be easily parseable and usable by agents

# Norm Representation (II)

- *Unconditional norms about predicates*
  - the norms on the value of *P* are active at all times:

$$\text{OBLIGED}(a, P) \qquad \text{PERMITTED}(a, P) \qquad \text{FORBIDDEN}(a, P)$$

- *Unconditional norms about actions*
  - the norms on the execution of A are active at all times:

$$\text{PERMITTED}(a \text{ DO } A) \qquad \text{FORBIDDEN}(a \text{ DO } A)$$

- *Conditional norms*
  - the activation of the norms is conditional under *C*
  - *C* may be a predicate about the system or the state of an action:

$$\text{OBLIGED}((a, P) \text{ IF } C) \qquad \text{OBLIGED}((a \text{ DO } A) \text{ IF } C)$$
$$\text{PERMITTED}((a, P) \text{ IF } C) \qquad \text{PERMITTED}((a \text{ DO } A) \text{ IF } C)$$
$$\text{FORBIDDEN}((a, P) \text{ IF } C) \qquad \text{FORBIDDEN}((a \text{ DO } A) \text{ IF } C)$$

Why shall we do this?

# Norm Representation (III)

- *Conditional norms with Deadlines*
  - the activation of norms is defined by a deadline

$$\text{OBLIGED}((a, P) \text{ BEFORE } D)$$
$$\text{PERMITTED}((a \text{ DO } A) \text{ AFTER } D)$$
$$\text{FORBIDDEN}((a, P) \text{ BEFORE } D)$$

  - absolute and relative deadlines:

$$23{:}59{:}00\ 09/05/2004$$
$$time(done(assign(organ, recipient))) + 5min$$

- Examples:

$$\text{OBLIGED}((allocator \text{ DO } assign(heart, recipient))$$
$$\text{BEFORE } (time(done(extraction(heart, donor))) + 6hours))$$

$$\text{FORBIDDEN}((allocator \text{ DO } assign(organ, recipient))$$
$$\text{IF NOT}(hospital \text{ DONE } ensure\_quality(organ)))$$

# Norm Representation
## Abstraction problem

- Problems:
  - Norms are more abstract than the procedures (in purpose)
  - Deontic expressions do not have operational semantics

***Example:***

*Regulation: "It is forbidden to discriminate potential recipients of an organ based on their age (race, religion,...)"*

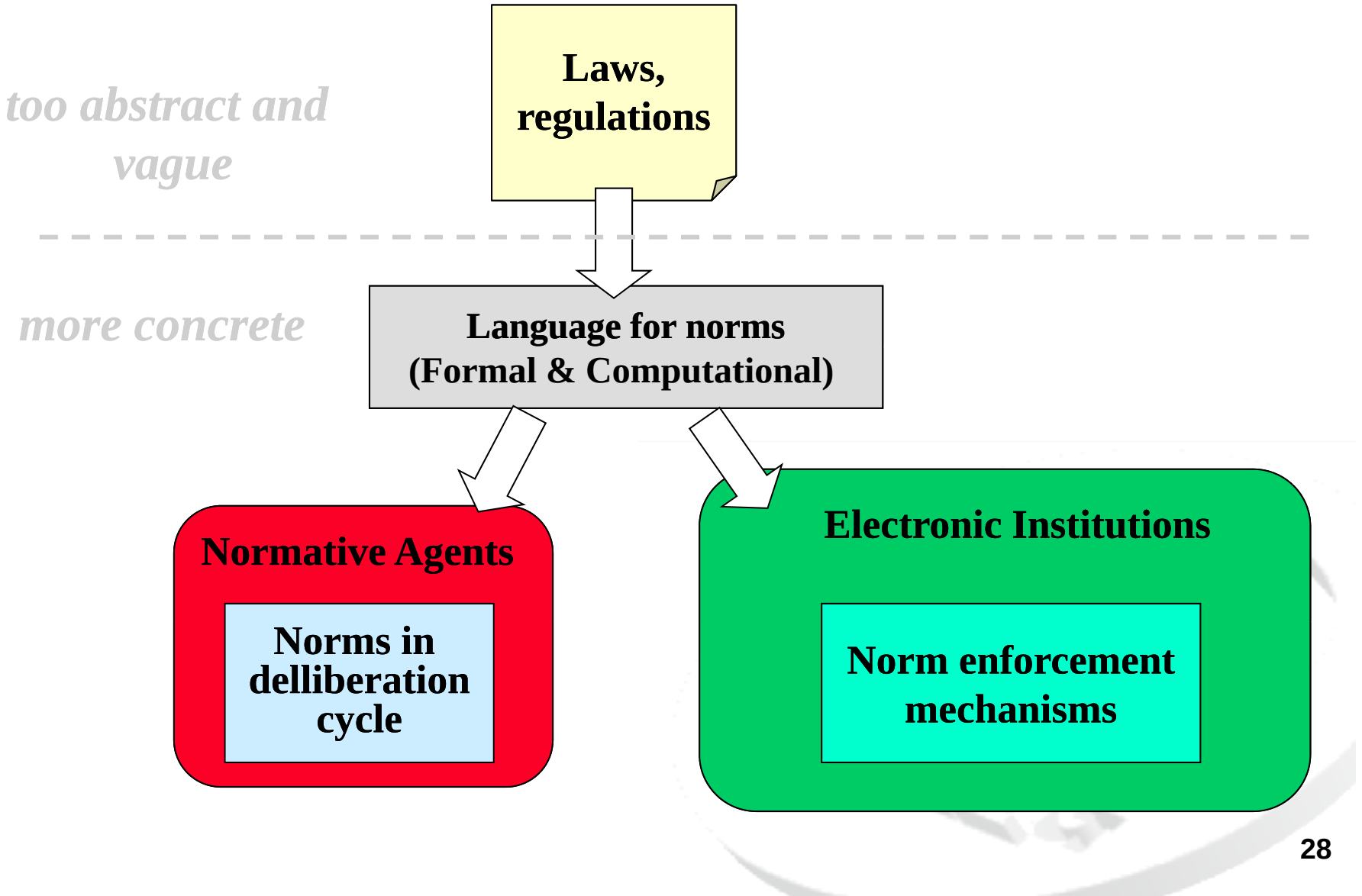
*Formal norm: F(discriminate(x,y,age))*

*Procedure: does not contain action "discriminate"*

Why shall we do this?

# Norm Representation

## Filling the gap



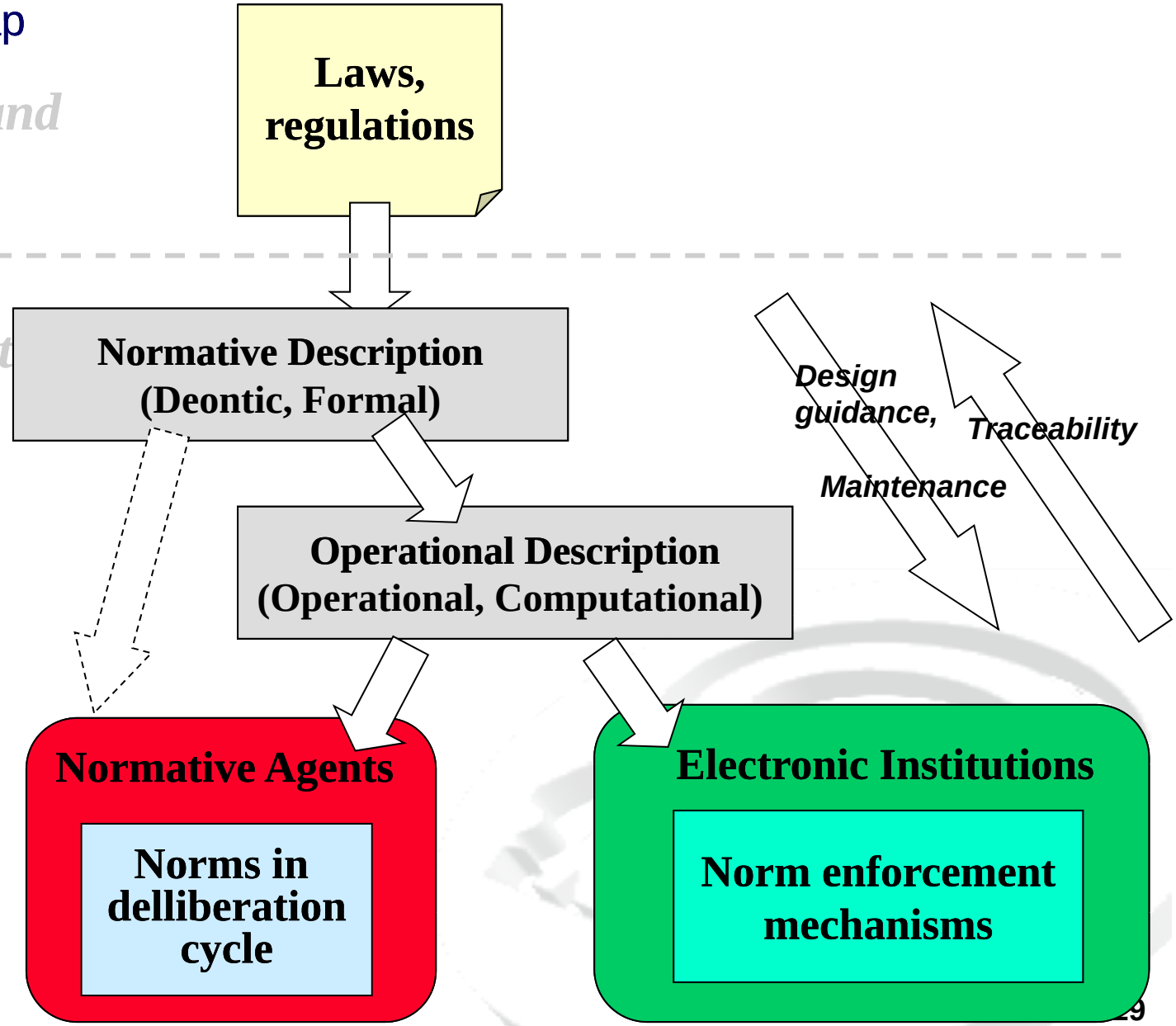

Why shall we do this?

# Norm Representation

## Filling the gap

*too abstract and vague*

*more concret*

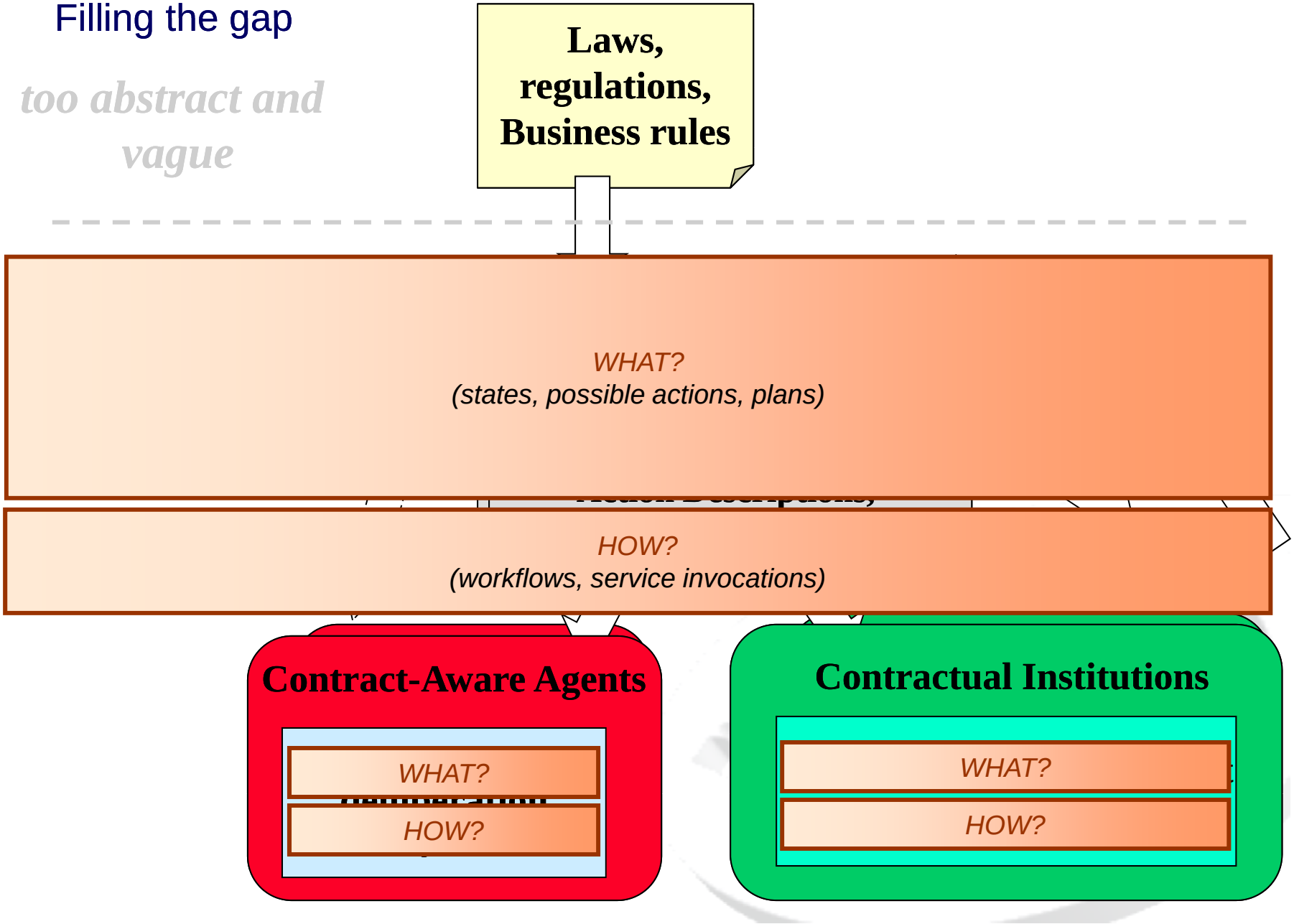



Why shall we do this?

