



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

Method Fragments from the PASSI process

rel. 0.1

Massimo Cossentino, Luca Sabatucci, Valeria
Seidita

RT-ICAR-03-21

December 2003



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*

Method Fragments from the PASSI process

rel. 0.1

Massimo Cossentino¹, Luca Sabatucci¹, Valeria
Seidita²

Rapporto Tecnico N.:
RT-ICAR-03-21

Data:
Dicembre 2003

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

² Università degli Studi di Palermo. Dipartimento di Ingegneria Informatica. Viale delle Scienze, 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

1	Domain Description	6
1.1	Introduction	6
1.2	Fragment Description	6
1.2.1	Portion of Process	7
1.3	Deliverables	9
1.3.1	Domain Description Diagram	9
1.3.2	System Requirements document	9
1.3.3	Glossary	9
1.3.4	Deliverables relationships with MAS model	10
1.4	Preconditions and concepts to be defined	10
1.5	Relationship with MAS meta-model	11
1.6	Guideline	11
1.7	Composition Guideline	11
1.8	Aspects of fragment	11
1.9	Dependency Relationships with other fragments	12
1.10	Glossary	12
2	Agents Identification	14
2.1	Introduction	14
2.2	Fragment Description	15
2.2.1	Portion of process	15
2.3	Deliverables	16
2.4	Preconditions and concepts to be defined	18
2.5	Relationship with MAS meta-model	19
2.6	Guideline	19
2.7	Composition Guideline	20
2.8	Aspects of Fragment	20
2.9	Dependency Relationships with other fragments	20
2.10	Glossary	20
3	Roles Identification	21
3.1	Introduction	21
3.2	Fragment Definition	22
3.3	Notation	24
3.3.1	Domain Description Diagram	24
3.4	Relation with MAS meta-model	25
3.5	Input/Output	26
3.6	Glossary	26
4	Task Specification	27
4.1	Introduction	27
4.2	Fragment Definition	28
4.3	Notation	29
4.3.1	Task Specification Diagram	29
4.4	Relation with MAS meta-model	30
4.5	Input/Output	31
4.6	Glossary	31
5	Domain Ontology Description	32
5.1	Introduction	32
5.2	Fragment Definition	33

5.3	Notation	35
1.1.	Domain Ontology Description Diagram	35
5.4	Relation with MAS meta-model	36
5.5	Input/Output	36
5.6	Glossary	37
6	Communication Ontology Description	38
6.1	Introduction	38
6.2	Fragment Definition.....	39
6.3	Notation	41
1.1.	Communication Ontology Description.....	41
6.4	Relation with MAS meta-model	42
6.5	Input/Output	43
6.6	Glossary	43
7	Roles Description	44
7.1	Introduction	44
7.2	Fragment Definition.....	45
7.3	Notation	46
1.1.	Roles Description Diagram	46
7.4	Relation with MAS meta-model	47
7.5	Input/Output	48
7.6	Glossary	48
8	Protocol Description.....	49
8.1	Introduction	49
8.2	Fragment Definition.....	50
8.3	Notation	51
1.2.	Protocol Description	51
8.4	Relation with MAS meta-model	52
8.5	Input/Output	53
8.6	Glossary	53
9	Multi-Agent Structure Definition (MASD)	54
9.1	Introduction	54
9.2	Fragment Definition.....	55
9.3	Notation	57
1.3.	Multi-Agent Structure Definition	57
9.4	Relation with MAS meta-model	58
9.5	Input/Output	59
9.6	Glossary	59
10	Multi-Agent Behaviour Description (MABD)	60
10.1	Introduction	60
10.2	Fragment Definition.....	61
10.3	Notation	63
1.4.	Multi-Agent Behaviour Description	63
10.4	Relation with MAS meta-model	64
10.5	Input/Output	65
10.6	Glossary	65
11	Single-Agent Structure Definition (SASD)	66
11.1	Introduction	66
11.2	Fragment Definition.....	67
11.3	Notation	68
1.5.	Single-Agent Structure Definition	68
11.4	Relation with MAS meta-model	69

11.5	Input/Output	69
11.6	Glossary	70
12	Single-Agent Behaviour Description (MABD).....	71
12.1	Introduction	71
12.2	Fragment Definition.....	72
12.3	Glossary	74
13	Code Reuse Library	75
13.1	Introduction	75
13.2	Fragment Definition.....	76
13.3	Notation	77
	Code Reuse	77
14	Code Completion Baseline	79
14.1	Introduction	79
14.2	Fragment Definition.....	80
15	Deployment Configuration	82
15.1	Introduction	82
15.2	Fragment Definition.....	83
15.3	Notation	84
	Deployment Configuration	84

1 Domain Description

Version: December 9, 2003

1.1 Introduction

The PASSI process is composed of five different phases: System Requirements, Agent Society, Agent Implementation, Code and Deployment.

Each phase produces a document that is usually composed aggregating the UML models and work products of the work definitions that are inside each phase .

We will define a method fragment Domain Description, extracted from PASSI System Requirements phase. The PASSI process is represented in the following figure. The System Requirements phase covers all the phases related to Req. Elicitation, analysis and agents/roles identification.

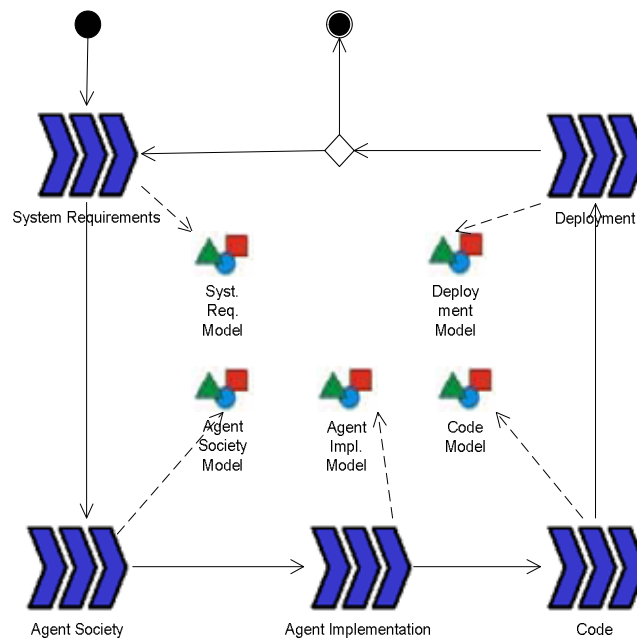


Fig. 1. The complete PASSI process

1.2 Fragment Description

We describe requirements in terms of use case diagrams. The Domain Description fragment, as a result, produces a functional description of the system composed of a hierarchical series of use case diagrams.

Starting from the PASSI System Requirement phase activities reported in the following Figure 2, let us consider the work definition “Domain Description” (the blue oval) whose aim is to identify the system requirements through the UML Domain Description Diagram and the (textual) Requirements document.

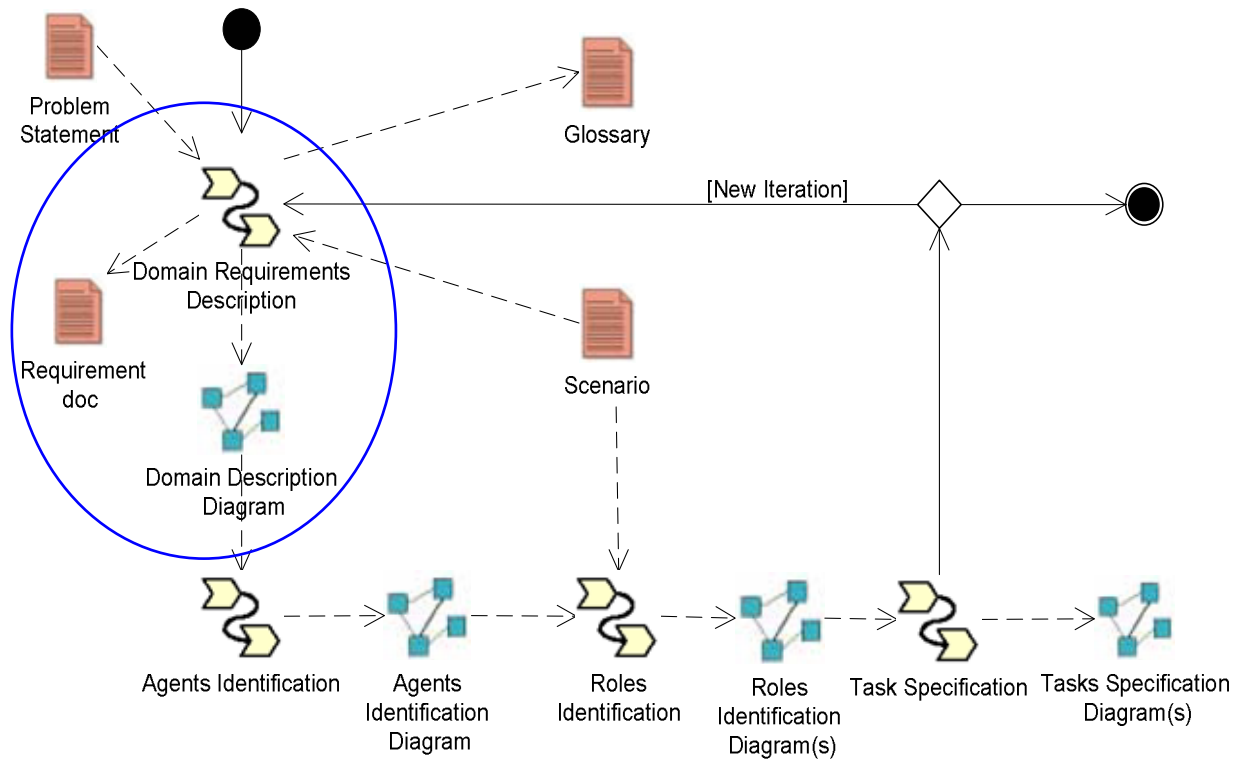


Fig.2. The System Requirements phase

1.2.1 Portion of Process

The process that is to be performed in order to obtain the result is represented in fig. 3 as a SPEM diagram

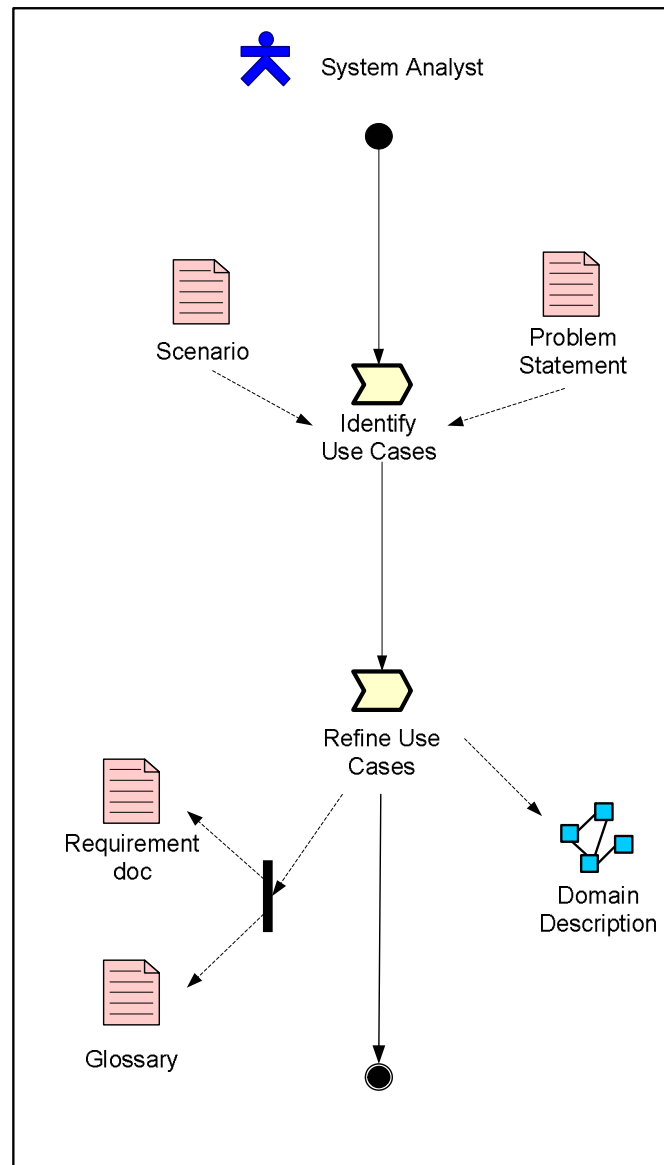


Fig.3. Requirements description fragment-Procedural aspect

Activities description:

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

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

1.2.1.1.1 System Analyst

He is responsible of:

1. Use cases identification during the DD sub-phase. **Errore. Il collegamento non è valido..**
2. Use cases refinement during the DD sub-phase. Use cases are refined with the help of a Domain Expert.

1.2.1.1.2 Domain Expert

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

1.3 Deliverables

1.3.1 Domain Description Diagram

Common UML use case diagram(s) are used to represent the system requirements.

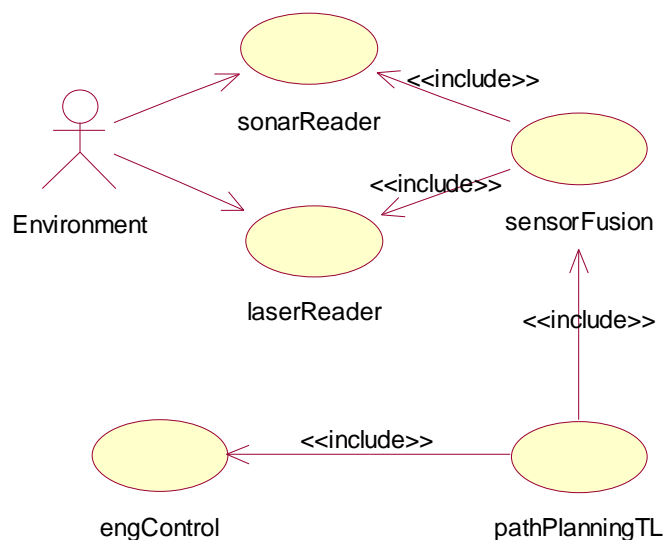


Fig. 4. The Domain Description Diagram

1.3.2 System Requirements document

It is a textual document containing the complete documentation of the use cases in terms of: name, participating actors, entry condition, flow of events, exit condition, exceptions and special requirements.

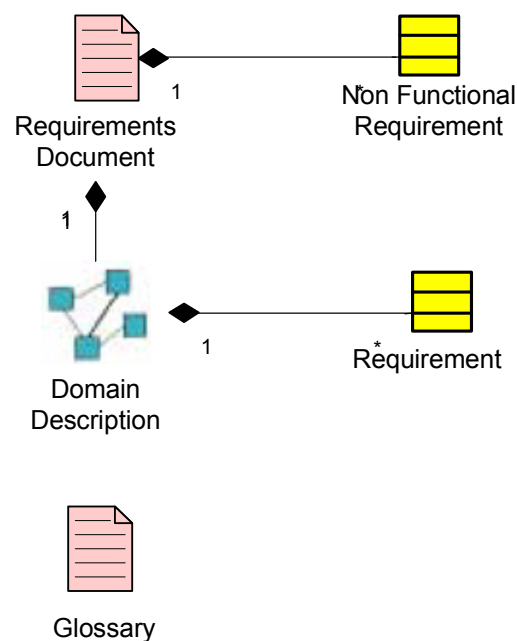
It also reports the non functional requirements identified for the system.

1.3.3 Glossary

A glossary of terms coming from the system domain

1.3.4 Deliverables relationships with MAS model

The following figure describes the structure of this fragment work products in relationship with the MAS model elements:



Note that the symbol:  represents an element of the MAS model.

In the Requirements document, use cases are documented in terms of: name, participating actors, entry condition, flow of events, exit condition, exceptions and special requirements.

1.4 Preconditions and concepts to be defined

Input, output and element to be designed in the fragment are detailed in the following table:

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

Problem Statement	Requirements (both functional and non functional)	System Requirements document
Scenarios		Domain Description diagram
		Glossary

1.5 Relationship with MAS meta-model

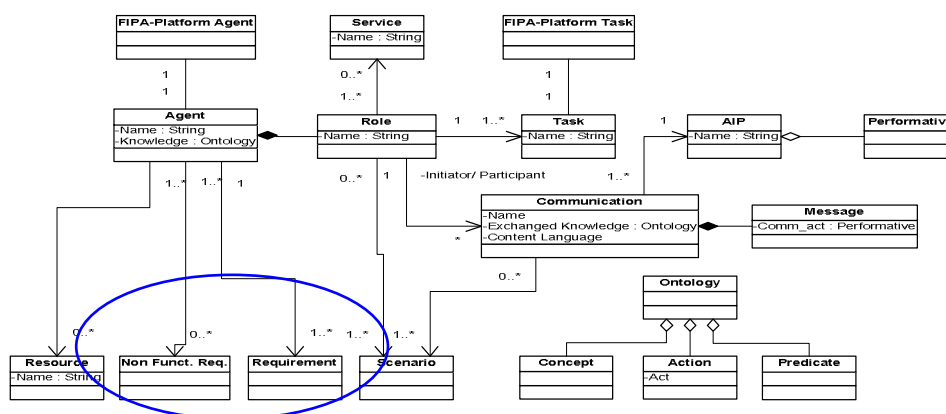


Fig.5. The MAS meta-model adopted in PASSI

This fragment refers to the MAS meta-model adopted in PASSI and contributes to define and describe functional and non functional requirements.

1.6 Guideline

None specific of an agent oriented approach

1.7 Composition Guideline

None

1.8 Aspects of fragment

None

1.9 Dependency Relationships with other fragments

In most approaches, this fragment is intended to be the first of the design process but it can also be preceded by a requirements elicitation fragment.

1.10 Glossary

This Fragment refers this terms:

Requirement - A requirement represents a feature that the system to be must exhibit, it can be a functional requirement that describes the interactions between the system and its environment independent of its implementation, or a non-functional requirement such as a constraint on the system (or a specific part of it) performance.

