

# Modeling Notation Source

## Prometheus

Version: 03-04-02

Document author: Radovan Cervenka

### 1 Introduction

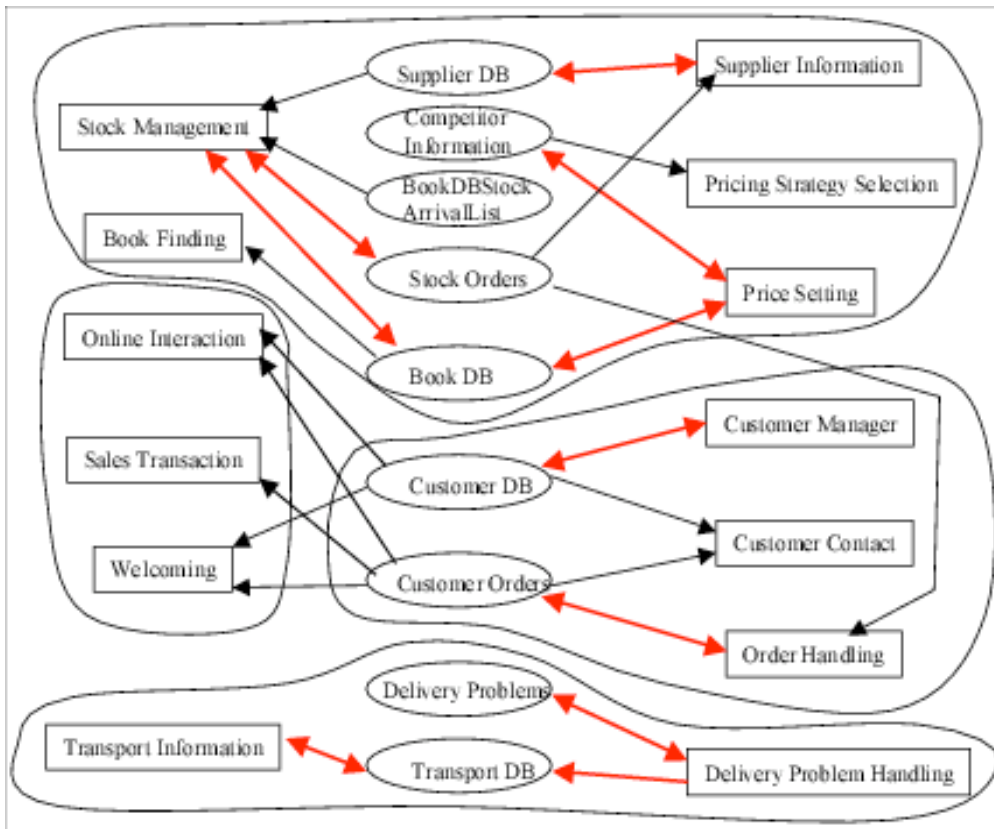
*Prometheus* [PW02A] [PW02B] [Prom] is a methodology for developing intelligent agent systems. It defines a modeling language that is generic to any MAS architecture and implementation environment, even if it has mostly been used in designing BDI agent systems, as the authors says.

The Prometheus methodology has been developed in collaboration with Agent Oriented Software, and a modified version of the Prometheus modeling language has been partially implemented in their JACK Intelligent Agents™ development environment [JACK], as a tool for visual modeling of the architectural design and plans.

### 2 Notation Overview

#### 2.1 Textual Descriptors

Prometheus uses structured textual descriptions to specify details of particular modeling elements. There are defined descriptor forms e.g. for list of goals, functionalities, scenarios, actions, percepts, agents, etc. For detailed description and examples see [PW02A].



**Figure 1:** Example of data coupling diagram

## 2.2 Data Coupling Diagram

One technique that is used to systematically examine the properties which lead to coupling is the *data coupling diagram*. A data coupling diagram consists of the functionalities and all identified data (not only persistent data, but also data the functionalities require to fulfill their job). Rectangles are used to depict functionalities and ovals are used to depict data. Directed links are inserted between functionalities and data, where an arrow pointing towards the data indicates the data is produced or written by that functionality, whereas an arrow pointing towards the functionality indicates the data is used by the functionality. A double-headed arrow indicates that the functionality both uses and produces the data. An example data coupling diagram is shown in Figure 1.

