



Consiglio Nazionale delle Ricerche
Istituto di Calcolo e Reti ad Alte Prestazioni

SPEM 2.0 description of the PASSI process

Massimo Cossentino, Valeria Seidita

RT-ICAR-

November 2008



Consiglio Nazionale delle Ricerche, Istituto di Calcolo e Reti ad Alte Prestazioni (ICAR) –
Sede di Cosenza, Via P. Bucci 41C, 87036 Rende, Italy, URL: www.icar.cnr.it
– Sezione di Napoli, Via P. Castellino 111, 80131 Napoli, URL: www.na.icar.cnr.it
– Sezione di Palermo, Viale delle Scienze, 90128 Palermo, URL: www.pa.icar.cnr.it



Consiglio Nazionale delle Ricerche
Istituto di Calcolo e Reti ad Alte Prestazioni

SPEM 2.0 description of the PASSI process

Massimo Cossentino¹, Valeria Seidita²

Rapporto Tecnico N.:
RT-ICAR-

Data:
November 2008

¹ Istituto di Calcolo e Reti ad Alte Prestazioni, ICAR-CNR, Sezione di Palermo, Viale delle Scienze edificio 11, 90128 Palermo (Italy).

² Dipartimento di Ingegneria Informatica, Università degli Studi di Palermo, Viale delle Scienze edificio 6, 90128 Palermo (Italy).

I rapporti tecnici dell'ICAR-CNR sono pubblicati dall'Istituto di Calcolo e Reti ad Alte Prestazioni del Consiglio Nazionale delle Ricerche. Tali rapporti, approntati sotto l'esclusiva responsabilità scientifica degli autori, descrivono attività di ricerca del personale e dei collaboratori dell'ICAR, in alcuni casi in un formato preliminare prima della pubblicazione definitiva in altra sede.

Index

Introduction.....	4
The PASSI Process.....	5
The PASSI Metamodel.....	6
Phases of the PASSI Process.....	9
The System Requirements Phase.....	9
Process roles.....	10
System Analyst.....	10
Domain Expert.....	10
Activity Details.....	10
Domain Requirements Description.....	10
Agent Identification.....	11
Roles Identification.....	12
Task Specification.....	12
Work Products.....	14
Work Product Kinds.....	15
Domain Requirements Description.....	15
Agent Identification.....	17
Roles Identification.....	17
Task Specification.....	18
Work Products Dependencies.....	20

1. Introduction

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.

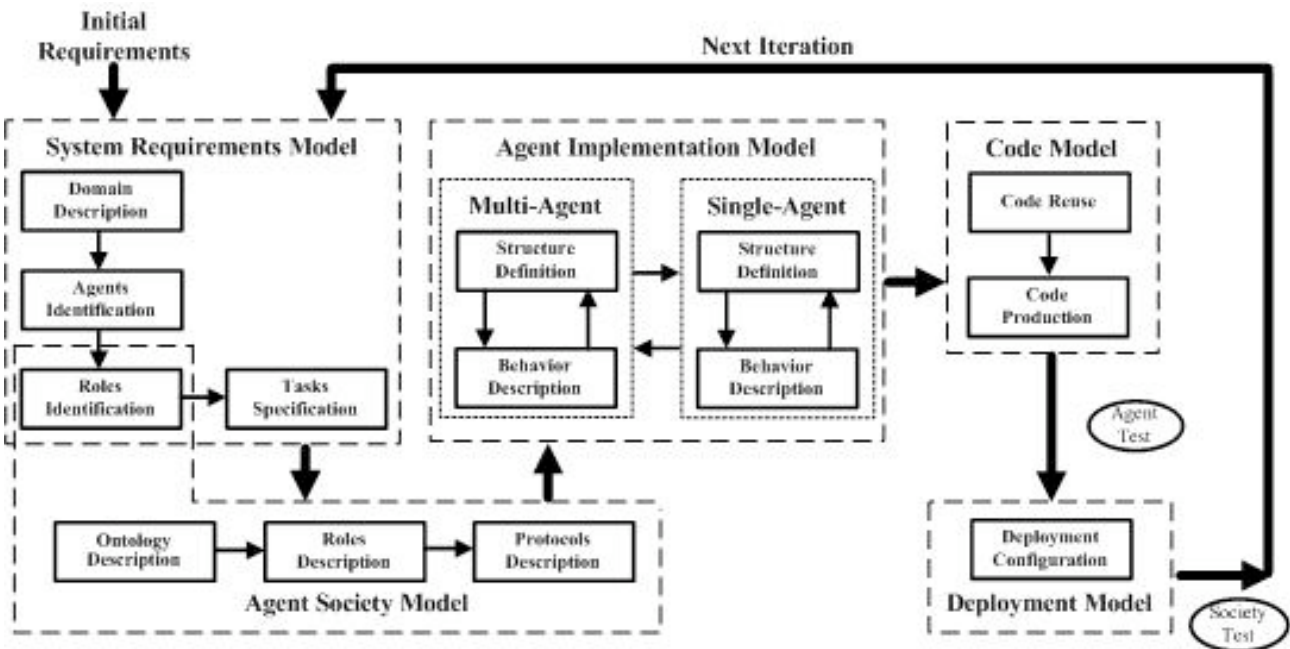


Figure 1. The PASSI design process

The design process is composed of five models (see Fig. 1): 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 (agents) across hardware processing units, and their movements across the different available platforms.

In the following the PASSI process will be described by initially considering its whole process and then its five components, each of them representing a phase, a portion of work for which a specific outcome and milestones can be identified and represented in the following diagram.