Scenario – A scenario represents a concrete sequence of interaction between the system and the actors.

2 Agents Identification

2.1 Introduction

The PASSI process is composed of five different phases: System Requirements, Agent Society, Agent Implementation, Code and Deployment.

Each phase produces a document that is usually composed aggregating the UML models and work products of the work definitions that are inside each phase .

We will define a method fragment we call “Agent Identification”, extracted from PASSI methodology whose process is completely represented in the following figure

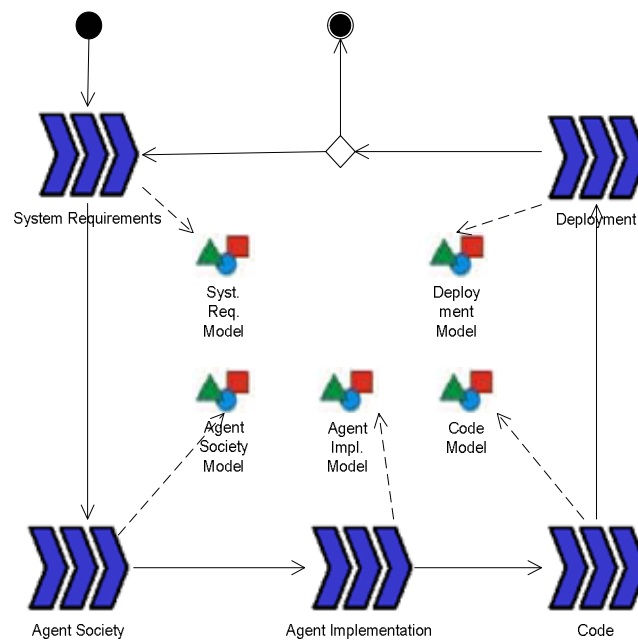


Fig. 1 The complete PASSI process

2.2 Fragment Description

The fragment here described is one of the peculiarities that distinguish the PASSI process from other approaches. The designer skill in capturing system requirements has been capitalized in order to produce an initial representation of the system functionalities (Domain Description Fragment) and now this model is used to identify agents and designate their responsibilities in terms of requirements to satisfy.

More in detail the System Requirements phase:

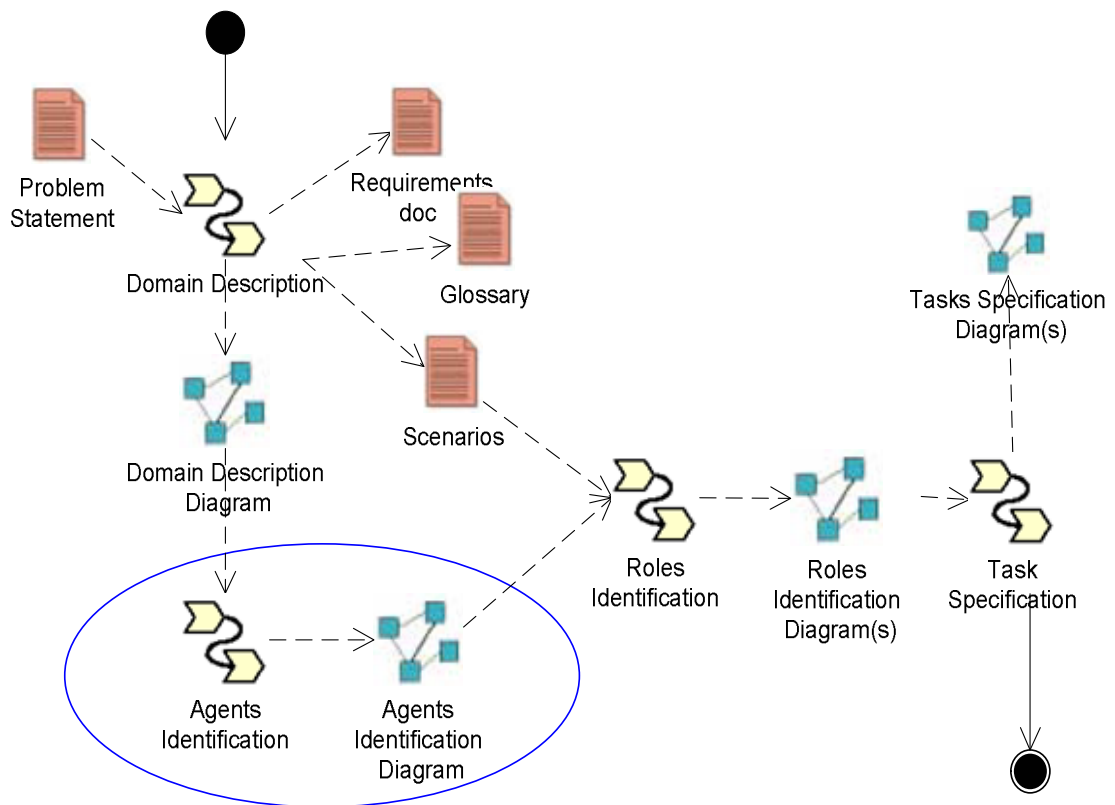


Fig.2 The System Requirements phase

Let us consider the “Agent Identification” sub-phase (the blue oval) .This fragment aims to identify all the agents involved in the system to be developed.

2.2.1 Portion of process

The process that is to be performed in order to obtain the result is represented in fig. 3 as a SPEM diagram

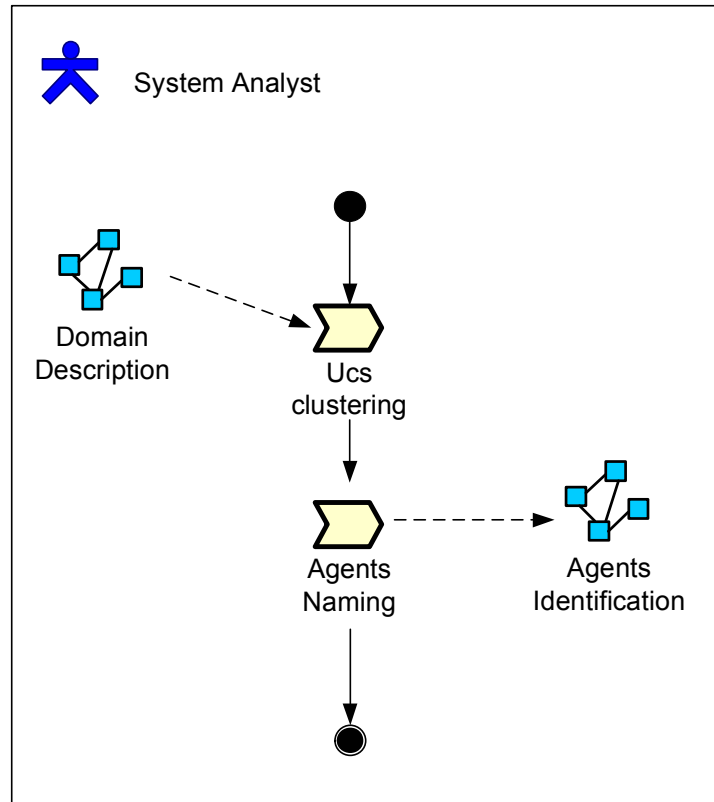


Fig 3 Agents Identification description fragment-Procedural aspect

Activities description:

Activity Name	Description	Roles involved
Use Cases Clustering	The System Analyst analyzes the use case diagrams resulting from the previous phase and attempts their clustering in a set of packages	System Analyst (perform)
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)

System Analyst Role

In this fragment, he is responsible of performing all of the above described activities

2.3 Deliverables

The resulting artifact of this phase is an use case diagram (Agent Identification diagram) reporting the same use cases of the previous phase now clustered inside a set of packages, each one representing one agent. As it is common, we represent external entities interacting with our system (people, devices, conventional software systems) as actors.

Relationships between use cases of the same agent follow the usual UML syntax and stereotypes, whereas relationships between use cases of different agents are stereotyped as *communication* as described below.

Our assumptions about agent interaction and knowledge play an important role in the understanding of this phase and they are as follows:

- An agent acts to achieve its objectives on the basis of its local knowledge and capabilities;
- Each agent can request help from other agents that are collaborative if this is not in contrast with their own objectives;
- Interactions between agents and external actors consist of *communication* acts; this implies that if some kind of *include/extend* relationship exists between two use cases belonging to different agents, this stereotype is to be changed to *communication* since a conversation is the unique interaction way for agents. This is a necessary extension of the UML specifications that allow communication relationships only among use case and actors. The direction of the relationships goes from the initiator of the conversation to the participant. This stereotype change is, however, not in contrast with the spirit of the definition of the communication relationship since an agent is a proactive entity that could initiate an interaction just like an actor. An exception exists to this change in the relationship stereotype: it is possible that an agent in requiring some collaboration from another will not use a communication but instead will instantiate the other one; in this case, that is however not frequent, we use an *instantiate* stereotype to distinguish this situation from the others.
- An agent's knowledge can increase through communication with other agents or exploration of the real world.

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).

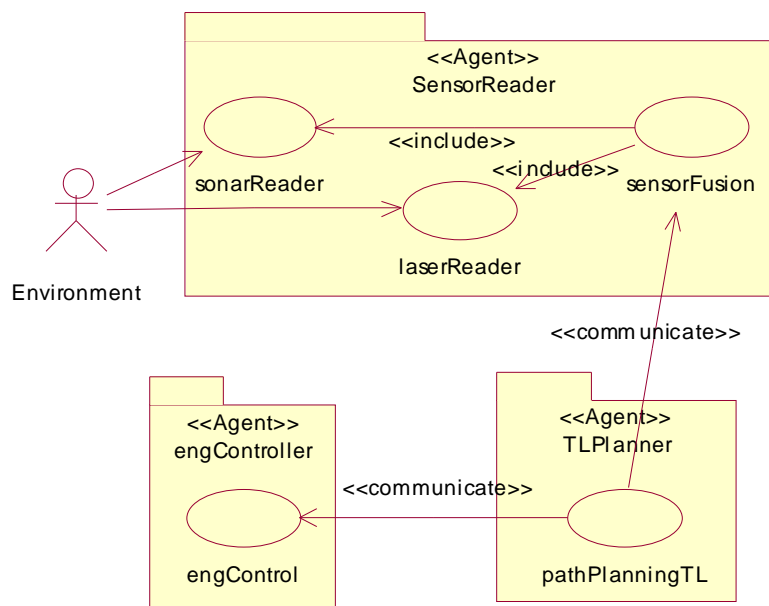
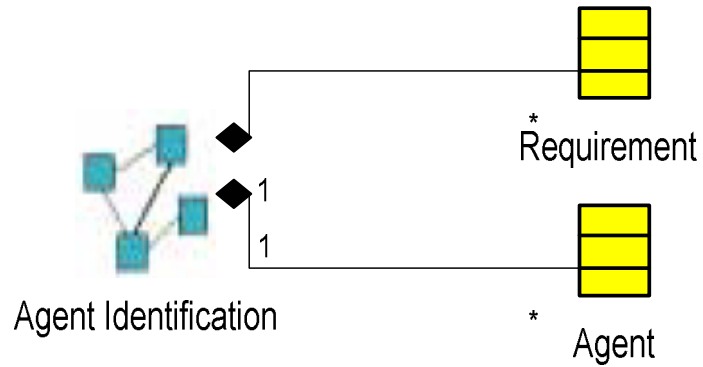



Fig. 4 The Agent Identification Diagram

The following figure describes the structure of the Agent Identification work product:



Note that the symbol:  represents an element of the MAS model.

The agent element is defined only by specifying its name and relationships with existing requirements.

2.4 Preconditions and concepts to be defined

Input, output and element to be designed in the fragment are detailed in the following table

Input	To Be Designed	Output
Use Case diagram from the system requirements elicitation (Domain Description in PASSI)	Agent	Agent Identification (UML diagram)

2.5 Relationship with MAS meta-model

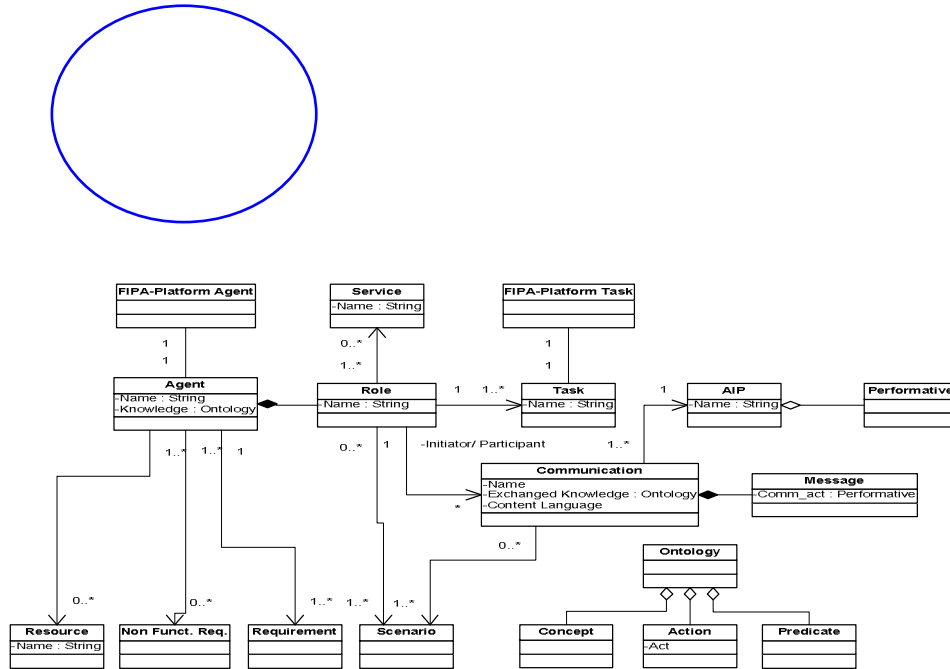


Fig5.. The MAS meta-model adopted in PASSI

This fragment refers to the MAS meta-model adopted in PASSI and contributes to define the agent element of it.

2.6 Guideline

This phase is usually performed by a system analyst whose work is described in the SPEM activity diagram reported in Figure 3; the first activity consists in analyzing the use case diagrams resulting from the previous phase and attempt their clustering in a set of packages. Not precise rules exist to guide this operation but some guidelines could be drawn:

- It is better to group use cases that have inner logical commonalities because probably this will bring to implementations that have several common elements
- Data flow could represent an important problem for intrinsically distributed systems like MASs and therefore it could be useful to group together use case that will probably exchange a significant amount of data
- This activity produces a sort of architectural decomposition of the future system (at least at the functionality level but being each agent a consistent element of the implementation this partition also guides some kind of structural decomposition for the following solution). This suggests the observance of some common sense rules for agents identification:
 - When possible (and if evident at this stage), agents that could be deployed in special devices (like PDA or cellular phones) should be fine grained in order to optimize their performance.

- Human interaction functionalities could be assigned to specific agents in order to prepare the option for a multi-device implementation (web-based, cell phone interfaces, and so on) via different categories of agents implementing these functionalities.
- In order to facilitate agents mobility, functionalities that strictly depend on hardware devices or databases should that could not be accessed by everywhere should be divided by the remaining part of the system eventually using a wrapping solution.

2.7 Composition Guideline

The fragment can be used after a functional-oriented requirements elicitation (performed with use case diagrams) in order to identify a system decomposition into agents. It is not good for goal-oriented approaches.

2.8 Aspects of Fragment

Behind this fragment there is only the basic assumption that the system is to be modelled in terms of (functional) requirements.

2.9 Dependency Relationships with other fragments

None specific, obviously as already discussed in section 2.7 and 2.4, an use case diagram representing the system requirements is necessary as an input.

2.10 Glossary

Agent Identification Fragment uses this list of model elements:

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.

Requirement - A requirement represents a feature that the system to be must exhibit, it can be a functional requirement that describes the interactions between the system and its environment independent of its implementation, or a non-functional requirement such as a constraint on the system (or a specific part of it) performance.

3 Roles Identification

3.1 Introduction

The PASSI process is composed of five different phases: System Requirements, Agent Society, Agent Implementation, Code and Deployment.

Each phase produces a document that is usually composed aggregating the UML models and work products of the work definitions that are inside each phase .

