

4. Multiagent Systems Design

Part 2:

The PROMETHEUS methodology.

Javier Vázquez-Salceda
SMA-UPC 2006

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) is gaining 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.

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

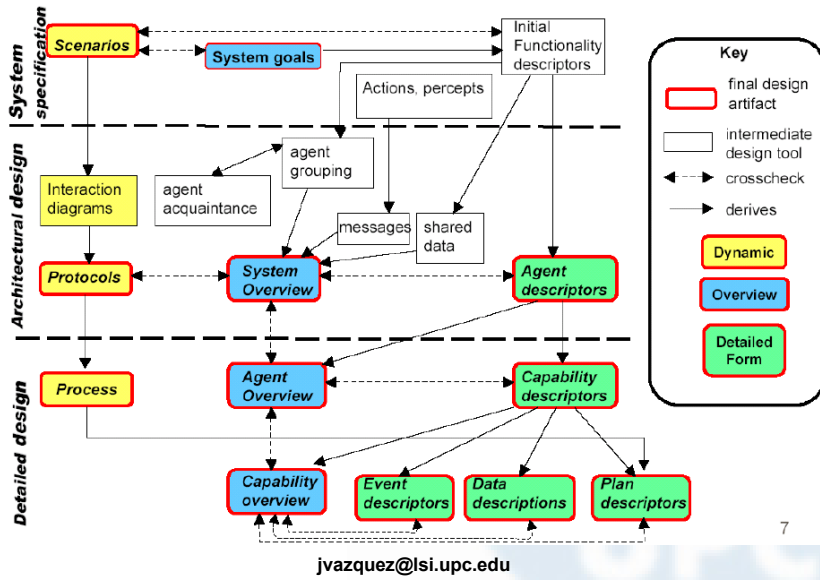
Prometheus

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

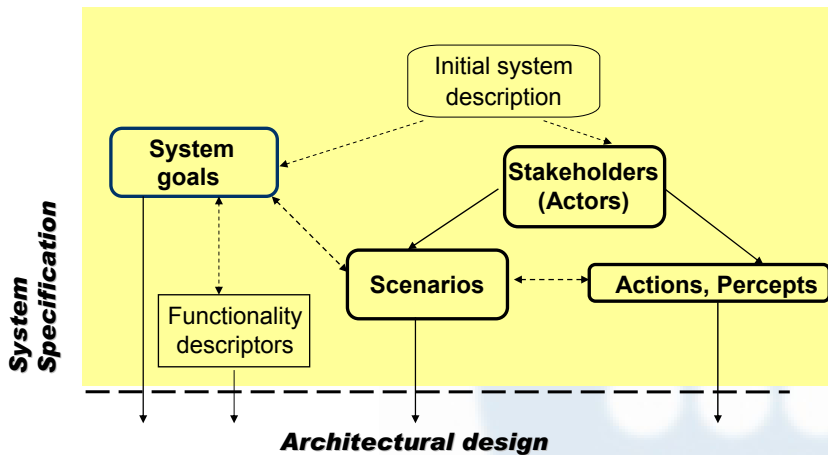
Process Overview



jvazquez@lsi.upc.edu

Prometheus

System Specification Phase



jvazquez@lsi.upc.edu

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!)

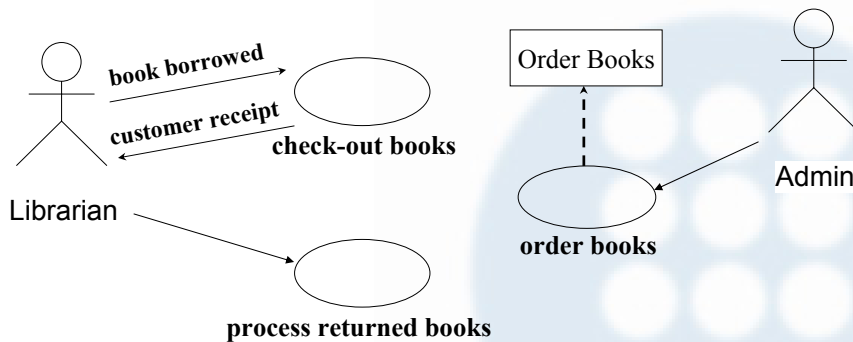
- Start with high-level description of the system (textual)
- Identify actors
- Identify top-level scenarios for each actor
- Identify inputs/outputs (actions/percepts)

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!)