2 The PASSI Process

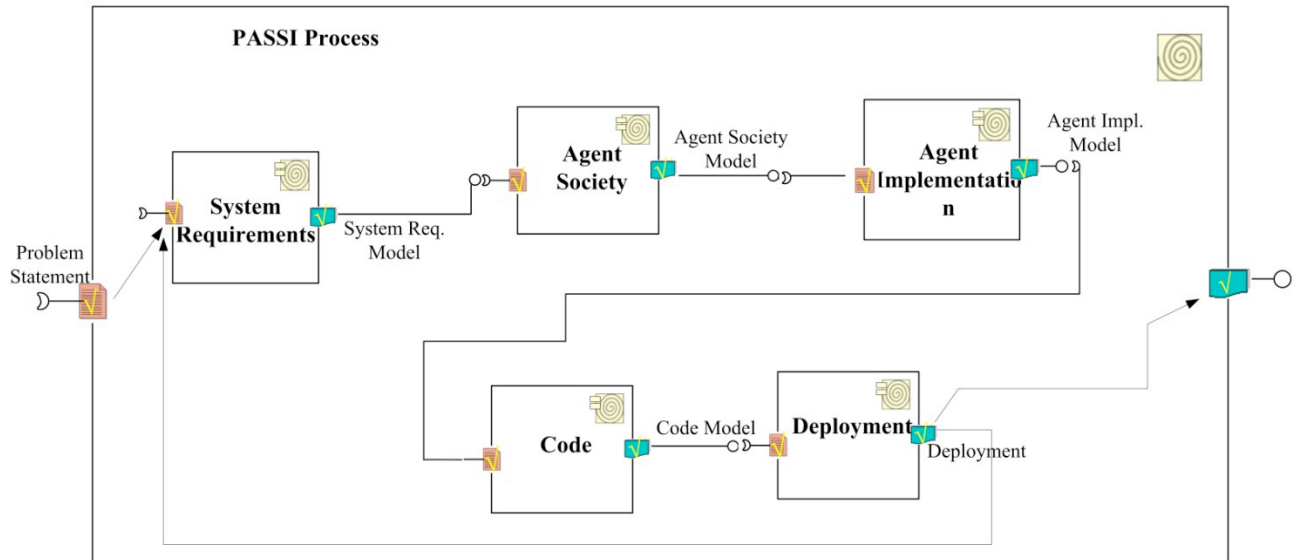


Figure 2. The PASSI process 5 phases

PASSI includes five components (see Figure 2):

- 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.

We can draw a direct correspondence between the above cited components and the five phases that characterise the procedural representation of PASSI represented in Figure 1.

In fact, the PASSI process is composed of five different phases: System Requirements, Agent Society, Agent Implementation, Code and Deployment (the same names of the components).

Each phase produces a document that is usually composed aggregating the 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).

The details of each phase and its related sub-phases will be discussed in the following section.

