3. Agent-Oriented Methodologies

Part 2: The PROMETHEUS methodology.

Javier Vázquez-Salceda

MASD

Methodological Extensions to Object-Oriented Approaches

- A means for agent technologies to gain traction within industrial settings may be by being introduced through well-established technologies
- The Unified Modeling Language (UML) has gained wide acceptance for the representation of engineering artifacts using the object-oriented paradigm
- There are several attempts to extend UML so as to encompass agent concepts
- In general, building methods and tools for agent-oriented software development on top of their object-oriented counterparts seems appropriate
  - It lends itself to smoother migration between these different technology generations
  - It improves accessibility of agent-based methods and tools to the object-oriented developer community which, as of today, prevails in industry.

jvazquez@lsi.upc.edu
The Prometheus Methodology

- Phases
- Tools
- From Prometheus to ROADMAP

Prometheus

- Prometheus is an iterative methodology covering the complete software engineering process
  - Analysis, Design, Detailed design, Implementation

- Aims at the development of intelligent agents (in particular BDI agents)
  - Uses goals, beliefs, plans, and events.

- The resulting specification can be implemented in any agent implementation software that covers such abstractions
  - Specially aimed for implementation with JACK

- It is evolved out of practical experiences

- It is aimed at industrial software development, not researchers
Prometheus Overview

- Methodology developed over 7-8 years in collaboration with industry partner (Agent Software). Feedback from many students and industry partner clients.
- Focus on detailed guidance and structure to facilitate tool support.
- Mixture of
  - graphical notation for overview
  - (structured) text notation for detail.
- Hierarchical and modular.
- Prototype tool available and used externally.

Prometheus Phases

- The Prometheus methodology covers three phases
  - The system specification focuses on identifying the basic functions of the system, along with inputs (percepts), outputs (actions) and their processing (for example, how percepts are to be handled and any important shared data sources to model the system’s interaction with respect to its changing and dynamic environment)
  - The architectural design phase subsequent to system specification determines which agents the system will contain and how they will interact
  - The detailed design phase describes the internals of each agent and the way in which it will achieve its tasks within the overall system. The focus is on defining capabilities (modules within the agent), internal events, plans and detailed data structures.
Prometheus
Process Overview

Prometheus
System Specification Phase
Prometheus
System Specification phase

- System defined by
  - Goals: *goal diagram*
  - Scenarios: *user case scenarios*
  - Functionalities: *functionality descriptors*

- System interface with environment described in terms of
  - actions,
  - percepts
  - external data

Prometheus
System Specification phase: Steps (non-sequential!)

- Start with high-level description of the system (textual)
- Identify actors
- Identify top-level scenarios for each actor
- Identify inputs/outputs (actions/percepts)

The online bookstore’s percepts and events include customers visiting the website, selecting items, placing orders using forms, and receiving email from customers, delivery services and book suppliers. Actions include bank transactions, sending email, and placing delivery orders.
Prometheus
System Specification phase: Steps (non-sequential!)

- Add a corresponding system goal for each use-case

Prometheus
System Specification phase: Goal Overview Diagram

- Apply Goal Abstraction to system goals
- Refine Goal (OR/AND refinement)
- Link goals to (sub)scenarios

Maintain large range of books

Borrow books from other libraries

Find cheapest price

Order books

Organise delivery

Log Order

Scenario

how?

why?

OR

AND
Prometheus
System Specification phase: Goal Overview Diagram

Maintain large range of books

- Why? OR - How?

Order books
- Find cheapest price
- Organise delivery
- Log Order

Borrow books from other libraries

Scenario

- Good practices:
  - Except in extreme situations, the goal diagram should be a fully-connected graph.
  - The abstraction level should be balanced in the different branches.
  - All (sub)goals should be linked to scenarios.

Prometheus
System Specification phase: Steps (non-sequential!)

- Identify the functionalities of the system
  - Idea: identify roles and activities

NAME: Welcoming
Description: Welcomes a new visitor to the world wide web site (with personalised information if possible).
Percepts/events/messages: Customer:Arrived (message), Customer:Information (message)
Messages sent: Customer:InformationRequest (message), Customised:WWWPage (message)
Actions: Display:Customised:WWWPage
Data used: Customer:DB, Customer:Orders
Prometheus
System Specification phase: Steps (non-sequential!)