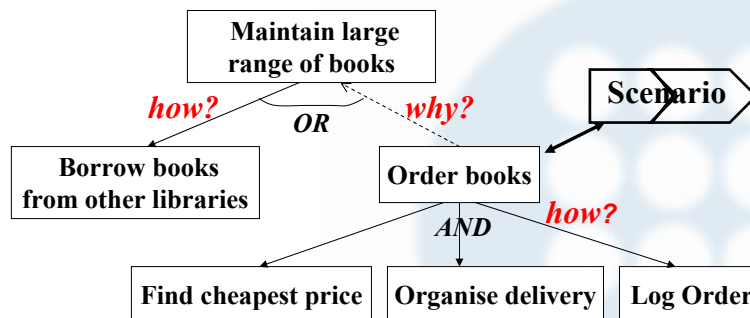
- Add a corresponding system goal for each use-case



Prometheus

System Specification phase: Steps (non-sequential!)

- Apply Goal Abstraction to system goals
- Refine Goal (OR/AND refinement)
- Link goals to (sub)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: CustomerArrived (message), CustomerInformation (message)

Messages sent: CustomerInformationRequest (message), CustomisedWWPage (message),

Actions: DisplayCustomisedWWPage

Data used: CustomerDB, CustomerOrders

Interactions: CustomerManager (via CustomerInformationRequest, CustomerInformation) OnlineInteraction (via CustomisedWWPage, CustomerArrived)

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.

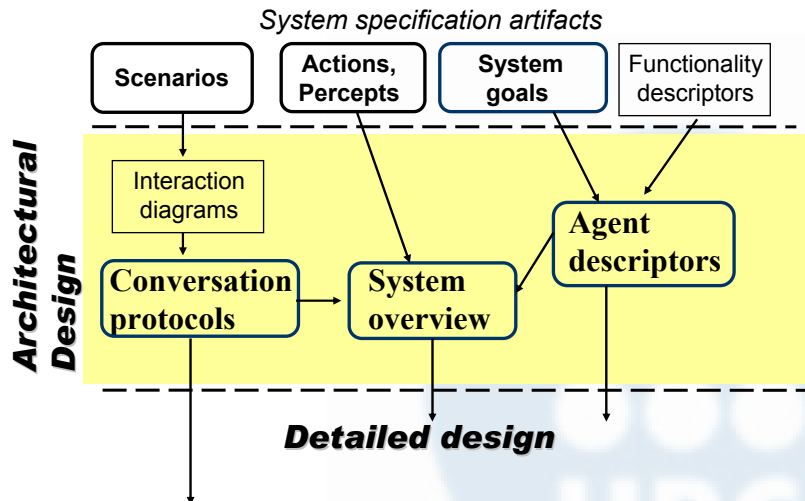
Steps:

1. EVENT BookOrder (→ Online Interaction)
2. DeliveryOptionQuery (Online Interaction → Transport Information)
3. DeliveryOptions (Transport Information → Online Interaction) Data read: Transport DB
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 EmailCourierCompany (Order Handling)
10. DecreaseStock (Order Handling → Stock Manager)

Variations: steps 9 (email courier) 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



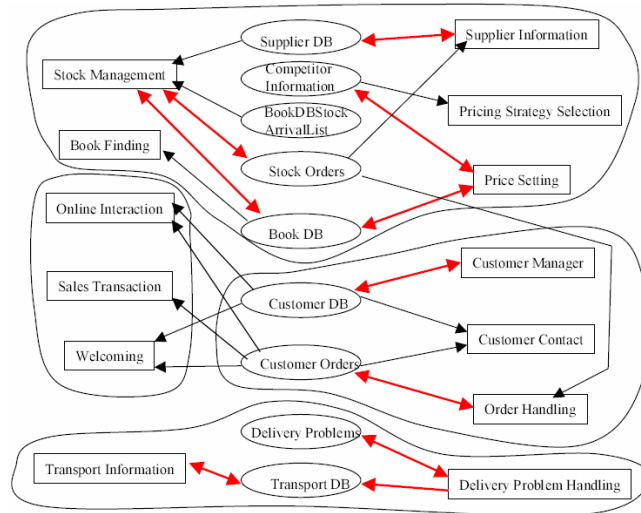
Prometheus

Architectural Design Phase: Agent types

- Identify the **agent types** in the system
 - Group functionalities to agent types based on cohesion and coupling
 - Group 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



jvazquez@lsi.upc.edu

17

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?

