IEEE FIPA Design Process Documentation and Fragmentation Working Group

# Fragment Definition and Documentation Template

Author(s): M. Cossentino, V. Seidita
Contributors:

Last saved on: 10/12/10 17:24

# Index

# 1. Revision History

03-12-2010: First initial draft by M. Cossentino and V Seidita. Status: No fragment definition introduced so far. Complete description of the fragment documentation template section in terms of goal and structure of each subsection.

# 2. Problem / Motivation

# 3. Introduction to the specification

## *3.1. Scope*

## *3.2. Assumptions*

This document assumes several underlying ideas, which are fundamental for the understanding of the proposal. We try to make them explicit in this section.

The first assumption concerns the way of layering the design process representation. The work to be done in the process is supposed to be divided into three main levels: phase, activity and task.
Phases are composed of activities, which in turn are composed of other activities or individual, and atomic tasks. This is only a simplification used for allowing an easy catching of the correct abstraction level when documenting the process.
From a work product point of view, phases are supposed to deliver a major artefact (for instance a requirement analysis document or a detailed design document). Activities are supposed to produce finer grained artefacts (like a diagram possibly complemented by a text description of the elements reported within it). Tasks are supposed to concur to the definition of activity-level artefacts.
Such a classification is not too tight and although useful for aligning the description of very different processes, it is still open enough to accommodate all needs.
Notes:

- ADD explicit reference to process documentation template specs
- Mention the role of MAS metamodel in the proposed approach

## *3.3. Notation*

In this specification, notation is not considered fundamental, although the use of standards is important. In particular, SPEM 2.0 is suggested for modelling some process aspects.
Because of agent-oriented specific needs, some SPEM extensions are also proposed along with a few new diagrams besides the SPEM ones.
In any case, this does not mean that other standards cannot be used with the template as far as the concepts implied and the underlying view of the system proposed by the work product is reflected in the notation used.
Neither specification nor suggestion is provided in this document about the modelling notation to be adopted by the documented design process. Its workflow will produce documents, diagrams and other artefacts according to the notation preferred by its designer. What is strongly advised is to think the system modelling notation as one of the

possible notations to be adopted in the process and to separate its description from the description of the work product where it is adopted.

### *3.4. Dependencies and references to other Standards*

# 4. Fragment definition

A *process fragment* is a portion of design process adequately created and structured for being reused during the composition of new design processes both in the field of agent oriented software engineering and in other ones (model driven engineering-based approaches are preferred fields of application for the proposed definition). The process fragment is, generally (this is the most common case), extracted from an existing design process and it is stored in a repository. Two things to be noted at this point: the process fragment definition together with the specific SME process (see for instance [5]) used for retrieving and composing fragments notably influence how the repository is conceived and constructed. Conversely, constructing from scratch a process fragment can be done in the same way a design process is composed using the SME approach

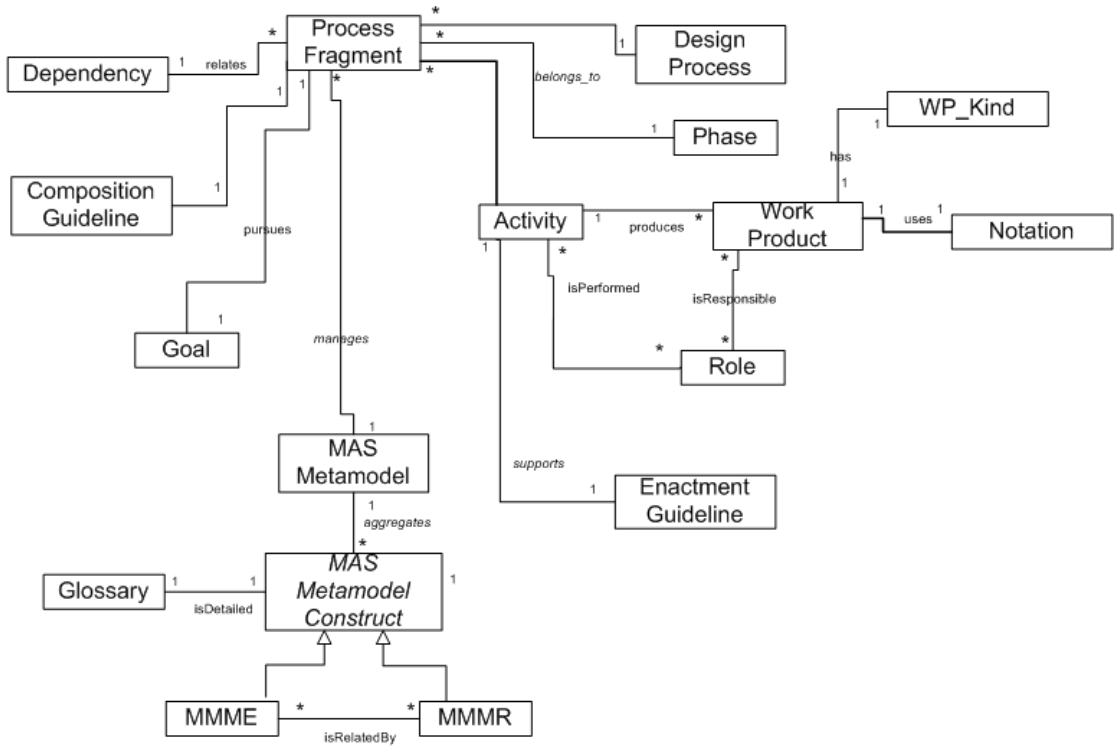



**Figure 1. The Process Fragment Metamodel.**

Figure 1 shows the metamodel of the process fragment proposed in this paper, it contains all the elements useful for representing and documenting the fragment under the process, product and reuse point of view; the proposed fragment documentation template slavishly follows the proposed metamodel, its elements and their definition.

The root element, the *Fragment*, has been generally extracted from an existing design process, therefore an important information to be stored in the repository is the *Design Process* the fragment refers to, this serves for the designer to set the application context and the particular features the fragment would exhibit.

The process fragment is composed of activities, each of them is a portion of work that has to be performed by one or more stakeholders (*Process Roles*) and can be decomposable in other activities or can be atomic in the sense that it is a single design action performed by

only one process role. It is also important to indicate if the process role, while performing the activity, is the main performer, the main responsible, or he is assisting another role.

Design process usually is decomposed into phases, let us think for instance to the Unified Process (UP) [4], it presents different iterations during which Analysis, Design etc. phases are repeated with different levels of details and each phase is characterized by the fact that the activities here performed have a common scope, specific deadlines and constraints: for instance, Design phase cannot start if Analysis phase is not finished. For that reason we need to identify the *Phase* the fragment refers to. In order to make easy storing and reusing fragments we identified in the past a taxonomy [6] for classifying phases, process roles and work products.