- Develop and refine the Scenarios and sub-scenarios
  - Steps inside a scenario consist of:
    - Incoming event/percept (→ receiving functionality)
    - Message (sender → receiver)
    - Activity or actions (→ functionalities)

Scenario: Book Order
Overview: The user orders a book. Delivery options are explored and then confirmed (with an OrderRequest). The books are shipped, stock updated, and the user notified.
Context: Assumes the book is in stock.
Steps:
1. **EVENT:** BookOrder (→ Online Interaction)
2. **DeliveryOptionQuery** (Online Interaction → Transport Information)
3. **DeliveryOptions** (Transport Information → Online Interaction) Data read: Transport FWR
4. **Obtain preferred delivery option** (Online Interaction)
5. **MakePayment** (Online Interaction → Sales Transaction)
6. **ACTION:** BankTransaction (Sales Transaction)
7. **PlaceOrder** (Sales Transaction → Order Handling)
8. **Register order** (Order Handling) Writes data: CustomerOrders
9. **ACTION:** TransferContactCompany (Order Handling)
10. **DecreaseStock** (Order Handling → Stock Manager)
Variations: steps 6 & 7 (general overview) and 10 (decrease stock) replaced with notification of delay (Order Handling to Customer Contact) and then placing an order for more stock (Order Handling to Stock Manager).

Prometheus
Architectural Design Phase

System specification artifacts
- Scenarios
- Actions, Percepts
- System goals
- Functionality descriptors

Architectural Design
- Interaction diagrams
- Conversation protocols
- System overview
- Agent descriptors

Detailed design

jvazquez@lsi.upc.edu
Prometheus
Architectural Design Phase: Agent types

- Option 1: The domain already identifies agent types
- Option 2: Identify the **agent types** in the system by
  - Grouping functionalities to agent types based on cohesion and coupling
  - Grouping functionalities that are
    - related based on common sense
    - group functionalities that require a lot of the same information:
      - **Data Coupling Diagram**
  - Do not group functionalities that are
    - clearly unrelated
    - exist on different hardware platform
    - security and privacy
    - Modifiable by different people
  - Evaluate grouping:
    - Simple descriptive names (heuristic)
    - Generate agent acquaintance diagram

Prometheus
Architectural Design Phase: Data Coupling Diagram
Prometheus
Architectural Design Phase: Agent Descriptors

- Generate Agent Descriptors based on the agent types
  - How many agents of a each agent type (one, many, dynamic)?
  - What is the life time of the agent?
  - What is the initial state of the agent?
  - What should be done when agent is killed?
  - What is the data used/produced by the agent?
  - To which event the agent should react?

Name: Sales Assistant agent
Description: greets customer, follows through site, assists with finding books
Cardinality: one/customer.
Lifetime: instantiated upon customer arrival at site. Demise when customer logs out or after inactivity period.
Initialisation: Obtains cookie. Reads Customer DB.
Demise: Closes open DB connections.
Functionality included: Online Interaction, Sales Transaction, Welcome, Book Finder.
Uses data: Customers DB, Customers Orders, Book DB.
Produces data: Customer preferences, orders, queries
Goals: Welcome customer, Update customer details, Respond to queries; Facilitate purchases;
Events responded to: new arrival, customer query, customer purchase; credit check response customer response;
Actions: Display information to customer (greetings, book info, info requests, Display customised WWW page, RequestCreditCheck messages
Interacts with: Warehouse Manager (book request protocol), Delivery Manager (order protocol), order query protocol, Customer Manager (customer information query protocol, customer information update protocol)
Design Tip: When agent communication?

- Any protocol interaction should come from some agent communication needs.

- Goals for Agent Communication:
  - Agents able to request (to other ags.) actions or services that they cannot perform by themselves
  - Agents able to ask for information (to other ags.)
  - Agents able to share their beliefs with other ags.
  - Agents able to coordinate with other ags. To solve complex tasks.

- Design Tip:
  - In Prometheus any protocol interaction should be connected to a (sub)goal.
Prometheus
Detailed Design Phase

Architectural design artifacts
- Conversation protocols
- System overview
- Agent descriptors
- Process diagrams
- Agent overview
- Capability descriptors
- Capability overview
- Event descr.
- Data descr.
- Plan descr.