jvazquez@lsi.upc.edu

18

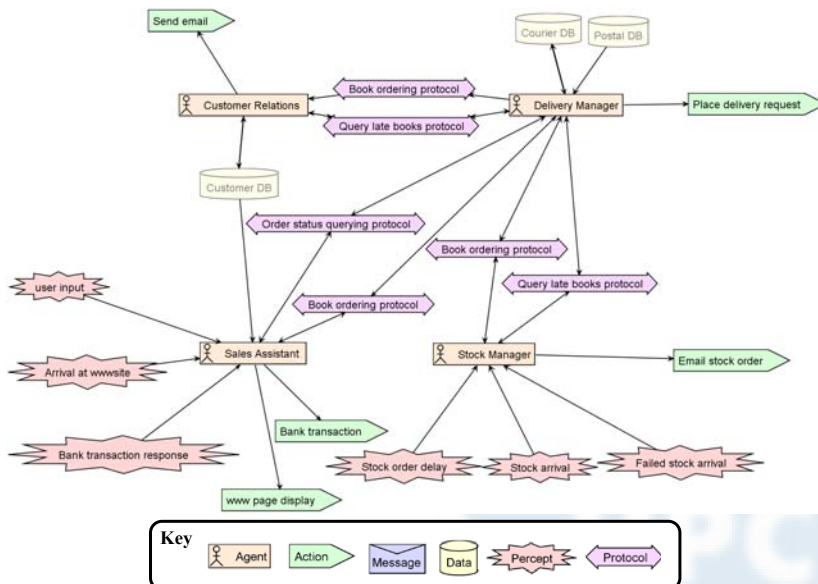
Prometheus

Architectural Design Phase: Agent Descriptors

Name: Sales Assistant agent
Description: greets customer, follows through site, assists with finding books
Cardinality: one/customer.
Lifetime: Instantiated on customer arrival at site. Demise when customer logs out or after inactivity period.
Initialisation: Obtains cookie. Reads Customer DB.
Demise: Closes open DB connections.
Functionalities included: Online Interaction, Sales Transaction, Welcomer, Book Finder.
Uses data: Customer DB, Customer 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)

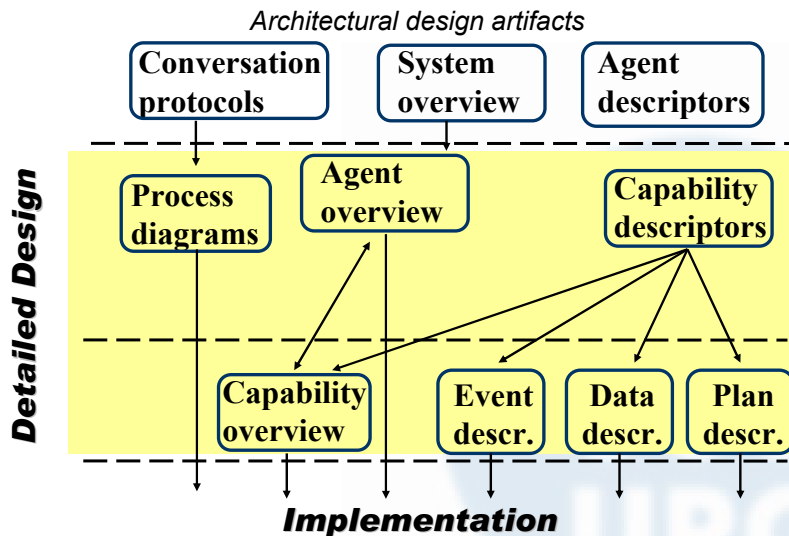
Prometheus

Architectural Design Phase: System Overview Diagram



Prometheus

Detailed Design Phase



jvazquez@lsi.upc.edu

21

Prometheus

Detailed Design Phase

- The details of the agent internals are developed
 - Defined in terms of capabilities, data, events and plans
 - Process diagrams are used as stepping stone between interaction protocols and plans
- 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