3 The PASSI Metamodel

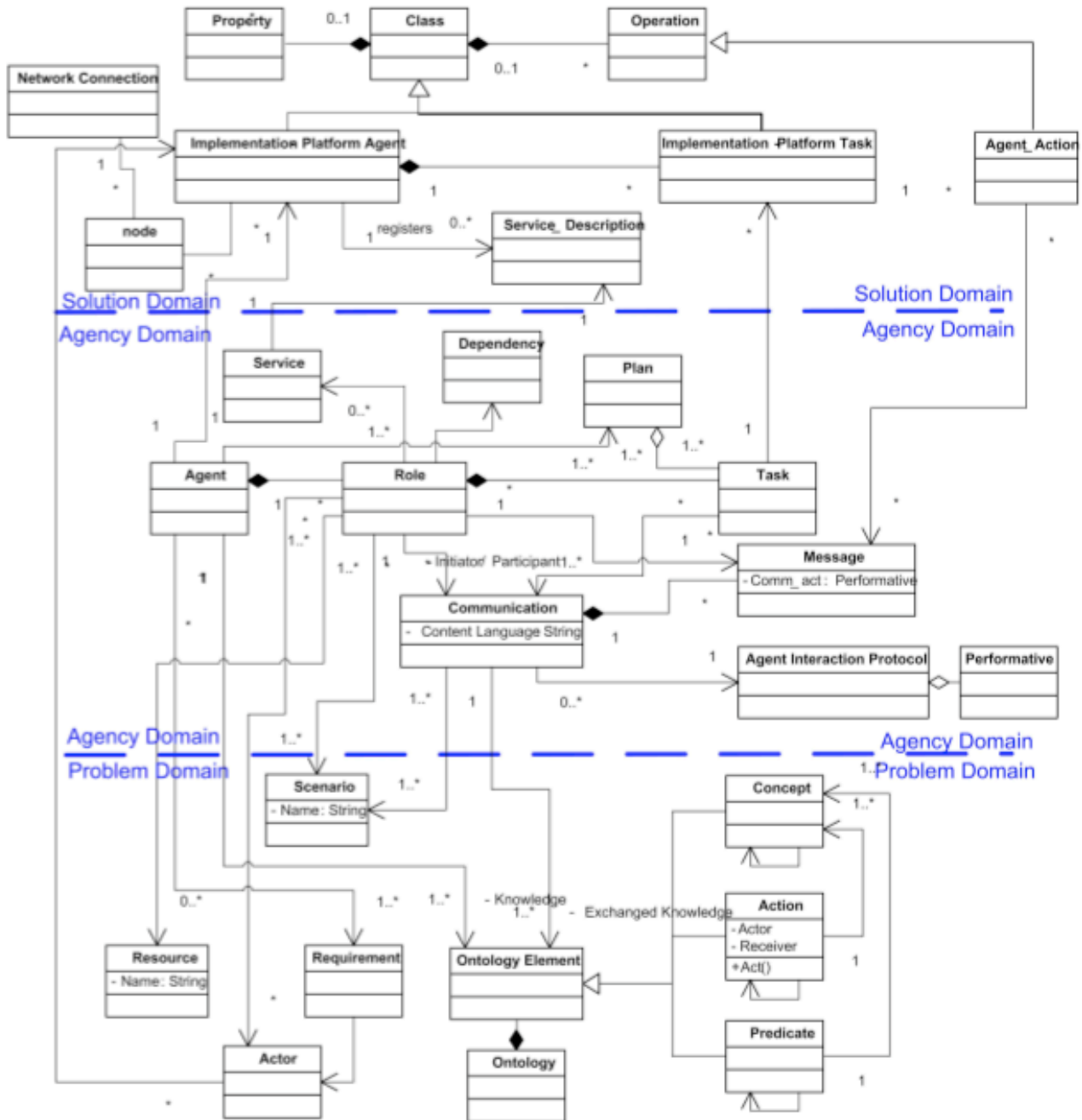


Figure 3: The PASSI metamodel

The description of the PASSI MAS meta-model (Figure 3) addresses three logical areas: (i) the problem domain, (ii) the solution domain and (iii) the agency domain; they are introduced in an order that reflects our choice of an agent approach for solution refinement and modeling.

In the problem domain we include components describing the requirements the system is going to accomplish: these are directly related to the requirements analysis phase of the PASSI process. Then we introduce the agency domain components; they are used to define an agent solution for the problem.

Finally, in the PASSI MMM solution domain, agency-level components are mapped to the adopted FIPA-compliant implementation platform elements (we suppose the platform supports at least the concepts of agent and task); this represents the code-level part of the solution and the last refinement step.

Going into the details of the model, we can see that (Figure 3), the Problem Domain deals with the user's problem in terms of scenarios, requirements, ontology and resources. Scenarios describe a sequence of interactions among actors and the system to be built.

The ontological description of the domain is composed of concepts (categories of the domain), actions (performed in the domain and effecting the status of concepts) and predicates (asserting something about a portion of the domain, i.e. the status of concepts).

Resources are the last element of the problem domain. They can be accessed/shared/manipulated by agents. A resource could be a repository of data (like a relational database), an image/video file or also a good to be sold/bought.

The Agency Domain contains the components of the agent-based solution. None of these components is directly implemented; they are converted to the correspondent object-oriented entity that constitutes the real code-level implementation.

In PASSI an agent is responsible for realizing some functionalities descending from one or more functional requirements. It also has to respect some non functional requirement constraints (like for instance performance prescriptions). It lives in an environment from which it receives perceptions (the related knowledge is structured according to the designed domain ontology). Sometimes an agent has also access to available resources and it is capable of actions in order to pursue its own objectives or to offer services to the community.

Each agent during its life plays some roles. A role is a peculiarity of the social behavior of an agent. When playing a role, an agent may provide a service to other agents.

The role concept is tightly related to the agent communication capability (interactions among agents are uniquely based on communications composed of messages seen as speech acts).

In PASSI, a task specifies the computation that generates the effects of a specific agent behavioral feature. It is used with the significance of atomic part for defining the overall agent's behavior. This means that an agent's behavior can be composed by assembling its tasks and the list of actions that are executed within each task cannot be influenced by the behavior planning. Tasks are structural internal components of an agent and they contribute to define the agent's abilities; these cannot be directly accessed by other agents (autonomy) unless the agent offers them as a set of services.

A communication is an interaction among two agents and it is composed of one or more messages. The information exchanged during a communication is composed of concepts, predicates or actions defined in the ontology. The flow of messages and the semantic of each message are ruled by an agent interaction protocol (AIP).

The last Agency Domain element (Service) describes a set of coherent functionalities exported by the agent for the community.

The Implementation Domain describes the structure of the code solution in the chosen FIPA-compliant implementation platform and it is essentially composed of three elements: (i) the FIPA-Platform Agent that is the base class catching the implementation of the Agent entity represented in the Agency domain; (ii) the FIPA-Platform Task that is the implementation of the agent's Task, (iii)



the ServiceDescription component that is the implementation-level description (for instance an OWL-S file) of each service specified in the Agent Domain.

4 Phases of the PASSI Process

4.1 The System Requirements Phase

The System Requirements phase involves two different process roles, eight work products (four UML models and four text documents) and four guidance documents (one for each UML model) as described in the following Figure 4.

The phase is composed of four activities (Domain Requirements Description, Agents Identification, Roles Identification and Task Specification), each of them composed of one or more tasks (for instance Identify use cases and Refine Use Cases).

