Ingeniería de software orientado a agentes

Javier Béjar

ECSDI - 2023/2024 2Q

CS-GEI-FIB

Objetivos del tema

- Software y sistemas complejos
- Ingeniería de Software orientada a agentes
  - Abstracciones, Autonomía, Interacción, Organización
- Medodologías orientadas a agentes - Proceso/fases

Ingeniería de software

- El desarrollo de aplicaciones software cada vez más complejas hace que se necesiten mayores niveles de abstracción
- Conceptos, modelos, metodologías, tecnologías y herramientas evolucionan forzando a afrontar cambios radicales en las formas de desarrollar software
- El paradigma de orientación a objetos no puede ser considerado la última respuesta en esta tendencia, es solo un paso más

Sistemas computacionales

- **Pasado:** Sistemas centralizados, modelo de programación monolítico
- **Presente:** Sistemas distribuidos, heterogéneos, escalables, abiertos, modelo de programación distribuido

Desarrollo de software

- **Pasado:** Modelo de desarrollo en cascada
- **Presente:** Modelo incremental, ágil, procesos de desarrollo experimentales

- El desarrollo de software basado en agentes se plantea como una nueva perspectiva para el desarrollo de sistemas software
- La **orientación a agentes** subsume los conceptos soportados por los previos paradigmas de programación y en particular los de la programación orientada a objetos.
  - Elevan el nivel de abstracción
  - Son una aproximación más adecuada para el desarrollo de software complejo

Sistemas Complejos

- Los sistemas complejos están formados por una jerarquía de subsistemas interrelacionados
- La elección de los componentes primitivos es relativamente arbitraria (objetivos y necesidades)
- En la jerarquía podemos distinguir entre:
  - Interacciones intra-sistema (más frecuentes y predecibles)
  - Interacciones inter-sistema (menos frecuentes)

- Esto hace que sean **casi-separables**
- Lo que no los hace totalmente separables son las interacciones inter-sistema
- Algunas de estas interacciones no son predecibles en tiempo de diseño

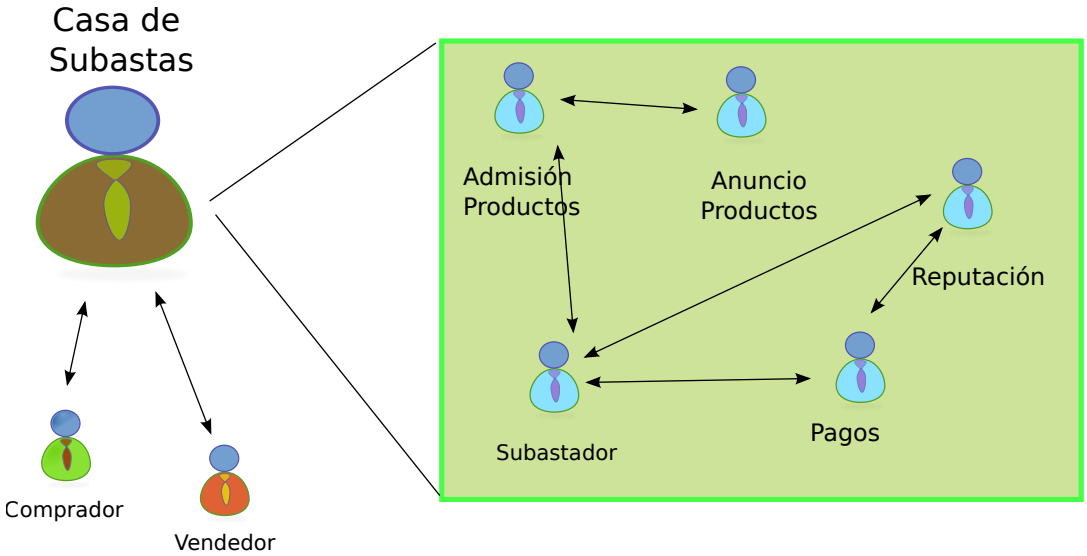
- Los problemas complejos son descentralizados
- Con múltiples puntos de control de ejecución (subproblemas)
- Con múltiples perspectivas sobre el problema y múltiples objetivos (según los subsistemas)
- Los diferentes subsistemas deben interaccionar para obtener sus objetivos y resolver sus dependencias
  - A través de mensajes (de alto a bajo nivel)
  - Mediante interacciones sociales (coordinación, cooperación, negociación)