We will define a method fragment Roles Identification, extracted from PASSI methodology whose process is completely represented in the following figure

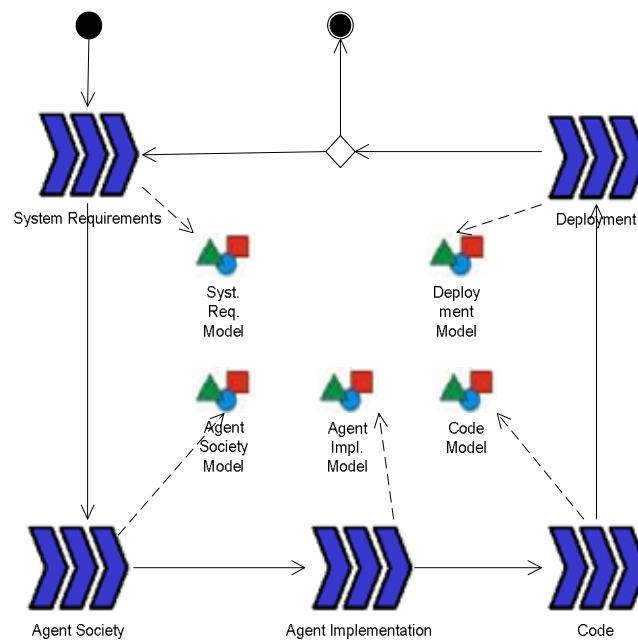


Fig. 1 The complete PASSI process

3.2 Fragment Definition

More in detail the System Requirements phase:

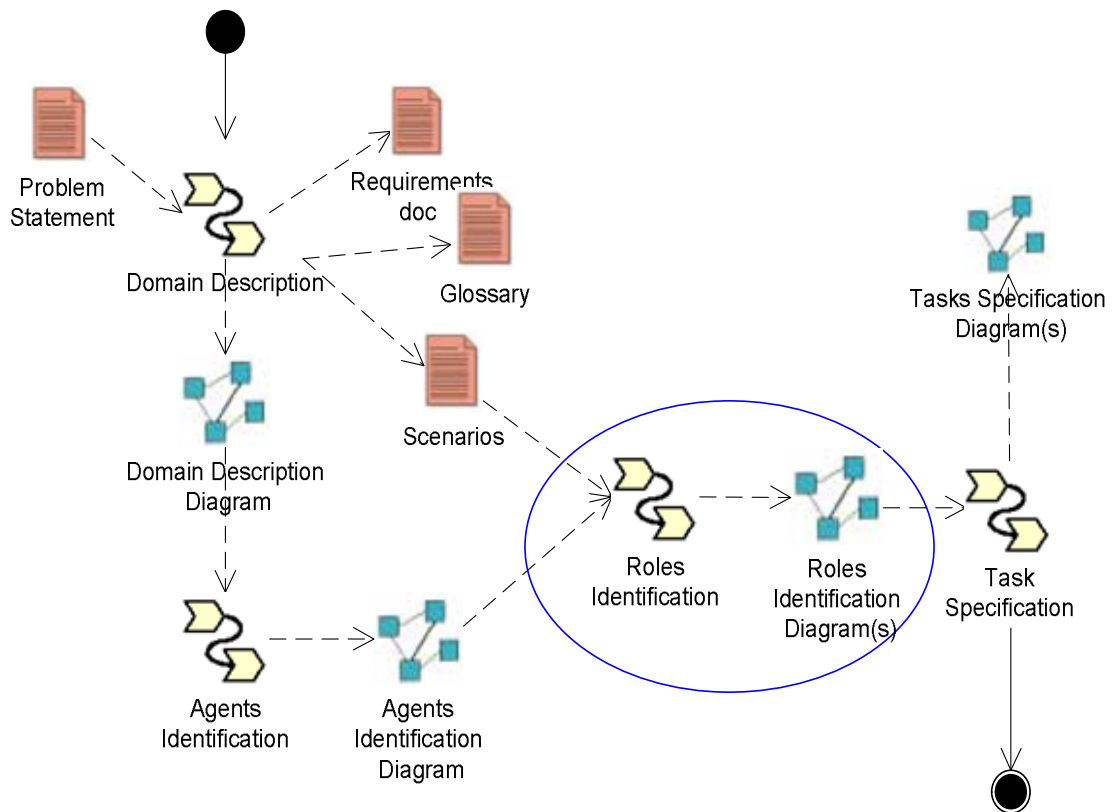


Fig.2 The System Requirements phase

Let us consider the work definition “Roles Identification” (the blue oval) whose aim is to describe all possible scenario of interacting agents working to achieve a required behaviour of the system. The UML Model of this portion of process, Roles Identification Diagram, is designed following a standard UML notation.

The process that is to be performed in order to obtain the result is represented in fig. 3 as a SPEM diagram

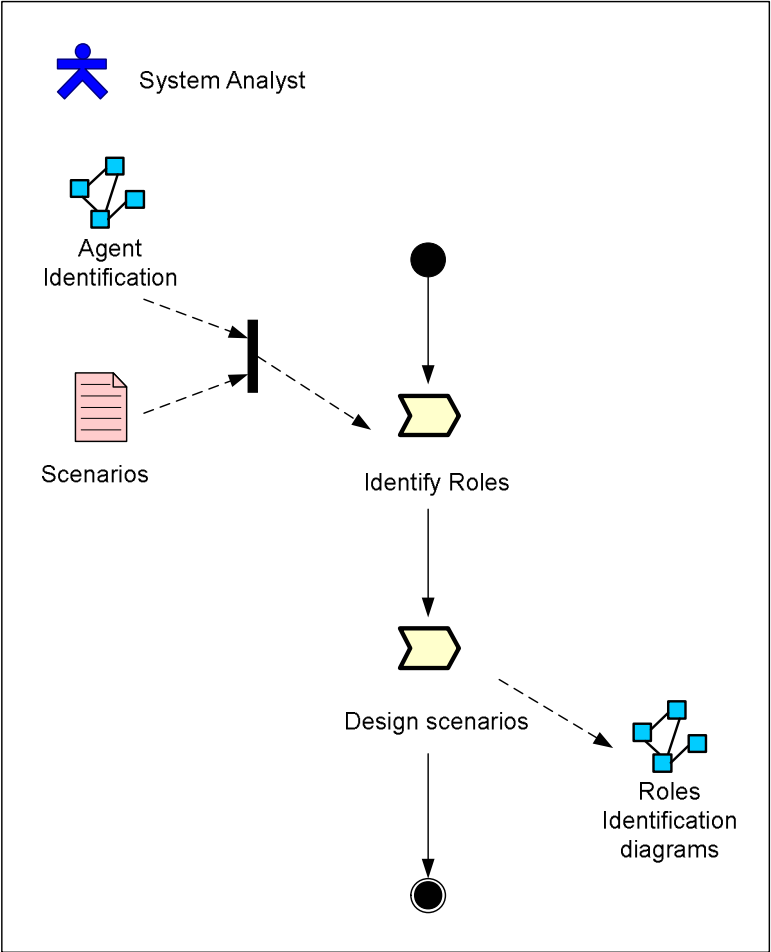


Fig.3 Roles Identification fragment-Procedural aspect

3.3 Notation

3.3.1 Domain Description Diagram

Sequence diagrams describe all the possible communication paths between agents. 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 symbolize roles.

The name of each class is in the form: <role name>:<agent name>

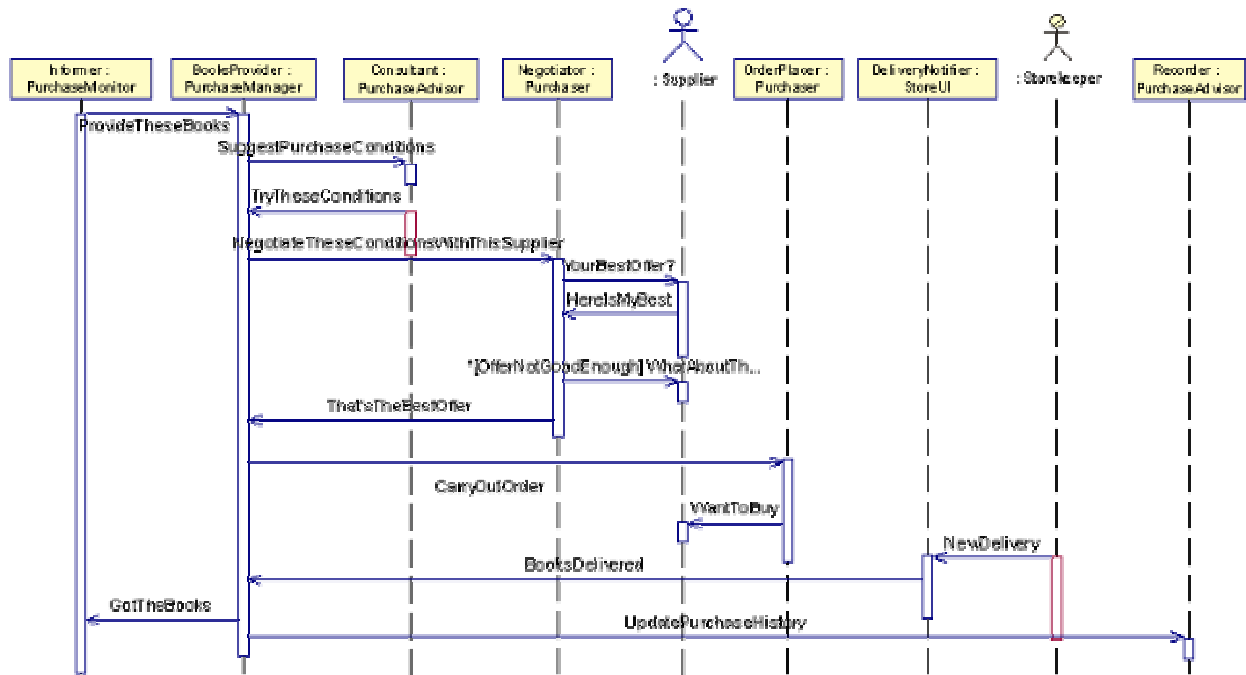


Fig. 4 The Role Identification Diagram

3.4 Relation with MAS meta-model

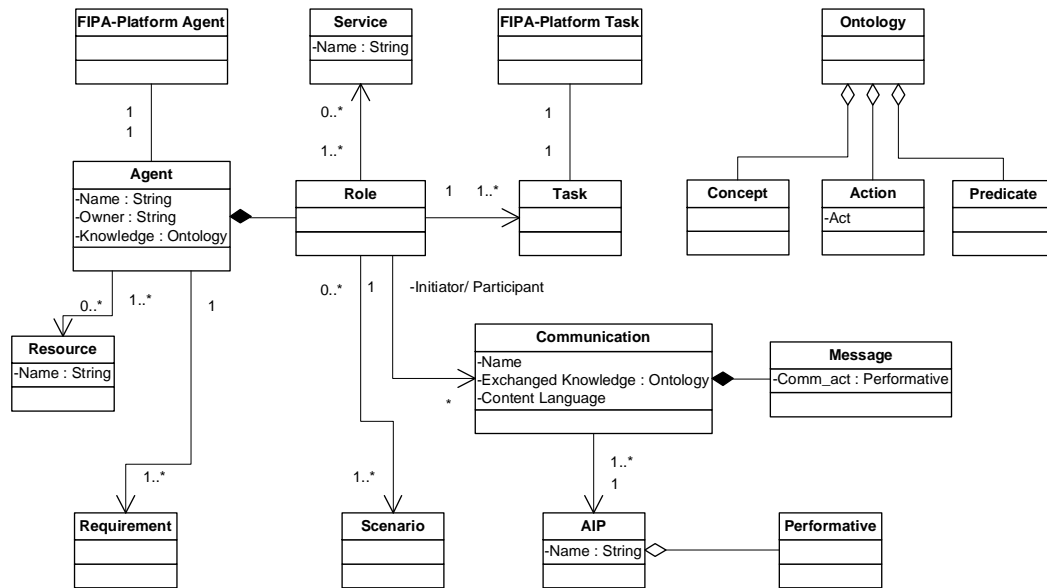


Fig.5. The MAS meta-model adopted in PASSI

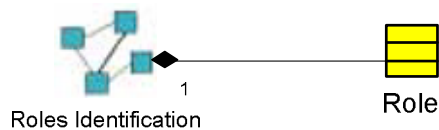
This fragment refers to the MAS meta-model adopted in PASSI and contributes to define and describe the concepts of role in relation with it .

The following figure describes the structure of the different work products, in the fragment, and their composition with respect to the MAS model.

Here the symbol:



represents an element of the MAS model .



3.5 Input/Output

Input, output and element to be designed in the fragment are detailed in the following table

Input	To Be Designed	Output
Scenario, Agent Identification	Role	Role Identification

3.6 Glossary

Roles Identification Fragment uses this list of model element:

Role – A role is a collection of tasks performed by agent in pursuing a sub-goal; an agent could play one or more roles in the system. Each role describes an aspect of agent life cycle and it is often related to a service offered by the agent to the society or to the achievement of one of its goals.

4 Task Specification

4.1 Introduction

The PASSI process is composed of five different phases: System Requirements, Agent Society, Agent Implementation, Code and Deployment.

Each phase produces a document that is usually composed aggregating the UML models and work products of the work definitions that are inside each phase .

We will define a method fragment Task Specification, extracted from PASSI methodology whose process is completely represented in the following figure

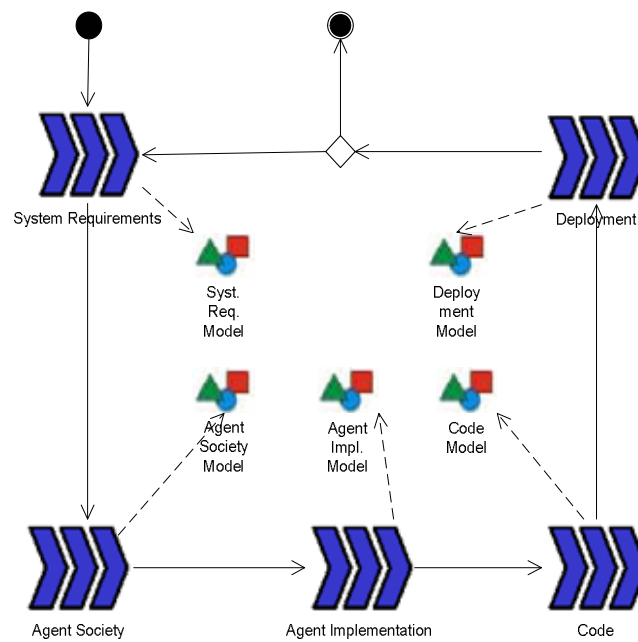


Fig. 1 The complete PASSI process

4.2 Fragment Definition

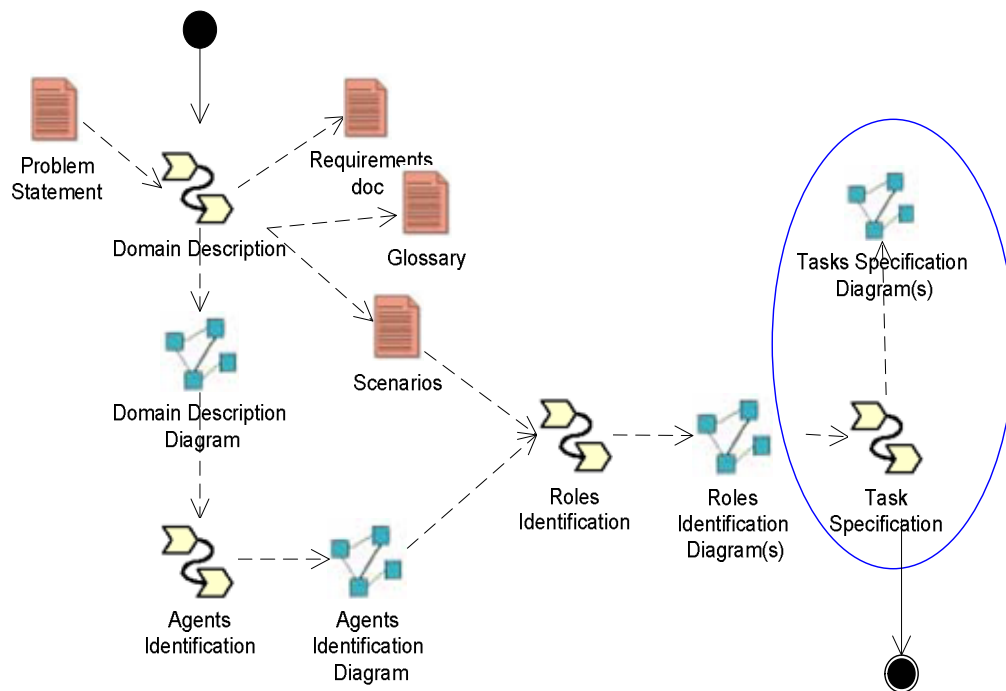


Fig.2 The System Requirements phase

This fragment aims is to describe the behaviour of each agent . The UML Model of this portion of process, Task Specification Diagram, is designed following a standard UML notation.

The process that is to be performed in order to obtain the result is represented in fig. 3 as a SPEM diagram

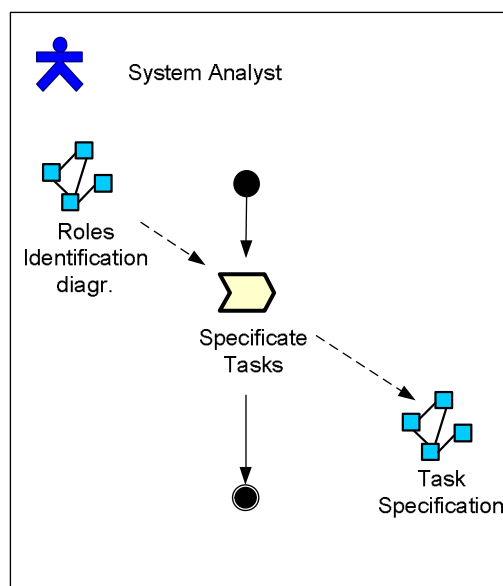


Fig.3 Task Specification fragment-Procedural aspect

4.3 Notation

4.3.1 Task Specification Diagram

One different activity diagram is drawn for each agent. This diagram describes how the agent can use its tasks to execute its plans.

Each diagram is composed of two swimlanes and contains activities that usually represent tasks of the agent. The right swimlane contains tasks of the agent we are describing (Purchase Manager in the figure above), in the left one we can find tasks of other agents that interact with this one.

Transitions in the same swimlane describe the flow of control from different tasks while transitions from one swimlane to the other represent communications.