Activity delivers *Work Products*, where the results of design activities are drawn by using a specific *Notation* and each work product is developed under the responsibility of one process role. The notation to be used greatly influences the flow of work to be done for producing a work product and for this reason a fragment has to be supplied with a set of *Guidelines*. It is not mandatory to follow a specific notation, the same kind of diagram (for instance a structural one) may be expressed by using different notations without significant differences in the resulting expressiveness. Moreover, different kinds (*WP_Kind*) of work products can be delivered, we identified two main work product kinds: graphical and textual, the former when an activity results in a diagram the second when designers produce textual documents.
Finally a work product can be of composite kind if it is a composition of the previous said kinds, for instance a document with a diagram and the text explaining it [7].
As well as in the design process definition, one of the most important elements in the fragment definition is the *MAS Metamodel* (Multi-Agent System Metamodel); each fragment underpins a metamodel that is obviously part of the metamodel of the design process it comes from. The metamodel contains the set of elements representing the system to be designed using a specific process fragment. In the case of fragment definition we have to consider that MAS metamodel are composed of elements (*MMME* - the concepts to be designed) or relationships among them (*MMMR*).

The main aim of process fragment is to instantiate one or more MAS metamodel elements, one fragment should at least instantiate one MMME/MMMR and in so doing it may be requested to define relationships with other elements or to quote other elements and/or relationships; besides the result of defining an element or a relationship might be the refinement of existing elements or relationships. This fact led to the definition of the kinds of action to be done on a MAS metamodel element (see the following section for details). Finally the MAS metamodel element has a definition to be listed in a glossary; the definition is mainly useful during selection when the method designer must know which kind of MAS metamodel element better fits with the MAS metamodel element s/he is dealing with.

Until now the process and product part of the fragment metamodel has been explored though a set of elements that has to be necessarily present in the fragment documentation, now let us focus on the elements that principally deal with the reuse aspect of the fragment: *Goal*, *Dependency* and *Composition Guideline*.

The fragment goal is the objective the process part of the fragment wants to pursue and it is to be used during fragment selection from the repository. For this reason it is related to the new design process requirements, in other words, a goal describes the contribution a fragment may give to the accomplishment of some design process requirements. The

dependency aims at describing specific constraints, if they exist, for the fragment to be composed with other ones, for instance, there can be fragments dealing with MAS metamodel elements that are very specific to particular application domains, in this case it should be possible that such fragments can be composed with fragments coming from the same classes of design processes.
Figure 2 resumes the process fragment metamodel elements and their definitions.

| **THE FRAGMENT METAMODEL** | |
|---|---|
| ***Name*** | ***Description*** |
| Activity | A portion of work assignable to a performer (role). An activity may be atomic (sometimes addressed as Action) or composed by other activities. |
| Composition Guideline | A set of guidelines for assembling/composing the fragments with others. This may include notational specifications, and constraints (also addressing issues like platform to be used for system implementation and application area) |
| Work Product | The resulting product of the work done in the fragment; it can be realized in different ways (diagram, text,..) also depending on the specific adopted notation. |
| Work Product Kind | Represents the specific kind one work product can be; it strictly depends on the means the adopted notation provides. One work product can be |
| Dependency | The description of specific dependencies of this fragment from other ones; it is useful for composition. |
| Design Process | It is the design process from which the fragment has been extracted. |
| Glossary | A list of definitions for the mas metamodel constructs. |
| Goal | The process-oriented objective of the fragment. |
| Phase | A specification of the fragment position in the design workflow. Usually referring to a taxonomy (i.e. Requirements elicitation, Analysis, Design, etc.) |
| Enactment Guideline | The description of how to perform the prescribed activity. This may include best practices and specific techniques for achieving the expected results. |
| MAS Metamodel | The structure of concepts underpinned by the fragment. |
| MAS Metamodel Construct | *(abstract class)* The main constructs the fragments deals with, both in terms of outputs and inputs, for instance a fragment aiming at defining the system requirements has to define and to identify the concept of "requirements". It stands for Multi agent system metamodel element and relationship and represents the items the fragment's metamodel is composed of. Each metamodel construct has to be defined during, at least, one portion of process work and has to appear in at least one work product. |
| MMME/MMMR | A MAS metamodel construct can be an element (MMME) or a relationship among elements (MMMR); both the two have to be designed in a portion of work and represented in a work product. |

| | |
|---|---|
| Notation | Each deliverable can be drawn by using a specific notation. Concepts dealt by the fragment have to find a mapping in the notation. Notation usually includes a metamodel and a set of pictorial prescriptions used to represent the instantiation of metamodel elements. |
| Role | The stakeholder performing the work in the process and responsible of producing a deliverable (or a part of it). Usually referring to a taxonomy (i.e. System Analyst, Test Designer, etc.) |

**Figure 2. Definitions of the Process Fragment Metamodel Elements**

## *4.1. Fragment Granularity*

Process fragments may be extracted from a design process on the base of different levels of granularity; three different levels have been identified: phase, composed and atomic.
Figure 3 shows the relationships among design process and fragments of different level of granularity.

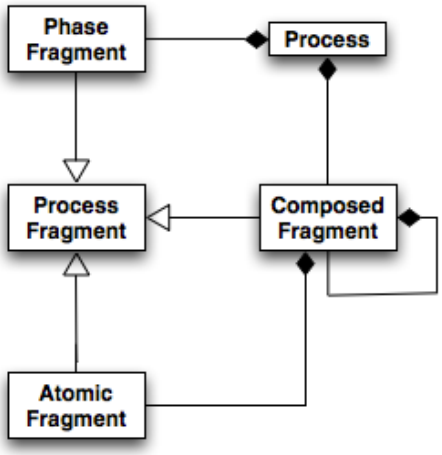



**Figure 3. Process Fragment Granularity**

### 4.1.1. Phase-level fragment

A phase-level (process) fragment delivers a set of work products related to the same abstraction level of the design flow.
Such work products may belong to any of the cited work product types.

#### 4.1.1.1. Example

An examples of deliverable may be a system analysis document.

### 4.1.2. Composed fragment

A composed (process) fragment delivers a work product. Such a work product may belong to any of the cited work product types.
Composed (process) fragments may be nested.
For instance a composed fragment delivering a composite work product may be composed by other composed (or atomic) fragments.

#### 4.1.2.1. Example

An example of composed fragment may consists in the portion of the UP process where the designer models use cases. This fragment delivers a composite work product (use case

diagrams and description text document) that is part of the System Analysis document produced by the System Analysis phase fragment.

### 4.1.3. Atomic fragment

An atomic (process) fragment delivers a portion of a work product or a set of system metamodel construct(s) (in terms of its/their instantiation or refinement).