Ingeniería de software orientada a agentes

- **Abstracciones:**
  - Componentes autónomos que buscan unos objetivos
  - Componentes que interaccionan a alto nivel
  - Componentes que se organizan socialmente
  - Componentes que pueden cambiar su relaciones dinámicamente
  - Componentes que se pueden ver a diferentes niveles de granularidad
- El desarrollo se basa en la agregación de componentes de manera jerárquica con una filosofía **de abajo a arriba**

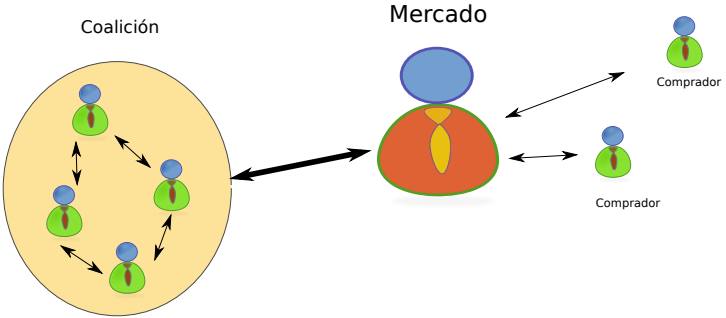
- Asignamos TAREAS al software (p. ej.: Software para subasta de mercancías)
  - El **qué** y el **cómo** se especifican por adelantado (casos de uso, escenarios)
  - No son tolerables los cambios en los requerimientos
- Asignamos ROLES a los agentes (p. ej.: Agente subastador)
  - El **qué** es especificado por adelantado, el **cómo** se determina dinámicamente (librería de métodos)
  - Son tolerables cambios en los requerimientos

- La inter-acción se ve como organización social:
  - De relaciones entre pares a relaciones jerárquicas
  - De relaciones puntuales a relaciones a largo plazo
- Esto permite:
  - Caracterizarlas y describirlas de manera abstracta
  - Agrupar diferentes componentes y usarlos como una unidad facilitando la descomposición
  - Usar en el análisis y desarrollo métodos/protocolos bien conocidos en organizaciones sociales


Casa de Subastas
Comprador
Vendedor
Admisión Productos
Anuncio Productos
Reputación
Subastador
Pagos

- No es posible determinar todas las inter-acciones en el diseño
- Los agentes son capaces de decidir como reaccionar y resolver las interacciones en ejecución (flexibilidad)
  - Tomando decisiones sobre situaciones imprevistas
  - Tomando decisiones sobre interacciones erróneas
  - Pidiendo asistencia a otros agentes
- Su comportamiento no esta predeterminado totalmente, resulta de la interacción dinámica entre los participantes

- En un sistema de comercio electrónico:
  - Agentes compradores determinan que pueden obtener un mejor precio si se coaligan y hacen compras más grandes en lugar de compras individuales



- Comunicación:
  - Software: Comunicación a nivel de señal
  - Agentes: Comunicación a nivel simbólico
- Los agentes
  - persiguen objetivos y necesitan que otros agentes persigan objetivos relacionados (delegación)
  - necesitan llegar a acuerdos
  - necesitan tomar decisiones organizacionales
  - necesitan de comunicarse sus creencias/conocimiento

- Para que un mensaje sea entendido, se deben asignar un significado a los elementos de su contenido
- Requeriremos una **Ontología** para hacer la correspondencia entre una codificación y un significado
- **Paso de mensajes dinámico**
  - Los agentes han de descubrir si comparten un conocimiento mutuo del dominio (la ontología) para poder continuar la comunicación

- Los sistemas software necesitan poseer un conocimiento mutuo completo para poder interaccionar (todo está predeterminado)
- Los agentes software pueden tener (o no) conocimiento mutuo completo acerca de:
  - Los objetivos de otros agentes
  - Sus estrategias (acciones que pueden usar)
  - Sus utilidades (beneficio de sus acciones)

- Interacciones que asumen el completo conocimiento se clasifican como cooperación y coordinación
  - Todos los agentes buscan un fin común
  - Los agentes necesitan delegar objetivos para cumplir su cometido
- En muchos casos la suposición de conocimiento completo puede no ser cierta: competición
  - Solo algunos agentes pueden cumplir su cometido (p.ej.: recursos limitados)