# Contract Representation

## Filling the gap

*too abstract and vague*




Why shall we do this?

# Bringing flexibility to contractual interactions

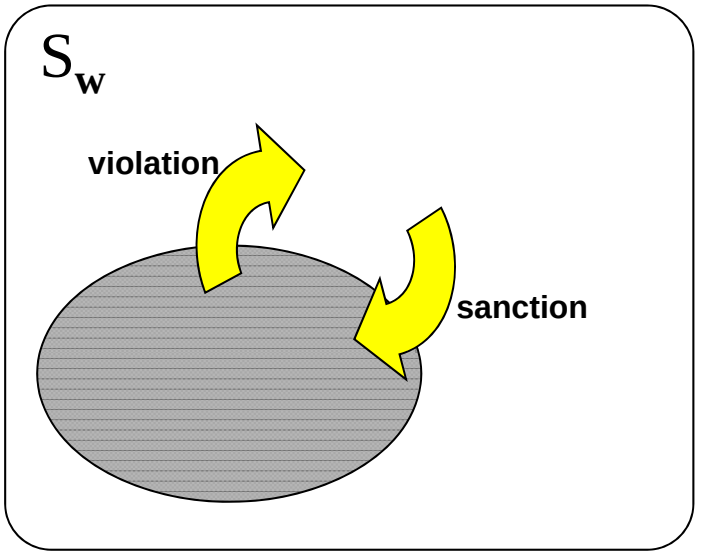
## Norm enforcement and violations

- Implementation of a safe environment (***norm enforcement***)
- 2 options depending on control over agents
  - Defining constraints on unwanted behaviour
  - Defining violations and reacting to these violations
- our assumptions:
  - Norms can be sometimes violated by agents
  - The internal state of agents is neither observable nor controlable
    - actions cannot be imposed on an agent´s intentions
    - agents as black boxes
    - only their observable behaviour and actions

Why shall we do this?

# Bringing flexibility to contractual interactions

## Boundaries for Safety and Soundness




- In our view Norms define the boundaries for acceptable behaviour
  - wanted (legal) and unwanted (illegal) behaviour
  - acceptable (safe) and unnacceptable (unsafe) states

- ***Violations*** when agents breaks one or more norms, entering in an illegal (unsafe) state.

- ***Sanctions*** are actions to make agents become legal (safe) again.

- Sanctions include the actions to recover the system from a violation

Safety

Soundness

Why shall we do this?

# Bringing flexibility to contractual interactions
## Landmarks

- Problem: if we ennumerate all the states of the system, divide them in *acceptable* and *unaceptable*, and define an ordering…
  - We will have something as expressive and fragile as a WS workflow!
  - We have again the problem of foreseeing all states!

- Idea: not define acceptable behaviour at the level of system states, but at the level of landmarks
  - *Landmarks* as meaningful (i.e. important) states in the system
  - *Landmark patterns*: partial accessibility relations from landmark to landmark

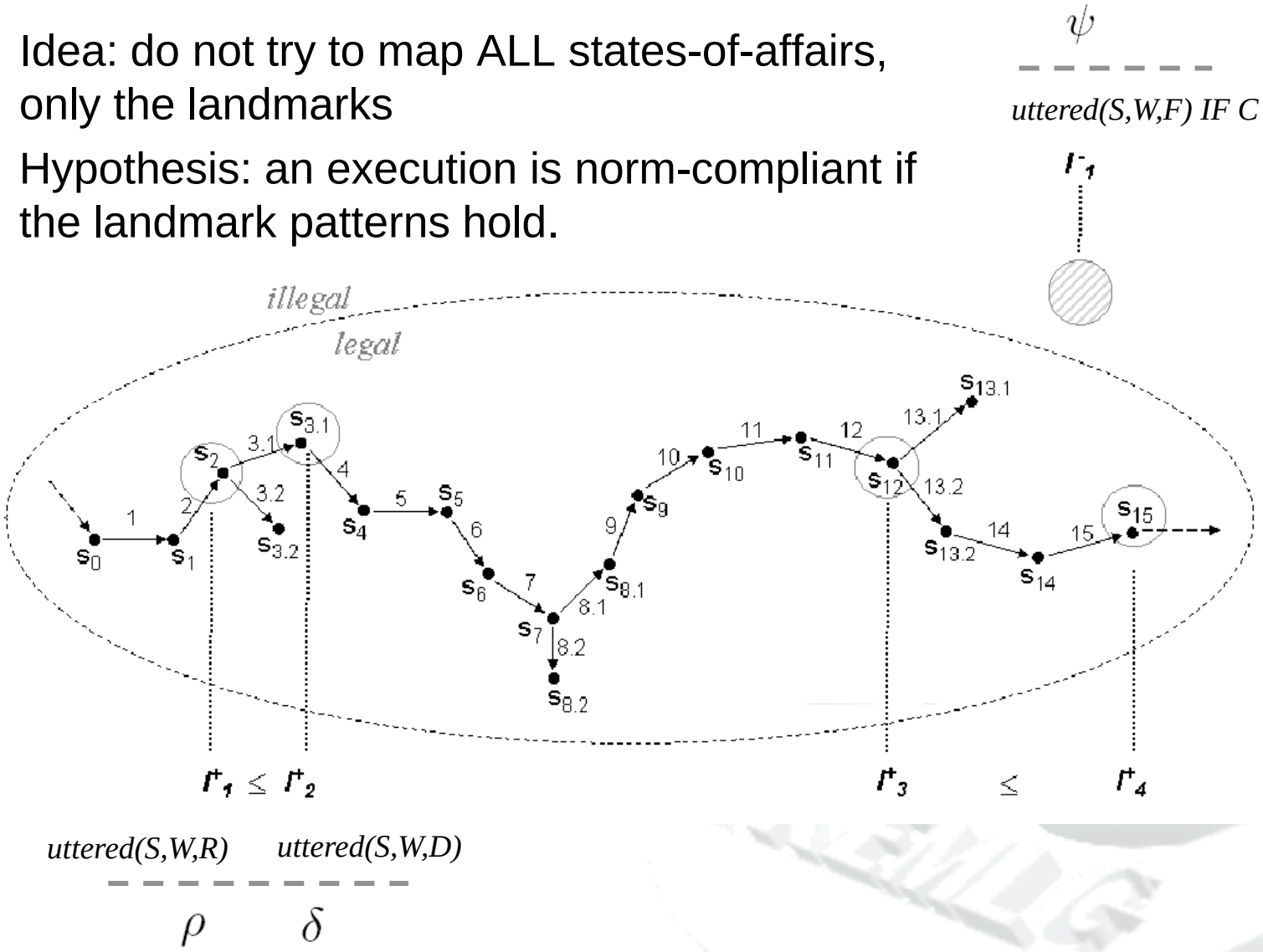
- Contracts usually define only those important states, and what should/should never happen among them
  - We can define landmarks in the normative level in terms of acceptable/unacceptable states of affairs
  - We can define landmarks in the operational level as states in the state machine

Why shall we do this?

# Bringing flexibility to contractual interactions

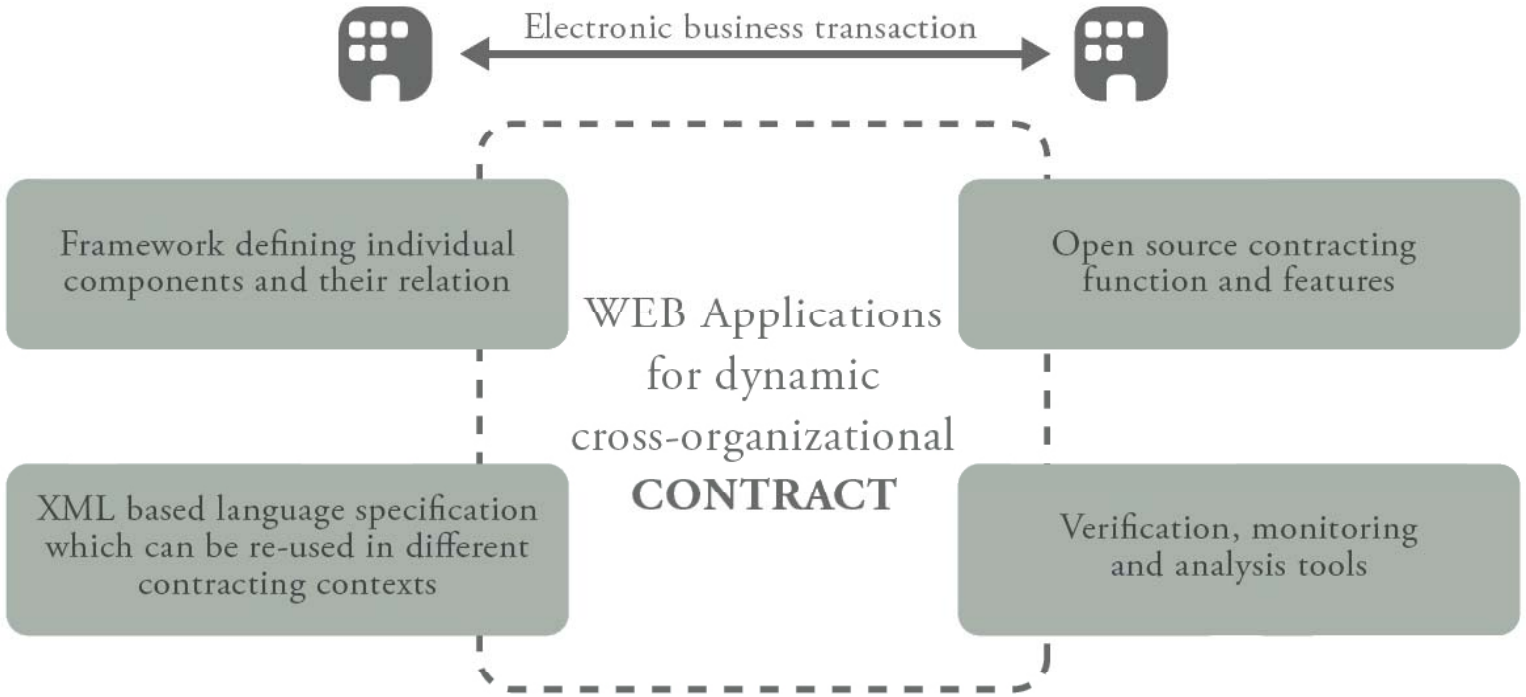
## Landmark Patterns and monitorization

- Idea: do not try to map ALL states-of-affairs, only the landmarks
- Hypothesis: an execution is norm-compliant if the landmark patterns hold.

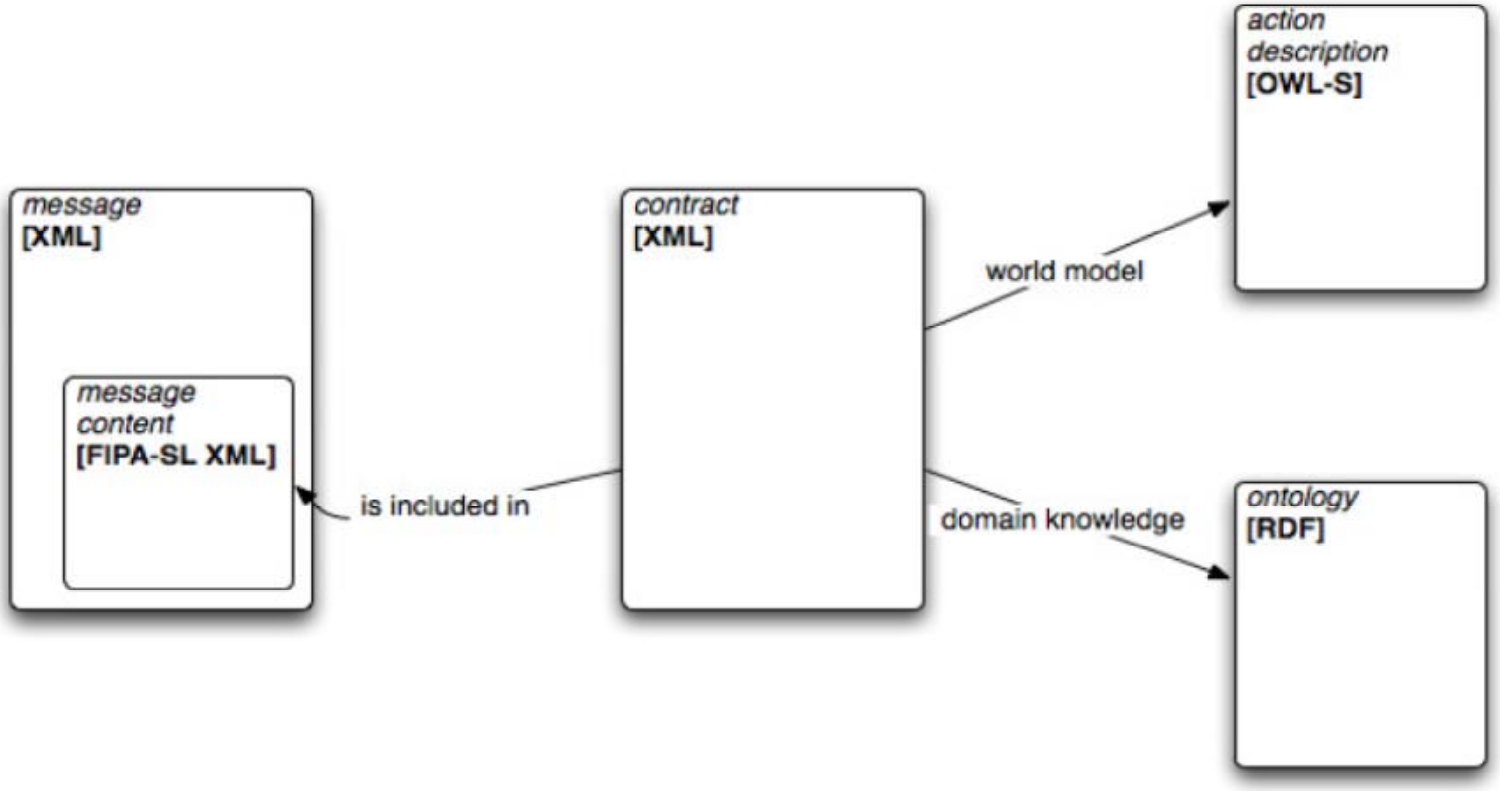



Why shall we do this?