A portion of a work product is here intended never to be a whole work product, in other words, step fragments never deliver entire work products.
A step fragment may also not deliver a portion of work product but rather it may deliver a system metamodel construct thus concurring to the instantiation/refinement of such a construct without reporting that in any artefact. The designer performing the corresponding work, just keeps in her/his mind the result of some brain work and will later depict it in some kind of work products while performing the work prescribed in another atomic fragment.

#### 4.1.3.1. Example

An example of atomic fragment may be the identification of actors to be used for modelling use cases by starting from the analysis of some text describing system functionalities.

# 5. Fragment documentation outline

The proposed documentation outline is composed of eight main sections: Description, System Metamodel, Stakeholders, Workflow, Deliverables, Guidelines, Glossary, References. This structure will be detailed in the following sections according to a specific format including (for each element of the process documentation template):

- **Goal** describing the goal addressed in this part of the document. Example of goals include the documentation of the general philosophy that is behind a fragment or the description of the involved stakeholders.
- **Structure** describing what is to be reported in this part of the process document. This may include diagrams as well as the textual description of specific fragment elements.
- **Guidelines** describing best practices suggested for a good application of the fragment documentation template or techniques about how to perform the prescribed work.
- **Example** addressing an existing example, possibly reported in this document.

# 6. Fragment Documentation Template

## *6.1. Description*

### 6.1.1. Fragment Goal

Goal: the aim of this section is to provide the reader with a quick understanding of the goal pursued by the fragment, possibly relating the description to common-sense in software engineering (i.e.: the aim of this fragment is collecting requirements in a text form)
Structure: Free text
Guidelines: --
Example: See Section …

### 6.1.2. Fragment Granularity

Goal
Structure
Guidelines
Example: See Section …

Granularity of this fragment: (Phase/Composed/Atomic)

#### 6.1.2.1. Composing fragments

This fragment is composed of the following fragments:

| Composing Fragment name | Granularity | Process of Origin |
|---|---|---|
| | | |

### 6.1.3. Fragment Origin

GOAL
This section aims at introducing the philosophy, basic ideas, scope, and limits of the process.

STRUCTURE
This section should discuss:

- concepts at the basis of the process
- a 'classic' figure of the process
- a quick description of the process (using the original process terminology if useful)
- scope of the process (kind of MAS, size, architecture, type of problems, Implementation platforms supported, ...)
- limits of the process
- reference materials and documents

EXAMPLE
See section 7.

#### 6.1.3.1. The Process lifecycle

GOAL
The aim of this section is to organize the process phases according to the selected lifecycle (or process model) in order to provide a bird-eye view of the whole process at the highest level of detail.

STRUCTURE
This section should include:

- a picture depicting the process lifecycle at the phase level and clearly showing the adopted process model (waterfall, iterative, ...)
- a description of the process phases

EXAMPLE
See section **Error! Reference source not found.**.

### 6.1.4. Fragment Overview

GOAL
STRUCTURE
GUIDELINES
EXAMPLE: See Section …

## *6.2. System metamodel*

GOAL: Provide the user with a detailed description of the portion of system metamodel instantiated/refined/quoted by the fragment.
STRUCTURE: A structural diagram (usually a UML class diagram) depicting the metamodel elements and their relationships
GUIDELINES
EXAMPLE: See Section …

### 6.2.1. Definition of System metamodel elements

GOAL: Providing a precise definition of all system metamodel elements reported in the system metamodel structural diagram
STRUCTURE: a table with two columns (name and definition) or a structured text (vocabulary style)
GUIDELINES
EXAMPLE: See Section …

### 6.2.2. Definition of System metamodel relationships

GOAL: Providing a precise definition of all system metamodel relationships reported in the system metamodel structural diagram
STRUCTURE a table with two columns (name and definition) or a structured text (vocabulary style)
GUIDELINES
EXAMPLE: See Section …

### 6.2.3. Fragment Input/Output in Terms of System Metamodel Constructs

GOAL
Categorizing constructs of the metamodel according to the action performed on them in the fragment (instantiation, refinement, …)

STRUCTURE
a table containing four main columns:

- Input: it deals with the system metamodel constructs that are input to the fragment but they are not designed nor refined nor quoted in the fragment work products.
- To be designed: it deals with the system metamodel contructs that are designed (instantiated) in the fragment and are represented in its work products.
- To be refined: it deals with the system metamodel contructs that are refined in the fragment and are represented in its work products.
- To be quoted: it deals with the system metamodel contructs that are quoted in the fragment, this means such constructs are not instantiated nor refined in this fragment workflow and they are only reported in its work products for providing a complete representation of a specific system view.

Each of these four columns is split in two sub-columns:

- SMME: it lists the system metamodel elements
- SMMR: it lists the system metamodel elements

GUIDELINES
EXAMPLE: See Section …

#### 6.2.3.1. Definition of input system metamodel elements and relationships

GOAL
Providing a precise definition of all system metamodel constructs that are input of this fragment and not reported in any fragment work product

STRUCTURE
a table with two columns (name and definition) or a structured text (vocabulary style)

GUIDELINES

EXAMPLE: See Section 7.

## *6.3. Process Roles*

GOAL
The aim of this section is listing the roles involved in the work of this fragment and clarifying their level of involvement in the job to be done.

STRUCTURE
This subsection should describe the responsibilities of each process role in the activities composing this phase. Roles can be responsible for activities or assist in them. The different levels of responsibility (responsible/assistant) should be clearly stated.

GUIDELINES
Adopting a common taxonomy of process roles could encourage process sharing and the reuse of their portions (fragments). A list of roles has been proposed in the paper by V. Seidita et al. (2008)[1].

EXAMPLE
See Section 7.

#### 6.3.1.1. Role 1

*(description of Role 1 as discussed above)*

EXAMPLE
See section **Error! Reference source not found.**

## *6.4. Workflow*

### 6.4.1. Workflow description

GOAL
The aim of this section is to detail the work to be done at each activity (decomposing it with further elements of a lower level of abstractions if needed)

STRUCTURE
One subsection for each work breakdown element describing phases/activities/tasks/steps composing the selected element by using a SPEM activity diagram including the involved roles (as swimlanes). Further details about each element can be provided in additional

[1] V. Seidita, M. Cossentino, S. Gaglio. Using and Extending the SPEM Specifications to Represent Agent Oriented Methodologies. In Agent-Oriented Software Engineering 2008. Lecture Notes in Computer Science, volume 5386-0086, pp. 46-59. Springer-Verlag GmbH. 2009.

sections.

GUIDELINE

EXAMPLE
See section 7.

### 6.4.2. Work Breakdown Elements description

GOAL
Describe the work to be done within this activity

STRUCTURE
Details of tasks and sub-activities are specified with a table that includes the following columns:
- Name: name of the activity studied in this subsection.
- Kind: It can be: phase, activity, task, steps as reported by SPEM specifications.
- Description: a free text description of the work to be done.
- Roles involved: the list of involved roles also specifying the responsibility (performs, assist) as specified by SPEM.

