

# FOUNDATION FOR INTELLIGENT PHYSICAL AGENTS

## Method Fragment Definition

|                        |  |                            |                     |
|------------------------|--|----------------------------|---------------------|
| <b>Document title</b>  | Method Fragment Definition                                     |                            |                     |
| <b>Document number</b> | ????   | <b>Document source</b>     | FIPA Methodology TC |
| <b>Document status</b> | Preliminary  | <b>Date of this status</b> | 2003/11/21          |
| <b>Supersedes</b>      | None   |                            |                     |
| <b>Contact</b>         | <a href="mailto:methodology@fipa.org">methodology@fipa.org</a> |                            |                     |
| <b>Change history</b>  |  |                            |                     |
|                        |  |                            |                     |

© 2004 Foundation for Intelligent Physical Agents - <http://www.fipa.org/>

Geneva, Switzerland

### Notice

Use of the technologies described in this specification may infringe patents, copyrights or other intellectual property rights of FIPA Members and non-members. Nothing in this specification should be construed as granting permission to use any of the technologies described. Anyone planning to make use of technology covered by the intellectual property rights of others should first obtain permission from the holder(s) of the rights. FIPA strongly encourages anyone implementing any part of this specification to determine first whether part(s) sought to be implemented are covered by the intellectual property of others, and, if so, to obtain appropriate licenses or other permission from the holder(s) of such intellectual property prior to implementation. This specification is subject to change without notice. Neither FIPA nor any of its Members accept any responsibility whatsoever for damages or liability, direct or consequential, which may result from the use of this specification.

## Foreword

The Foundation for Intelligent Physical Agents (FIPA) is an international organization that is dedicated to promoting the industry of intelligent agents by openly developing specifications supporting interoperability among agents and agent-based applications. This occurs through open collaboration among its member organizations, which are companies and universities that are active in the field of agents. FIPA makes the results of its activities available to all interested parties and intends to contribute its results to the appropriate formal standards bodies.

The members of FIPA are individually and collectively committed to open competition in the development of agent-based applications, services and equipment. Membership in FIPA is open to any corporation and individual firm, partnership, governmental body or international organization without restriction. In particular, members are not bound to implement or use specific agent-based standards, recommendations and FIPA specifications by virtue of their participation in FIPA.

The FIPA specifications are developed through direct involvement of the FIPA membership. The status of a specification can be either Preliminary, Experimental, Standard, Deprecated or Obsolete. More detail about the process of specification may be found in the FIPA Procedures for Technical Work. A complete overview of the FIPA specifications and their current status may be found in the FIPA List of Specifications. A list of terms and abbreviations used in the FIPA specifications may be found in the FIPA Glossary.

FIPA is a non-profit association registered in Geneva, Switzerland. As of January 2000, the 56 members of FIPA represented 17 countries worldwide. Further information about FIPA as an organization, membership information, FIPA specifications and upcoming meetings may be found at <http://www.fipa.org/>.

## Contents

|  |   |
|--|---|
| 1 Scope.....                                   | 1 |
| 2 Method Fragment Definition.....              | 2 |
| 3 Example of Method Fragment.....              | 3 |
| 3.1 Process.....                               | 3 |
| 3.1.1 Notation.....                            | 5 |
| 3.2 Deliverables.....                          | 5 |
| 3.3 Preconditions.....                         | 6 |
| 3.4 Relationships with the MAS meta-model..... | 7 |
| 3.5 Guidelines.....                            | 7 |
| 3.6 Glossary.....                              | 7 |
| 3.7 Composition guidelines.....                | 8 |
| 3.8 Aspects of fragment.....                   | 8 |
| 3.9 Relationships with other fragments.....    | 8 |
| 3.10 Summary .....                             | 8 |
| References.....                                | 9 |



## 1 Scope

Existing development methodologies have different advantages when applied to specific problems. It is therefore possible to think that the developer of a MAS would like to use phases or models or elements coming from different methodologies in order to build up a personalized approach for his own problem.