# The IST CONTRACT Project results

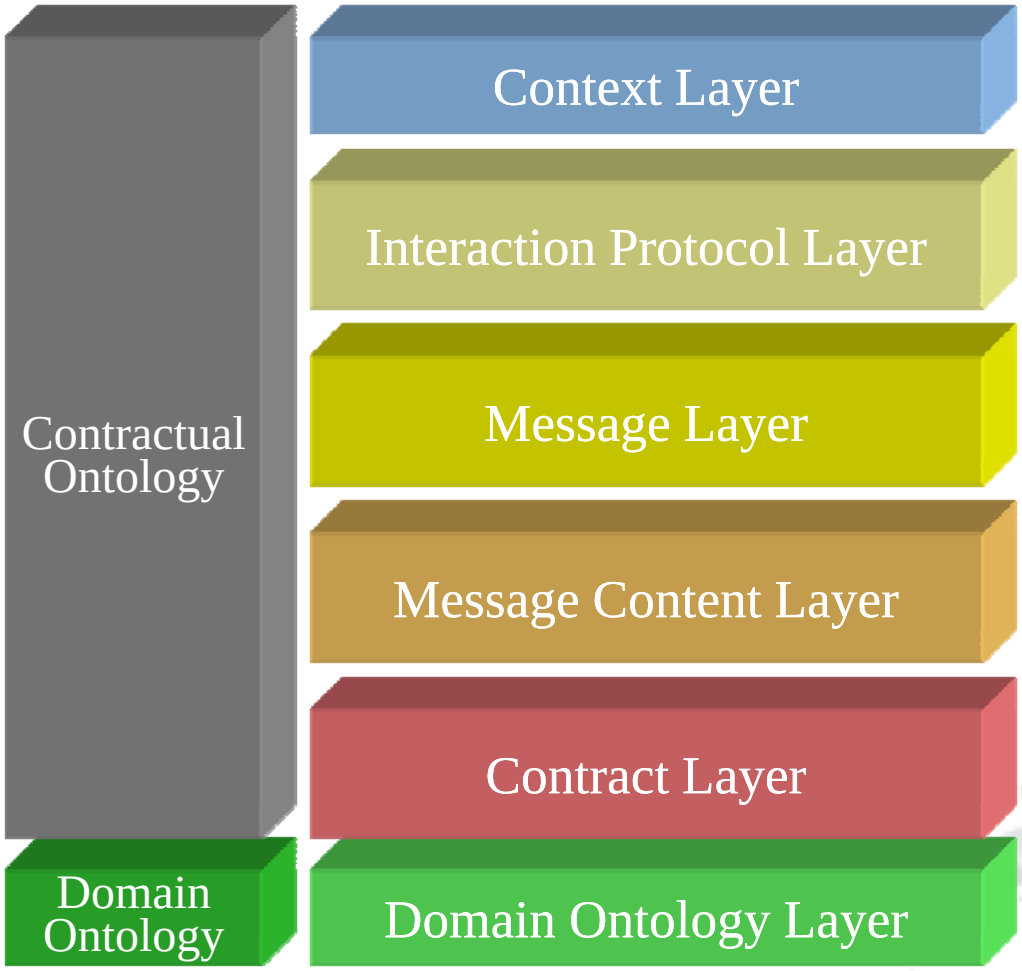


# Contracting language(I)
## Contract Language components

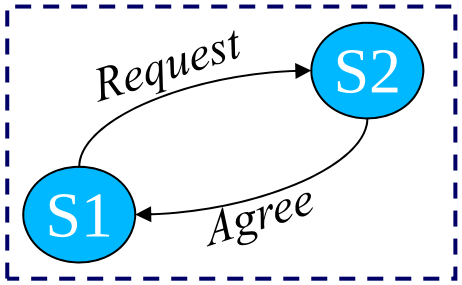



Why shall we do this?

# Contracting language(II)
## Communication Model




**Interaction context**:




**Protocol handling**:

**Message envelope + intentionality**:
*from service S1 to service S2 …*
*Request*[cancel(contract C1)]

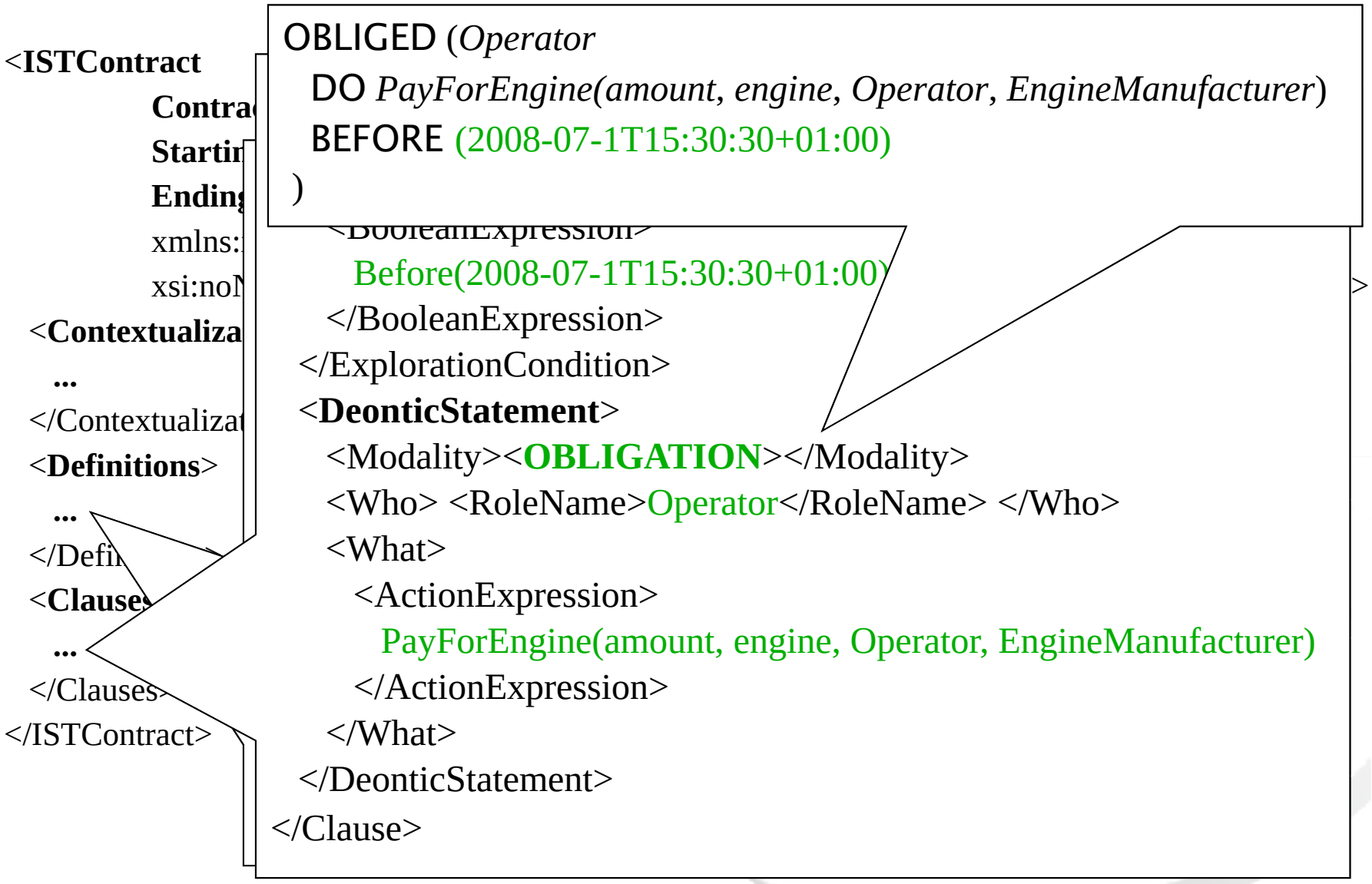
**Statements / actions related to contracts**:
*cancel*(contract C1)

**A contract**:
"the workshop is *obliged to* repair the car in 2 days"

**Domain terms**: *car, workshop, repair*

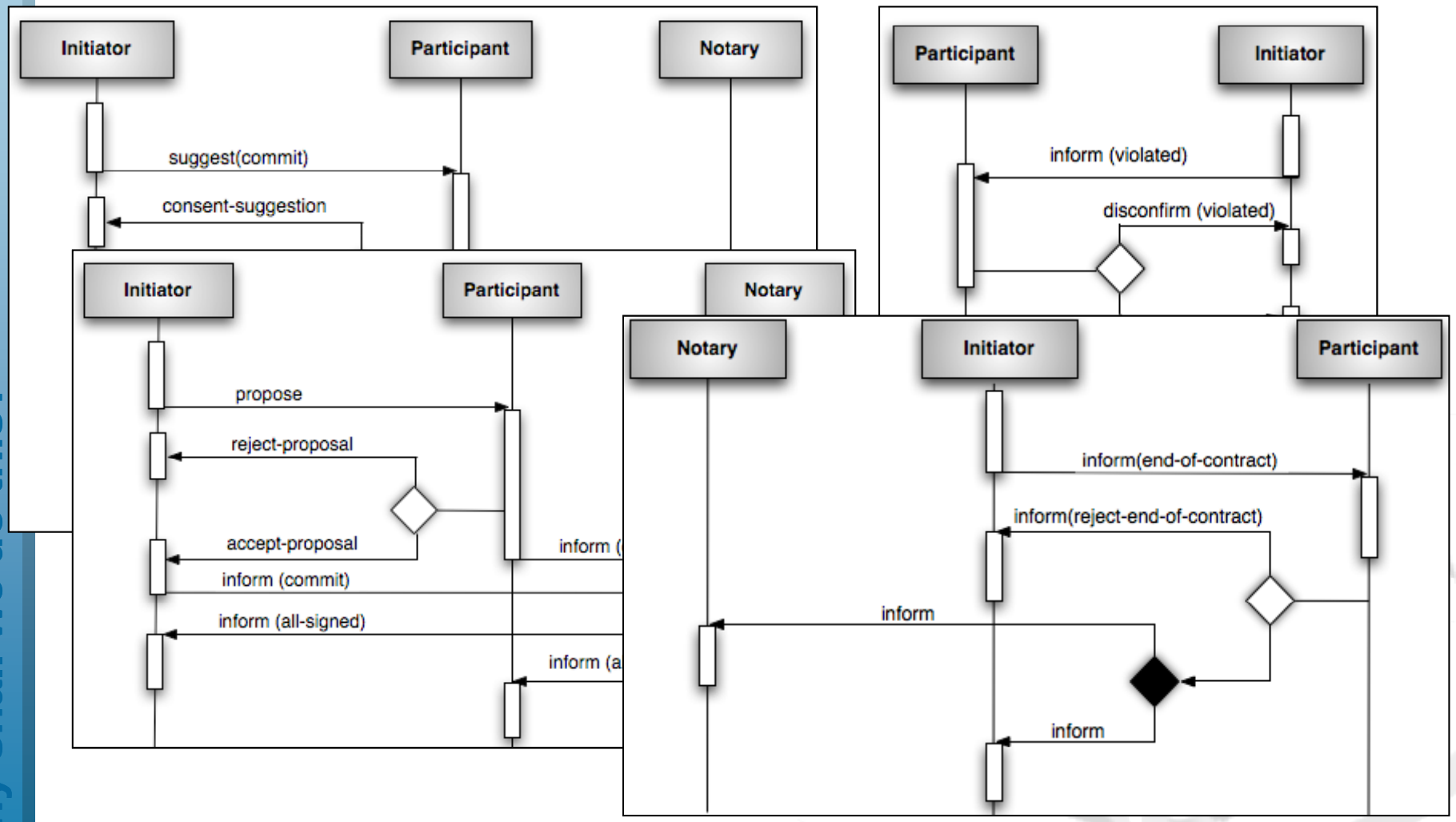
Why shall we do this?

# Contracting language (III)
## Contract expressions




Why shall we do this?