EXAMPLE
See section 7**Error! Reference source not found.**.

### 6.4.3. Input/Output of Work Breakdown Elements in Terms of System Metamodel Constructs

GOAL
Defining input and output data in terms of system metamodel elements for work breakdown elements

STRUCTURE
A table composed as follows:

| | **Input** | | **Output** | |
|---|---|---|---|---|
| **Activity/Task /Step Name** | **SMME** | **SMMR** | **SMME** | **SMMR** |
| *Name of the work breakdown element* | *List of input system metamodel elements* | *List of input system metamodel relationships* | *List of output system metamodel elements* | *List of output system metamodel relationships* |

GUIDELINE

EXAMPLE
See section 7.

### 6.4.4. Work Products Input/Output

GOAL

STRUCTURE

GUIDELINE

EXAMPLE
See section **Error! Reference source not found.**.

Input, output work products to be designed in the fragment are detailed in the following tables.

| Input | Output |
|---|---|
| | |
| | |

## *6.5. Work Products*

### 6.5.1. Document's name

GOAL
This section aims at detailing the information content of each work product.

STRUCTURE
Work products produced in this phase are described by using a free text.

GUIDELINE

EXAMPLE
See section **Error! Reference source not found.**.

#### 6.5.1.1. Work product kind

A classification of the document according to the five categories prescribed in Seidita et al.: Free Text, Structured Text, Behavioural, Structural, and Composite.
If the document is composite, more details may be provided about the composing elements (for instance a behavioural diagram and a structured text description part).

#### 6.5.1.2. [Document's name] notation

GOAL
The aim of this subsection is allowing an easy adoption of the modelling approach proposed by the original process to the reader of this document or conversely, providing the creator of

a new modelling notation with a clear idea of what is required in order to properly support the modelling demands of this document.

STRUCTURE
The structure of this subsection includes:
1) A description of the document with a specific mention of the notation adopted and an explicit reference to adopted standards (if any).
2) Examples of the document parts. These examples also include tables, diagrams, and outlines of specific portions of text used for describing the design.

#### *6.5.1.2.1.* Example

GOAL
Exemplifying the adopted notation and documentation approach.

STRUCTURE
An example of work product (a portion if the whole is too large), including tables and other templates that could help in understanding and reusing the notation and documentation approach.
The structure of this subsection includes:
1) A description of the document with a specific mention of the notation adopted.
2) Examples of the document parts. These examples also include tables, diagrams, and outlines of specific portions of text used for describing the design.

### 6.5.2. Deliverable relationships with the system metamodel

GOAL
Describing which system metamodel constructs are reported in the work products delivered by this fragment and which operations are performed on them (define, refine, relate, quote).

STRUCTURE
Work products produced in this fragment are described by using a work product structure diagram.
This diagram (an extension to SPEM specifications proposed by V. Seidita et al.[2]) is a structural (i.e. class) diagram reporting the work products delivered by the fragment.
The structure of each work products is then expressed in terms of the system metamodel constructs defined/related/refined/quoted in it.



## *6.6. Guidance*

### 6.6.1. Enactment Guidance

GOAL

[2] V. Seidita, M. Cossentino, S. Gaglio. Using and Extending the SPEM Specifications to Represent Agent Oriented Methodologies. In Agent-Oriented Software Engineering 2008. Lecture Notes in Computer Science, volume 5386-0086. Springer-Verlag GmbH. 2009.

Describing all guidelines that are useful to the involved process roles in correctly performing their work.

STRUCTURE
Free text

#### 6.6.2. Reuse Guidance

##### 6.6.2.1. Composition

GOAL
Providing composition guidelines for reusing the current fragment

STRUCTURE
Free text

##### 6.6.2.2. Dependency Relationship with other fragments

GOAL
Defining common dependency relationship of the current fragment with others (usually in the process of origin)

STRUCTURE
Free text

### *6.7. References*



# 7. Example of process fragment

## *7.1. Process Fragment Name*

Communication Ontological Description

## *7.2. Fragment Description*

### 7.2.1. Fragment Goal

Describing semantic agent communications in terms of exchanged knowledge (referred to an ontology), content language and interaction protocol.

## 7.1. Fragment Granularity

Granularity of this fragment: Composed

### 7.1.1. Composing fragments

This fragment is composed of the following fragments:

| Composing Fragment name | Granularity | Process of Origin |
|---|---|---|
| Identify Communications | Atomic | PASSI |
| Describe Communications | Atomic | PASSI |
| Refine Communication Relationships | Atomic | PASSI |

### 7.1.2. Fragment Origin

The presented fragment has been extracted from *PASSI* (Process for Agent Societies Specification and Implementation) design process.

*PASSI* (Process for Agent Societies Specification and Implementation) is a step-by-step requirement-to-code methodology for designing and developing multi-agent societies. The methodology integrates design models and concepts from both Object-Oriented software engineering and artificial intelligence approaches.

PASSI has been conceived in order to design FIPA-compliant agent-based systems, initially for robotics and information systems applications.

Systems designed by using the PASSI process are usually composed of peer-agents (although social structures can be defined). According to FIPA specifications agents are supposed to be mobile, and they can interact by using semantic communications referring to an ontology and an interaction protocol.

PASSI is suitable for the production of medium-large MAS (up to a hundred agent-kinds each one instantiated in an unlimited number of agents in the running platform).

The adoption of patterns and the support of specific CASE tools (PTK) allows a quick and affordable production of code for the JADE platform. This encourages the use of this process even in time/cost-constrained projects or where high quality standards have to be met.