In order to reuse contributions coming from existing methodologies we will adopt the method engineering as the referring paradigm [BRI96], [SAE94], [TOLV]. In this context the development methodology is constructed by the developer assembling pieces of the process (method fragments) from a method base. In this way he could obtain the best process for his specific need/problem.

We will take method fragments (the composing elements of the development methodology) from several existing methodologies, if these fragments are coherent with the FIPA architecture. Other novel and specific contributions will be considered as well.

The first step in this work consisted in the identification of the meta-model that could be used to describe the existing methodologies and the multi-agent system structure (MAS meta-model). An important contribution to this approach came from an OMG specification, the Software Process Engineering Metamodel [SPEM]. This was the natural candidate to be the meta-model adopted for describing FIPA processes since it has been already an accepted standard in the OO context (and OO process are not too different from the AO ones).

Some existing methodologies have been described with this SPEM and method fragments will be extracted from the descriptions of these processes. A method fragment is a reusable part of a design process that taking some already designed pieces of the system produces a new part of the design following a precise procedure. These fragments will be collected in a method base. This introduces another step of the methodology TC plan: the study of possible technological solutions for the implementation of this database in order to obtain a representation of the fragments that could be easily supported in a CASE/CAME (Computer Aided Software Engineering/Computer Aided Method Engineering) tool.

The last crucial phase of the work will be the study of the method fragments composition strategies. Each method fragment produces an artifact that contributes to the construction of the complete MAS design model.

Composing fragments coming from different methodologies implies considering that they may refer to different models of the system. For example they could address the concept of role in a slightly different way and as a consequence, reusing roles defined with one methodology in another context could bring to inconsistent or incomplete models. Anyway, this artifact structure is only one of the aspects of the problem we are dealing with. There is also a procedural point of view. In taking two method fragments from a repository and reusing them, the designer could find that they do not exactly match. It could be necessary to integrate them with some more activities that should complete the process.

## 2 Method Fragment Definition

A fragment is a portion of the development process, composed as follows:

1. A portion of process (what is to be done, in what order), defined with a SPEM diagram
2. One or more deliverables (artifacts like (A)UML/UML diagrams, text documents and so on). The result of the work could also be some kind of product/artifact that is not be delivered to anyone outside the development process. It includes a reference to a recommended notation/language/structure to be used to represent AUML/UML/other diagrams if they are part of the deliverables (as it is common)
3. Some preconditions (they are a kind of constraint because it is not possible to start the process specified in the fragment without the required input data or without verifying the required guard condition)
4. A list of concepts (related to the MAS meta-model) to be defined (designed) or refined during the specified process fragment.
5. Guideline(s) that illustrates how to apply the fragment and best practices related to that
6. A glossary of terms used in the fragment (in order to avoid misunderstandings if the fragment is reused in a context that is different from the original one)
7. Composition guidelines – A description of the context/ problem that is behind the methodology from which the specific fragment is extracted.
8. Aspects of fragment. Textual description of specific issues like for example: platform to be used, application area,...
9. Dependency relationships useful to assemble fragments

It should be noted that not all of these elements are mandatory. Some of them (for instance notation, guideline or inputs) could be not applicable or not necessary for some specific fragment.

The fragment refers to a MAS meta-model and its aim is to contribute in increasing the definition of the instance of this meta-model that will solve the problem the designer is facing

### 3 Example of Method Fragment

In the following, an example of method fragment will be proposed, extracting it from the PASSI methodology [PASSI]. Not all the aspects of the fragment will be discussed but only the most important ones in order to provide a quick introduction to the concept of fragment and its description.

#### 3.1 Process

Consider the PASSI process represented in fig . 1. and particularly the Agent Society phase.