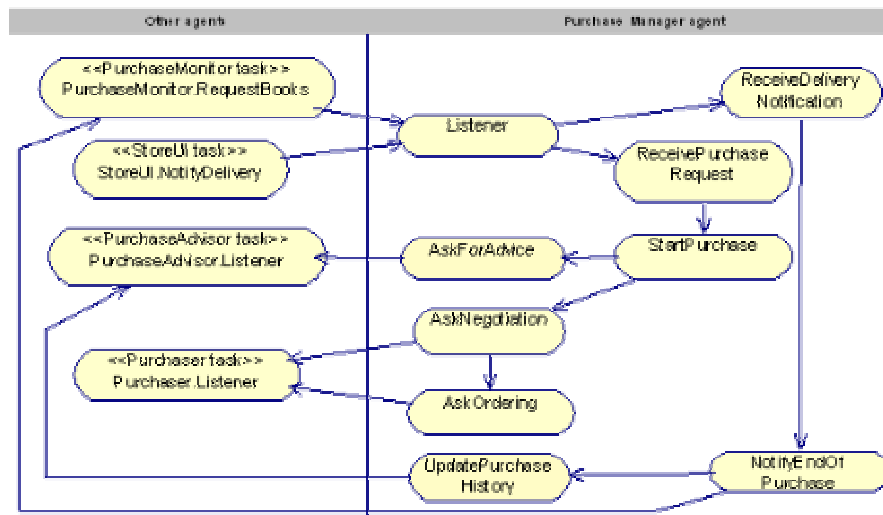


Fig. 4 The Task Specification Diagram

4.4 Relation with MAS meta-model

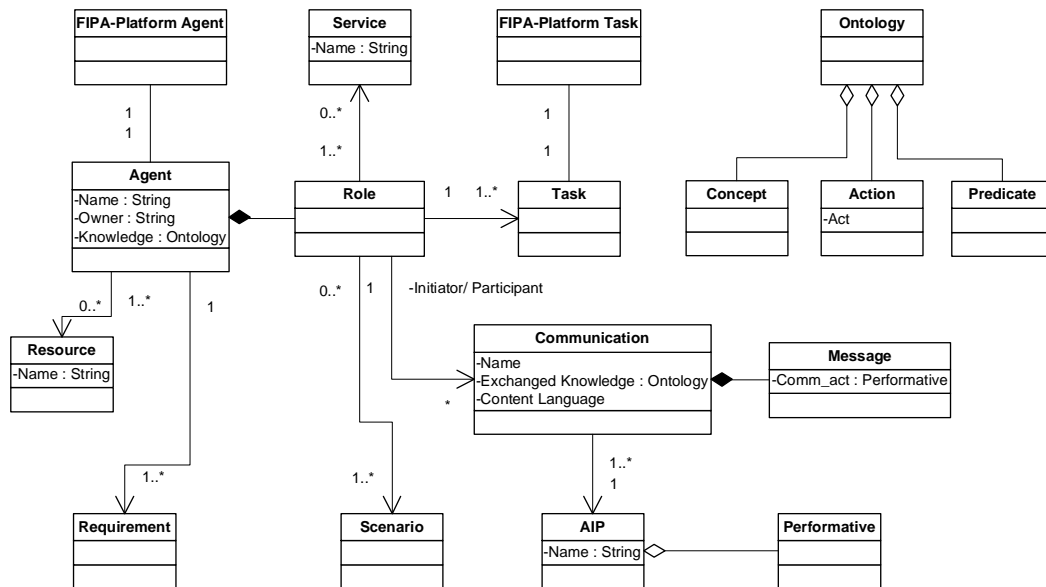


Fig.5. The MAS meta-model adopted in PASSI

This fragment refers to the MAS meta-model adopted in PASSI and contributes to define and describe a set of concepts in relation with it : requirement, scenarios.

The following figure describes the structure of the different work products, in the fragment, and their composition with respect to the MAS model.

Here the symbol:



represents an element of the MAS model .

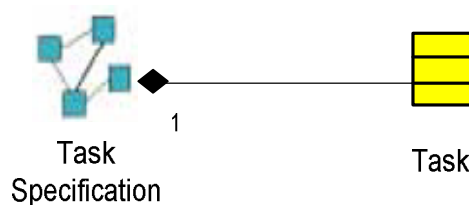


Fig.6. MAS Metamodel concepts

4.5 Input/Output

Input, output and element to be designed in the fragment are detailed in the following table

Input	To Be Designed	Output
Roles Identification	Task	Task Specification
Scenario	Requirement	Requirement doc

4.6 Glossary

Requirement Fragment uses this list of model element:

Task – It is a logical unit of individual or interactive behaviour. An agent uses tasks to execute its plan(s). Each task is an entity that aims to reach a sub-goal (for example dealing with a communication or executing some transformations on a specific resource).

5 Domain Ontology Description

5.1 Introduction

The PASSI process is composed of five different phases: System Requirements, Agent Society, Agent Implementation, Code and Deployment.

Each phase produces a document that is usually composed aggregating the UML models and work products of the work definitions that are inside each phase .

We will define a method fragment Domain Ontology Description, extracted from PASSI methodology whose process is completely represented in the following figure

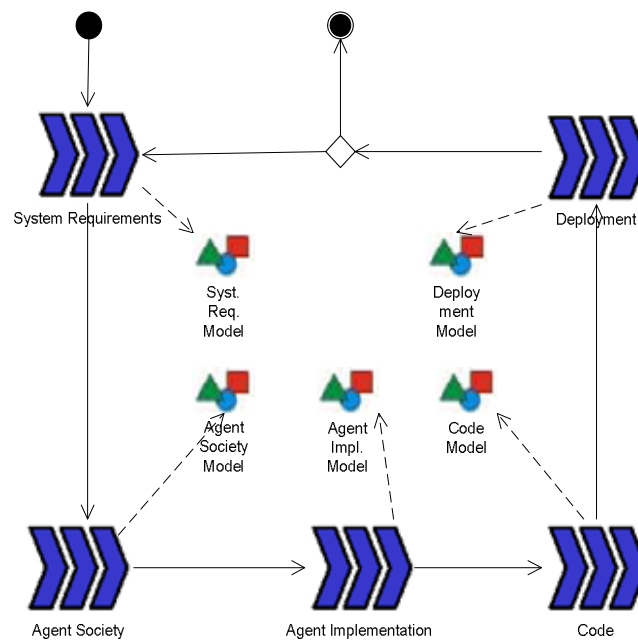


Fig. 1 The complete PASSI process

5.2 Fragment Definition

Consider the PASSI process (Fig. 1) and the phase “Agent Society” with its outcome “Agent Society Model”,

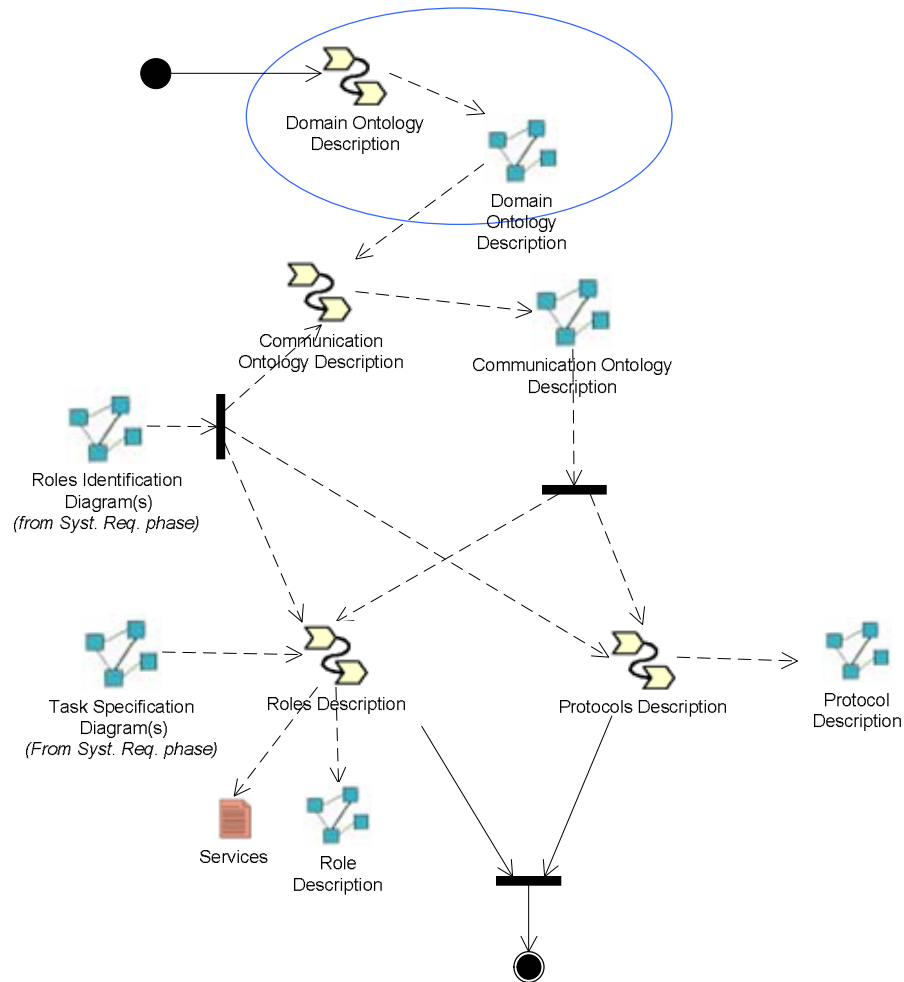


Fig.2 The Agent Society Phase

Let us 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. 3 as a SPEM diagram

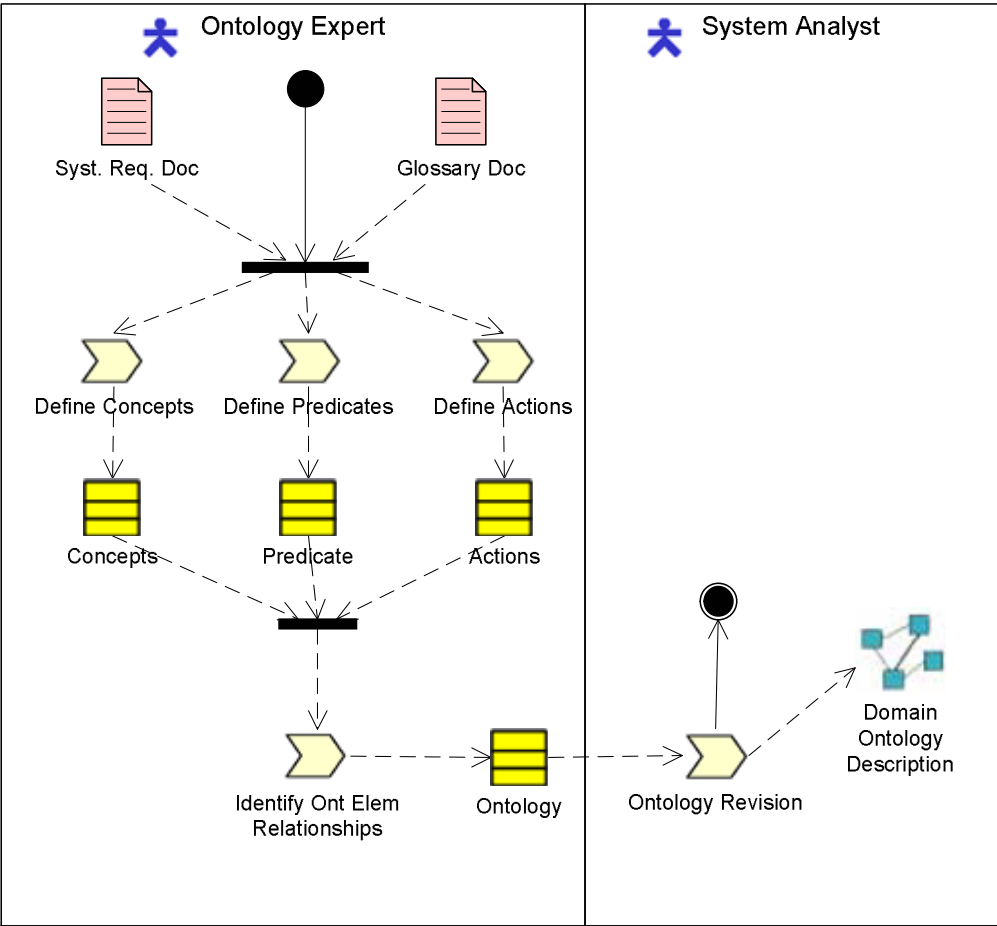


Fig 3 Domain Ontology description fragment-Procedural aspect

5.3 Notation

5.3.1 Domain Ontology Description Diagram

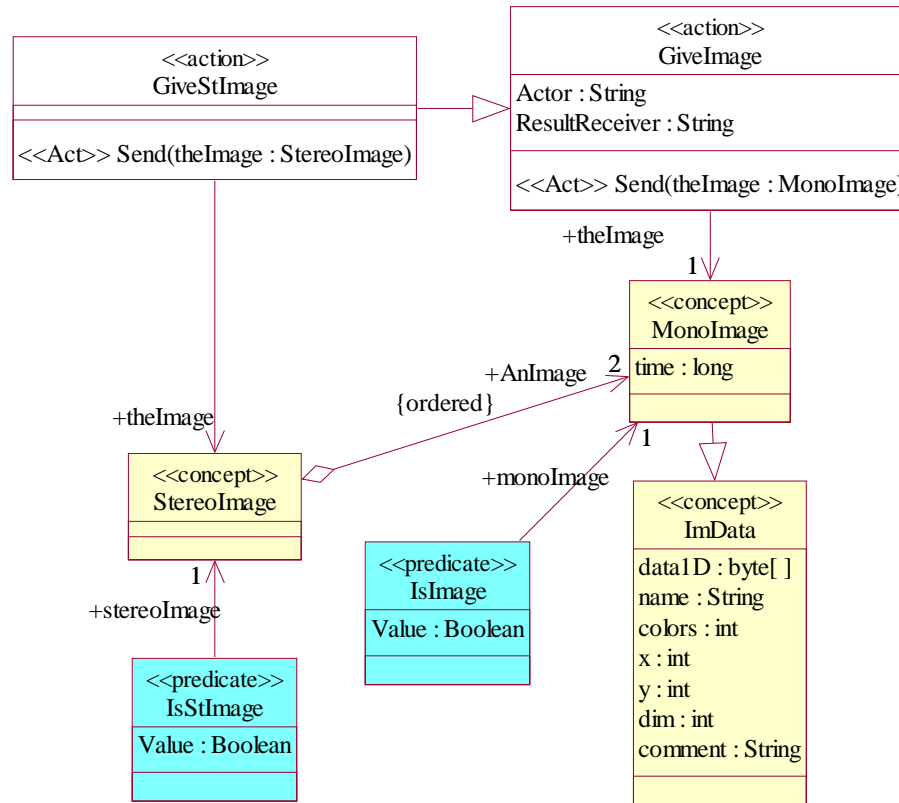


Fig. 4. The Domain Ontology Description diagram

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 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.

In the previous figure we have a small portion of a robotic vision ontology. *MonoImage* is a specialization of the *ImData* concept with a time stamp (grabbing time). The ordered aggregation of

two mono images gives the *StereoImage*. We define the *GiveImage* action in order to allow a robot to ask for an image. The image should be provided by the *Actor* and sent to the *ResultReceiver* (both agents). Predicates are also defined in relation to some existing concepts (*IsImage*, *IsStImage*).

5.4 Relation with MAS meta-model

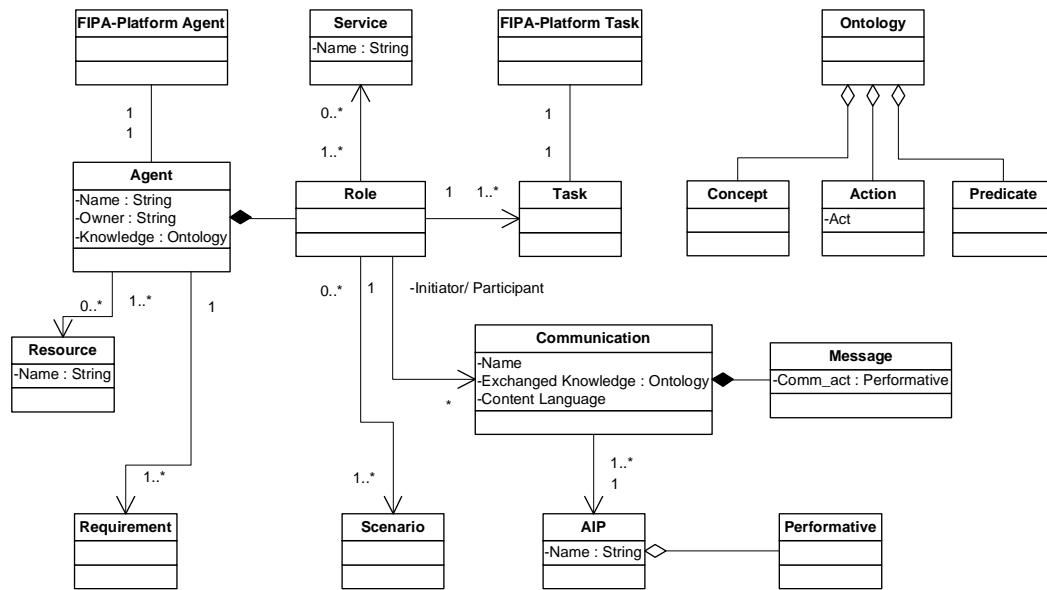


Fig.5. The MAS meta-model adopted in PASSI

This fragment refers to the MAS meta-model adopted in PASSI and contributes to define and describe a set of concepts in relation with it : ontology (concept, action, predicate)

5.5 Input/Output

Input, output and element 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	Ontology (MAS meta-model component)
	Predicates	Ontology (MAS meta-model component)
	Ontology elements Relationships	D.O.D. diagram

5.6 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.

6 Communication Ontology Description

Version: December 10, 2003

6.1 Introduction

The PASSI process is composed of five different phases: System Requirements, Agent Society, Agent Implementation, Code and Deployment.

Each phase produces a document that is usually composed aggregating the UML models and work products of the work definitions that are inside each phase .