- Los puntos de decisión de los sistemas software son deterministas
- Los agentes software (inteligentes) están dotados de mecanismos de razonamiento
- La **toma de decisiones** involucra múltiples flujos de control:
  - Pensamos también en consecuencias inmediatas
  - Reaccionamos a estímulos
  - Planteamos objetivos a corto plazo
  - Replanteamos objetivos a largo plazo

- **Agentes**, entidades autónomas, elementos de control independientes, situación en un entorno, interacción
- **Entorno**, mundo de entidades y recursos que el agente percibe, controla, explota o consume.
- **Roles** e **interacciones**, funcionalidades, actividades, responsabilidades y patrones de interacción.
- **Reglas de organización**, restricciones a roles e interacciones, o relaciones entre roles y entre protocolos
- **Estructuras** y **patrones de organización**, topología de interacción, régimen de control de actividades

Inteligencia Artificial (fuerte)

Un sistema multiagente es una sociedad de individuos (agentes software inteligentes) que interaccionan intercambiando conocimiento y negociando entre ellos para lograr sus propios intereses o un objetivo global.

Ingeniería de Software (débil)

Un sistema multiagente es un sistema software compuesto por múltiples elementos de control independientes y encapsulados (agentes) interaccionando entre ellos en el contexto de una aplicación específica

- Se focaliza en las características de los agentes que tienen impacto en el desarrollo de software:
  - Concurrencia, interacción, múltiples elementos de control
  - La *inteligencia* puede verse como una forma particular de control independiente, las *conversaciones* como una forma particular de interacción.
- Es más general:
  - Diferentes sistemas software, incluso si no se conciben como basados en agentes, pueden caracterizarse en términos de sistemas multiagente débiles

Metodologías de software orientadas a agentes

- Una metodología de software tiene como objetivo introducir una disciplina en el desarrollo:
  - qué producir y cuando
- Define el marco conceptual del desarrollo
- Define las abstracciones a usar para modelar el software:
  - Orientada a datos, flujos, objetos...

- **Análisis**, qué debería hacer el sistema y cuáles son las restricciones de desarrollo
- **Diseño**, especificación del sistema cumpliendo los objetivos y restricciones
- **Desarrollo**, proceso de producción del sistema software
- **Validación**, comprobar que el software es lo que el cliente quiere
- **Evolución**, cambiar el software en respuesta a los cambios

¿Cuál es el proceso del software ideal?

No existe un proceso ideal

- AOSE es el paso siguiente en la evolución de orientación a objetos, patrones de diseño y diseño basado en componentes
- Apropiado para sistemas abiertos y sistemas distribuidos
- Sus características están más alineadas con este tipo de entornos (p. ej.: Internet, Cloud computing, IoT...)

- La fase de **análisis** consiste en entender:
  - Cuáles son los actores principales que interaccionan con el sistema
  - Cómo el sistema interacciona con esos actores
  - Qué tiene que hacer el sistema (globalmente)
- En la fase de análisis vemos el sistema como una entidad cerrada para no anticipar decisiones de diseño

- Asociamos agentes con las entidades de los escenarios que se analizan
- Dentro de esos escenarios asignamos:
  - Roles, responsabilidades y capacidades
  - Patrones de interacción entre agentes
- El objetivo es tener una visión neutra del problema
- Algunas metodologías no usan la palabra agente para denotar las entidades en esta fase

Analogía cinematográfica

- Agentes software = Actores representando papeles
- Casos de uso = Guión
- Ingeniero de software = Productor/director

- Cuáles son los principales componentes que interaccionan dentro del sistema
- Cuáles son las responsabilidades y capacidades de cada componente del sistema
- Cómo los componentes interaccionan para implementar el sistema (su arquitectura)

- Se asocia agentes con los componentes que se usan para construir el sistema
- A partir de ahí se refinan:
  - Roles, responsabilidades y capacidades
  - Patrones de interacción entre agentes
- A diferencia del análisis, hay que escoger qué agentes usar y como interaccionan

- **GAIA**: desarrollo como el diseño de una organización
- **TROPOS**: enfocada en el análisis de requerimientos
- **PASSI**: metodología paso a paso de requerimientos a código que integra modelos de diseño y conceptos de orientación a objetos e inteligencia artificial
- **Prometheus**: se focaliza en el diseño organizacional y en el de la arquitectura interna del agente (diseño de sistemas de agentes BDI)