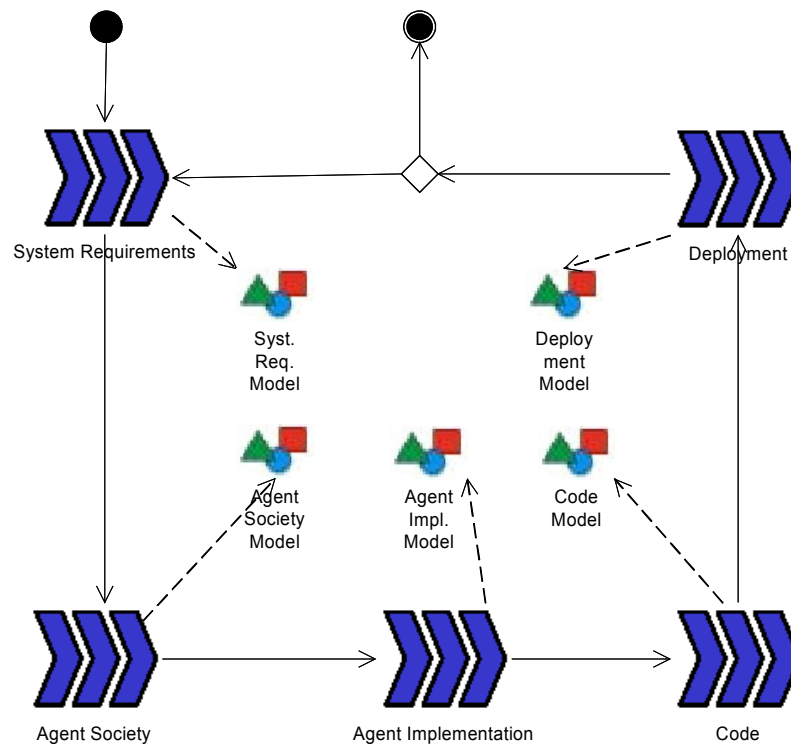


Fig. 1 The complete PASSI process

Within this phase we perform the activities specified in fig .2. In this context we can identify our example fragment (red oval).

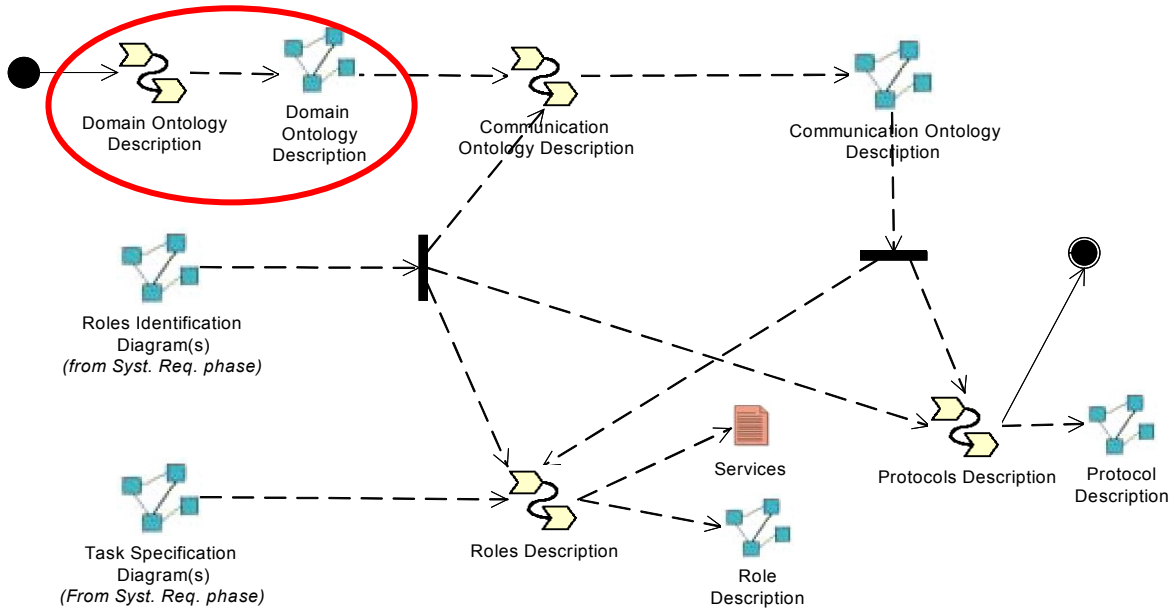


Fig.2: Activities of the Agent Society phase

Consider the work definition “Domain Ontology Description” and the consequent outcome (UML model “Domain Ontology Description”). This is a fragment whose aim is to design the ontology of the system.