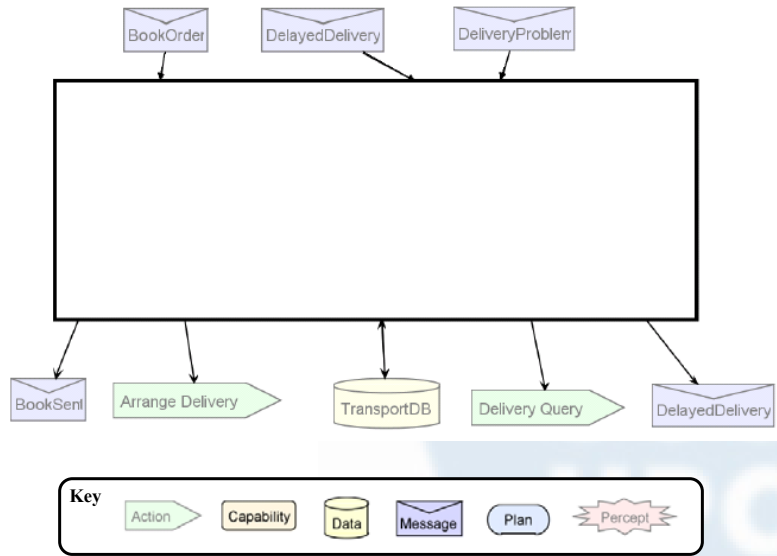
- Generate *agent overview diagrams*

jvazquez@lsi.upc.edu

22

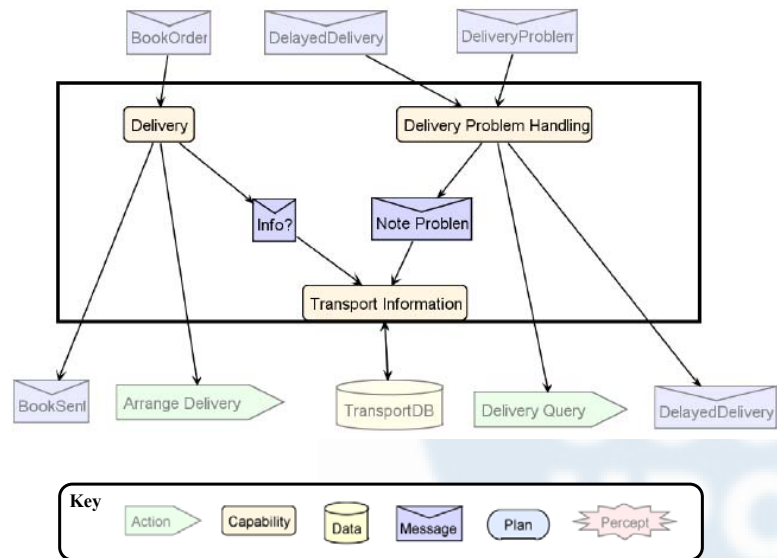
Prometheus

Detailed Design Phase: Agent Overview Diagrams



Prometheus

Detailed Design Phase: Agent Overview Diagrams



Prometheus

Detailed Design Phase: Event, Data & Plan Descriptions

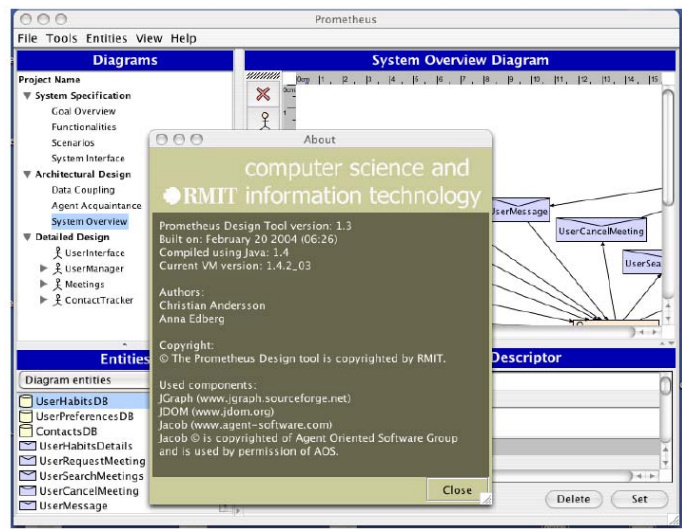
- Steps (II)
 - *Plan descriptions*

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

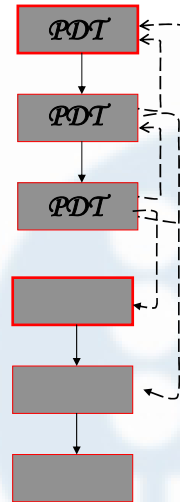
Tools: the Prometheus Design Tool (PDT)



Prometheus

Tools: the Prometheus Design Tool (PDT)