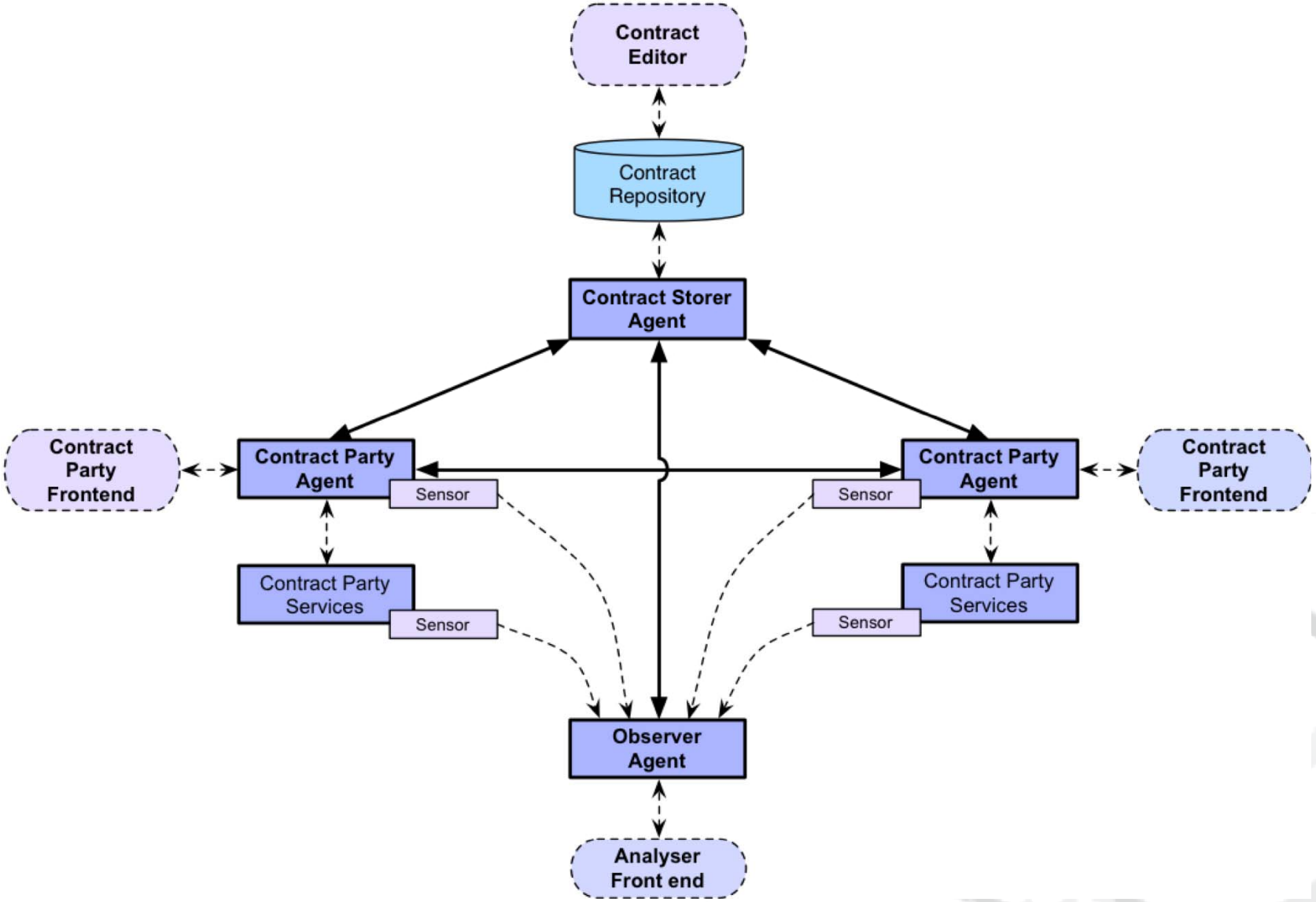
# Contracting language (IV)
## Predefined protocols



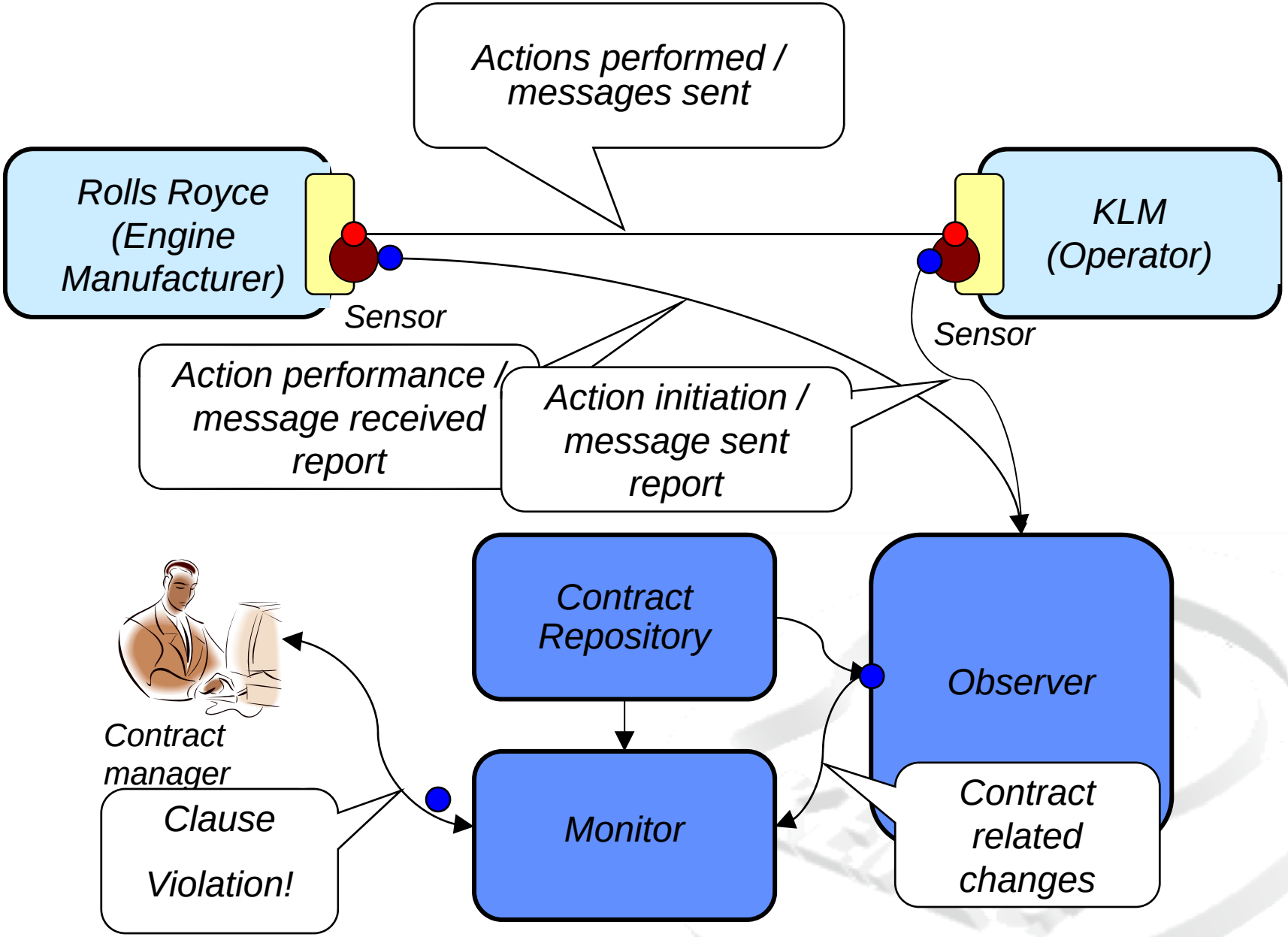

Why shall we do this?

Why shall we do this?

# Contracting Architecture



# Contract Monitorization



Why shall we do this?

# Novel features

- **Contracting Language** based in Normative Systems research
  - Includes semantic-rich service-to-service interaction, based on intentions and commitments
  - This allows the definition of formal semantics → ease verification
- Language covers all levels of communication
  - Not only centered in the expression of electronic contracts
  - A language to express statements about contracts
  - Protocols for contract handling
  - Includes connection with domain (context) models and ontologies

# …But we need more!

- **CONTRACT** has created concrete methods and tools which enable the use of contracts, obligations and agreements in order to structure the design and execution of sound applications in Digital Business environments

- But this is not enough:
  - not clear ***WHY*** to do things (other than to fulfill the terms of the contract)
  - Cannot adapt to changes in the environment

# Step 2: Distinguishing between why do things and what to do

*(Context-awareness through Organisational-awareness)*

Knowledge Engineering and Machine Learning Group
UNIVERSITAT POLITÈCNICA DE CATALUNYA
*https://kemlg.upc.edu*

Why shall we do this?

# The problem: Engineering flexible, adaptive Service Oriented applications for the Future Internet

- New generations of networked service applications should be able to:
  - *communicate and reconfigure* at runtime
  - *adapt* to their environment
  - *dynamically combine* sets of building block services into new applications

- This requires profound changes in the way software systems are designed, deployed and managed…
  - from existing, top-down, “design in isolation”...
  - ... to new approaches based on integrating new functionalities/behaviours into existing running systems

# Idea: bring experience from human societies/organisations

**The mechanisms used today to organise the vastly complex interdependencies found in human, social, economic behaviour will be essential to structuring future distributed software systems**

- Such mechanisms provide
  - *Robust descriptions* of distributed systems
  - Account for the *individual autonomous nature* of service providers/consumers
  - Define a wide range on strategies and mechanisms with *known properties*

Why shall we do this?

Why shall we do this?