The process that is to be performed in order to obtain the result is represented in fig. 4 as a SPEM activity diagram.

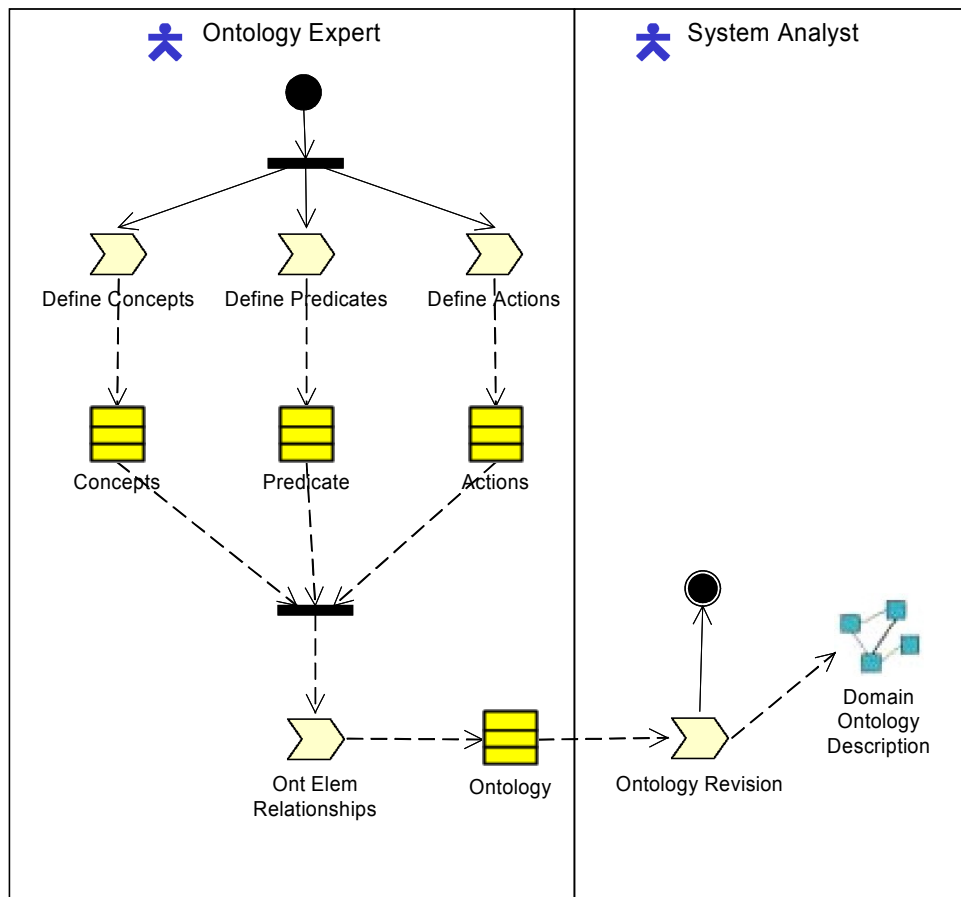


Fig.3 The Domain Ontology Description Fragment – Procedural aspects



### 3.1.1 Notation

The UML model “Domain Ontology Description” is designed according to a specific notation described below:

The ontology is described (using a class diagram) in terms of concepts (fill colour: yellow), predicates (fill colour: light blue) and actions (fill colour: white).