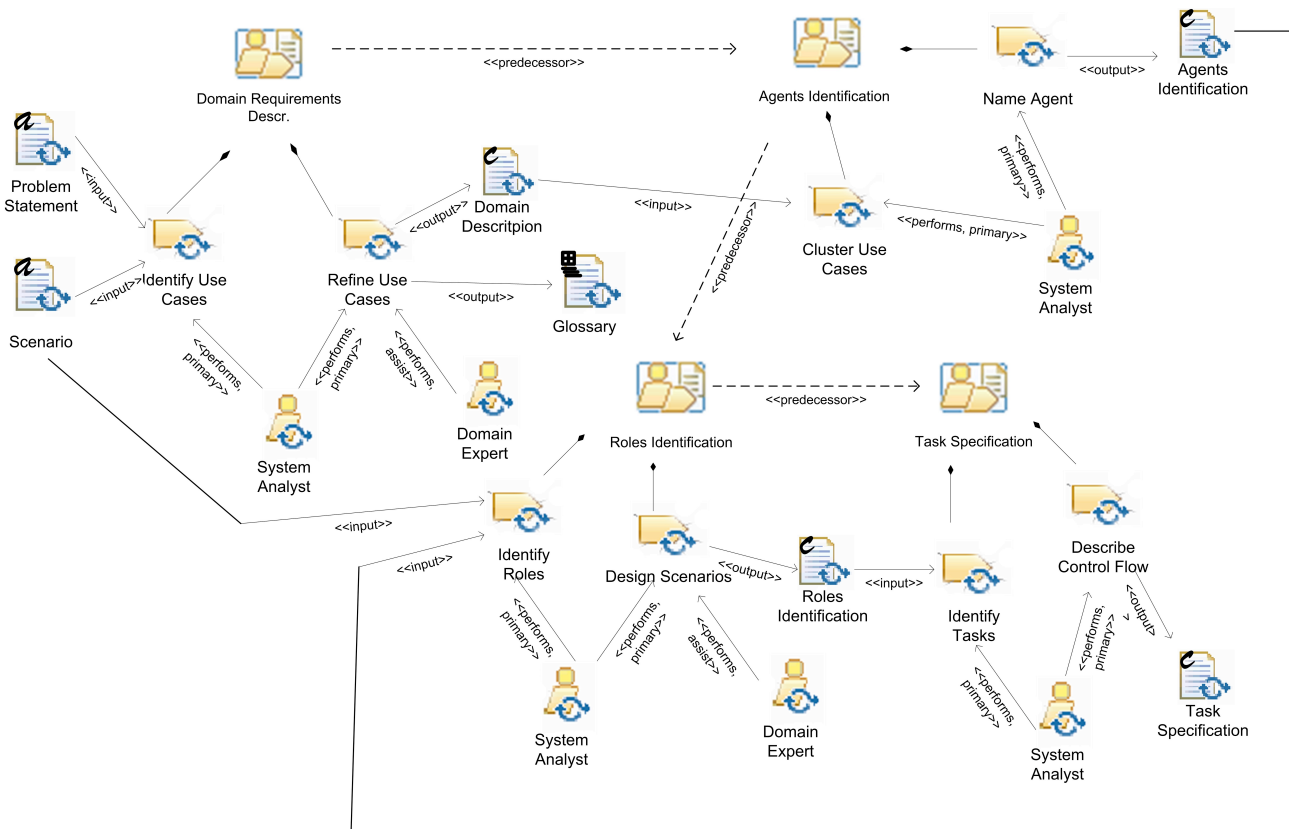


Figure 4. The System Requirement phase described in terms of activities and work products

The process flow at the level of activities is reported in Figure 5. The process flow inside each activity will be detailed in the following sub-sections (after the description of process roles).



Figure 5. The System Requirements Phase flow of activities.

4.1.1 Process roles

Two roles are involved in the System Requirements discipline: the System analyst and the Domain Expert. They are described in the following sub-sections.

4.1.1.1 System Analyst

He is responsible of:

1. Use cases identification during the DD sub-phase. Use cases are used to represent system requirements.
2. Use cases refinement during the DD sub-phase. Use cases are refined with the help of a Domain Expert.
3. Use cases clustering during the AId sub-phase. The System Analyst analyzes the use case diagrams resulting from the previous phase and attempts their clustering in a set of packages.
4. Naming agents during the AId sub-phase. After grouping the use cases in a convenient set of packages, the last activity of this phase consists in designing these packages with the names that will distinguish the different agents throughout all the project.
5. Roles identification during the RId sub-phase. The System Analyst studies (textual) scenarios and system requirements (as defined in the previous phase) and identifies the roles played by agents.
6. Designing scenarios during the RId sub-phase. Each scenario is designed in form of sequence diagram thus depicting the details of agents interactions
7. Tasks identification during the TSp. sub-phase. It consists in the identification of the behavioral capabilities that each agent needs to perform the specified roles and the fulfill the requirements that are under its responsibility.
8. Description of the control flow during the TSp. sub-phase. It consists in introducing the communication relationships among tasks of different agents and the control flow among tasks of the same agent.

4.1.1.2 Domain Expert

He supports the system analyst during the description of the domain requirements.

4.1.2 Activity Details

4.1.2.1 Domain Requirements Description

The flow of tasks inside this activity is reported in Figure 6 and the tasks are detailed in the following table.