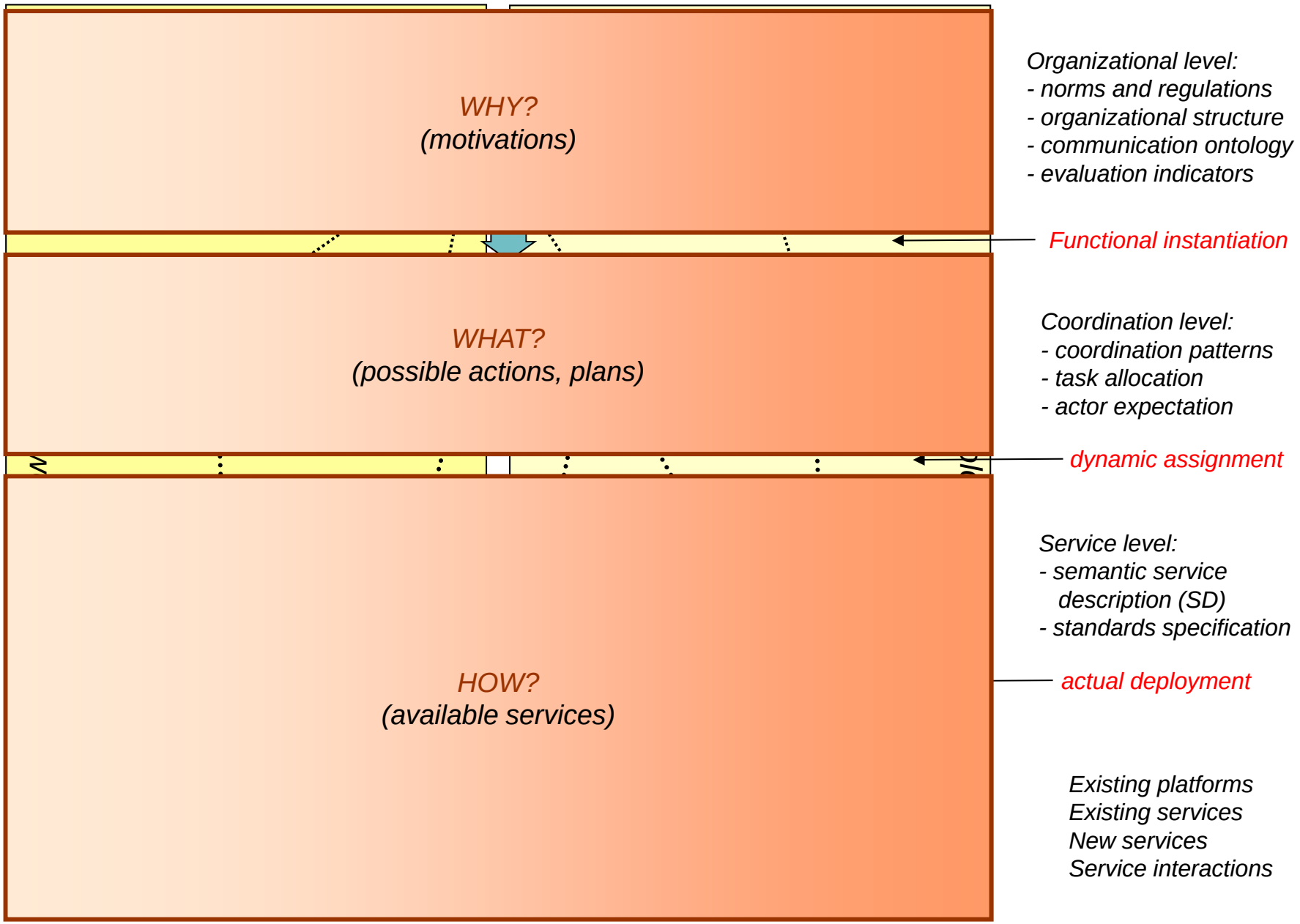
# The ALIVE Approach

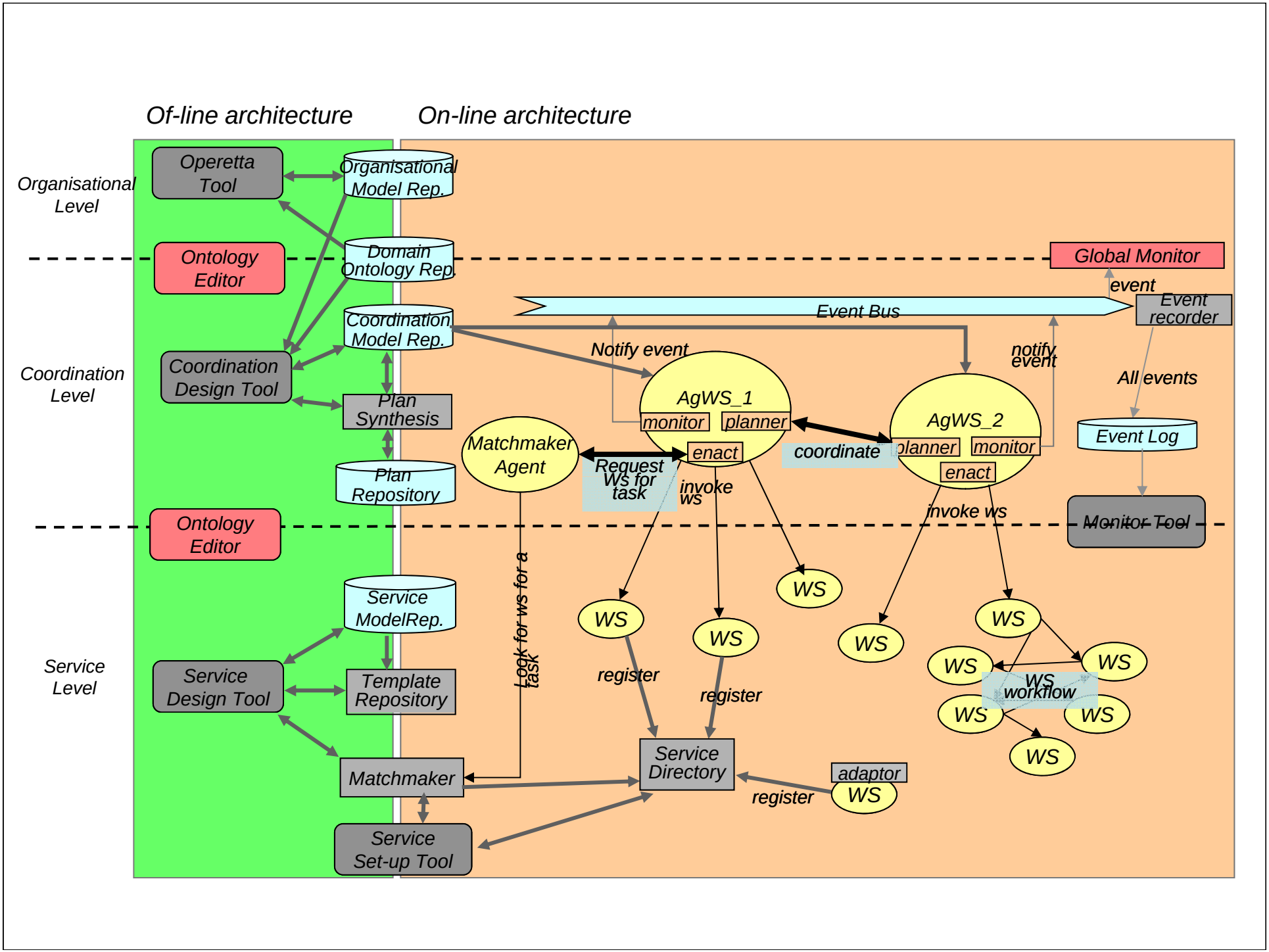
- To bring together the leading edge methods from *Coordination Technology*, *Organizational theory* with new technologies on *Model Driven* design to create a **framework for software and services engineering** addressing the new reality of “live”, open systems of active services.

- To *close the gap* between theoretical approaches and existing web services technologies

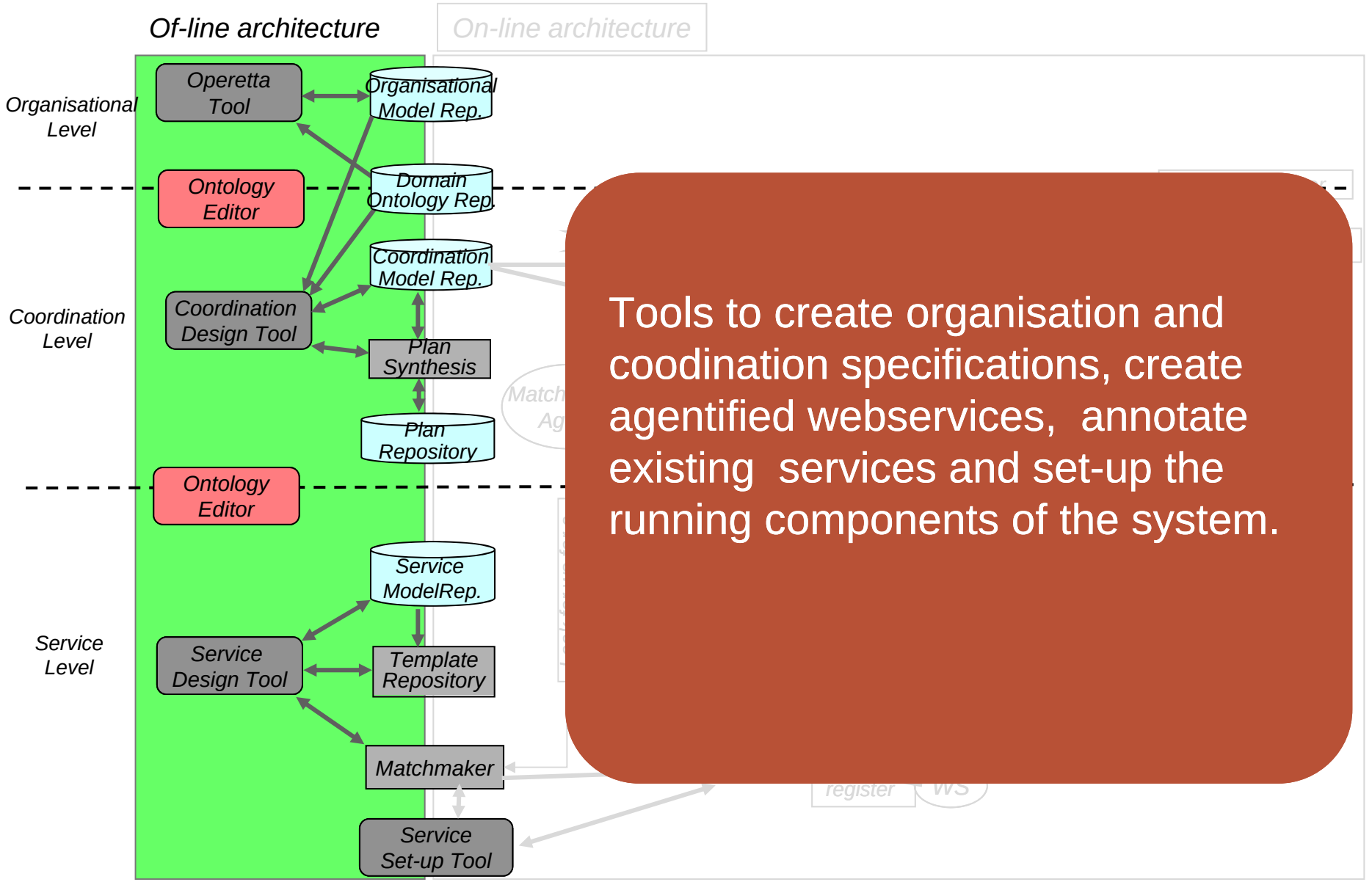
# The ALIVE Approach

- Splitting the design process in three separate layers
  - **Service layer**
    - augments service models to make components aware of their social context
  - **Coordination layer**
    - specifying patterns of interaction
  - **Organisational layer**
    - specifying organisational rules that govern interaction


WHY?
(motivations)
WHAT?
(possible actions, plans)
HOW?
(available services)
Organizational level:
- norms and regulations
- organizational structure
- communication ontology
- evaluation indicators
Functional instantiation
Coordination level:
- coordination patterns
- task allocation
- actor expectation
dynamic assignment
Service level:
- semantic service
description (SD)
- standards specification
actual deployment
Existing platforms
Existing services
New services
Service interactions

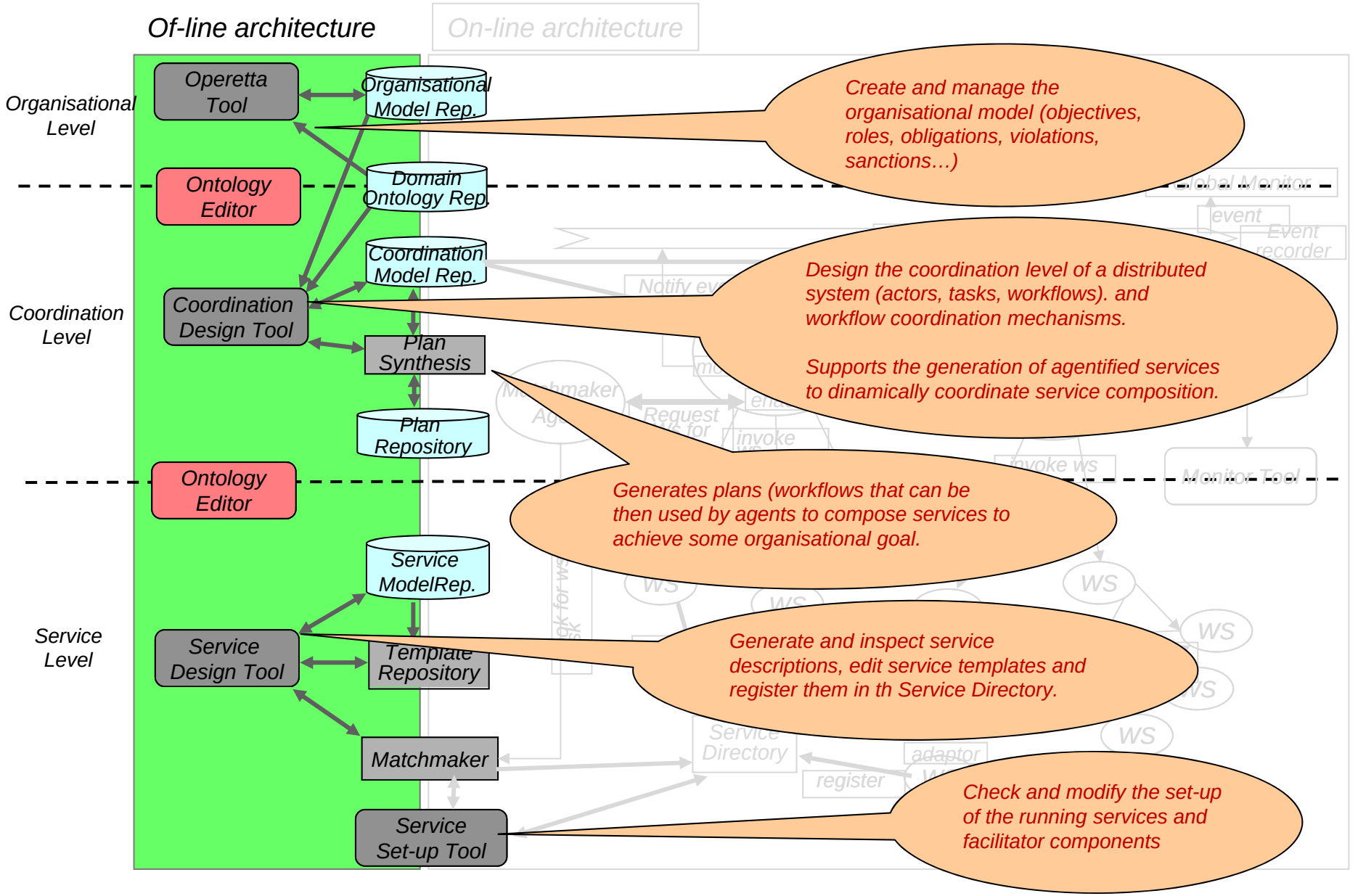

Of-line architecture
On-line architecture
Organisational Level
Coordination Level
Service Level
Operetta Tool
Organisational Model Rep.
Ontology Editor
Domain Ontology Rep.
Coordination Model Rep.
Coordination Design Tool
Plan Synthesis
Plan Repository
Ontology Editor
Service ModelRep.
Service Design Tool
Template Repository
Matchmaker
Service Set-up Tool
Global Monitor
event
Event recorder
Event Bus
Notify event
notify event
All events
AgWS_1
monitor
planner
enact
AgWS_2
planner
monitor
enact
coordinate
Event Log
Matchmaker Agent
Request Ws for task
invoke ws
invoke ws
Monitor Tool
look for ws for a task
WS
register
register
Service Directory
adaptor
register
workflow