Implementation

Steps (I)
- Develop the internal structure of individual agents
- Identify the capability of each agent (start with functionalities)
- Generate capability descriptors

Name: Delivery Problem Handling
External interface to the capability: events used/produced
Natural language description: Respond if books are not in stock
Interaction with other capabilities: Transport capability
Data used/produced by the capability: Note problem to transport capability
Inclusion of other capabilities: None

Generate agent overview diagrams
Prometheus
Detailed Design Phase: Agent Overview Diagrams

Key
Action  Capability  Date  Message  Plan

Prometheus
Detailed Design Phase: Agent Overview Diagrams

Key
Action  Capability  Date  Message  Plan
Prometheus
Detailed Design Phase: Event, Data & Plan Descriptions

• Steps (II)
  • Plan descriptions
    
    **Name:** Delivery Problem Handling
    **Natural language description:** Respond if books are not in stock
    **Triggering event type:** Delivery problem, Delayed delivery
    **Plan steps:** Delivery Query, Register problems
    **Context of performing the plan:** The delivery is delayed
    **Data used/produced:** Produce note problem

• Event descriptions
  • Identify the purpose of events and the data carried by it

• Data descriptions
  • Identify the data structure and operations on the data

Prometheus
Tools: the Prometheus Design Tool (PDT)

![Diagram of Prometheus Design Tool (PDT)]
Prometheus
Tools: the Prometheus Design Tool (PDT)

- System Specification
- Architectural Design
- Detailed Design
- Implementation
- Debugging
- Testing

Prometheus: summary

- Main strengths:
  - Structured processes to refine design.
  - Automated consistency checking between (some of) the design artefacts.
  - Hierarchical and modular views.

- Actively continuing development…
ROADMAP

• It is an evolution on Gaia v2 with some ideas coming from Prometheus and other methodologies

• Main characteristics:
  - More abstract and high level than Prometheus.
  - Concerned with high level view of models needed.
  - Adds elements to deal with requirements analysis in more detail by using use cases.
  - Aims to better model open systems (Gaia’s main limitation)
  - It merges the abstract design and detailed design phases into a single design phase

• There exists only partial tool support:
  - REBEL (Roadmap Editor Built for Easy Development) which is designed to help the developer to identify the Goal Models and the Role Models during the analysis stage.
ROADMAP
Models (I)

- **Use Case Model**: discovers requirements in an effective and sufficient way, by means of scenario identification
  - An important part of the requirement elicitation is made by the identification of the system goals in the **Goal Model**.
- **Environment Model**: derived from the use case model, provides a holistic description of the system environment
- **Knowledge Model**: derived from above, provides a holistic description of the domain knowledge used in the system
- **Role Model**: identifies the key roles of the system and usually correspond to individuals, groups or organizations. They are associated to the goals and are characterized by four attributes: **Responsibilities**, **Permissions**, **Activities** and **Protocols**.

ROADMAP
Models

- **Interaction Model**: describes the dependencies and relationships between various roles in a multi-agent organization. It is defined by means of AUML interaction diagrams.
  - Further detail of the patterns of interaction is given by the **Protocol Model** at the design phase.
- **Agent Model**: identifies the agent types that make up the system, and can be thought of as a set of agent roles
- **Services Model**: identifies the main services, defining the function of an agent as characterized by input, output, pre-conditions and post-conditions that are required to realize the agent’s role
- **Acquaintance Model**: documents the lines of communication between the different agents
ROADMAP
Example of new models: Goal Model

Librarian

User

Borrow book

Select book

Register borrower

Provide return date

Large choice

Friendly

Role

Goal

Soft goal

Key

3. Agent-Oriented Methodologies

Roadmap Overview of Models (I): comparison with GAIA

Added in ROADMAP

Major changes in ROADMAP

Use Case Model

Environment Model

Role Model

Knowledge Model

Interaction Model

Abstract model of interaction

Concrete model of protocol

Analysis

Design

Agent Model

Services Model

Acquaintance Model

jvazquez@lsi.upc.edu

jvazquez@lsi.upc.edu
ROADMAP
Overview of Models (II): comparison with PROMETHEUS

Domain specific
- Use Case Model
- Environment Model
- Knowledge Model

Application specific
- Goal Model
- Role Model
- Agent Model
- Interaction Model

Reusable service models
- Service Model

Prometheus provides details in these models - and a little in the environment model.

ROADMAP
Integration with Prometheus

- Since its creation there have been plans to integrate ROADMAP and Prometheus into a single methodology:
  - Prometheus actors/stakeholders and functionalities become external/internal roles
  - Can identify goals or scenarios at top level
  - Add soft goals as annotations on all entities
  - Percepts and actions possibly wait till architectural design

- The integration of both methodologies has been first described in 2002...

- ...However, there have been few advances, especially on the tool support.

- Now-a-days, ROADMAP is presented not as a methodology but as an agent-based meta-model.
References


These slides are based mainly in [2], [4], [5] and material from M. Winikoff, L. Padgham, M. Luck, M. d’Inverno, R. Ashri and M. Dastani