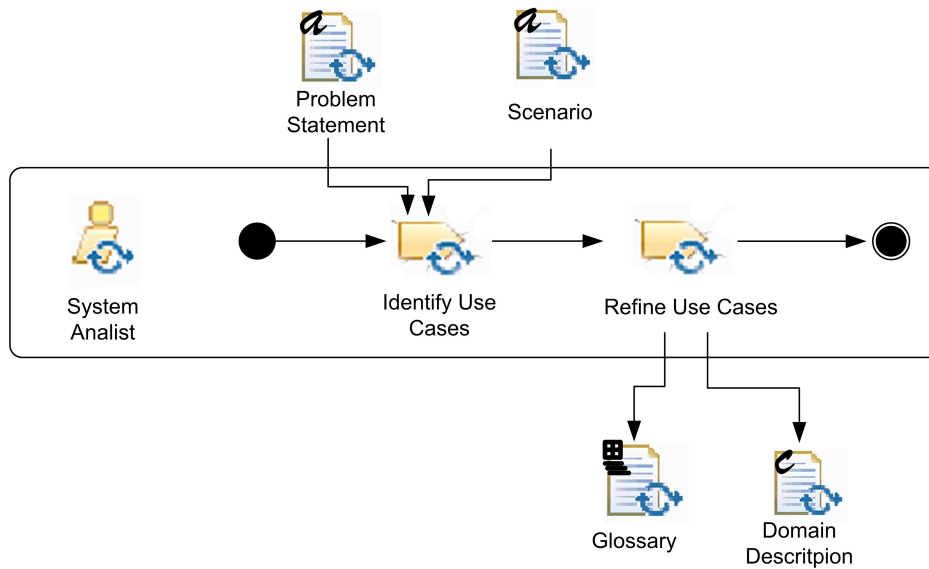


Figure 6: The flow of tasks of the Domain Requirements Description activity

Activity	Task	Task Description	Roles involved
Domain Requirements Description	Identify Use Cases	Use cases are used to represent system requirements	System Analyst (perform)
Domain Requirements Description	Refine Use Cases	Use cases are refined with the help of a Domain Expert	System Analyst (perform) Domain Expert (assist)

4.1.2.2 Agent Identification

The flow of tasks inside this activity is reported in Figure 7 and the tasks are detailed in the following table.

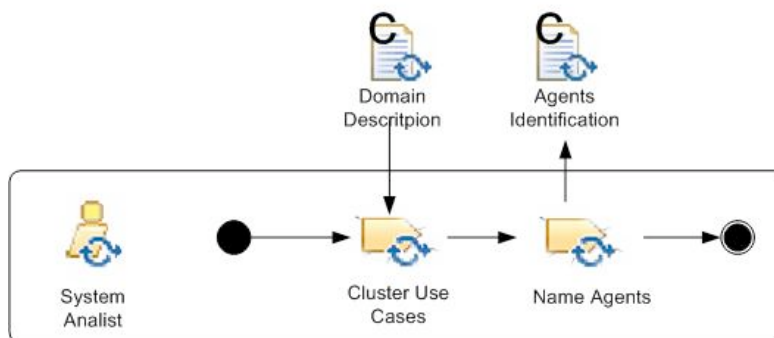


Figure 7: The flow of tasks of the Agent Identification activity

Activity	Task	Task Description	Roles involved
Agents Identification	Use Cases Clustering	The System Analyst analyses the use case diagrams resulting from the previous phase and attempts their clustering in a set of package.	System Analyst (perform)

Agents Identification	Agents Naming	After grouping the use cases in a convenient set of packages, the last activity of this phase consists in identifying these packages with the names that will distinguish the different agents throughout all the project	System Analyst (perform)
-----------------------	---------------	---	--------------------------

4.1.2.3 Roles Identification

The flow of tasks inside this activity is reported in Figure 8 and the tasks are detailed in the following table.

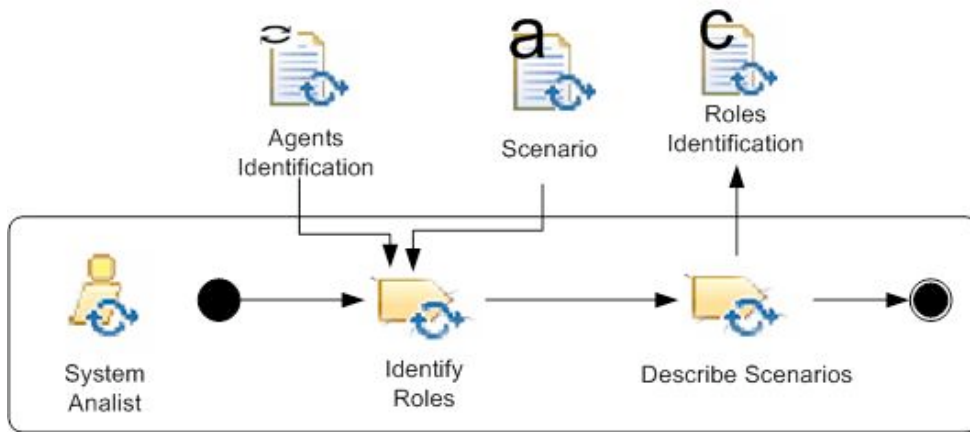


Figure 8: The flow of tasks of the Roles Identification activity

Activity	Task	Task Description	Roles involved
Role Identification	Identify Role	The System Analyst studies (textual) scenarios and system requirements (as defined in the previous phase) and identifies the roles played by agents	System Analyst (perform)
Role Identification	Design Scenarios	Each scenario is designed in form of sequence diagram thus depicting the details of agents interactions	System Analyst (perform) Domain Expert (assist)

4.1.2.4 Task Specification

The flow of tasks inside this activity is reported in Figure 9 and the tasks are detailed in the following table.