Elements of the ontology can be related using three UML standard relationships:

- Generalization: it permits the generalize/specialization relation between two entities that is one of the fundamental operator for constructing an ontology.
- Association: it models the existence of some kind of logical relationship between two entities. It is possible to specify the role of the involved entities in order to clarify the structure.
- Aggregation: it can be used to construct sets where value restrictions can be explicitly specified; in the W3C RDF standard [6] three types of container objects are enumerated: the bag (an unordered list of resources), the sequence (an ordered list of resources) and the alternative (a list of alternative values of a property). We choose of considering a bag as an aggregation without an explicit restriction, a sequence is qualified by the *ordered* attribute while the alternative is identified with the *only one* attribute of the relationship.

An example is:

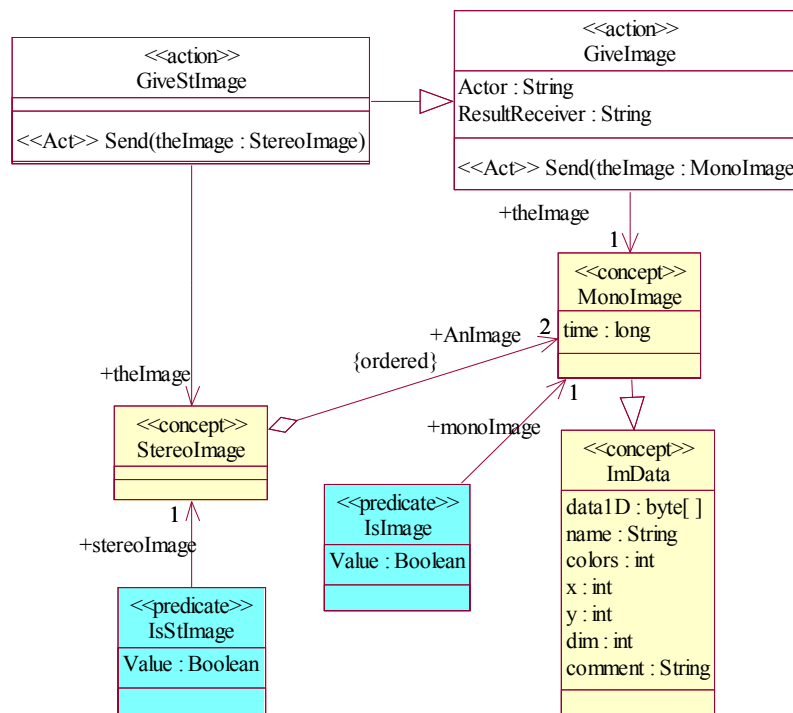



Fig. 4. An example of domain ontology diagram in PASSI

### 3.2 Deliverables

This fragment produces a class diagram whose classes represent concepts, actions and predicates with the following details:

- Concepts are described in terms of their attributes,
- Predicates report a value specified according to the chosen notation,

Actions have an Actor (that is responsible to complete the work), a ResultReceiver (that is to be notified of the action results) and an Act that describes the action to be done with the required input and prescribed outcome.

The overall output of this fragment is the application system domain ontology (considered as the aggregation of involved concepts, predicates and actions). The following figure illustrates the relationship between this fragment outcome and the MAS meta-model.  
 (The symbol:  represents an element of the MAS model)

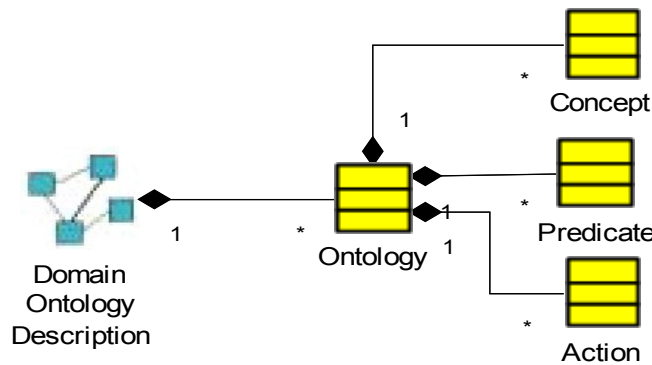


Fig.5. Structure of the DOD fragment work-product with respect to the MAS meta-model