## 2.3 Agent Acquaintance Diagram

*Agent acquaintance diagram* depicts interconnection of different agent types. The agent acquaintance diagram is simplified UML class diagram, where agents are represented by classes (having just name compartments) and agent connections are represented by binary associations with multiplicities.

## 2.4 System Overview Diagram

The *system overview diagram* ties together the agents, events and shared data objects. It uses the notation depicted in Figure 2. Example of system overview diagrams is shown in Figure 3.

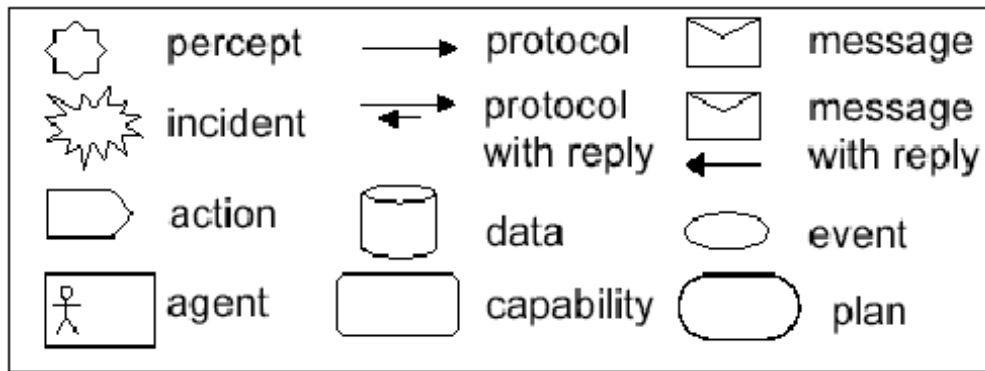


Figure 2: System overview diagram notation

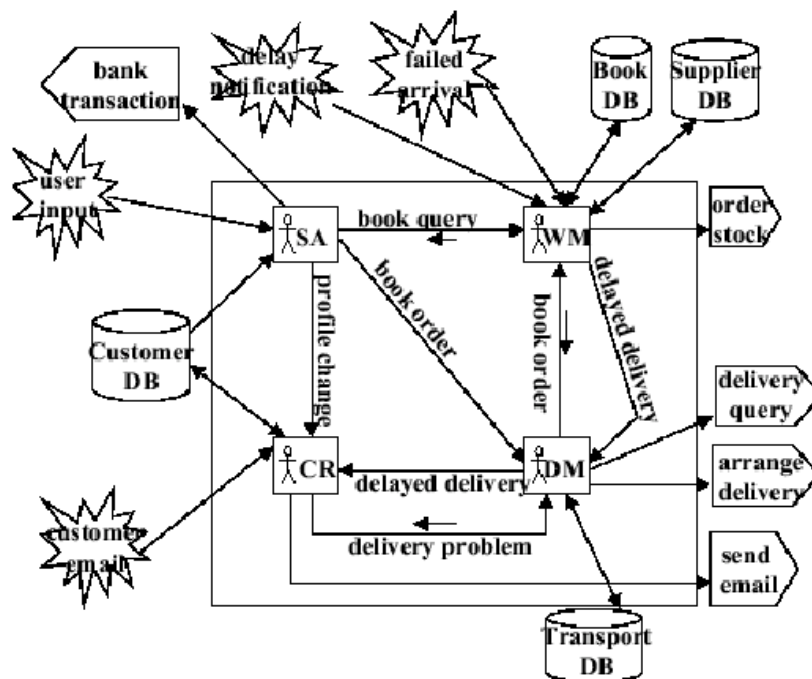


Figure 3: Example of system overview diagram

## 2.5 Interaction Diagrams and Protocols

*Interaction diagrams* shows interactions between agents. Prometheus exploits UML sequence diagrams to model interactions and AUML sequence diagrams to model interaction protocols.

## 2.6 Agent Overview Diagram

An *agent overview diagram* is quite similar to the system overview diagram, but it shows interactions between capabilities within an agent, rather than between agents within a system. Any messages or percepts that are incoming to an agent in the system overview, must be incoming to some capability (or plan) within that agent in the capability overview diagram. Similarly any actions or messages that are outgoing from an agent in the system overview, must be outgoing from some capability (or plan) within that agent in the capability overview diagram.

## 2.7 Capability Overview Diagram

A *capability overview diagram* is again similar to the system overview and agent overview diagrams, and has the same constraints about matching incoming and outgoing “events” and actions. The interactions at this level are between plans and nested capabilities.