We will define a method fragment Communication Ontology Description, extracted from PASSI methodology whose process is completely represented in the following figure

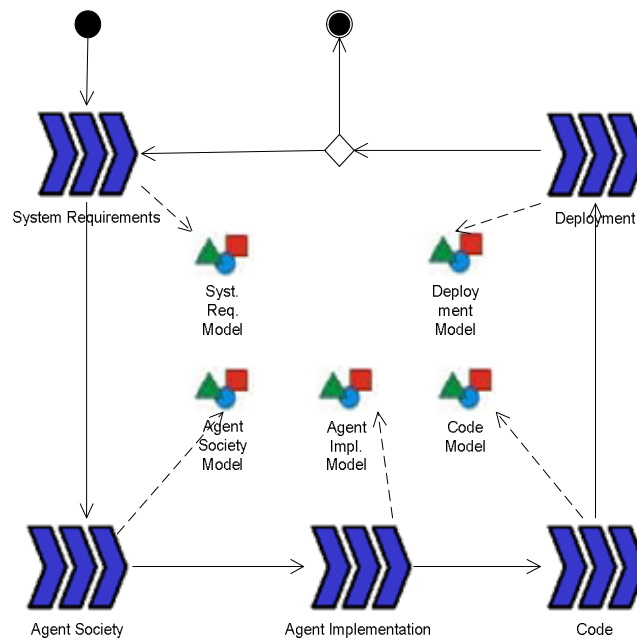


Fig. 1. The complete PASSI process

6.2 Fragment Definition

Consider the PASSI process (Fig. 1) and the phases “Agent Society” with its outcome “Agent Society Model”,

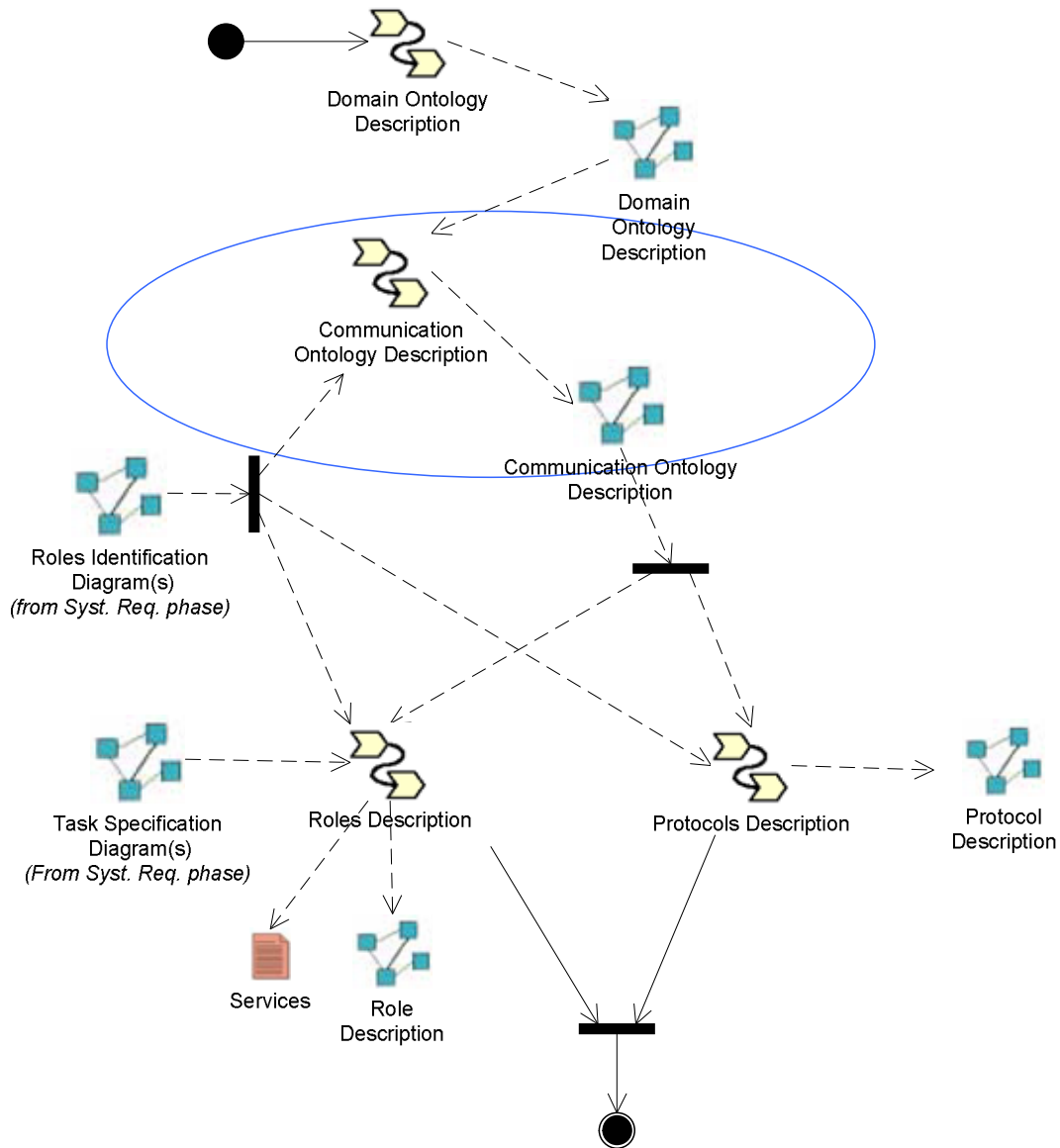


Fig. 2. The Agent Society phase

Let us consider the work definition “Communication Ontology Description” with their outcome (UML model “Communication Ontology Description”).

This fragment aims to model the social interactions and dependencies among the agents involved in the solution and the sequent agent society aspects are faced: communication and role description. The UML Model of this portion of process: Communication Ontology Description Diagram is designed following a standard UML notation. The process that is to be performed in order to obtain the result is represented in Fig.3 as SPEM diagram

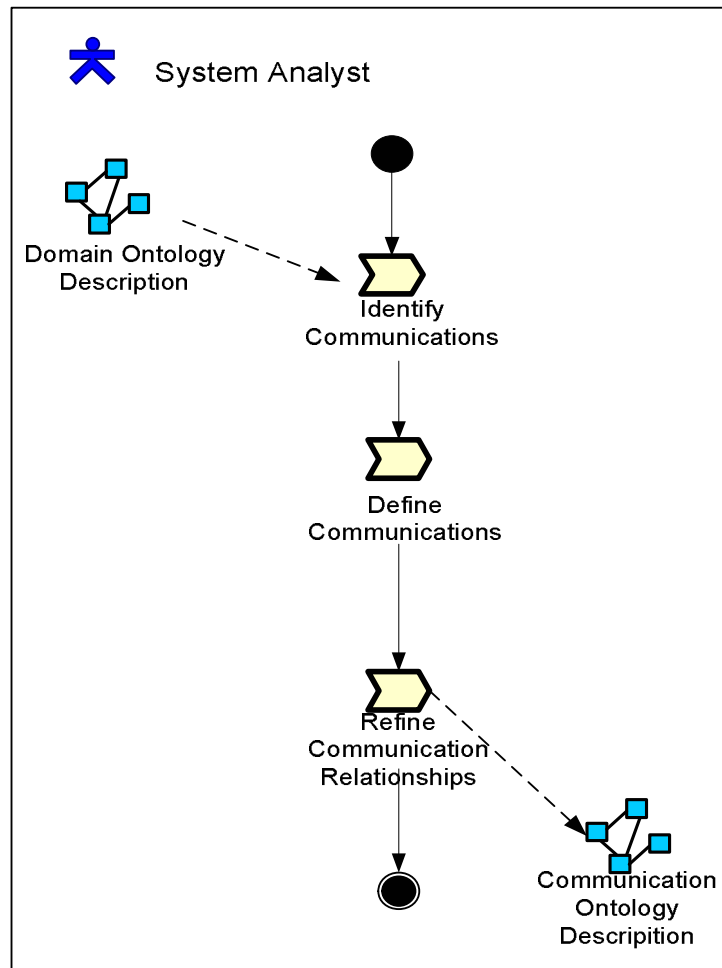


Fig. 3. Communication Ontology Description fragment-Procedural aspect

6.3 Notation

6.3.1 Communication Ontology Description

The COD diagram is a class diagram and it is mainly composed of two elements: agents and communications.

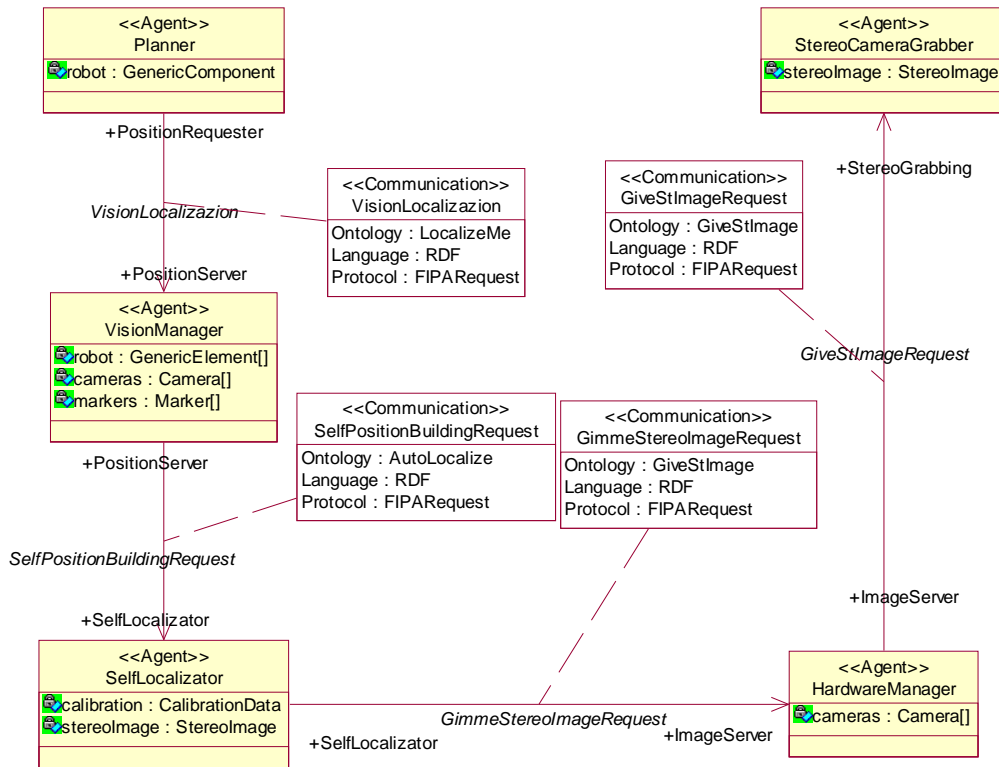


Fig.4. Communication Ontology Description diagram

Each agent (fill colour: yellow) is described in terms of its knowledge (pieces of the ontology described in the previous diagram). 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.

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 a unique name (also reported in the relationship among the two agents) and it is described by the *ontology*, *language* and *protocol* fields.

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

In the previous diagram we can see that the *HardwareManager* agent asks for a stereo image to the *StereoCameraGrabber* agent with the *GiveStImageRequest* communication.

This communication refers to the *GiveStImage* action defined in the previous seen DOD diagram, uses the RDF content language and the FIPA Request interaction protocol.

6.4 Relation with MAS meta-model

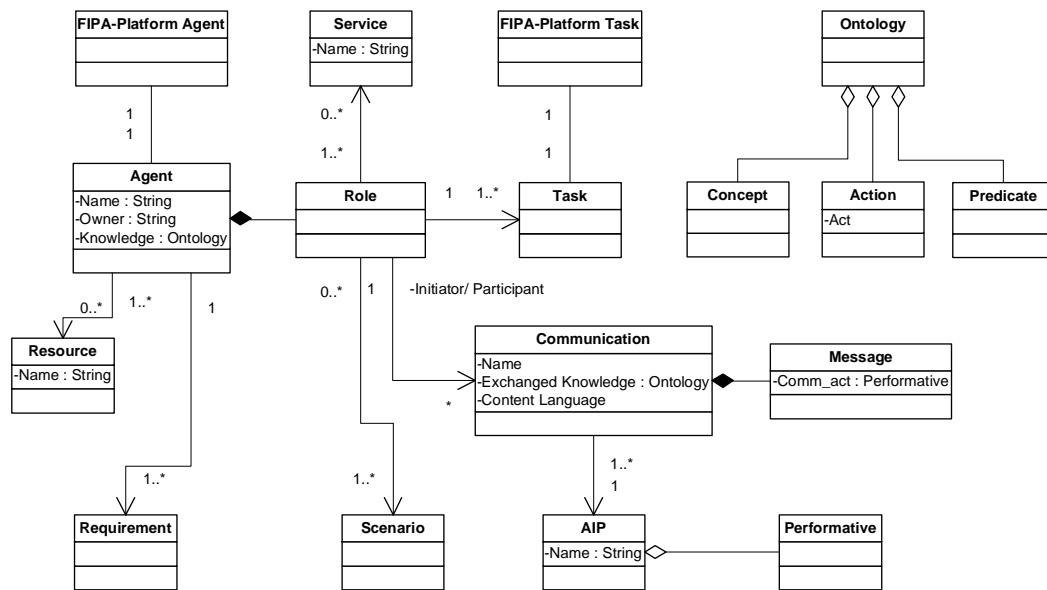


Fig.5. The MAS meta-model adopted in PASSI

This fragment refers to the MAS meta-model adopted in PASSI and contributes to define and describe a set of concepts in relation with it : communication and messages.

The following figure describes the structure of the different work products, in the fragment, and their composition with respect to the MAS model.

Here the symbol:



represents an element of the MAS model .

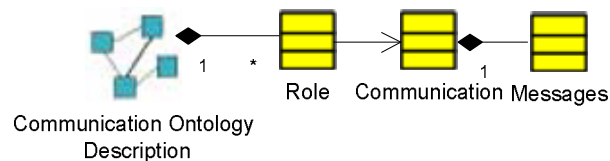


Fig.6. MAS Metamodel concepts

6.5 Input/Output

Input, output and element to be designed in the fragment are detailed in the following table

Input	To Be Designed	Output
D.O.D.	Communication and messages	Communication Ontology Description

6.6 Glossary

The Communication Ontology Description Fragment uses this list of model element:

Communication – a communication is an interaction between two agents. Each communication is described in terms of: ontology (related to the part of knowledge exchanged by the agents), content language and interaction protocol.

Message - an individual unit of communication between two or more agents that point out the standard FIPA message format.

7 Roles Description

Version: January 13, 2003

7.1 Introduction

The PASSI process is composed of five different phases: System Requirements, Agent Society, Agent Implementation, Code and Deployment.

Each phase produces a document that is usually composed aggregating the UML models and work products of the work definitions that are inside each phase .

We will define a method fragment Roles Description, extracted from PASSI methodology whose process is completely represented in the following figure

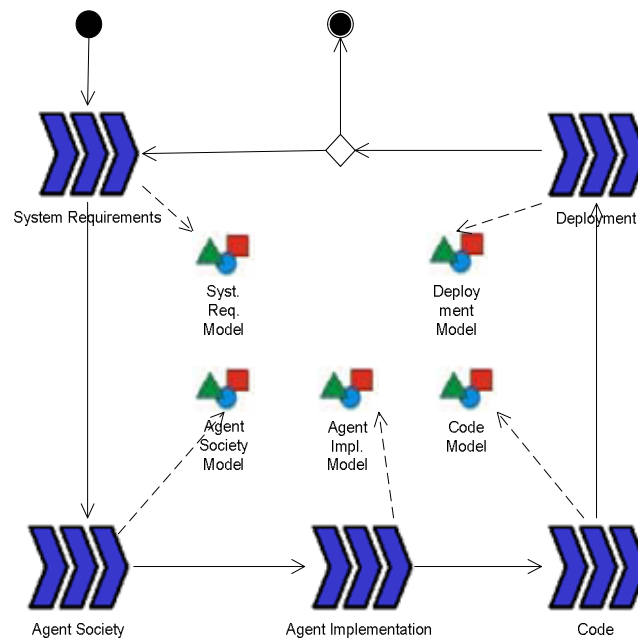


Fig. 1 The complete PASSI process

7.2 Fragment Definition

Consider the PASSI process (Fig. 1) and the phase “Agent Society” with its outcome “Agent Society Model”, the order of activities performed in this fragment is showed in the following SPEM diagram

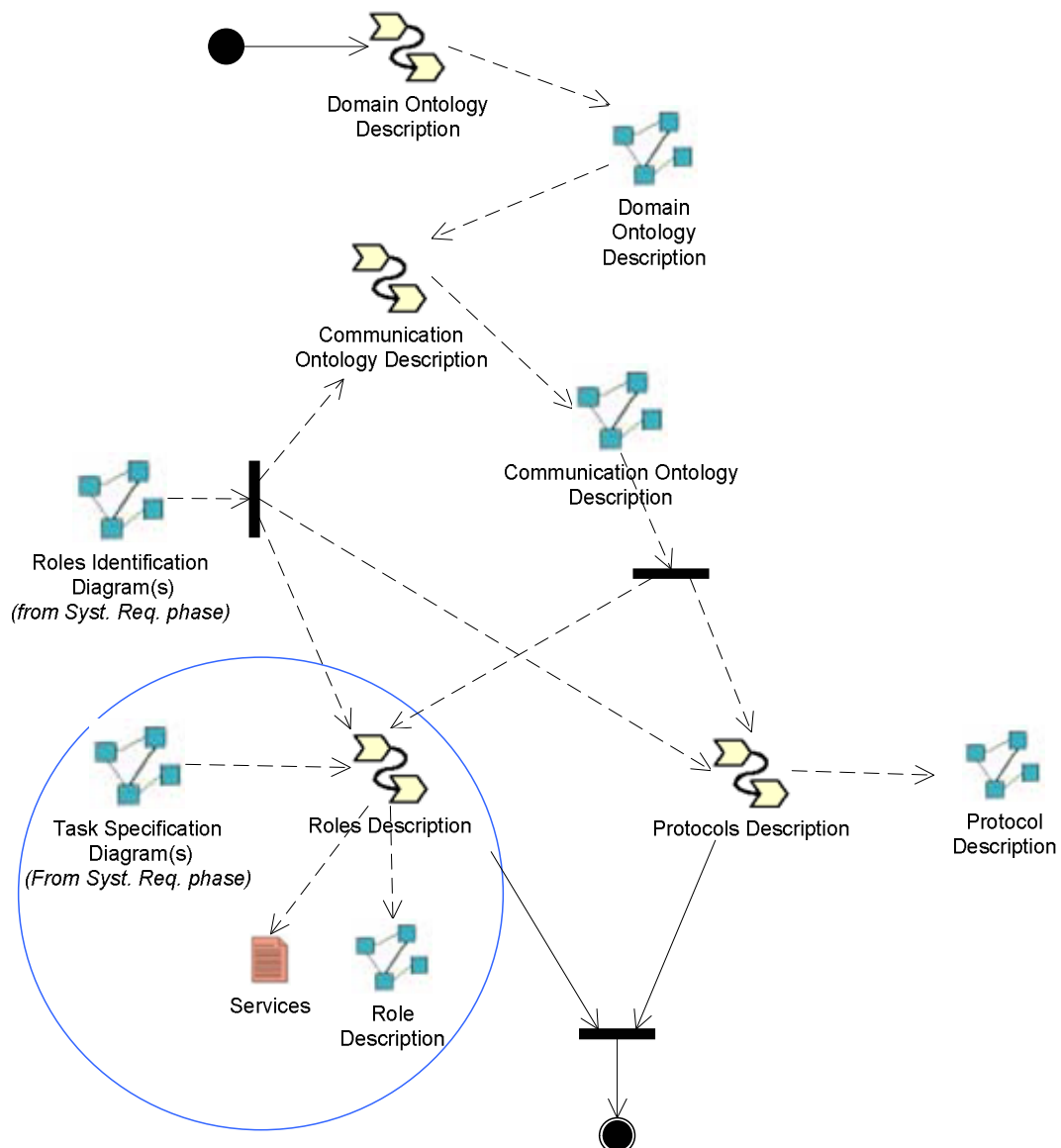


Fig.2 The Agent Society phase

This fragment's purpose is to model the lifecycle of each agent, looking at the roles it can play, at the collaboration it needs and the communications in which it participates.