Information described in the class diagram is (optionally) completed by a text document reporting the following data for each element (concept, action, predicate) of the ontology:

| Concept   | Description |             |
|-----------|-------------|-------------|
| Attribute | Type        | Description |
|           |             |             |

| Predicate | Return Type | Description |
|-----------|-------------|-------------|
|           |             |             |

| Action         | Description |             |
|----------------|-------------|-------------|
| Actor          | Description |             |
| ResultReceiver | Description |             |
| Parameter      | Type        | Description |
| Result         | Type        | Description |
|                |             |             |

### 3.3 Preconditions

Inputs, output and elements to be designed in the fragment are detailed in the following table.

| Input | To Be Designed | Output |
|-------|----------------|--------|
|       |                |        |

|                              |                        |                                     |
|------------------------------|------------------------|-------------------------------------|
| System Requirements document | Concepts               | Ontology (MAS Meta-model component) |
| Glossary                     | Actions                |                                     |
|                              | Predicates             |                                     |
|                              | Ontology Relationships | elements                            |

As it is obvious, in order to design an ontology we need to know the application domain described in the System Requirements document and we need a glossary of terms. The ontology expert (whose contribution is described in fig.3) is supposed to be an expert of the domain who can produce a formal representation of its categories according to the already described notation)

This fragment is suitable for describing ontology in an RDF-like way [RDF, RDF2]. If the desired outcome should be expressed in a different structure, this could not be the appropriate method to obtain it

### 3.4 Relationships with the MAS meta-model

The final result is the ontology that is an element of the MAS metamodel (see fig. 6)