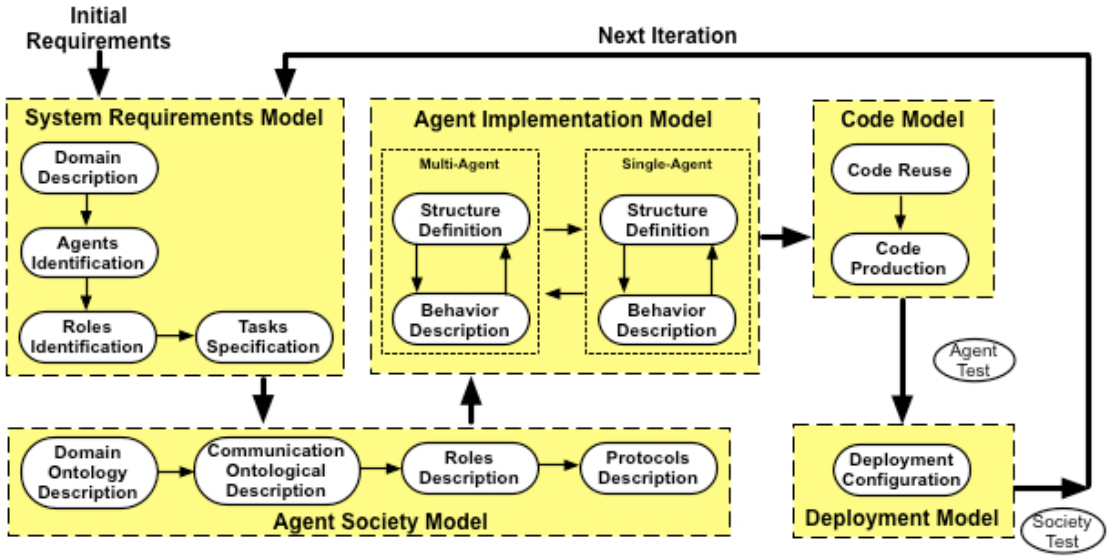



**Figure 4. The PASSI design process**

The design process is composed of five models (see Figure 4): the System Requirements Model is a model of the system requirements; the Agent Society Model is a model of the agents involved in the solution in terms of their roles, social interactions, dependencies, and ontology; the Agent Implementation Model is a model of the solution architecture in terms of classes and methods (at two different levels of abstraction: multi and single-agent); the Code Model is a model of the solution at the code level and the Deployment Model is a model of the distribution of the parts of the system (i.e. agents) across hardware processing units, and their movements across the different available platforms.

Useful references about the PASSI process are the following:

- M. Cossentino. From Requirements to Code with the PASSI Methodology. In Agent-Oriented Methodologies, B. Henderson-Sellers and P. Giorgini (Editors). Idea Group Inc., Hershey, PA, USA. 2005.

- M. Cossentino, S. Gaglio, L. Sabatucci, and V. Seidita. The PASSI and Agile PASSI MAS Meta-models Compared with a Unifying Proposal. Lecture Notes in Computer Science, vol. 3690. Springer-Verlag GmbH. 2005. pp. 183-192.
- M. Cossentino and L. Sabatucci. Agent System Implementation in Agent-Based Manufacturing and Control Systems: New Agile Manufacturing Solutions for Achieving Peak Performance. CRC Press, April 2004.
- M. Cossentino, L. Sabatucci, and A. Chella. Patterns reuse in the PASSI methodology. In Engineering Societies in the Agents World IV, 4th International Workshop, ESAW 2003, Revised Selected and Invited Papers, volume 3071 of Lecture Notes in Artificial Intelligence. Springer-Verlag, 2004. pp. 294-310
- M. Cossentino, L. Sabatucci, A. Chella - A Possible Approach to the Development of Robotic Multi-Agent Systems - IEEE/WIC Conf. on Intelligent Agent Technology (IAT'03). October, 13-17, 2003. Halifax (Canada)
- Chella, M. Cossentino, and L. Sabatucci. Designing JADE systems with the support of case tools and patterns. Exp Journal, 3(3):86-95, Sept 2003.


#### 7.1.2.1. The Process lifecycle

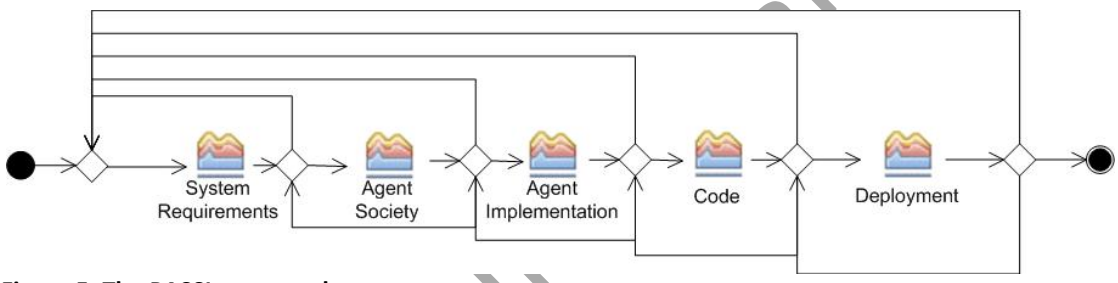



**Figure 5. The PASSI process phases**

PASSI includes five phases (see Figure 2) arranged in an iterative/incremental process model:

- System Requirements: It covers all the phases related to Req. Elicitation, analysis and agents/roles identification
- Agent Society: All the aspects of the agent society are faced: ontology, communications, roles description, Interaction protocols
- Agent Implementation: A view on the system's architecture in terms of classes and methods to describe the structure and the behavior of single agent.
- Code: A library of class and activity diagrams with associated reusable code and source code for the target system.
- Deployment: How the agents are deployed and which constraints are defined/identified for their migration and mobility.

Each phase produces a document that is usually composed aggregating UML models and work products produced during the related activities. Each phase is composed of one or more sub-phases each one responsible for designing or refining one or more artefacts that are part of the corresponding model. For instance, the System Requirements model includes an agent identification diagram that is a kind of UML use case diagrams but also some text documents like a glossary and the system use scenarios.

### 7.1.3. Fragment Overview

Consider the PASSI process (see Figure 5) and the “Agent Society” phase with its outcome “Agent Society Model”. Now, let us consider the “Communication Ontological Description” (red colored in Figure 6) activity and the consequent outcome (the “Communication Ontological Description” composite document).

This activity aims to model the social interactions and dependencies among the agents involved in the solution and to face the following agent society aspects are faced: communication and role description. The activity and its main outcome has been considered for being extracted from PASSI and for becoming a process fragment.