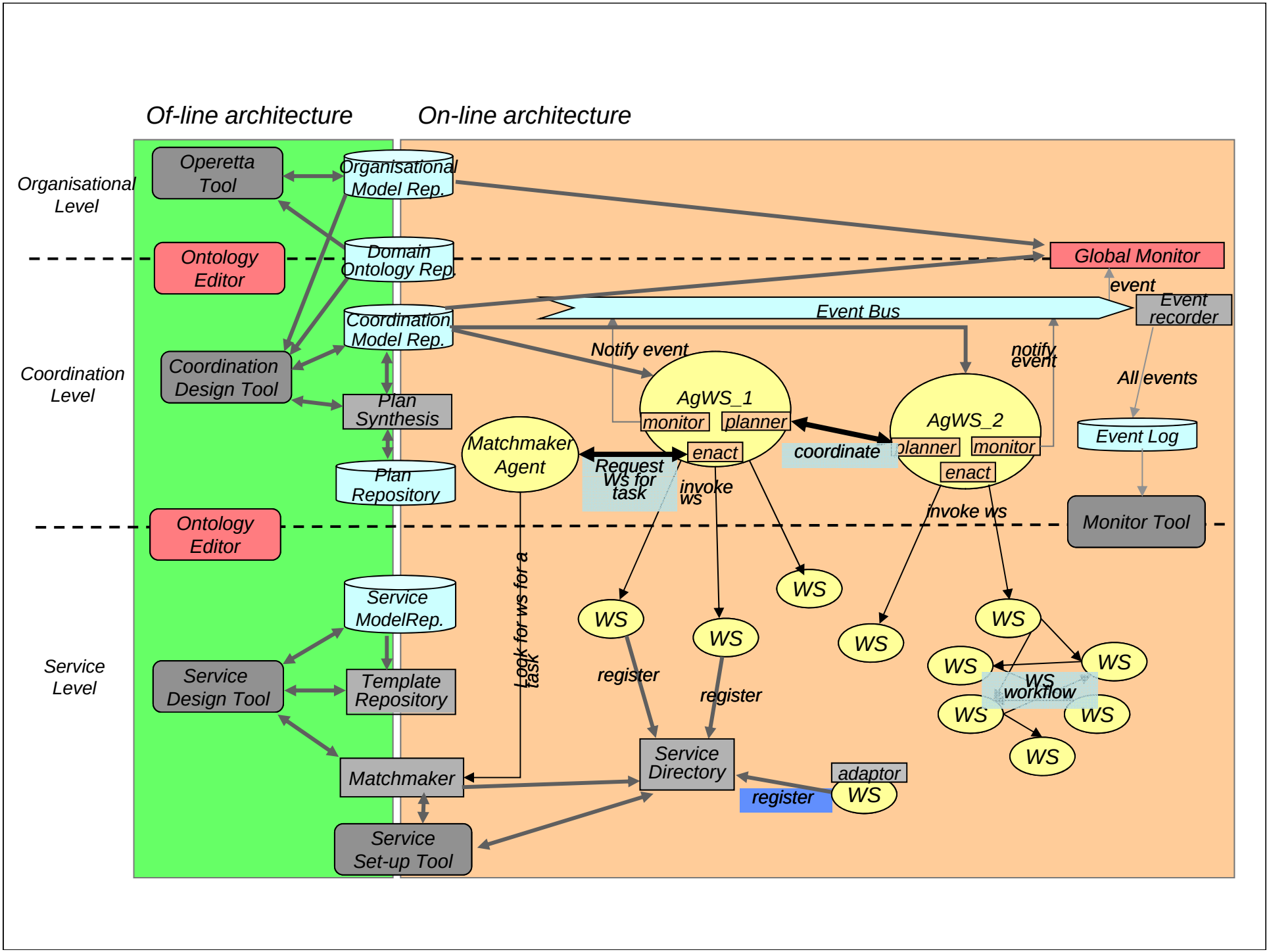
# ALIVE Off-line Architecture



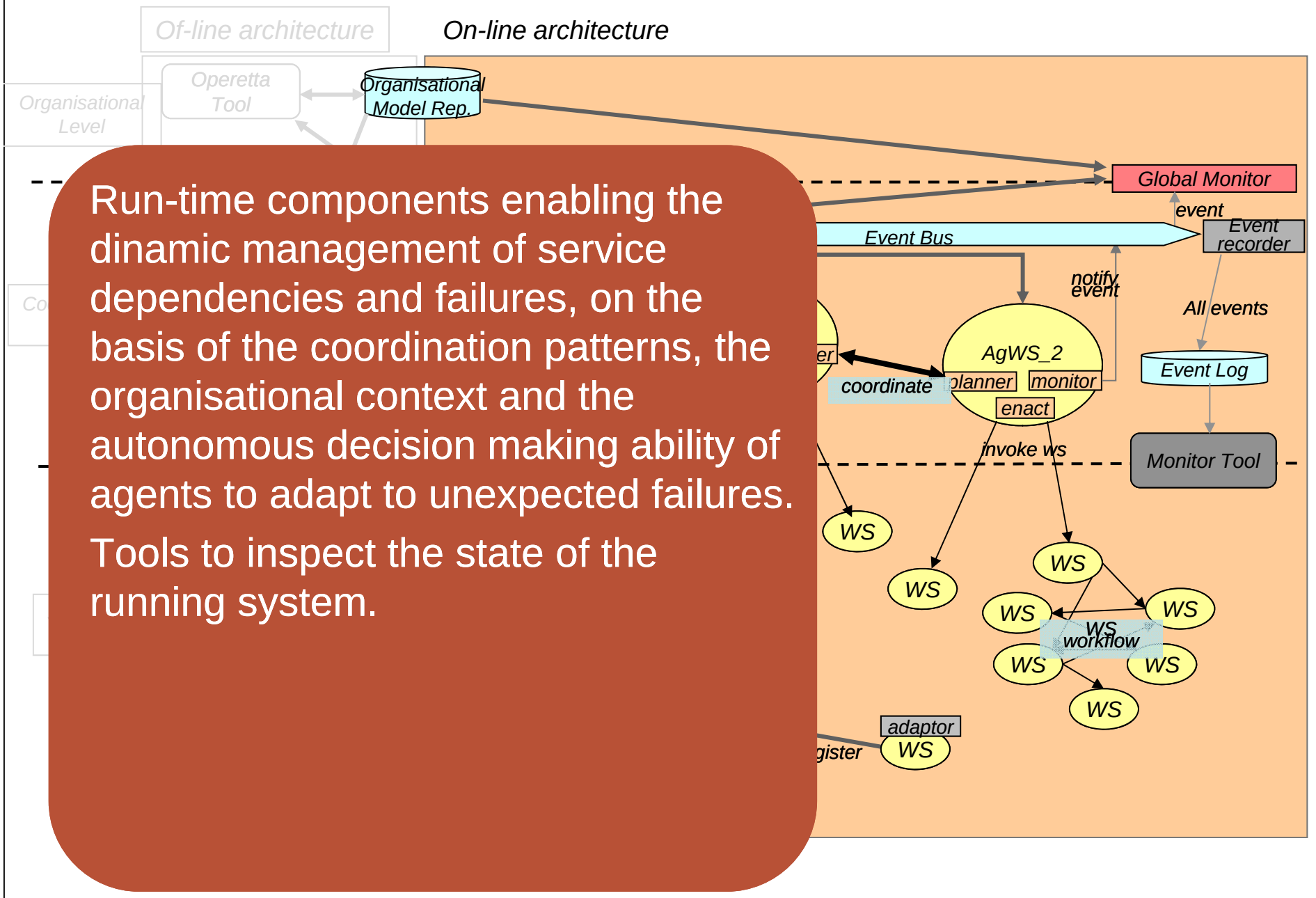

Tools to create organisation and coodination specifications, create agentified webservices, annotate existing services and set-up the running components of the system.

# ALIVE Off-line Architecture




Of-line architecture
On-line architecture
Organisational Level
Coordination Level
Service Level
Operetta Tool
Organisational Model Rep.
Ontology Editor
Domain Ontology Rep.
Coordination Model Rep.
Coordination Design Tool
Plan Synthesis
Plan Repository
Ontology Editor
Service ModelRep.
Service Design Tool
Template Repository
Matchmaker
Service Set-up Tool
Global Monitor
event
Event Bus
Event recorder
Notify event
notify event
All events
AgWS_1
monitor
planner
enact
AgWS_2
planner
monitor
enact
coordinate
Event Log
Matchmaker Agent
Request WS for task
invoke ws
invoke ws
Monitor Tool
look for ws for a task
WS
register
register
Service Directory
adaptor
register
WS
workflow

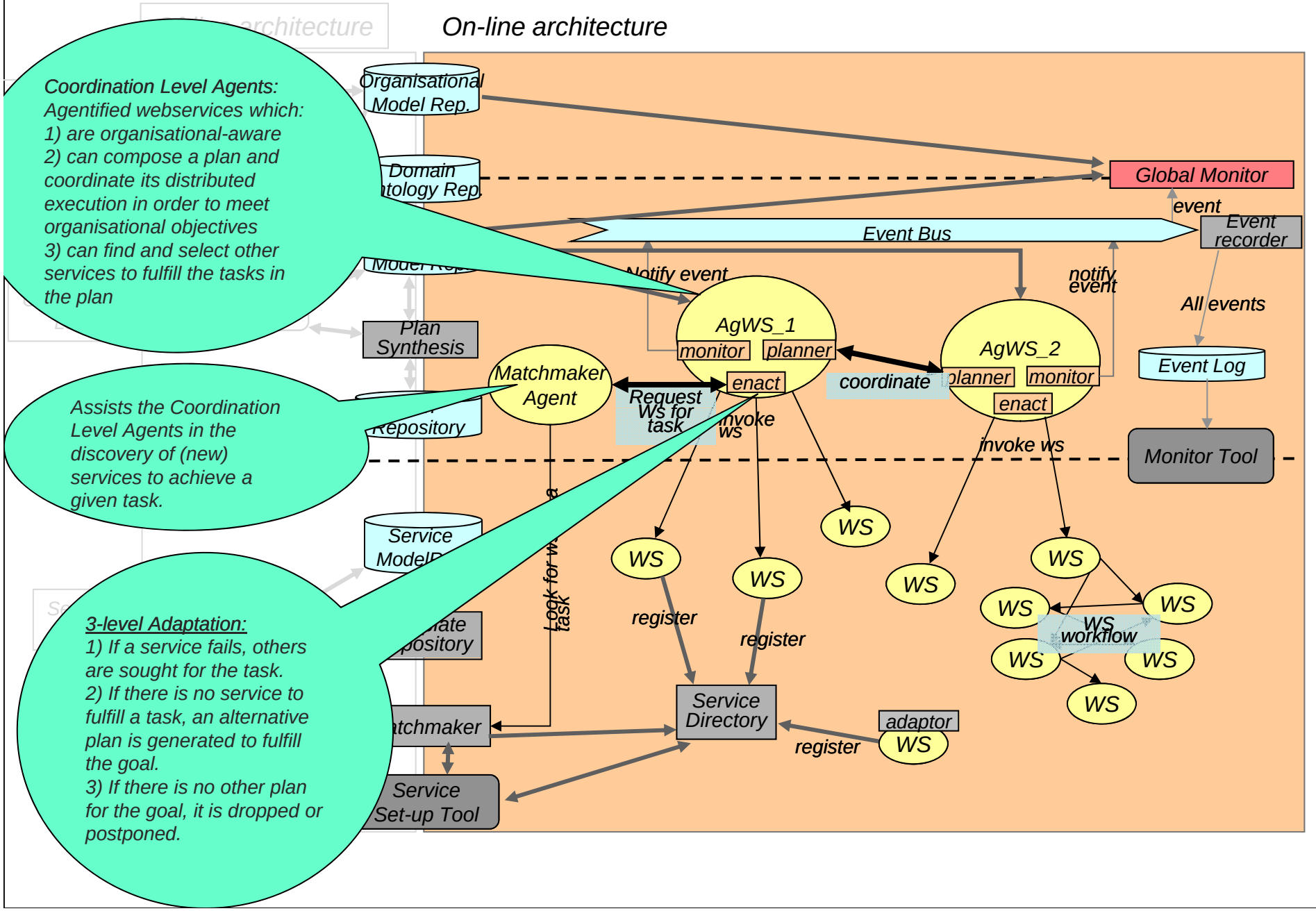
# ALIVE On-line Architecture



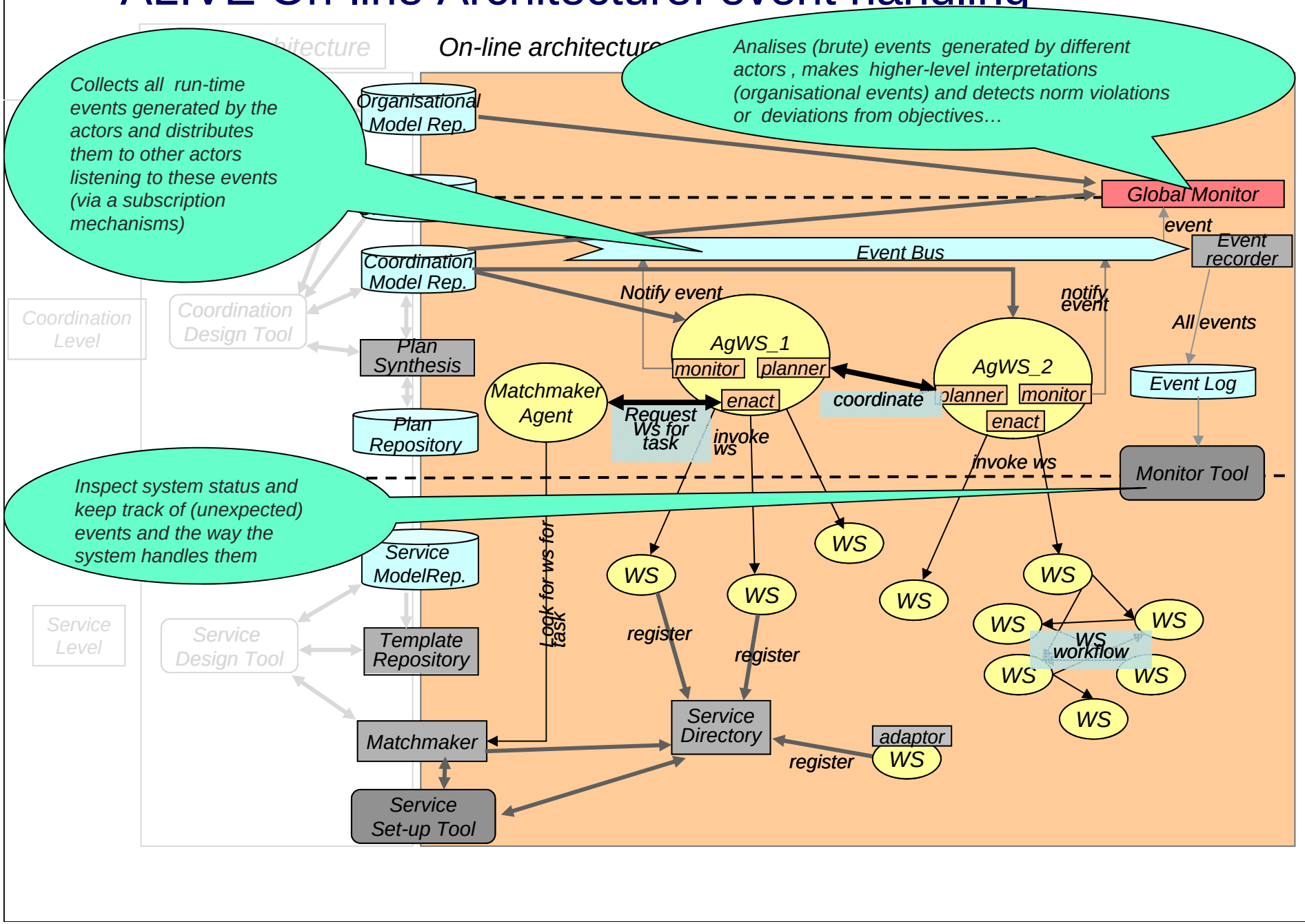

Run-time components enabling the dinamic management of service dependencies and failures, on the basis of the coordination patterns, the organisational context and the autonomous decision making ability of agents to adapt to unexpected failures.