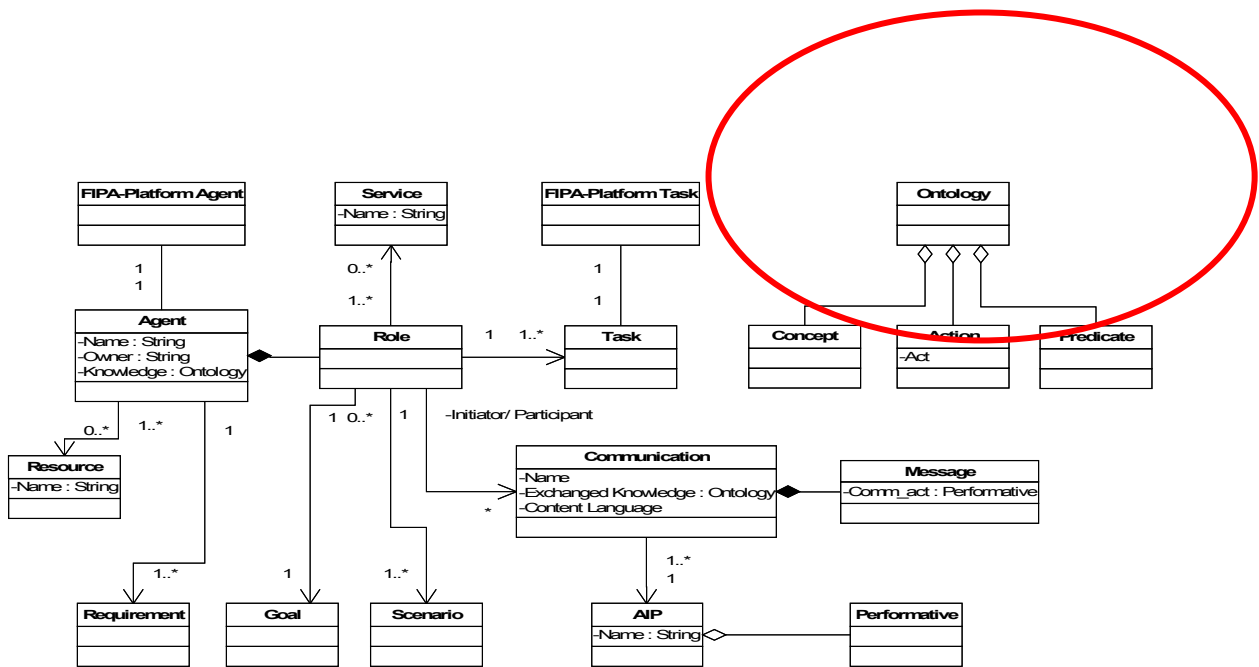


Fig.6. The MAS meta-model adopted in PASSI includes an ontology composed of concepts, predicates and actions

### 3.5 Guidelines

- 1) Consider that the suggested structure is based on the RDF specification [RDF, RDF2] and therefore different approaches are not recommended
- 2) The strategy of identification of the ontology elements is out of the scope of this fragment and it is left to the ontology expert

### 3.6 Glossary

Domain Ontology Description Fragment uses this list of model element:

**Agent** – an autonomous entity that is composed by roles and has a knowledge. An agent can be seen from different level of abstraction. In this fragment agents are a logical aggregation of functionalities (Use Case diagrams).

In general in PASSI, an agent is a significant software unit at both the abstract and concrete levels of design. According to this view, an agent is an instance of an agent class. So it is the software implementation of an autonomous entity capable of going after an objective through its autonomous decisions, actions and social relationships. An agent may undertake several functional roles during interactions with other agents to achieve its goals. A role is a collection of tasks performed by the agent in pursuing a sub-goal. A task, in turn, is defined as a purposeful unit of individual or interactive behaviour.

**Ontology** – An ontology is composed of concepts, actions and predicates.

**Concept** - Description of a certain identifiable entity of the domain

**Action** – It expresses an activity, carried out by an agent.

**Predicate** – Description of a property of an entity of the domain

### 3.7 Composition guidelines

The only inputs for this fragment are the system requirement descriptions (text document) and the glossary of terms; as a consequence, this fragment could be reused in almost all the stages of a design methodology, its aim is providing a description of the ontology elements that could be used for defining agents' communications and other knowledge related issues.

### 3.8 Aspects of fragment

This fragment is conceived to produce (possibly with the support of an automatic code generation tool) an RDF description of ontology categories. This makes it enough general but it could not be appropriate in some conditions.

Ontology designed with this fragment is supposed to be 'static'. It supports some kind of a-priori (design-time) ontology and no type of dynamic discovery (at run-time) of new categories/relationships.

### 3.9 Relationships with other fragments

None

### 3.10 Summary

The DOD fragment can be used to formalize the application domain ontology. It deals with a description of the domain ontology in terms of concepts, predicates and actions in terms that are inspired by the RDF syntax.

## References

- [BRI96] S. Brinkkemper. Method engineering : engineering of information systems development methods and tools. *Information and Software Technology* 38 (1996) 275-280.
- [PASSI] SPEM description of the PASSI process. ICAR-CNR Technical report. Dec. 2003.  
<http://www.csai.unipa.it/passi>
- [SAE94] Motoshi Saeki. Software Specification & Design Methods and Method Engineering. *International Journal of Software Engineering and Knowledge Engineering*. 1994
- [SPEM] Software Process Engineering Metamodel Version 1.0. OMG Document.  
<http://www.omg.org/technology/documents/formal/spem.htm>
- [RDF] W3C Resource Description Framework (RDF) specification. Online at:  
<http://www.w3.org/RDF/>
- [RDF2] FIPA RDF Content Language Specification, FIPA document n.00011, available online at:  
<http://www.fipa.org/specs/fipa00011/>
- [TOLV] Incremental Method Engineering with Modeling Tools: Theoretical Principles and Empirical Evidence. PhD Thesis of Juha-Pekka Tolvanen. University of Jyväskylä. Available online at:  
<http://www.cs.jyu.fi/~jpt/doc/index.html>