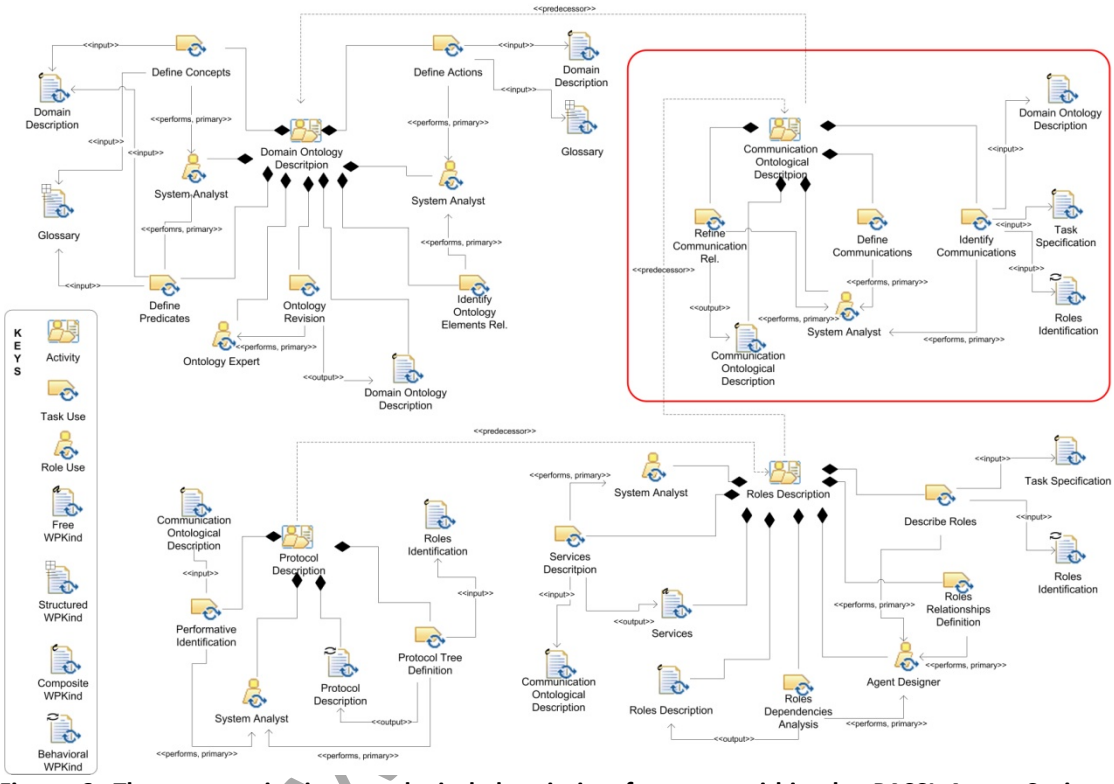



**Figure 6. The communication ontological description fragment within the PASSI Agent Society model structural view**

## *7.2. Fragment System metamodel*

The portion of metamodel of this fragment is:

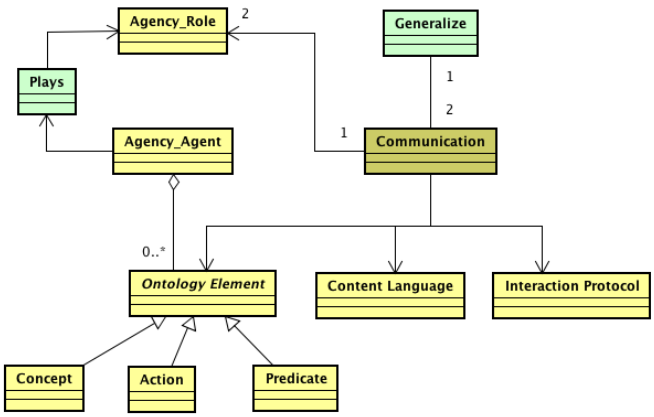



**Figure 7. The fragment System metamodel**

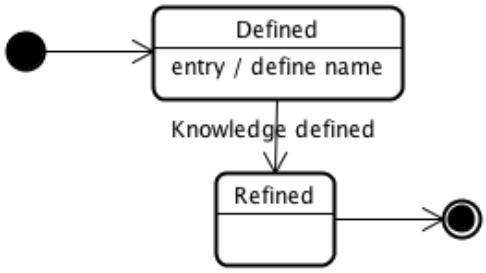
This fragment refers to the MAS meta-model adopted in PASSI and contributes to define and describe the elements reported in Figure 7.

### 7.2.1. Definition of System metamodel elements

This fragment underpins the following model elements:

**Agency_Agent –** an autonomous entity capable of pursuing an objective through its autonomous decisions, actions and social relationships. It is capable of performing actions in the environment it lives; it can communicate directly with other agents, typically using an Agent Communication Language; it possesses resources of its own; it is capable of perceiving its environment; it has a (partial) representation of this environment in form of an instantiation of the domain ontology (knowledge); it can offer services; it can play several, different (and sometimes concurrent or mutually exclusive) agency_roles.

Each agent may be refined by adding knowledge items necessary to store/manage communication contents. The Agency_agent statechart is:




Description of the Agency_Agent states:

*Defined*: An Agency_Agent is in this state once it is instantiated in the system model. The agent's unique name has to be defined.

*Refined*: An Agency_Agent moves in this state once its knowledge chunks are defined.

**Agency_Role** - A portion of the behaviour of an agent that is characterized by an objective (accomplishing some specific functionality) and/or that provides a service.

**Content Language –** A language with a precisely defined syntax semantics and pragmatics, which is the basis of communication between independently designed and developed agents. (from [1])

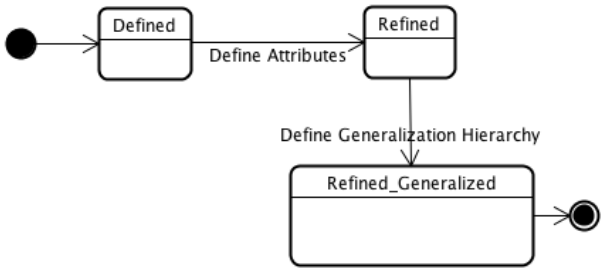
**Ontology Element** *(abstract class)* – An ontology is composed of concepts, actions and predicates. An Ontology element is an abstract class used as a placeholder for the ontology constituting elements (either concepts, predicates or actions).

**(Interaction) Protocol** – It is a pattern specifying the sequence of message types within a communication. Usually message types are identified by the performative (or speech act) associated to the message.
**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

### 7.2.2. Definition of System metamodel relationships

This fragment underpins the following relationships among the model elements:

- **Communication** *(SMMR Association Class)* – An interaction among two agents, referring an Agent Interaction Protocol and a piece of the domain ontology (knowledge exchanged during the interaction). Usually it is composed of several messages, each one associated with one communicative act (or performative).
  The communication construct is an association class and its life-cycle within this fragment is depicted by the following statechart:




  Description of the Communication class states:
  **Defined**: the construct is defined once instantiated in the new system model
  **Refined**: the construct is refined once the values of its attributes (content language, ontology, interaction protocol) have been defined
  **Refined_Generalized**: the construct enters the state Refined_Generalized once a specific generalization hierarchy is defined for reusing common elements in the system communications.

- **Generalize** - See standard UML semantics
- **Plays** (*SMMR Association*) – It specifies that an agent may play one (or more) role(s).

### 7.2.3. Fragment Input/Output in terms of System metamodel constructs

Input, output system metamodel constructs to be designed in the fragment are detailed in the following table.

| Input | | To Be Designed | | To Be Refined | | To Be Quoted | |
|---|---|---|---|---|---|---|---|
| **SMME** | **SMMR** | **SMME** | **SMMR** | **SMME** | **SMMR** | **SMME** | **SMMR** |
| Scenario | Message_RR | Agency_Agent* | Generalize | | Communi-cation (definition of (Concept | Concept | |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | | or Predicate or Action), Content Language, Interaction Protocol) | | |
| Role | AA -OntoRel | Agency_ Role* | Plays | | | Predicate | |
| Agent | CA-OntoRel | | Communi-cation | | | Action | |
| Concept | CC-OntoRel | | | | | Content Language | |
| Predicate | CP-OntoRel | | | | | Interaction Protocol | |
| Action | PP-OntoRel | | | | | | |
| Content Language | PA-OntoRel | | | | | | |
| Interaction Protocol | | | | | | | |

* Some obtained by 1:1 transformation from Problem Domain Agent and Role

#### 7.2.3.1. Definition of input system metamodel elements and relationships

**Role** - A portion of the behaviour of an agent that is characterized by a goal (accomplishing some specific functionality) and/or provides a service.