The UML Model of this portion of process, Roles Description Diagram, is designed following a standard UML notation.

7.3 Notation

7.3.1 Roles Description Diagram

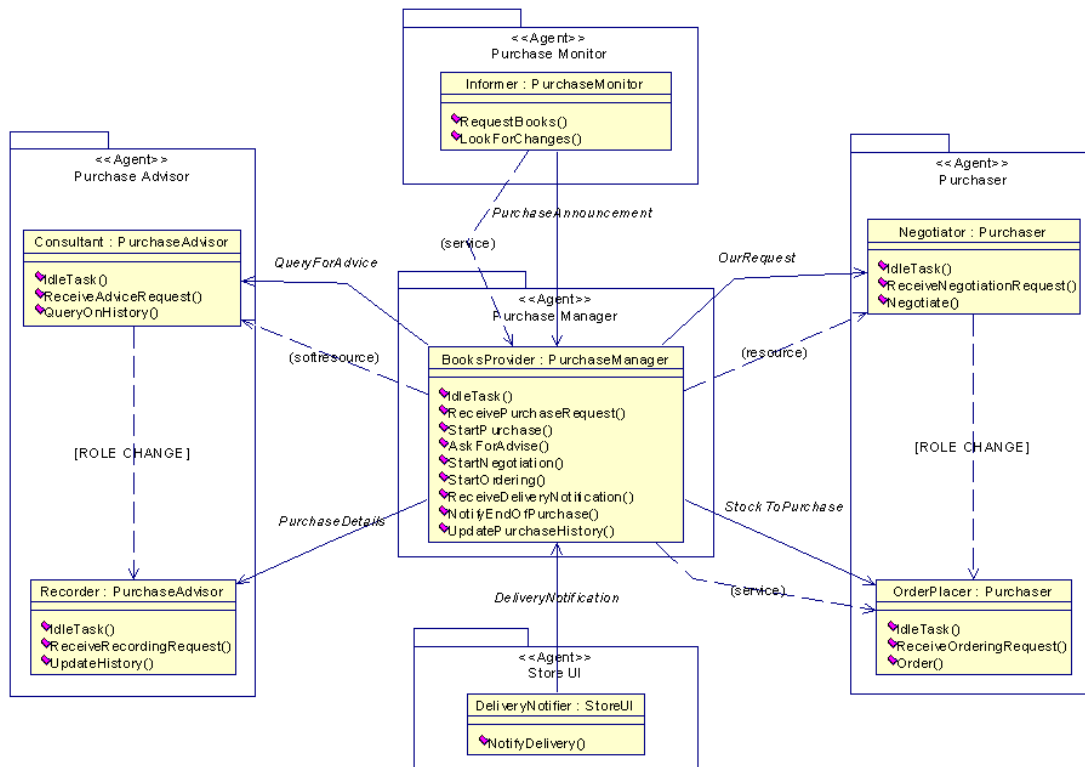


Fig.3 Roles Description diagram

We represent the Role Description diagram as a class diagram where roles are classes grouped in packages representing the agents.

Roles can be connected by relationships representing changes of role, dependencies for a service or the availability of a resource and communications. Each role is obtained composing several tasks for this reason we specify the tasks involved in the role using the operation compartment of each class.

More in details:

- Classes represent roles of the agent. They are grouped in packages that stand for the agent.
- Relationships among roles can be of 3 different kinds:
 - Communications. Represented by a solid line directed from the initiator to the participant. Names of communications come from the Communication Ontology Description diagram.
 - Dependencies. Like in i*, we can have service or resource dependencies. A *service* dependency shows that a role depends on another to bring about a goal (indicated by a dashed line with the *service* stereotype). In the *resource* dependency, a role depends on another for the availability of an entity (indicated by a dashed line with the *resource* stereotype). We can also have *soft-service* and *soft-resource*

dependencies; in this case the requested service/resource is helpful or desirable, but not essential to achieve a role's goal.

- Role changes. This connection is depicted as a dependency relationship because we want to signify the dependency of the second role on the first. Sometimes the trigger condition is not explicitly generated by the first role but its precedent appearance in the scenario justifies the consideration that it is necessary to prepare the situation that allows the second role to start. We use OCL or semi-formal text to express the trigger condition.

7.4 Relation with MAS meta-model

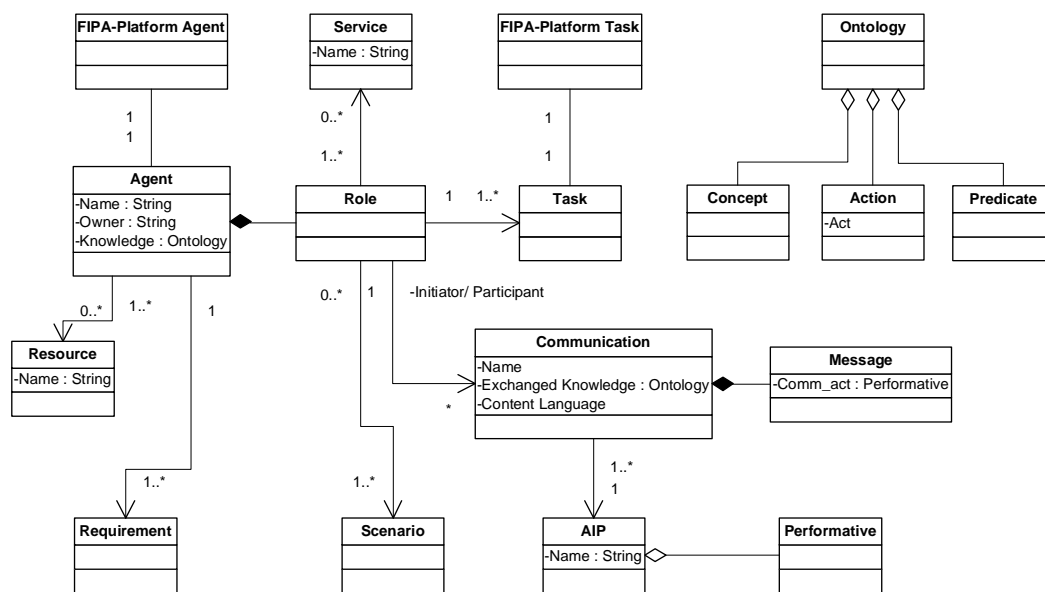


Fig.4. The MAS meta-model adopted in PASSI

This fragment refers to the MAS meta-model adopted in PASSI and contributes to define and describe the concept of roles.

The following figure describes the structure of the different work products, in the fragment, and their composition with respect to the MAS model.

Here the symbol:



represents an element of the MAS model .

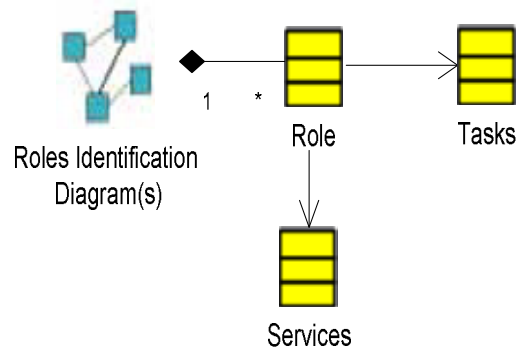


Fig.5. MAS Metamodel concepts

7.5 Input/Output

Input, output and element to be designed in the fragment are detailed in the following table

Input	To Be Designed	Output
R.Id. diagr, T.sp. diagr., C.O.D.	Roles	Role Description diagr. Services

7.6 Glossary

Agent Society Fragment uses this list of model element:

Role – an agent could play one or more roles in the system. Each role describes an aspect of agent life cycle and it is often related to a service offered by the agent to the society or to the achievement of one of its goals.

Task – An agent uses tasks to execute its plan(s). Each task is an entity that aims to reach a sub-goal (for example dealing with a communication or executing some transformations on a specific resource) .The term “task” can be used as synonymous of Behaviour but with the significance of atomic part of the overall agent behaviour.

8 Protocol Description

Version: January 11, 2004

8.1 Introduction

The PASSI process is composed of five different phases: System Requirements, Agent Society, Agent Implementation, Code and Deployment.

Each phase produces a document that is usually composed aggregating the UML models and work products of the work definitions that are inside each phase .

We will define a method fragment Protocol Description, extracted from PASSI methodology whose process is completely represented in the following figure

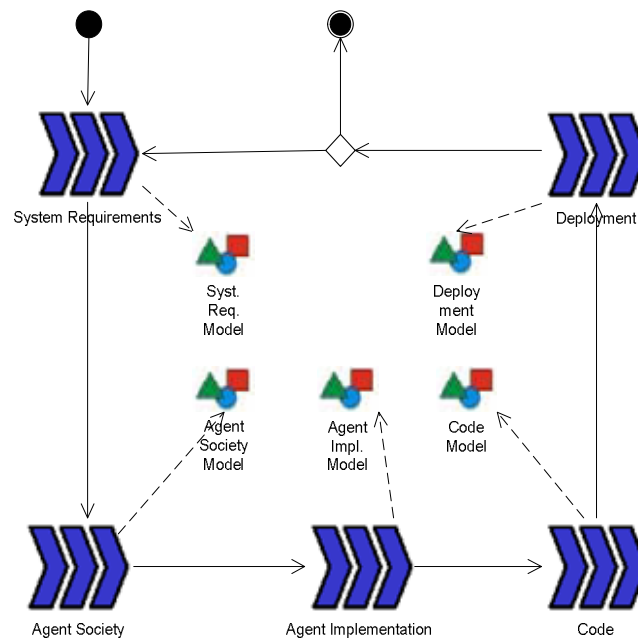


Fig. 1 The complete PASSI process

8.2 Fragment Definition

Consider the PASSI process (Fig. 1) and the phase “Agent Society” with its outcome “Agent Society Model”, the order of activities performed in this fragment is showed in the following SPEM diagram

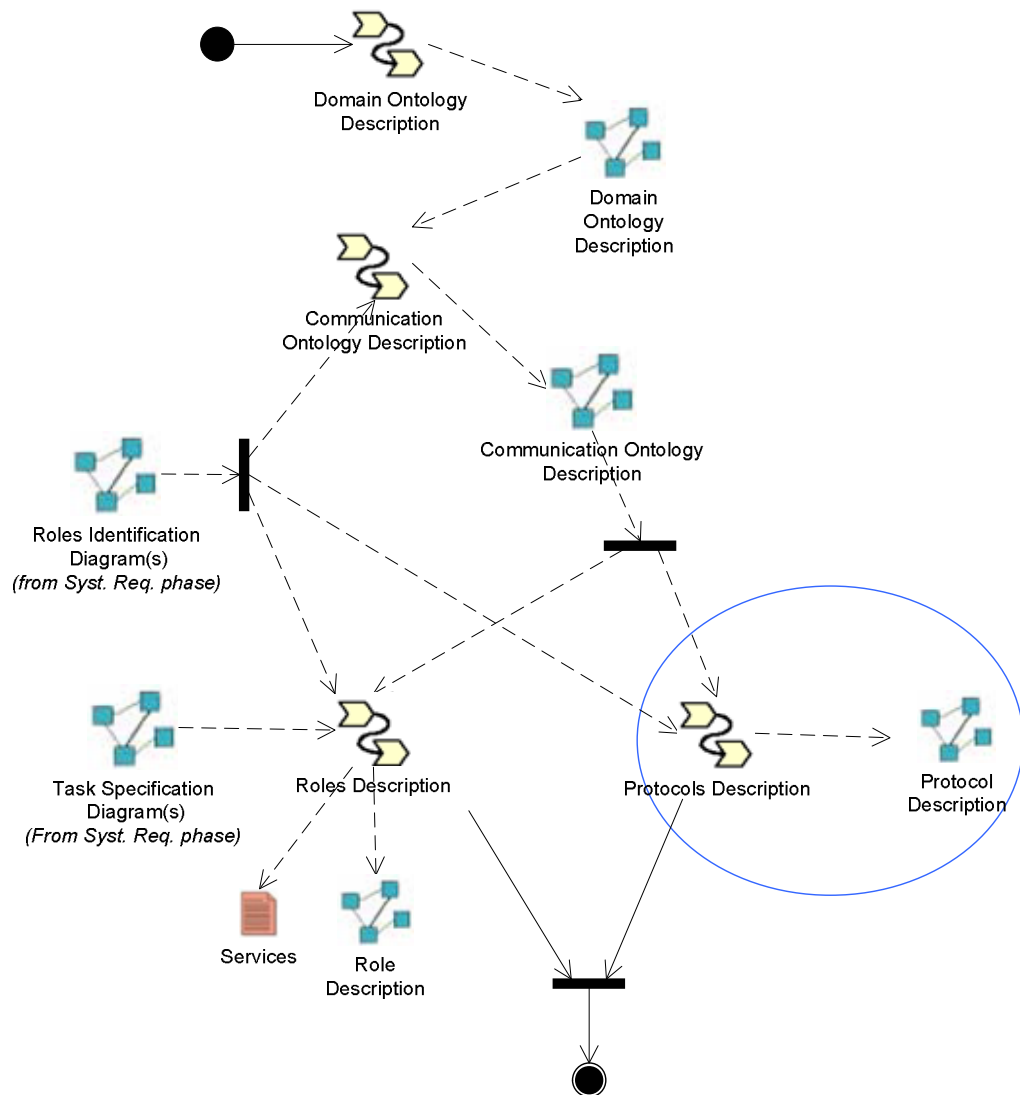


Fig.2 The Agent Society phase

This fragment aims to represent the protocol used for each communication as specified by FIPA architecture. The UML Model of this portion of process, Protocol Description, is designed following a AUML notation.