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

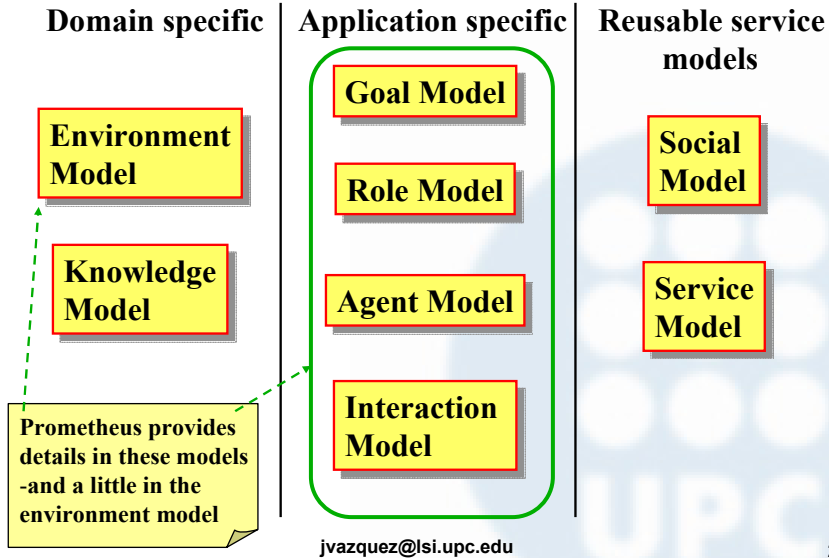


From Prometheus to ROADMAP

- Main strengths:
 - Structured processes to refine design.
 - Automated consistency checking between (some of) the design artefacts.
 - Hierarchical and modular views.
- Actively continuing development...
- Next step: the **ROADMAP methodology**
 - More abstract and high level than Prometheus.
 - Concerned with high level view of models needed.
 - Focusses particularly on requirements analysis.
 - Notation for new elements still being defined.

ROADMAP

Overview of Models



29

ROADMAP

Models

- **Goal hierarchy** (Requirements, propagates down)
- **Roles** associated with goals (Requirements)
- **Interaction model:**
 - Scenarios (Requirements).
 - Protocols (Architectural design)

Requiring more work:

- **Knowledge Model**
- **Environment Model**
- Possibly need a **Task Model**
- **Social Model**
- **Services Model**

jvazquez@lsi.upc.edu

30

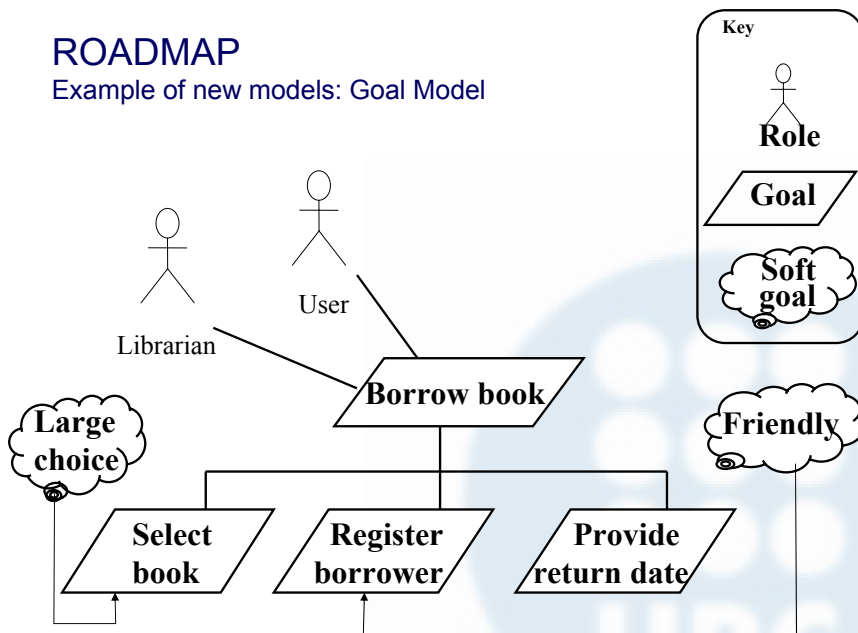
ROADMAP

Integration with Prometheus

- 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
- Still need to decide common notation

ROADMAP

Example of new models: Goal Model



References

- [1] N.R. Jennings, “On Agent-Based Software Engineering”, Artificial Intelligence, 117:227-296, 2000.
- [2] F. Zambonelli, N. Jennings, M. Wooldridge, “Organizational Abstractions for the Analysis and Design”, 1st International Workshop on Agent-oriented Software Engineering, LNAI No. 1957, 2001.
- [3] O. Shehory and A. Sturm, “Evaluation of Modelling Techniques for Agent-Based Systems”, Proceedings of The Fifth International Conference on Autonomous Agents, pp. 624-631, 2001.
- [4] L. Padgham, M. Winikoff. “Prometheus: A methodology for developing intelligent agents”. In Third Int. Workshop on agent-Oriented Software Engineering, July 2002.
- [5] L. Padgham, M. Winikoff. “Prometheus: A pragmatic methodology for engineering intelligent agents”. In proc. of the OOPSLA 2002 Workshop on Agent-Oriented Methodologies, page 97-108, Seattle, 2002.