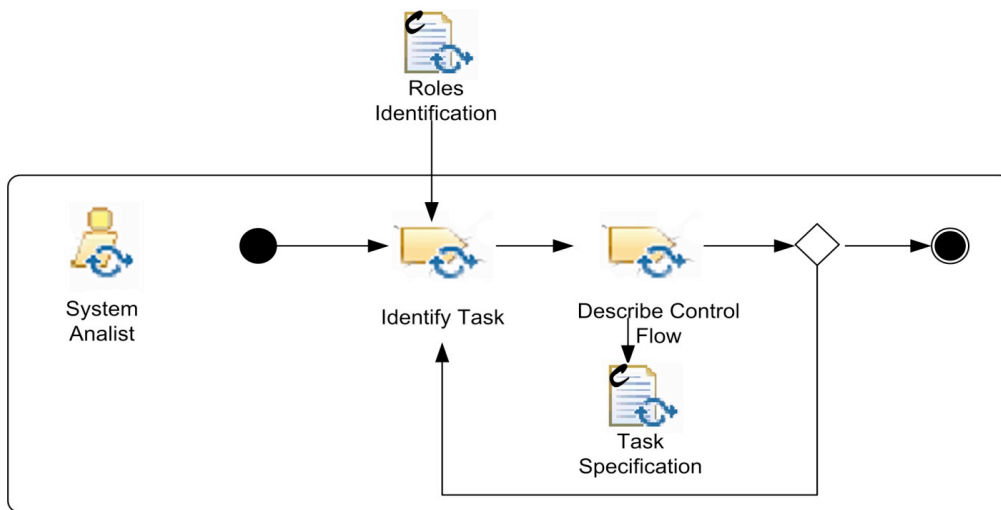


Figure 9: The flow of tasks of the Task Specification Activity

Activity	Task	Task Description	Roles involved
Task Specification	Identify Tasks	It consists in the identification of the behavioral capabilities that each agent needs to perform the specified roles and the fulfill the requirements that are under its responsibility	System Analyst (perform)
	Describe the control flow	It consists in introducing the communication relationships among tasks of different agents and the control flow among tasks of the same agent	

4.1.3 Work Products

The System Requirements Model generates four composed work products (text documents including diagrams). Their relationships with the MAS meta-model elements are described in following Figure 10.

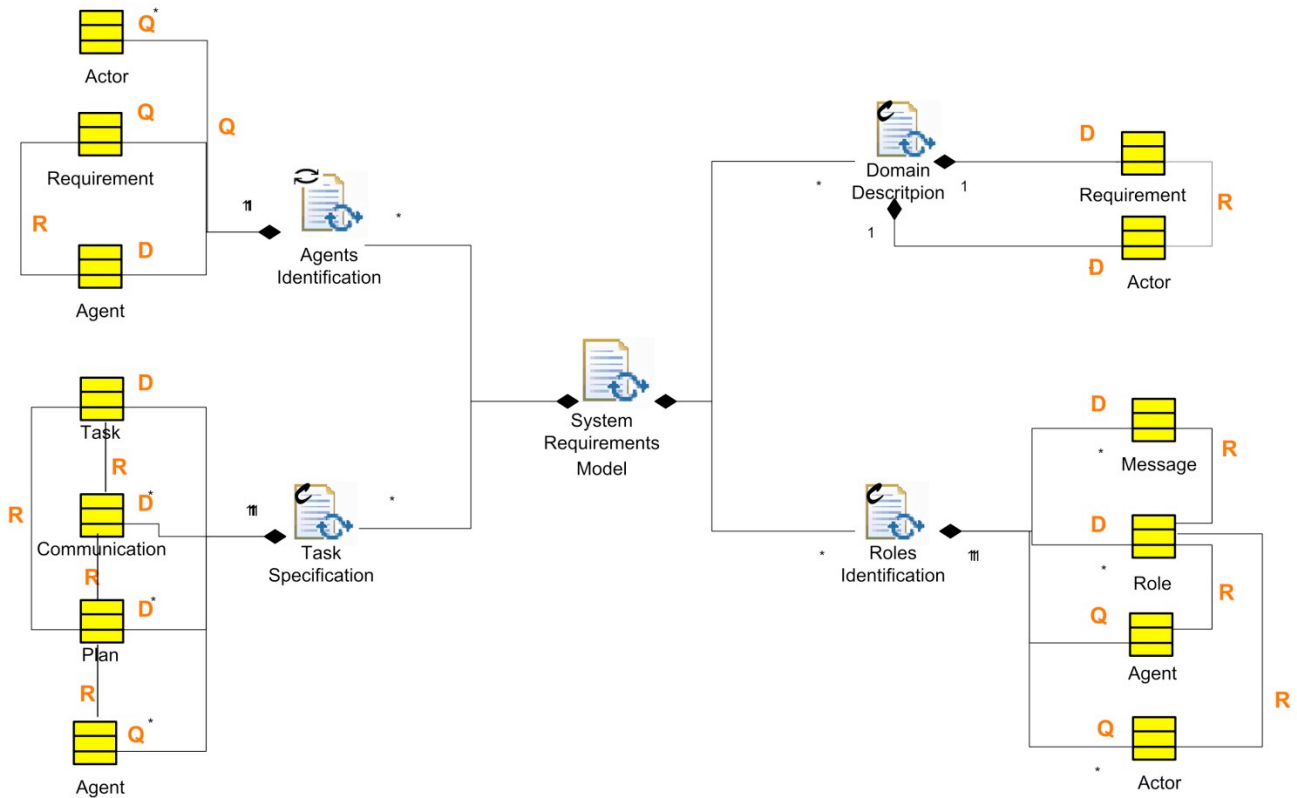


Figure 10. The System Requirements Model documents structure

This diagram represents the System Requirement model in terms of Work Products. Each of these reports one or more elements from the PASSI MAS meta-model; each MAS meta-model element is represented using an UML class icon (yellow filled) and, in the documents, such elements can be Defined, Quoted or Related as described below:

- *defined*: D label near the element symbol, this means that the element is introduced for the first time in the design in this artefact (the MAS metamodel element is instantiated in this artefact),
- *related*: R label, this means that an already defined element is related to another or from a different point of view that one of the MAS metamodel relationships is instantiated in the document)
- *quoted*: Q label, this means that the element has been already defined and it is reported in this artefact only to complete its structure but no work has to be done on it.