The process that is to be performed in order to obtain the result is represented in fig. 3 as a SPEM diagram

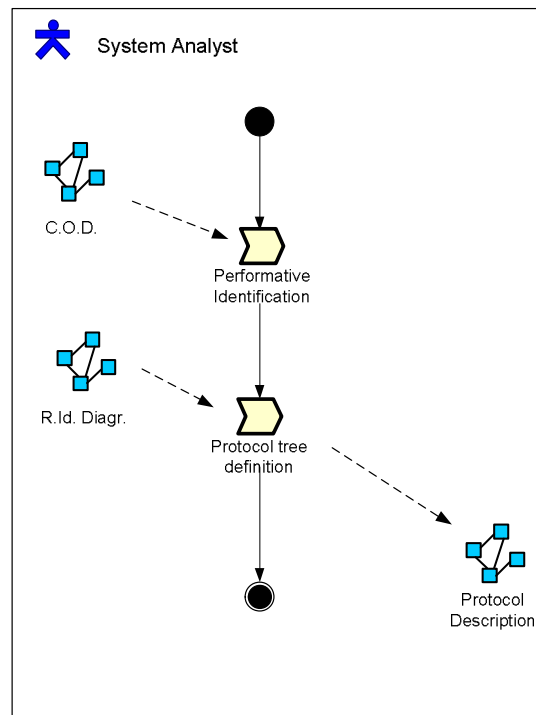
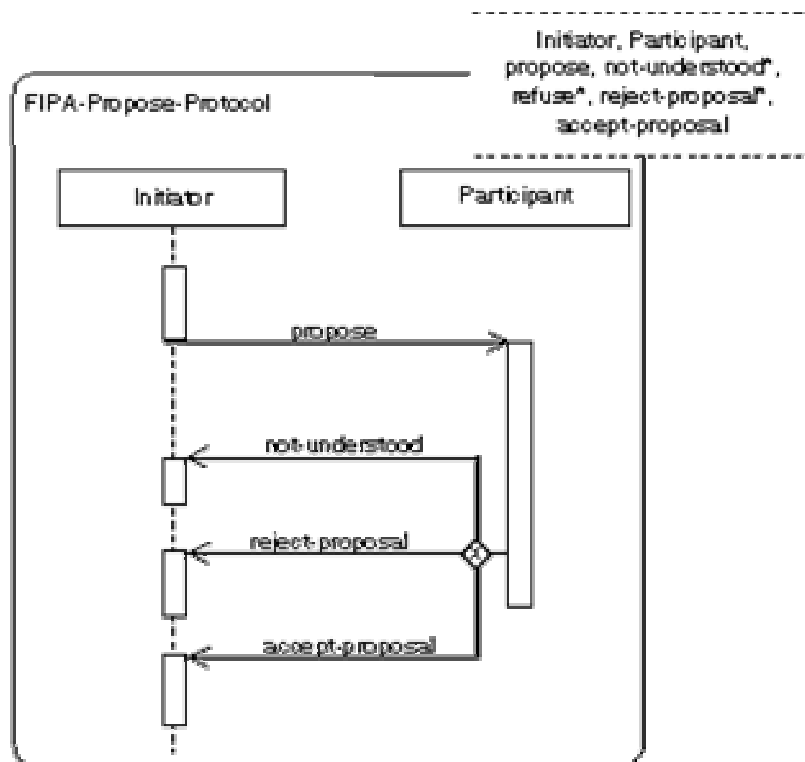


Fig.3 Protocols Description fragment-Procedural aspect

8.3 Notation

8.4 Protocol Description

An AUML sequence diagram for each (non standard) protocol



8.5 Relation with MAS meta-model

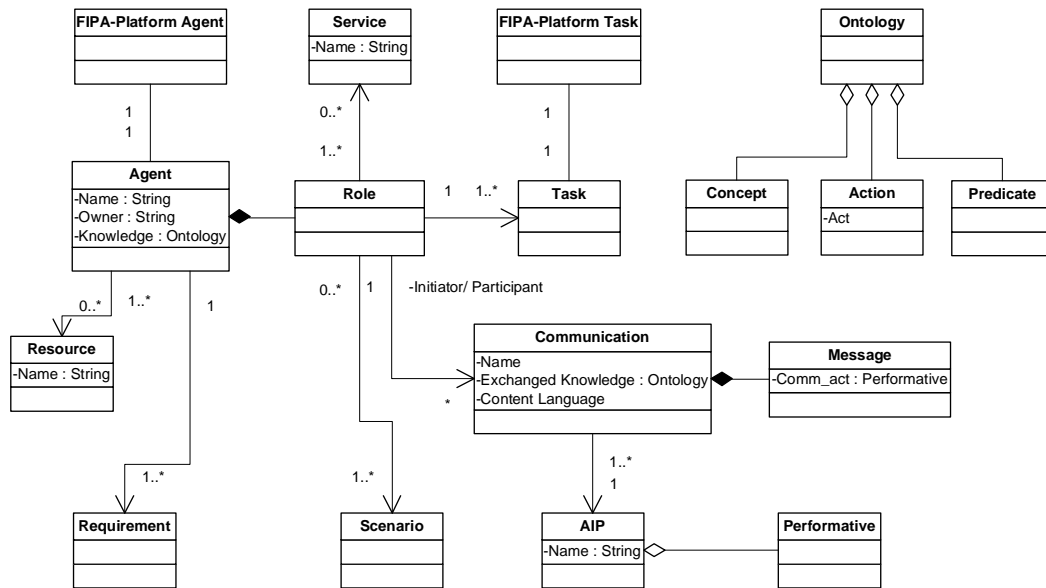


Fig.4. The MAS meta-model adopted in PASSI

This fragment refers to the MAS meta-model adopted in PASSI and contributes to define and describe a set of concepts in relation with it : AIP and performative .

The following figure describes the structure of the different work products, in the fragment, and their composition with respect to the MAS model.

Here the symbol:



represents an element of the MAS model .

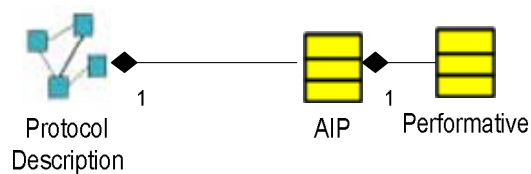


Fig. 5. MAS Metamodel concept

8.6 Input/Output

Input, output and element to be designed in the fragment are detailed in the following table

Input	To Be Designed	Output
R.Id. diagr., C.O.D. diagr.	Performative	Protocol description

8.7 Glossary

Protocol Description Fragment uses this list of model element:

Performtaive – message’s performative indicates the adopted FIPA Interaction Protocol.

9 Multi-Agent Structure Definition (MASD)

Version: January 11, 2004

9.1 Introduction

The PASSI process is composed of five different phases: System Requirements, Agent Society, Agent Implementation, Code and Deployment.

Each phase produces a document that is usually composed aggregating the UML models and work products of the work definitions that are inside each phase .

We will define a method fragment Multi-Agent Structure Definition, extracted from PASSI methodology whose process is completely represented in the following figure

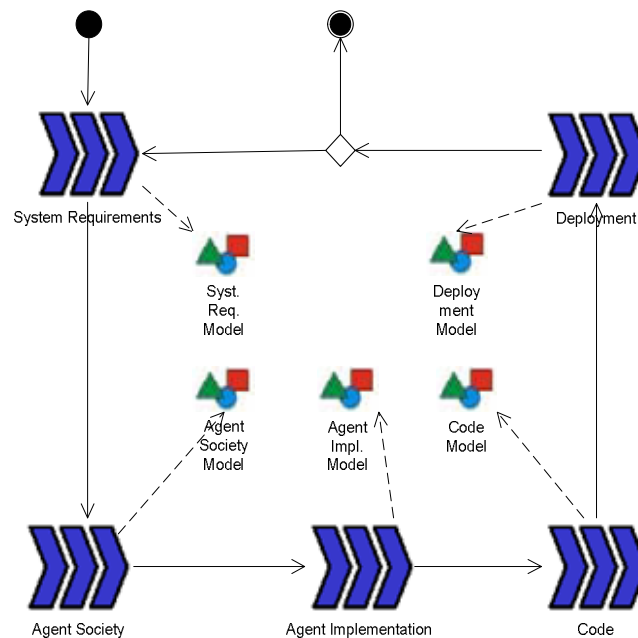


Fig. 1 The complete PASSI process

9.2 Fragment Definition

Consider the PASSI process (Fig. 1) and the phase “Agent Implementation” with its outcome “Agent Implementation Model”, the order of activities performed in this fragment is showed in the following SPEM diagram

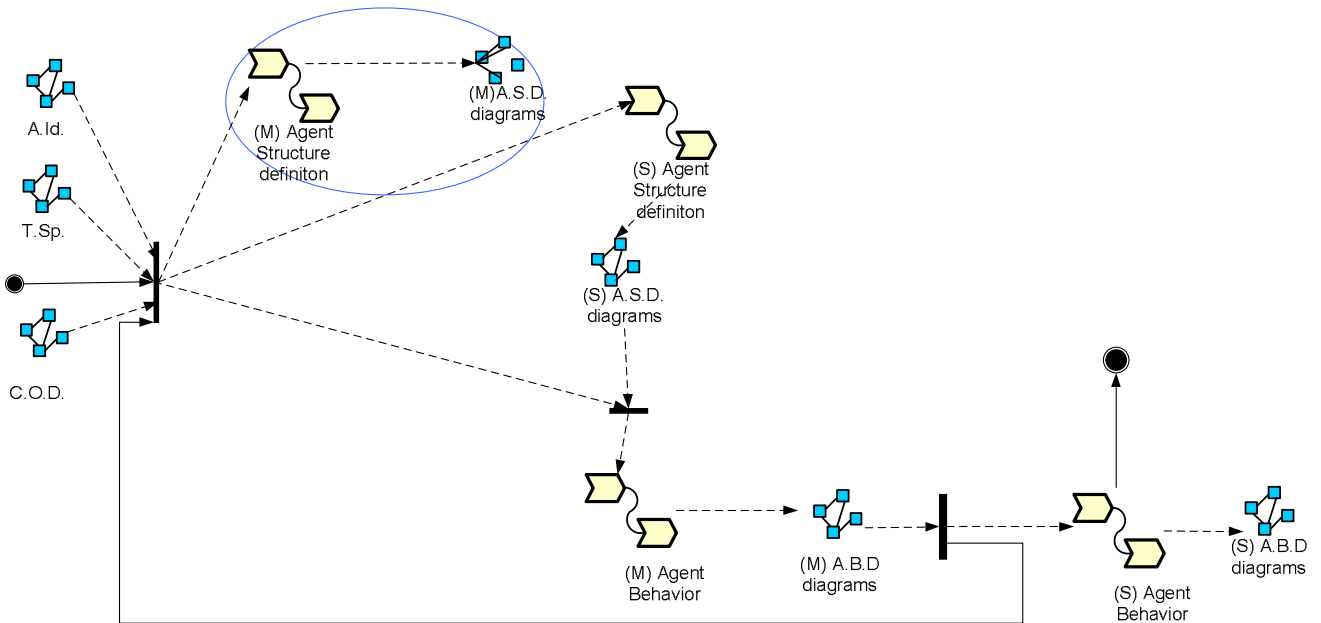


Fig.2 The Agent Implementation phase

This fragment aims to represent the general architecture of the system (agents their knowledge and their tasks). The UML Model of this portion of process, MASD diagram, is designed following a standard UML notation.

The process that is to be performed in order to obtain the result is represented in fig. 3 as a SPEM diagram

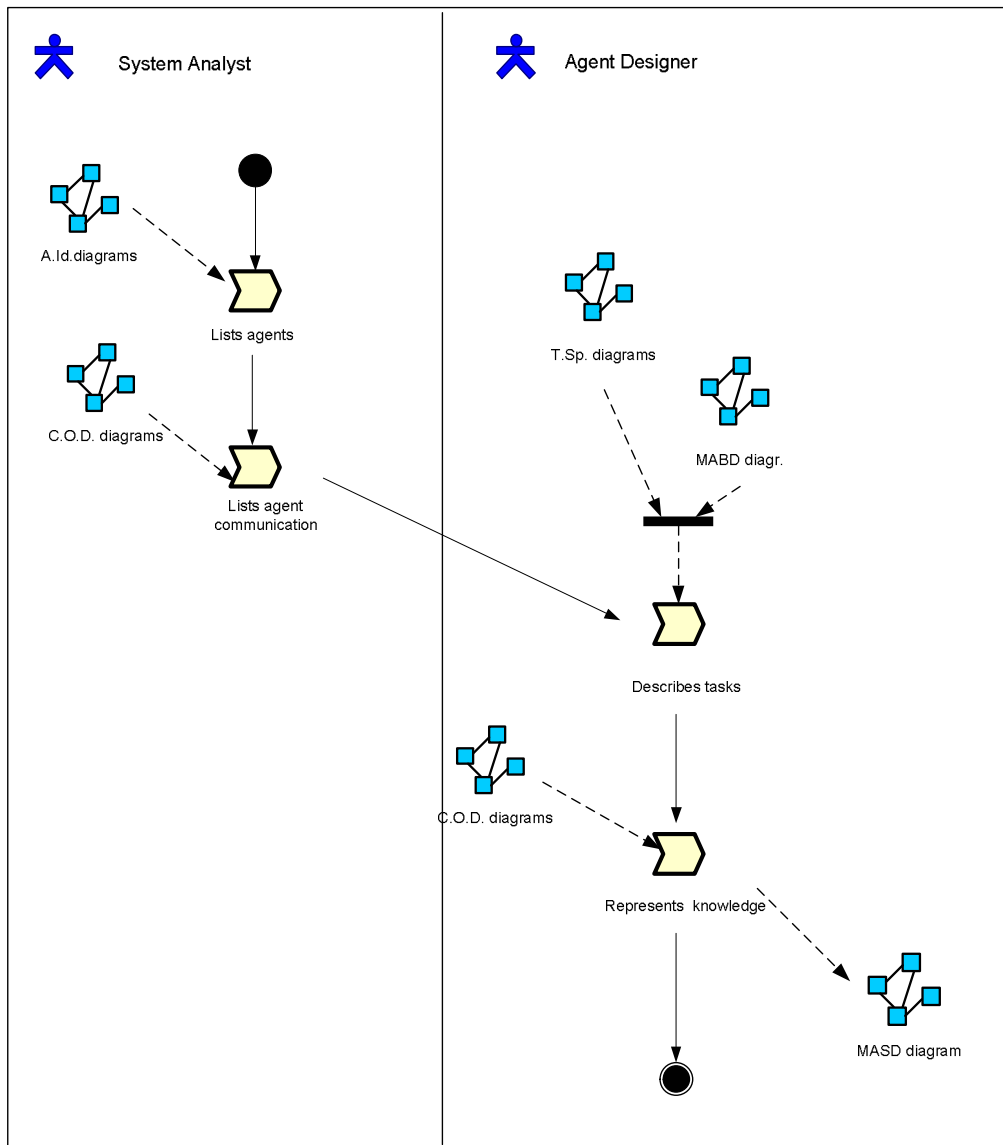


Fig.3. Multi-Agent Structure Definition fragment-Procedural aspect

9.3 Notation

9.4 Multi-Agent Structure Definition

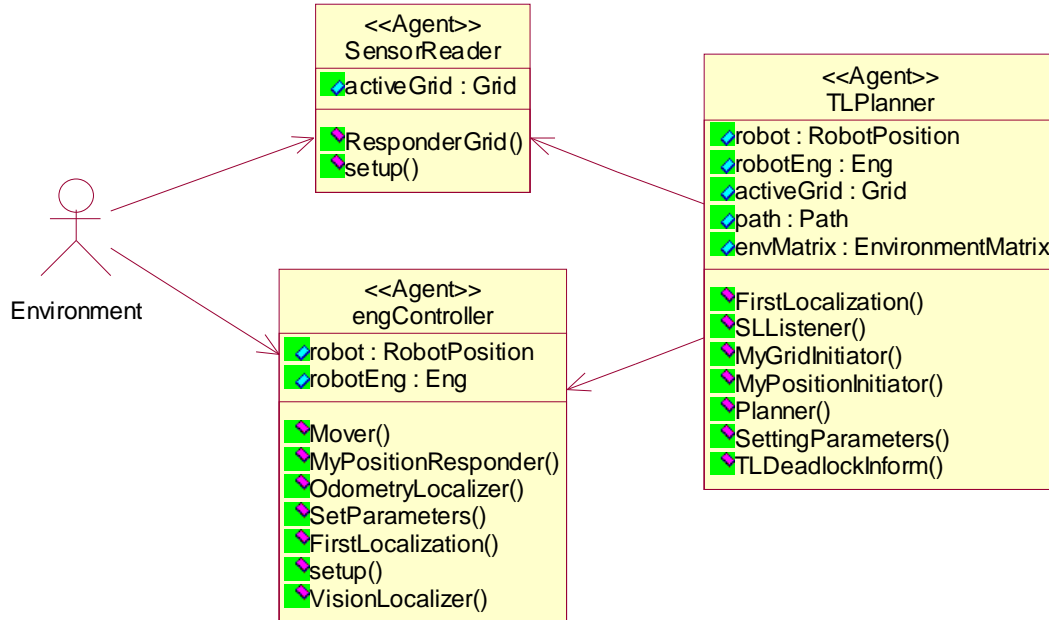


Fig.4. The MASD diagram

The class diagram contains classes and actors. Each class symbolizing one agent of the system. Attributes compartments can be used to represent the knowledge of the agent (referring to entities defined in the Domain Ontology Description), whereas operations compartments are used to signify the agent's tasks. The relations indicates the flow of exchanged information (communications)

9.5 Relation with MAS meta-model

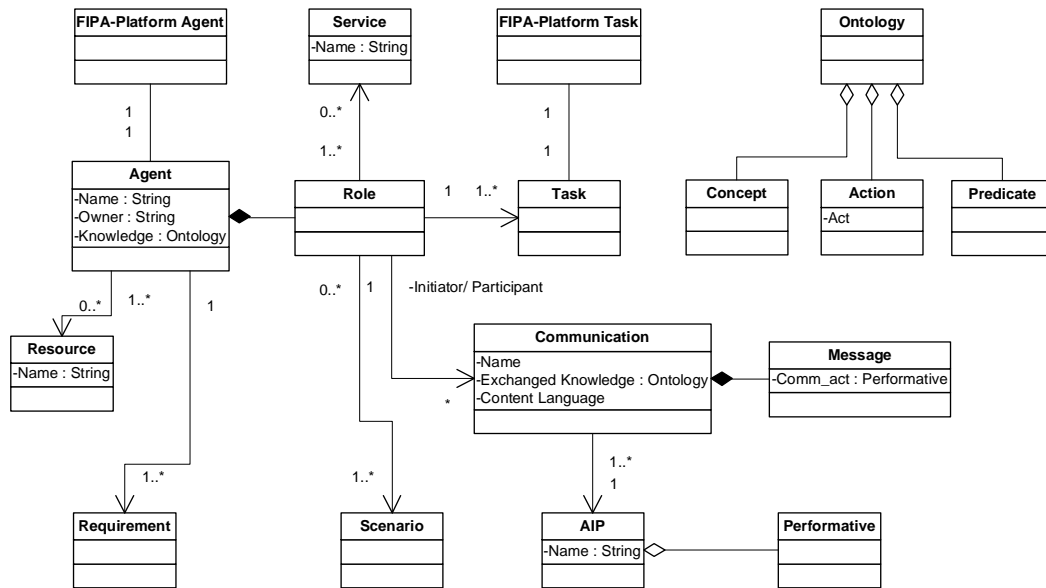


Fig.4. The MAS meta-model adopted in PASSI

This fragment refers to the MAS meta-model adopted in PASSI and contributes to define and describe a set of concepts in relation with it :agent, ontology, communication and tasks . The following figure describes the structure of the different work products, in the fragment, and their composition with respect to the MAS model. Here the symbol:



represents an element of the MAS model .