**Scenario** - An instance of a use case describing a concrete set of actions. A scenario is composed of the following fields:

- Name: used to identify the scenario
- Participating actors: the list of participating actors (frequently actor instances are used)
- Flow of events: describing the flow of events step by step.

**Ontology Element** *(abstract class)* – An ontology is composed of concepts, actions and predicates. An Ontology element is an abstract class used as a placeholder for the ontology constituting elements (either concepts, predicates or actions).

**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



## *7.3. Stakeholders*

Roles involved in this fragment are:

- System Analyst.

Their responsibilities are described in the following subsections.

#### 7.3.1.1. System Analyst

(S)He is responsible for:

1. Communications identification. It consists in introducing an association for each communication between two agents, looking at exchanged messages in the scenario.
2. Communications definition. The description of agents' communication in terms of ontology, content language and interaction protocol.

3. Communication relationships refinement. The identification of association classes in order to link each communication to the three fundamental element of communication itself (ontology, language and protocol).

## *7.4. Fragment workflow*

### 7.4.1. Workflow description

The process that is to be performed in order to obtain the result is represented in the following as a SPEM 2.0 diagram.

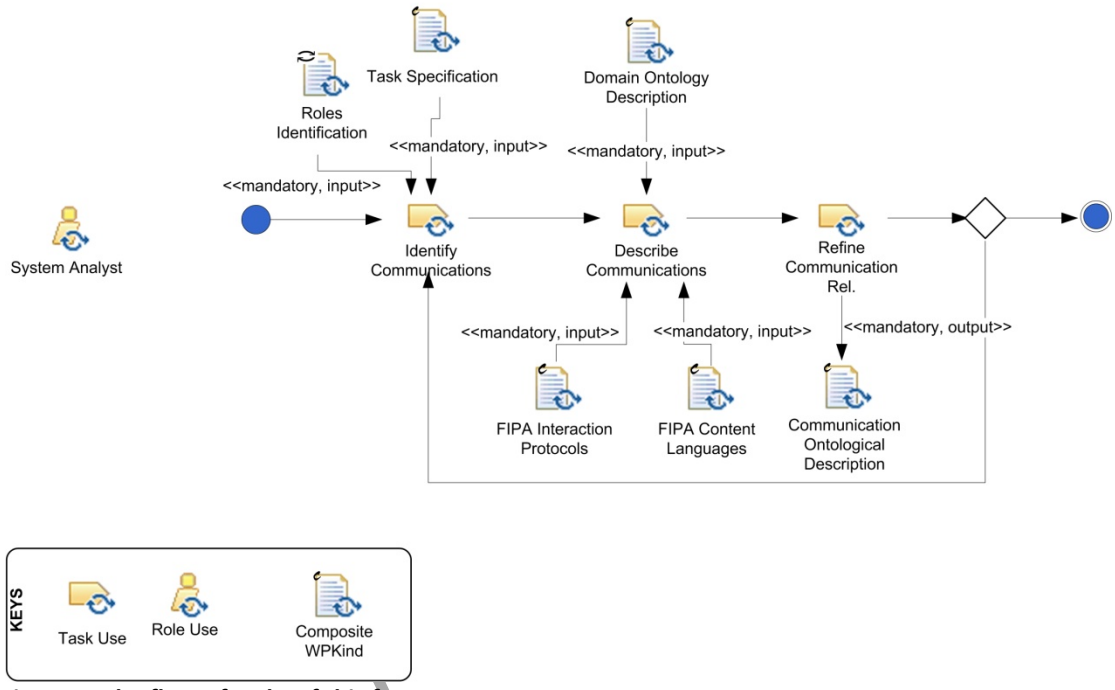



**Figure 8. The flow of tasks of this fragment**

### 7.4.2. Activity description

The fragment encompasses the following *work breakdown elements*:

| Name | Kind | Description | Roles involved |
|---|---|---|---|
| Identify Communications | Task | It consists in defining communications among agents looking at exchanged messages in the scenario. | System Analyst (performs) |
| Describe Communications | Task | It consists in the description of agents' communications in terms of ontology, content language and interaction protocol. Agents' knowledge structures necessary to deal with communication contents have to be introduced in the agents. | System Analyst (performs) |

| | | | |
|---|---|---|---|
| Refine Communication Relationships | Task | The identification of general communication association classes in order to enhance reuse and improve the architecture. Use of generalize association among general and specialized communications. | System Analyst (performs) |

### 7.4.3. Input/Output of Work Breakdown Elements in Terms of System Metamodel Constructs

The above described *work breakdown elements* have the following input/output in terms of system metamodel components.
In the Input column, system metamodel components utilization is completed by the name of the input document reporting them in the original design process.

| | Input | | Output | |
|---|---|---|---|---|
| **Activity/Task Name** | **SMME** | **SMMR** | **SMME** | **SMMR** |
| Identify Communications | Scenario, Role, Agent, Content Language, Interaction Protocol | Message_RR | Agency_Agent* Agency_Role* | Plays Communication[defined |
| Describe Communications | Concept, Predicate, Action, Interaction Protocol, Content Language | AA-OntoRel, CA-OntoRel, CC-OntoRel, CP-OntoRel, PP-OntoRel, PA-OntoRel | Agency_Agent* [refined] | Communication [refined] |
| Refine Communication Relationships | | | | Generalize, Communication [refined_generalized] |

### 7.4.4. WP Input/Output

Input, output work products to be designed in the fragment are detailed in the following tables.

| **Input** | **Output** |
|---|---|
| Agent Identification Document | Communication Ontological Description Document |
| Task Specification Document | |
| Domain Ontology Description Document | |

| | |
|---|---|
| FIPA Interaction Protocols | |
| FIPA Content Languages | |

## *7.5. Deliverable*

### 7.5.1. Communication Ontological Description (COD) Document

This fragment produces a composite document composed by a class diagram (whose classes represent agents and communications) and a text document describing the elements reported in the diagram.
The Communication Ontological Description (COD) diagram is a representation of the agents' (social) interactions; this is a structural diagram (for instance a class diagram) that shows all agents and all their interactions (lines connecting agents).
According to FIPA standards, communications consist of speech acts [1] and are grouped by FIPA in several interaction protocols [2] that define the sequence of expected messages. As a consequence, each communication is characterized by three attributes, which we group into an association class. The attributes are: ontology (a piece of the ontology defined in the PASSI DOD fragment), content language (see [3]), interaction protocol. This is the characterization of the communication itself (a communication with different ontology, content language or interaction protocol is certainly different from this one) and its name is used to uniquely refer this communication (which can have, obviously, several instances at runtime since it may be enacted more than once).