4.1.3.1 Work Product Kinds

Name	Description	Work Product Kind
Problem Statement	A description of the problem to be solved with the system. It is complemented by the Scenario document	Free Text
Scenarios	Textual description of the scenarios in which the system to be developed is involved.	Free Text
Domain Description	A text document composed by the Domain Description diagram, a documentation of use cases reported in it and the non functional requirements of the system	Composed (Structured + Behavioural)
Agent Identification	A document composed of: 1) a use case diagram representing agents and the functionalities assigned to them; 2) a structured text description of the agents	Composed (Structured + Behavioural)
Roles Identification	A document composed of several sequence diagrams (one for each scenario) and the roles description text	Composed (Structured + Behavioural)
Task Specification	A document composed of several Task specification diagrams (one for each agent) and a structured text description of each task	Composed (Structured + Behavioural)
Glossary	A glossary of terms	Structured Text
Code	Code of the software solution	Structured Text

4.1.3.2 Domain Requirements Description

Common UML use case diagram(s) are used to represent a functional description of the system.

Use cases documentation is inspired by the approach proposed in: I. Jacobson, M. Christerson, P. Jonsson, and G. Overgaard. Object-oriented software engineering: A use case driven approach. Reading, MA: Addison Wesley, 1992.

An example of documentation is reported in the following figure and table

Identifier: Unique identifier of the use case, may depend on reference number and modification history.

Name: Name of the use case.

Iteration: informs the reader of the stage the use case has reached.

Type: *Abstract* or *Concrete*

Description: Goal to be achieved by use case and sources of requirements.

Actors: List of actors involved in use case.

Pre-conditions: Conditions that must be true for use case to start successfully.

Post-conditions: Conditions that must be true at the end of the scenario. It summarizes the state of affairs after the scenario is complete.

Flow: The ordered list of interactions between actors and system that are necessary to achieve goal. Example of general flow of interaction:

1. Interaction x
2. Interaction y
- 3.a If condition then
 - 3.a.1 Interaction
 - 3.a.2 Interaction
- 3.b else If condition then
 - 3.b.1 Interaction
 - 3.b.2 Interaction
- 3.c else
 - 3.c.1 Interaction

⋮

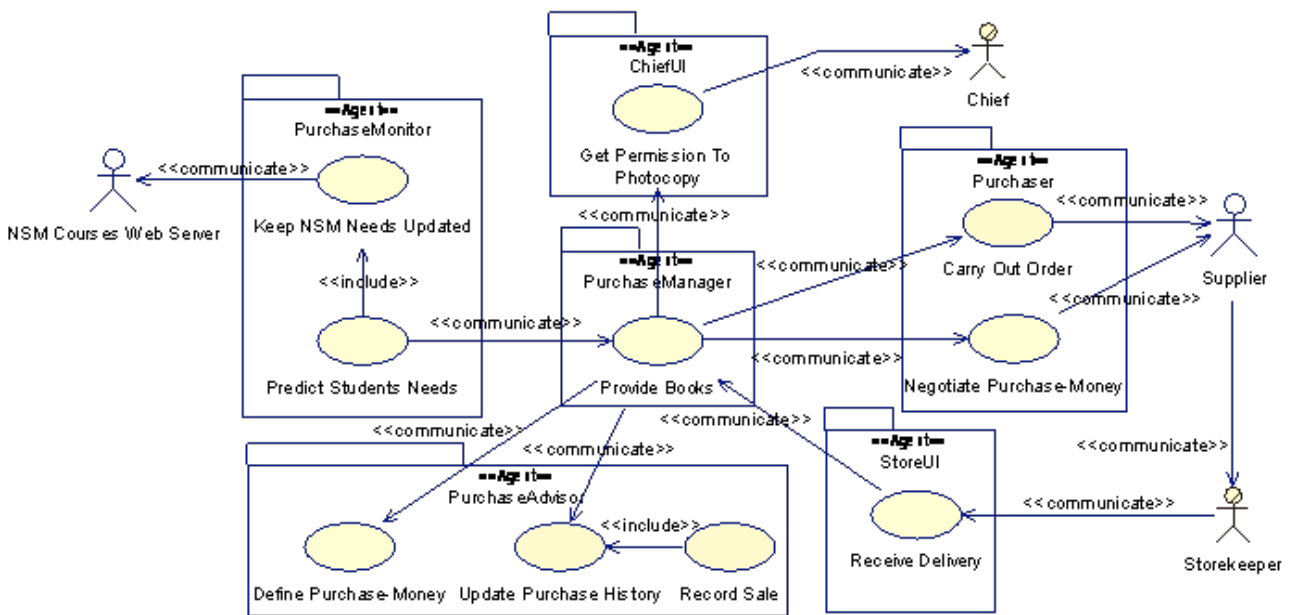
Variations/Alternative Flow: Any variations, alternative in the flow of a use case.

Non-functional: List of non-functional requirements that the use case must meet.

Authors and dates: Modifications history.

Actor Name	Description

4.1.3.3 Agent Identification



Starting from an use case diagram, packages are used to group functionalities that will be assigned to an agent (whose name is the name of the package).