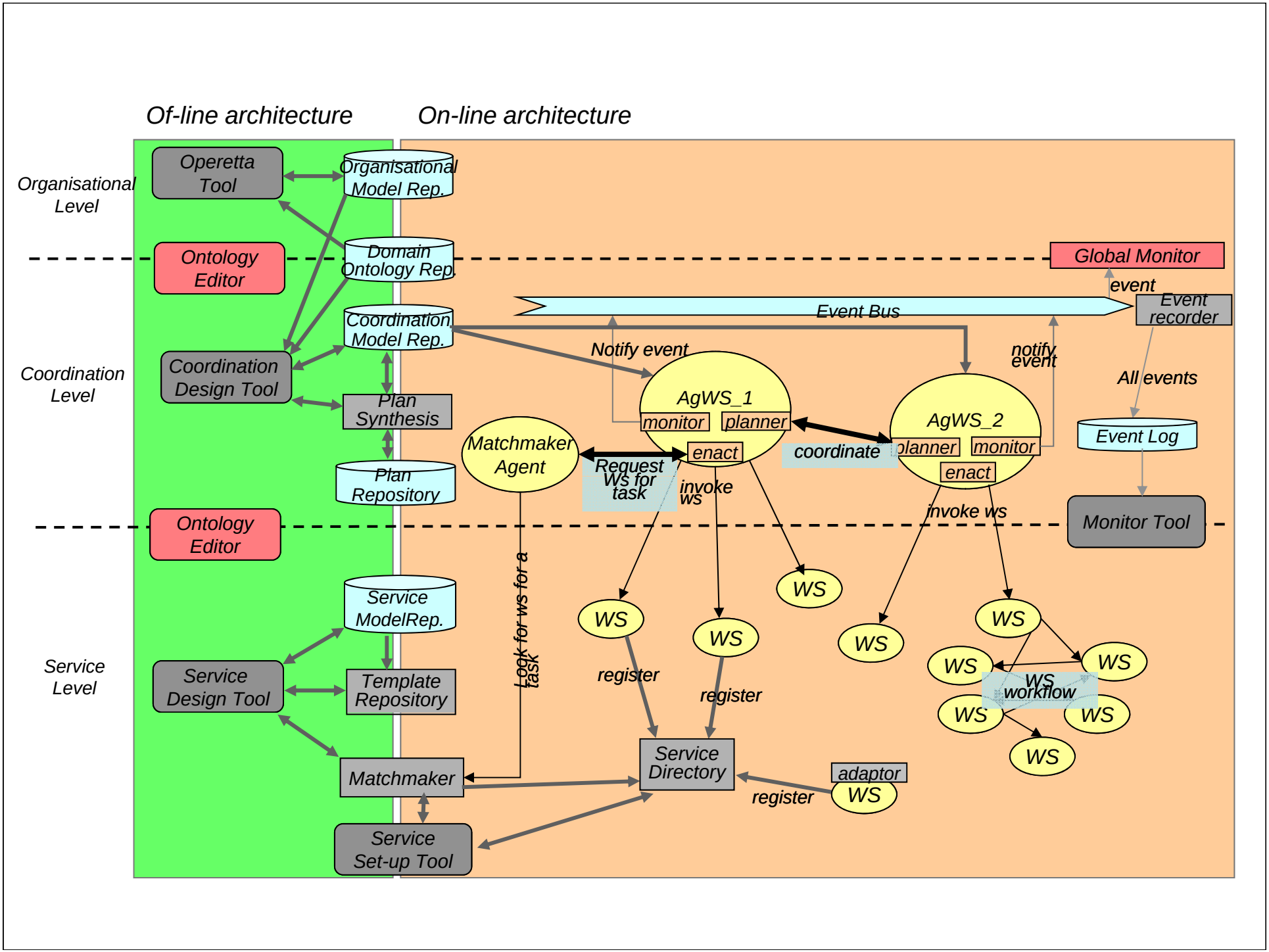
Tools to inspect the state of the running system.

# ALIVE On-line Architecture: service composition



# ALIVE On-line Architecture: event handling




Of-line architecture
On-line architecture
Organisational Level
Coordination Level
Service Level
Operetta Tool
Organisational Model Rep.
Ontology Editor
Domain Ontology Rep.
Coordination Model Rep.
Coordination Design Tool
Plan Synthesis
Plan Repository
Ontology Editor
Service ModelRep.
Service Design Tool
Template Repository
Matchmaker
Service Set-up Tool
Global Monitor
event
Event Bus
Event recorder
Notify event
notify event
All events
AgWS_1
monitor
planner
enact
AgWS_2
planner
monitor
enact
coordinate
Event Log
Matchmaker Agent
Request WS for task
invoke ws
invoke ws
Monitor Tool
look for ws for a task
WS
register
register
register
Service Directory
adaptor
WS
workflow

Why shall we do this?

# Benefits of ALIVE for SOA

- Mapping human organisations to service-based solutions
  - models are defined at a level of abstraction that allows non-expert end-users to support better the design and the maintenance of the system
- Provides an organisational context (such as, e.g., objectives, structures and regulations) that can be used to select, compose and invoke services dynamically.
- Multi-layer approach allows for:
  - ***Traceability*** (why is something done in this way on this level?)
  - ***Adaptivity*** (moving up in abstraction to solve problems at a specific level)

# Change and adaptation in ALIVE

- Adaptation in 3 levels:

*Service*

– Changes in system functionalities

*e.g., services that become unavailable or are not used correctly*

*Coordination*

– Changes in environmental conditions

*e.g., changes (sensed symptoms) that can lead to potential failure during the achievement of objectives*

*Organisation*

– Changes in stakeholders needs

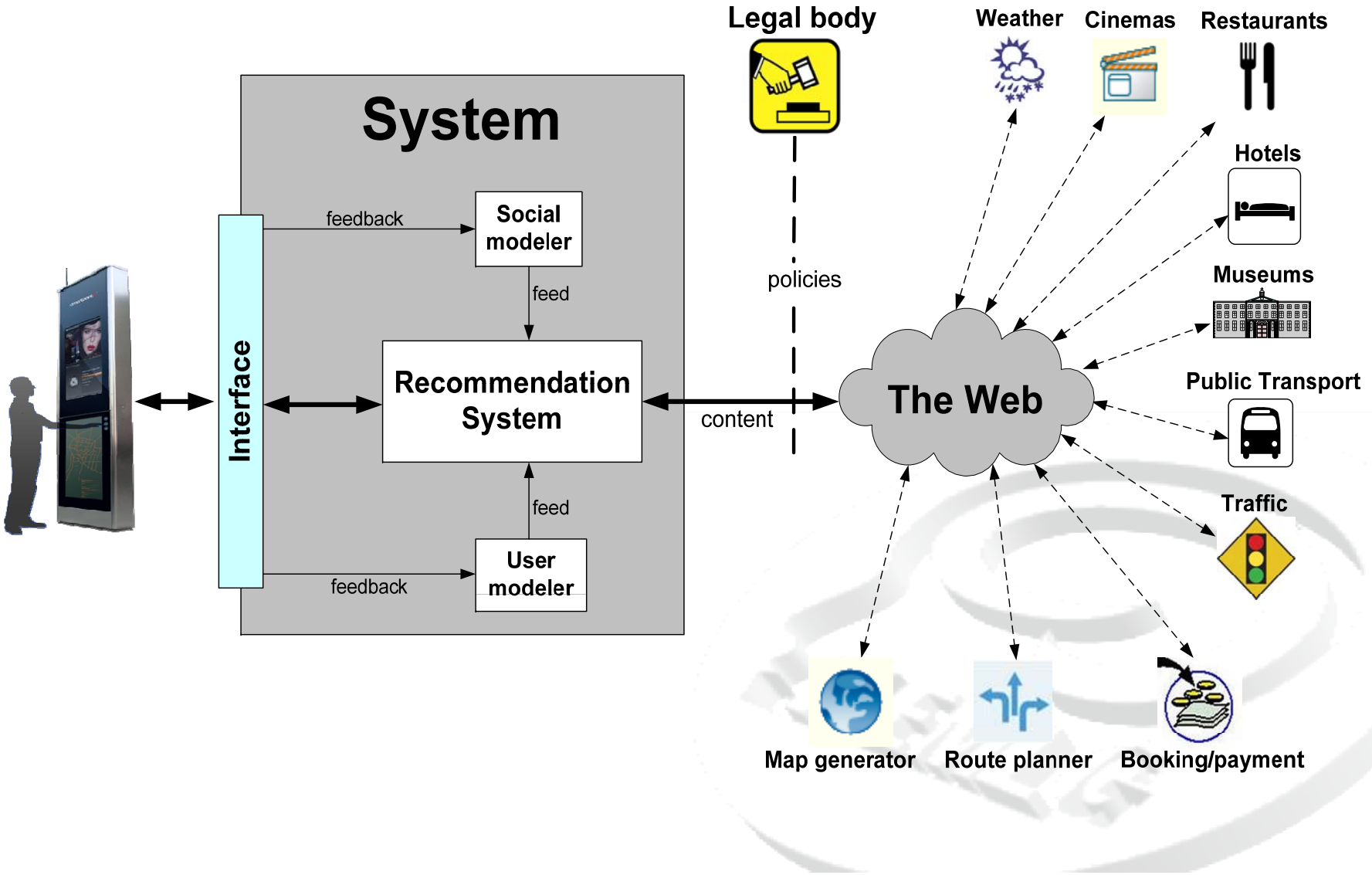
*e.g., changes in laws and norms that regiment particular organisational protocols and responsibilities*

Why shall we do this?

# Ex.1: Interactive Community Displays

:tmtfactory>

Why shall we do this?



# Ex.1: Interactive Community Displays

:tmtfactory>

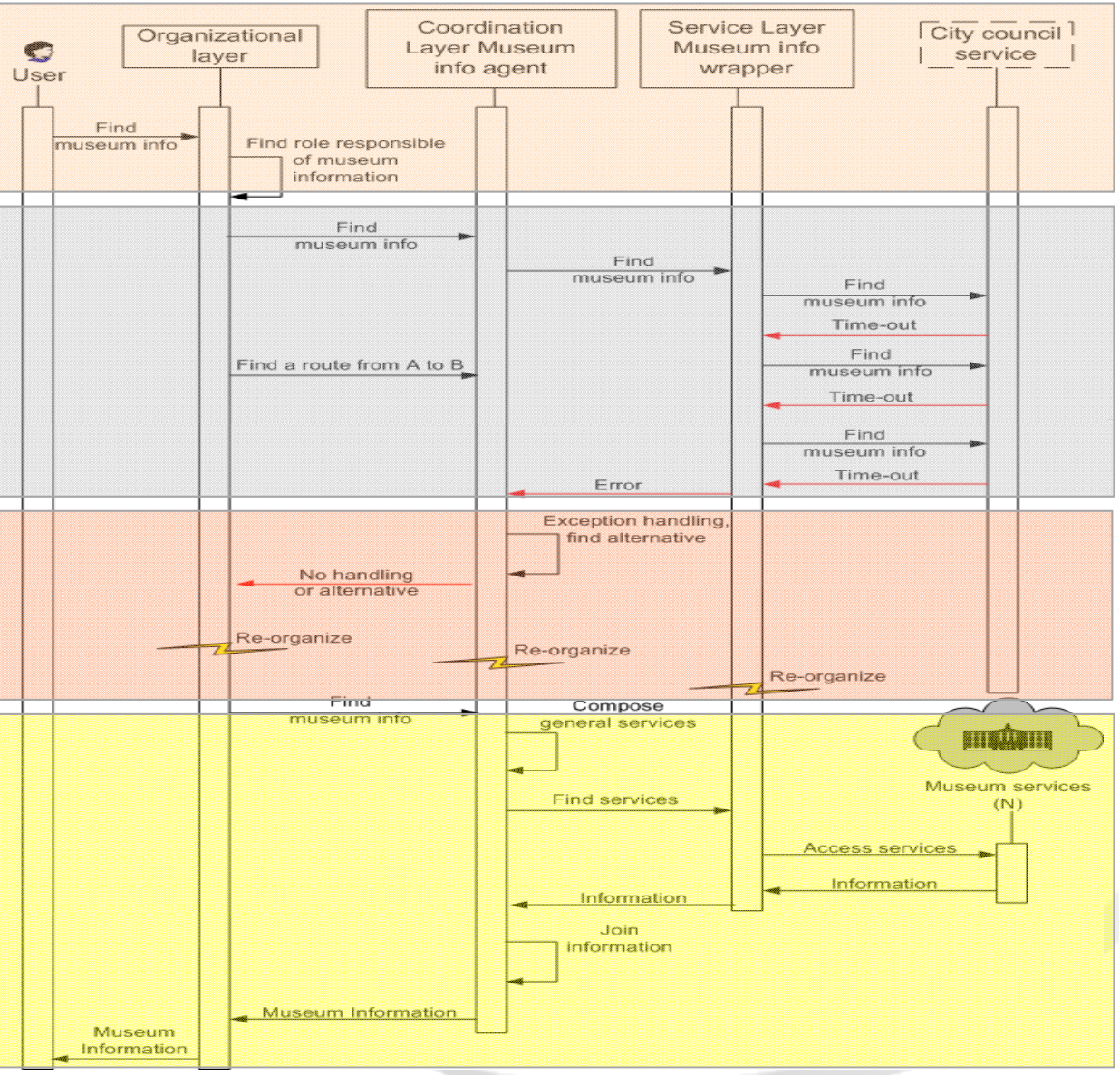
Why shall we do this?