The following table describes the knowledge items assigned to agents (as attributes) in order to conveniently store/manage the received/outgoing communication content.

#### 7.5.1.1. Agents' Knowledge

| Agent | Knowledge piece | Data Type | Description |
|---|---|---|---|
| <agent name> | <name used to instantiate a piece of the ontology in the agent> | <a piece of the system ontology> | <text description> |

The following table details the communication:

#### 7.5.1.2. Communication details

| From | To | Protocol | Content Language | Content (referred to ontology) | Description |
|---|---|---|---|---|---|
| <Initiator Agent>.<Initiator Role> | <Destination Agent>.<Destination Role> | <Interaction Protocol name> | <Content Language> | <Ontology element referred to in the communication content> | <description of the communication objective> |

#### 7.5.1.3. Communication Ontological Description Diagram notation

Each agent (fill colour: yellow) is described in terms of its knowledge (pieces of the ontology described in the Domain Ontology Description fragment). There is one relationship between two agents for each communication they are involved in. In each relationship the roles played by the agents during the communication are also reported as connection roles.

Each communication (fill colour: white) is represented by the relationship among the two agents and it is detailed in the relationship attribute class. The class is identified by an unique name and it is described by the following attributes: the ontology, the content language and the interaction protocol.

The ontology field refers to an element of the DOD (Domain Ontology Description); the language addresses for the adopted FIPA content language of the communication while the protocol points out the adopted FIPA Interaction Protocol.

##### 7.5.1.3.1. Example

In Figure 9, the *PurchaseManager* agent starts a communication (see *QueryForAdvice* association class) with the *PurchaseAdvisor* agent. The communication contains the *Course* ontology, the *Query* protocol and the RDF language. This means that the *PurchaseManager* wants to perform a speech act based on the FIPA query protocol in order to ask the *PurchaseAdvisor* advice on how to purchase (supplier, number of stocks, number of items per each, purchase-money) provided the Course information.

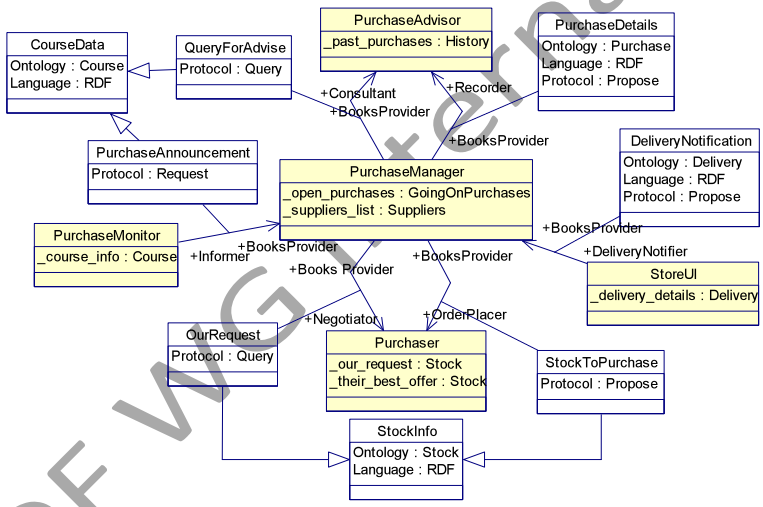
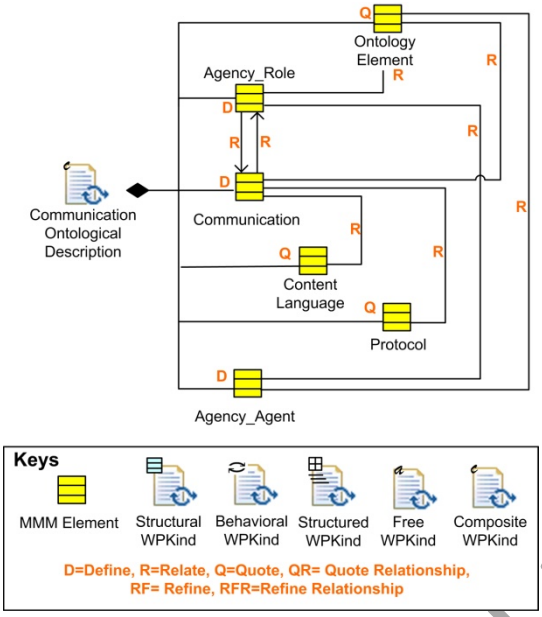



**Figure 9. An example of Communication Ontological Description diagram**

### 7.5.2. Deliverable relationships with the system metamodel




## 7.6. Guidelines

### 7.6.1. Enactment Guidelines

Agency_Roles are usually obtained by 1:1 transformation from Roles defined in the PASSI Problem Domain. This initially means Agency_Roles represent analysis roles, but new Agency_Roles may be defined if necessary/useful to improve design.
The same happens for Agency_agents.

### 7.6.2. Reuse Guidelines

#### 7.6.2.1. Composition

--

#### 7.6.2.2. Dependency Relationship with other fragments

This fragment usually is preceded by the PASSI Domain Ontology Description (composed) fragment.

## 7.7. Glossary

## 7.8. References


[1] J.R. Searle. Speech Acts. Cambridge Univ. Press. Cambrige. 1969.

[2] IEEE FIPA. Interaction Protocols Specifications. Available online at: http://www.fipa.org/repository/ips.php3

[3] IEEE FIPA. Content Languages Specifications. Available online at: http://www.fipa.org/repository/cls.php3

[4] Jacobson, I., Booch, G., Rumbaugh, J.: The uni_ed software development process. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA (1999).

[5] Seidita, V., Cossentino, M., Hilaire, V., Gaud, N., Galland, S., Koukam, A., Gaglio, S. (2010). The metamodel: a starting point for design processes construction.

International Journal of Software Engineering and Knowledge Engineering. Volume 20, No.4, pp. 575-608.
[6] Seidita, V., Cossentino, M., Gaglio, S. A repository of fragments for agent systems design. Proc. of the Workshop on Objects and Agents (WOA06). Catania, Italy. 26-27 September 2006.
[7] Seidita, V., Cossentino, M., Gaglio, S.: Using and Extending the SPEM Specifications to Represent Agent Oriented Methodologies. In: Agent-Oriented Software Engineering IX. pp. 46{59. Springer-Verlag (2009).