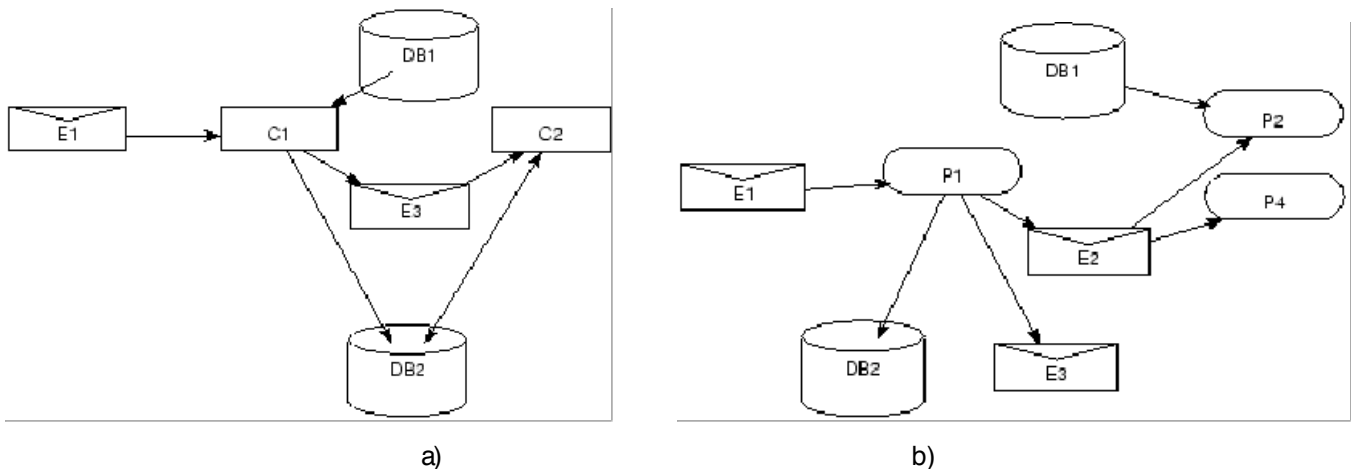


Figure 3: Example of a) agent overview diagram, and b) capability overview diagram

## 3 Unification Considerations

Unfortunately, the modeling language used by Prometheus has not been properly documented yet (i.e. meta-model or formal specifications are missing). Therefore users would have problems with understanding of precise semantics as well as proper using of particular modeling constructs. There is not documented explicit traceability relationship between overview diagrams and interaction diagrams. Overview diagrams do not provide mechanisms for bound parameterized protocols [Odell2001]. Furthermore, Prometheus covers just some aspects of MAS, e.g. explicit modeling of goals and their relationships, detailed structure of data (like in UML class diagrams), internal plan behavior (even if JACK provides a kind of activity diagram, this technique is not described by the Prometheus methodology), modeling of environments, modeling of ontology, etc. are either missing or covered just partially.

However, Prometheus uses modeling elements that represent fundamental principles of agent-oriented paradigm and form a basis for not only BDI architectures. After proper defining of their semantics and notation, they should be integrated into FIPA AUML.

## 4 References

- [PW02A] Lin Padgham and Michael Winikoff, *Prometheus: A Pragmatic Methodology for Engineering Intelligent Agents*. In proceedings of the workshop on Agent-oriented methodologies at OOPSLA 2002. November 4, 2002, Seattle.  
<http://www.cs.rmit.edu.au/agents/Papers/oopsla02.pdf>
- [PW02B] Lin Padgham and Michael Winikoff. *Prometheus: A Methodology for Developing Intelligent Agents*. In proceedings of the Third International Workshop on Agent-Oriented Software Engineering, at AAMAS'02.  
<http://www.cs.rmit.edu.au/agents/Papers/aamas02-aose-ws.pdf>
- [Prom] Prometheus home page  
<http://www.cs.rmit.edu.au/agents/SAC/methodology.shtml>
- [JACK] JACK home page.  
<http://www.agent-software.com/shared/products/index.html>
- [Odell2001] Odell, James, Van Dyke Parunak, H. and Bauer, B., Representing Agent Interaction Protocols in UML. In: Agent-Oriented Software Engineering, Ciancarini, P. and Wooldridge, M., Eds., Springer, pp. 121-140, Berlin, 2001.  
<http://www.fipa.org/docs/input/f-in-00077/>