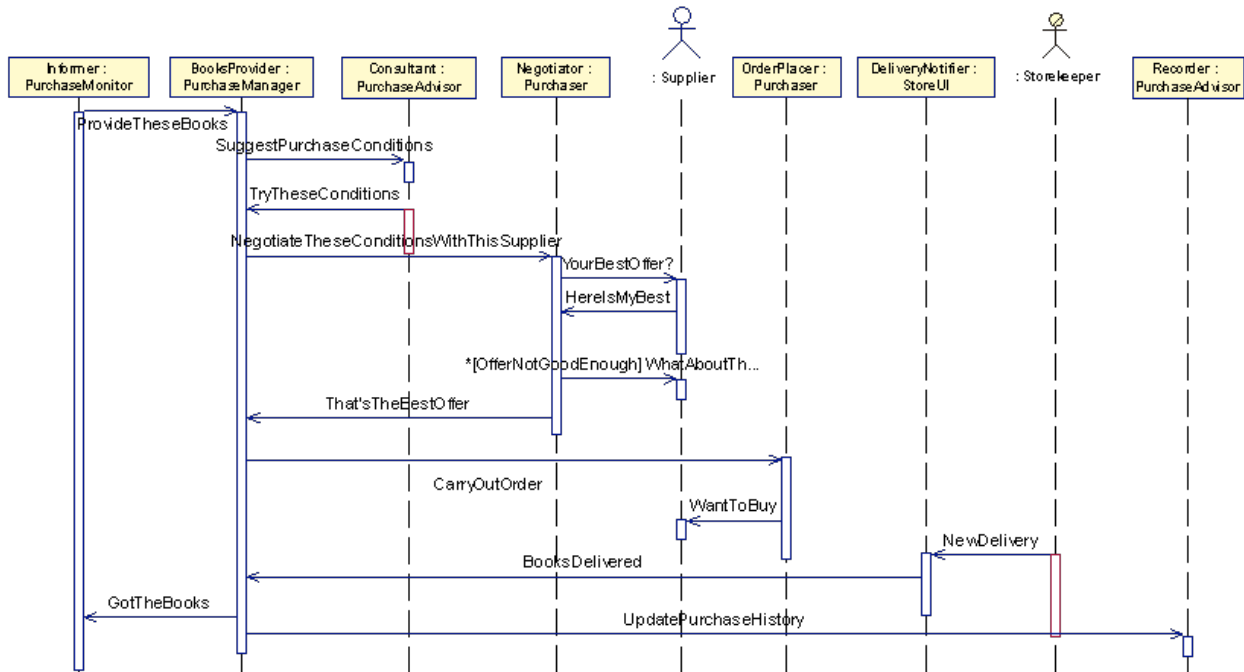
Stereotypes of relationships between use cases of different packages (agents) are converted to 'communicate' since different agents can interact only in this way. Direction of the relationships goes from the initiator to the participant.

The diagram is completed by the following information:

Agent	Description (Functional Req.)	Special Requirements
		<i>(security, other non funct. Req., pseudo-req, mobility, ...)</i>

4.1.3.4 Roles Identification

During this phase all the possible communication paths between agents are represented. A path describes a scenario of interacting agents working to achieve a required behaviour of the system. Each agent may belong to several scenarios, which are drawn by means of sequence diagrams in which objects are used to represent roles (collection of tasks performed by agent in pursuing a sub-goal).

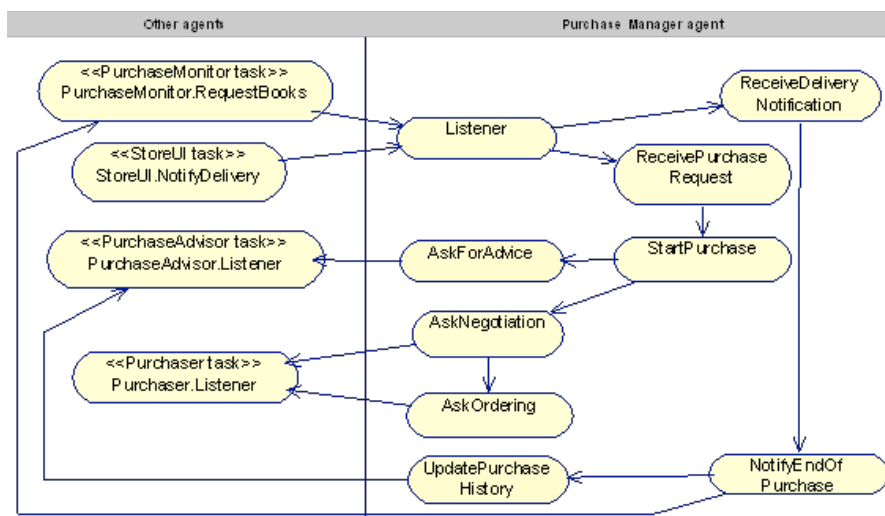


Each diagram is completed by the following information:

Role Name	Agent which plays it	Description	Responsibilities

4.1.3.5 Task Specification

Activity diagrams are used to show the behaviour of each agent pointing attention in what it is capable to do. Relationships between activities signify messages and communications between tasks of the same agent. A Task specification diagram represents the plan of the agent behavior. It shows the relationships among the external stimuli received by the agent and its behaviour (expressed in terms of fired tasks).



Each diagram is completed by the following information:

Task name	
Description	
Actions	
Data	
Behavior	

The Agency Society Phase

The Agent Implementation Phase

The Code Phase

The Deployment Phase

5 Work Products Dependencies

This diagram describes the dependencies among the different workproducts. For instance, the Communication Ontology diagram depends on the Domain Ontology diagram since during the communications parameters specification it is necessary to know the ontology elements (concepts, actions, predicates) defined in the previous phase.