*A set of services is selected to fulfill a user request.*

*The service selected for the "find museum info" task fails …*

*No alternate service is found for the task → re-plan*

*A new set of services is invoked and the results merged to fulfill the user request.*



Why shall we do this?

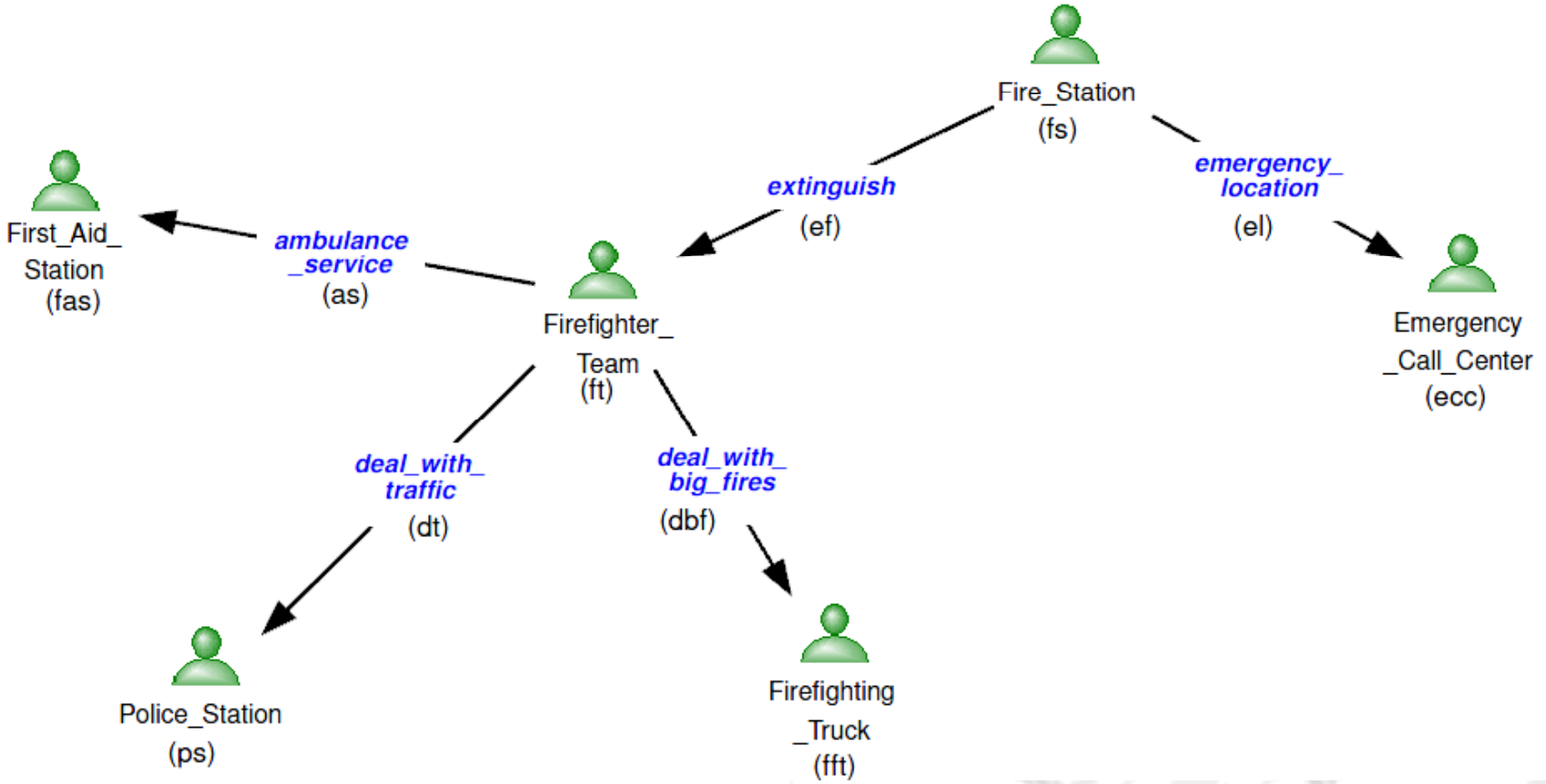
# *Ex.2: Dynamic Crisis Management*

THALES


- *(non-local) Inter-agency Cooperation*
- *Different services mean different priorities.*
- *Different policies for different crisis scenarios.*
- *Disaster profile changes*

# Ex.2: Dynamic Crisis Management

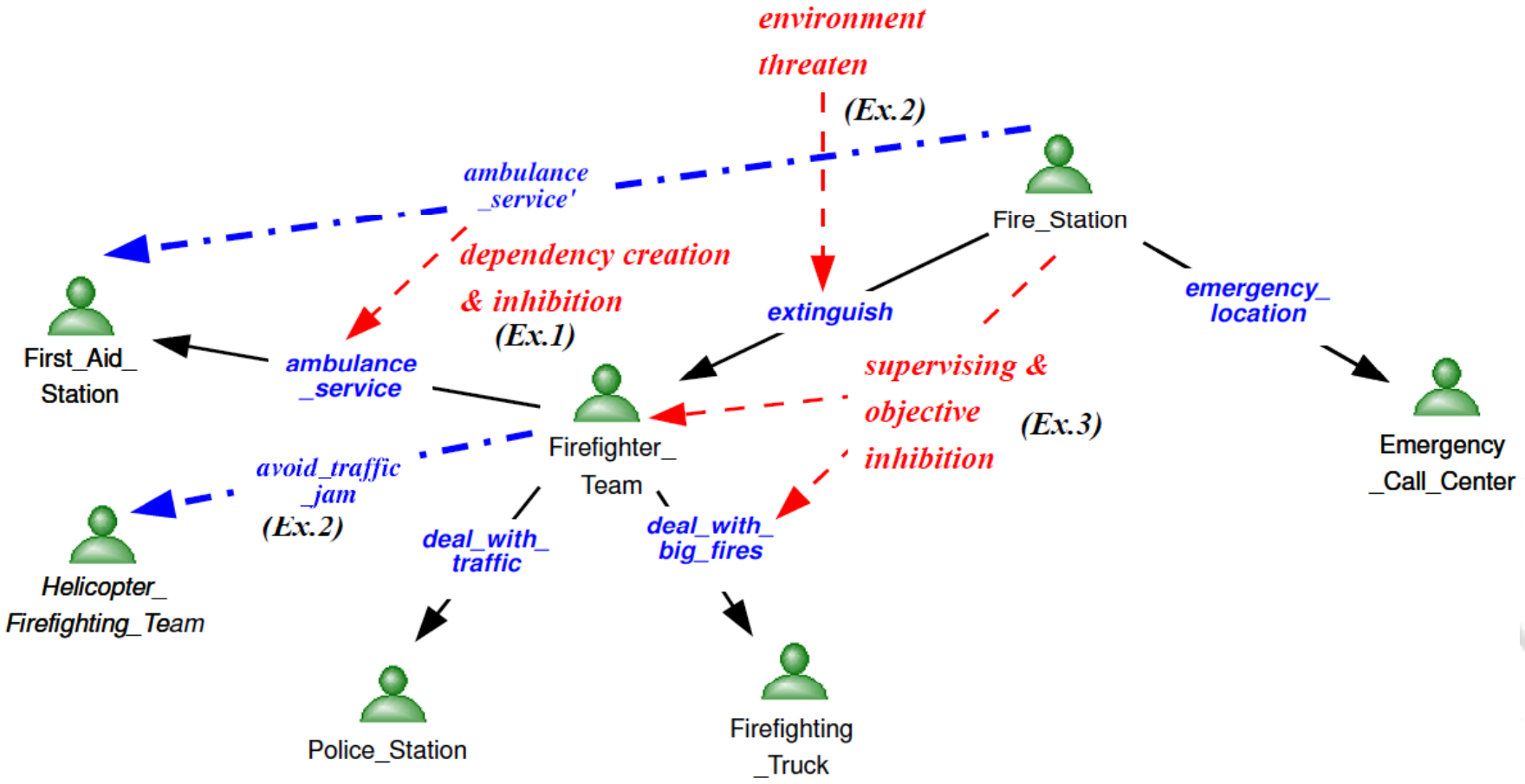
THALES




Why shall we do this?

# Ex.2: Dynamic Crisis Management

THALES




*Emergency Scalation Handling → Changes in Stakeholders' Relationships in Various Situations*

Why shall we do this?

# ALIVE contributions

- **Sound Organisational framework**
  New framework incorporates both organisational and institutional concepts for design, deployment and management of distributed systems.

- **New design and methodological approaches**
  Design methods and tools based on Model-Driven Engineering.

  - Automatic transformations from specifications in one level to the other levels, easing design and providing coherence among levels

# ALIVE contributions

- **New engineering techniques and components**
  Provide concrete modelling languages and their implementations to capture organisational, coordination and service levels, generating executable code from specifications.

  - Organisational Normative Agents: agents that can keep track of multiple instantiations of norms and use them in their goal-oriented task selection and plan formation.

  - Real-time, flexible Organisational Monitoring Architecture: a monitoring architecture capable of:
    - collecting great amounts of low-level events,
    - interpreting them in terms of the organisational concepts
    - detecting behavioural deviations and non-compliance to norms.

Why shall we do this?

# Conclusions


Knowledge Engineering and Machine Learning Group
UNIVERSITAT POLITÈCNICA DE CATALUNYA
*https://kemlg.upc.edu*

# Conclusions

- Most e-Business applications will require dynamic integration of a large number of complex services.
  - Need for technologies able to dynamically adapt and reconfigure in an ever-changing environment.
- Current SOA technology is not prepared for the challenge
  - Stateless services
  - Low-level, static (business) process models, prone to failure.
- Need to decouple
  - **WHAT to do from HOW to do it**
    - Action Descriptions vs Service Descriptions
    - Landmark patterns vs Workflows
    - Planning to bridge the gap.
  - **WHY do things from WHAT to do**
    - Organisational aims vs Actions
    - Organisational aims drive adaptation
- Have presented two approaches
  - **Institutional Approach:**
    - Contracts and a Contractual e-Institution
    - Contract agreement, deplyment and management
  - **Organisational Approach:**
    - Includes design, deployment, management and maintenance

# Some thoughts for the future

- Technological shift:
  - Future Internet Platforms will have new ways for functional discovery (things, services…) across multiple platforms (www, cell phone network, ad-hoc networks..)
    [Source: EU FP7 Future Internet Platform White Paper]
  - How will we be able to (describe and) discover business in this new setup?

- Business shift:
  - Shift from business- or product-centric view towards customer-centric view
    - Adapt to user needs, consumer-centric strategies
  - The new business services should be flexible to adapt to different markets within Europe.
    - It should be supported by ICT tools.
  - Main role of ICT in Future e-Business is not about chain optimisation or efficiency, or differenciation, but on innovation.
    [Source: EU Fines Cluster Position Paper]

# Vision Statement:
## Future Internet based Enterprise Systems 2025

- *"Specifically, the Future Internet will enable enterprises to:"*
  - *"Be empowered by a new participative Web, hosting a new wave of services and using userfriendly technologies."*
  - *"Create new value by leveraging the Internet as the platform through which knowledge is exploited dynamically, experienced in the business context and represented in a radically different way."*
  - *"Have the required capability that enables and supports collaboration with other enterprises, new dynamic relationships, discovery of partnerships, new opportunities and markets, and the management of the new risks and uncertainties involved."*
  - *"Operate in a new set of business environments that provide support for quality measures, guarantees, persistence, safety, trust, arbitration and other mechanisms for reducing risks on both the customer and the provider side."*
  - *"Become the WYSIWYG [What You See Is What You Get] enterprise, where Web-based applications become as rich as their desktop equivalents."*

Why shall we do this?


**http://www.lsi.upc.es/~jvazquez**