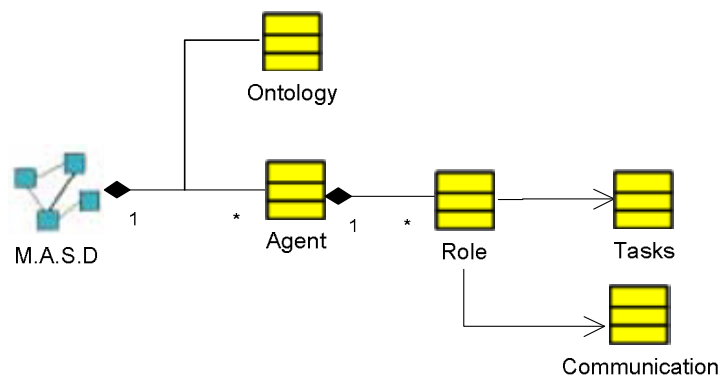


Fig. 5. MAS Metamodel concepts

9.6 Input/Output

Input, output and element to be designed in the fragment are detailed in the following table

Input	To Be Designed	Output
A.Id. diagram	Agent	MASD diagram
C.O.D. diagr.	Ontology	MASD diagram
C.O.D. diagr.	Communication	MASD diagram
MABD, Task Sp. diagram	Tasks	MASD diagram

9.7 Glossary

Multi-Agent Structure Definition 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.

Communication – a communication is an interaction between two agents. Each communication is described in terms of: ontology (related to the part of knowledge exchanged by the agents), content language and interaction protocol.

Task – It is a logical unit of individual or interactive behaviour. An agent uses tasks to execute its plan(s). Each task is an entity that aims to reach a sub-goal (for example dealing with a communication or executing some transformations on a specific resource).

10 Multi-Agent Behaviour Description (MABD)

Version: January 11, 2004

10.1 Introduction

The PASSI process is composed of five different phases: System Requirements, Agent Society, Agent Implementation, Code and Deployment.

Each phase produces a document that is usually composed aggregating the UML models and work products of the work definitions that are inside each phase .

We will define a method fragment Multi-Agent Structure Description, extracted from PASSI methodology whose process is completely represented in the following figure

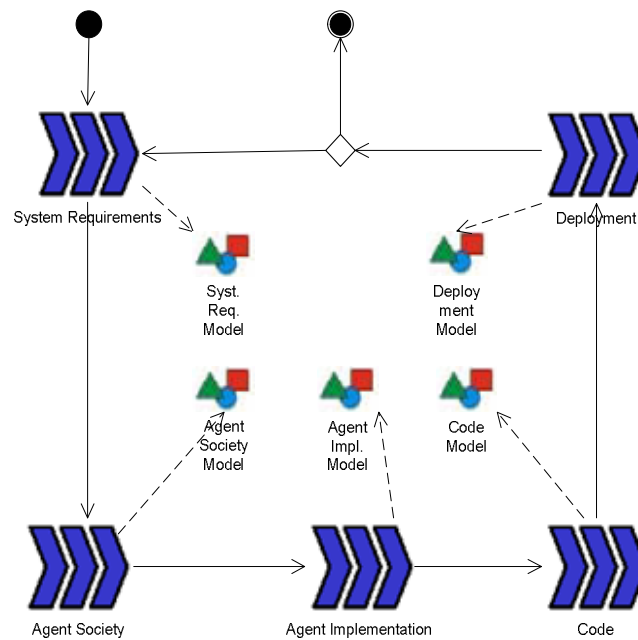


Fig. 1 The complete PASSI process

10.2 Fragment Definition

Consider the PASSI process (Fig. 1) and the phase “Agent Implementation” with its outcome “Agent Implementation Model”, the order of activities performed in this fragment is showed in the following SPEM diagram

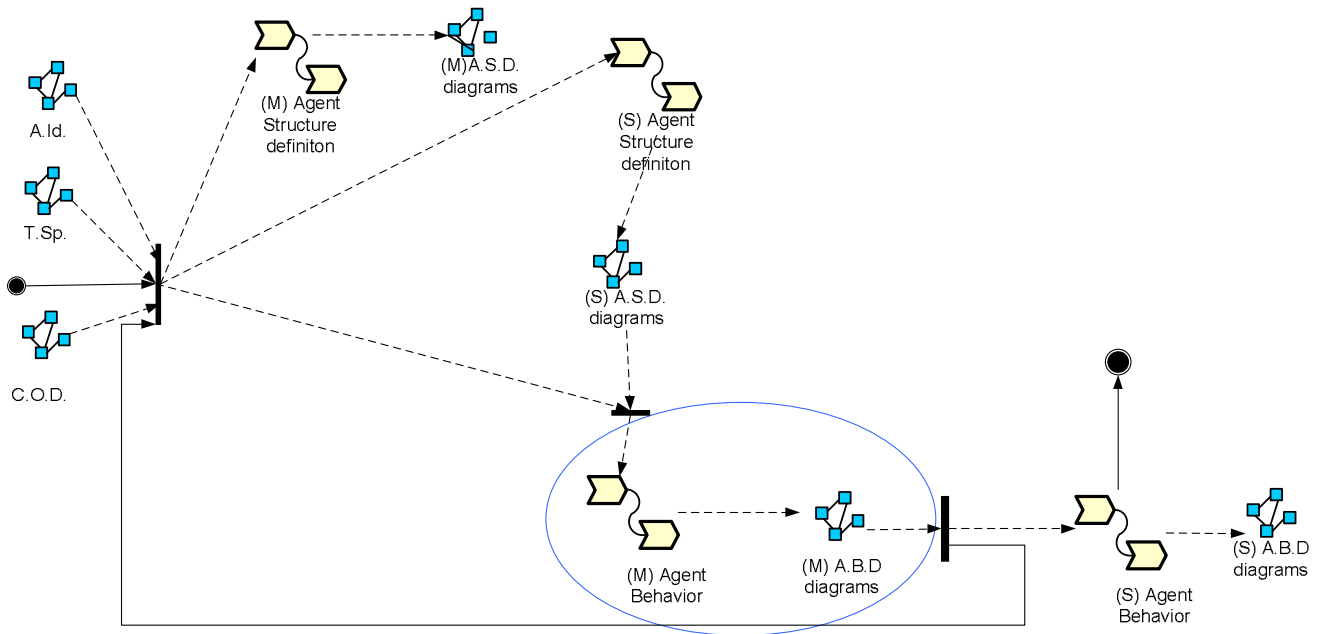


Fig.2 The Agent Implementation phase

The fragment, we are describing (blue oval), aims to show flow of events between and within both the main agents classes and their inner classes (representing their tasks). The UML Model of this portion of process, MABD diagram, is designed following a standard UML notation.

The process that is to be performed in order to obtain the result is represented in fig. 3 as a SPEM diagram

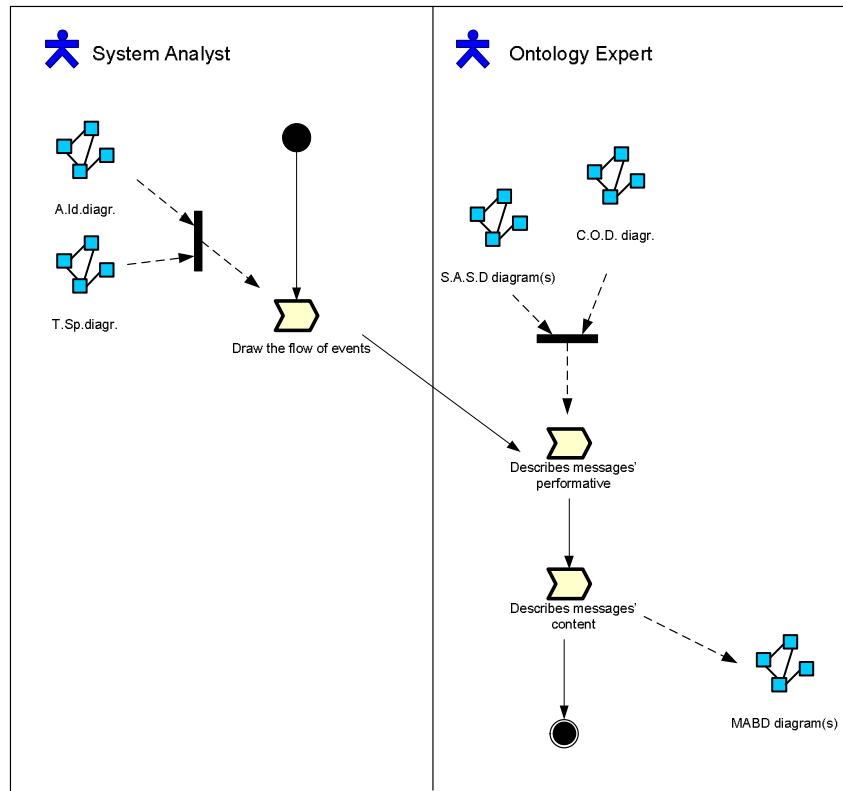


Fig.3. Multi-Agent Behaviour Description fragment-Procedural aspect

10.3 Notation

10.4 Multi-Agent Behaviour Description

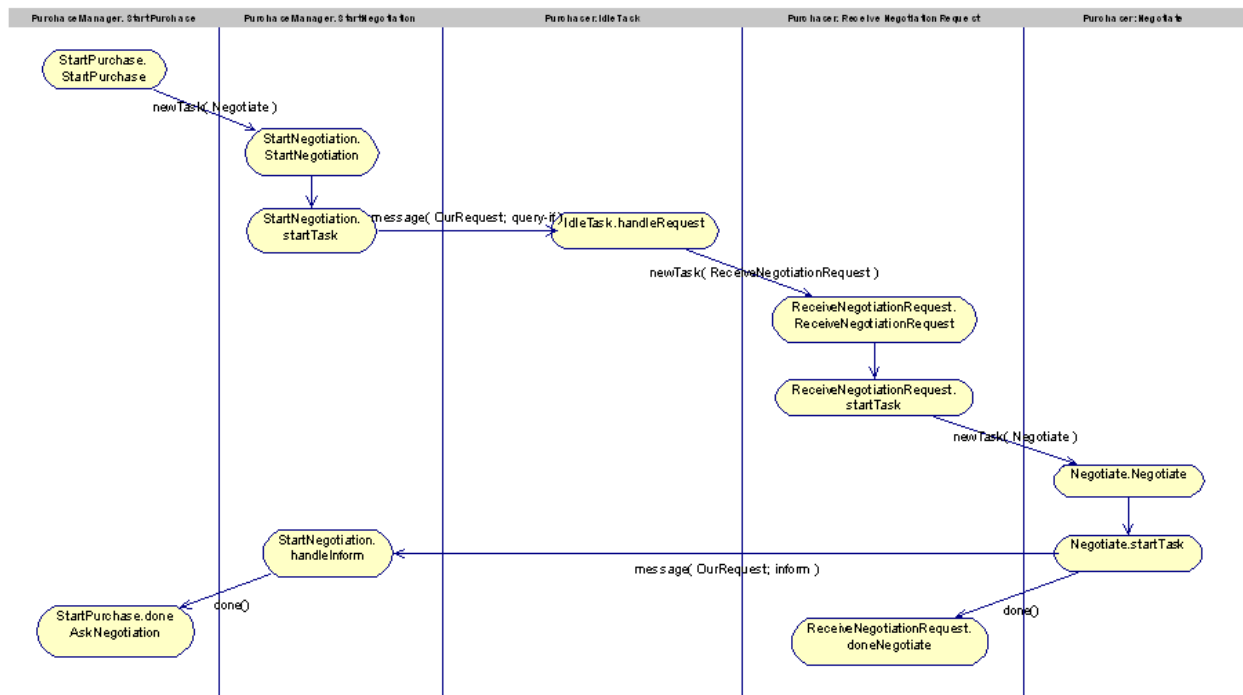


Fig.4. The MABD diagram

This activity diagram can be used to show the flow of events among and within both the main agents classes and their inner classes (representing their tasks).

We use one swimlane for each agent and for each of its tasks. The activities inside the swimlanes indicate the methods of the related class.

Usual transitions of the UML standard are here depicted to signify either events (e.g. an incoming message or a task conclusion) or invocation of methods.

If the transition is related to a conversation, the label reports the message's performative and content.

10.5 Relation with MAS meta-model

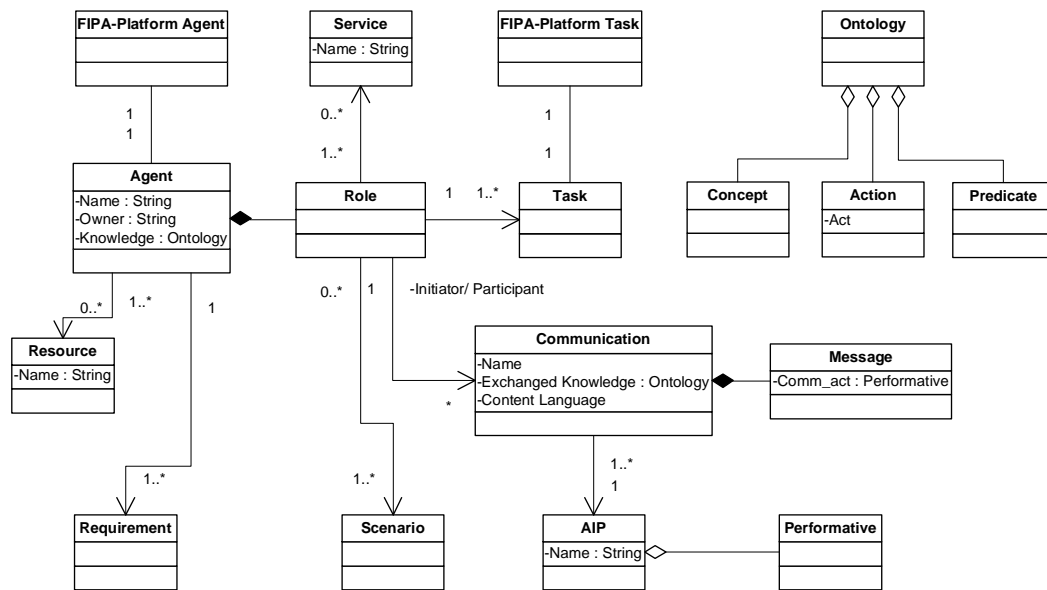


Fig.5. The MAS meta-model adopted in PASSI

This fragment refers to the MAS meta-model adopted in PASSI and contributes to define and describe a set of concepts in relation with it : tasks, communications, performative . The following figure describes the structure of the different work products, in the fragment, and their composition with respect to the MAS model. Here the symbol:



represents an element of the MAS model .

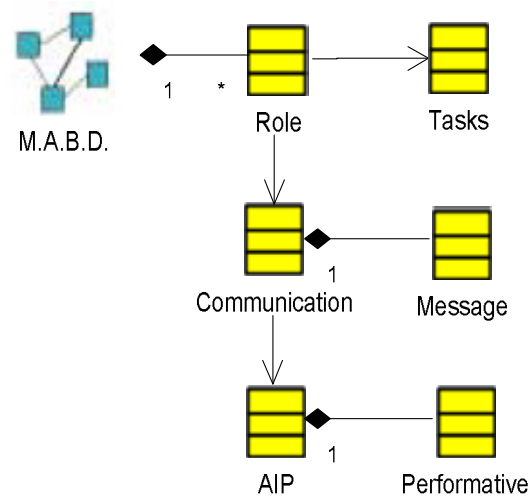


Fig. 7. MAS Metamodel concepts

10.6 Input/Output

Input, output and element to be designed in the fragment are detailed in the following table

Input	To Be Designed	Output
A.Id. diagram , Task Specification	Task	MABD diagram
COD	Communications	MABD diagram
COD, SASD	Performative	MABD diagram

10.7 Glossary

Multi-Agent Behaviour Description Fragment uses this list of model element:

Task – It is a logical unit of individual or interactive behaviour. An agent uses tasks to execute its plan(s). Each task is an entity that aims to reach a sub-goal (for example dealing with a communication or executing some transformations on a specific resource).

Communication – a communication is an interaction between two agents. Each communication is described in terms of: ontology (related to the part of knowledge exchanged by the agents), content language and interaction protocol.

Message - an individual unit of communication between two or more agents that point out the standard FIPA message format.

Performative – message's performative indicates the adopted FIPA Interaction Protocol.

11 Single-Agent Structure Definition (SASD)

Version: January 11, 2004

11.1 Introduction

The PASSI process is composed of five different phases: System Requirements, Agent Society, Agent Implementation, Code and Deployment.

Each phase produces a document that is usually composed aggregating the UML models and work products of the work definitions that are inside each phase .

We will define a method fragment Single-Agent Structure Definition, extracted from PASSI methodology whose process is completely represented in the following figure

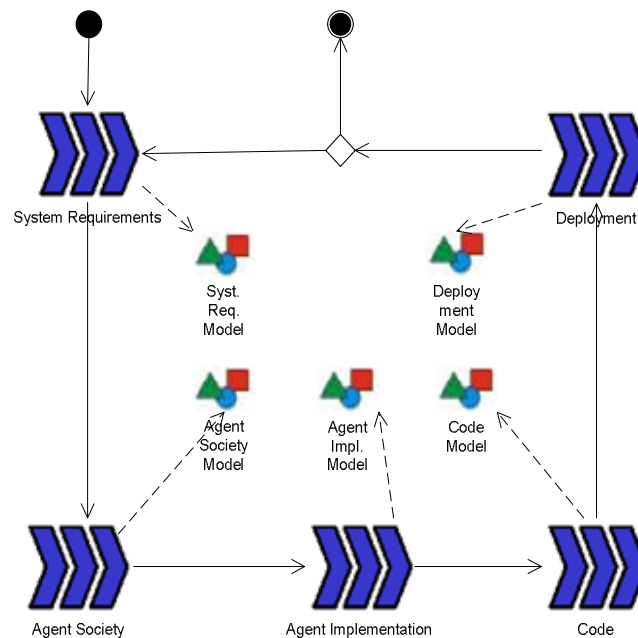


Fig. 1 The complete PASSI process

11.2 Fragment Definition

Consider the PASSI process (Fig. 1) and the phase “Agent Implementation” with its outcome “Agent Implementation Model”, the order of activities performed in this fragment is showed in the following SPEM diagram

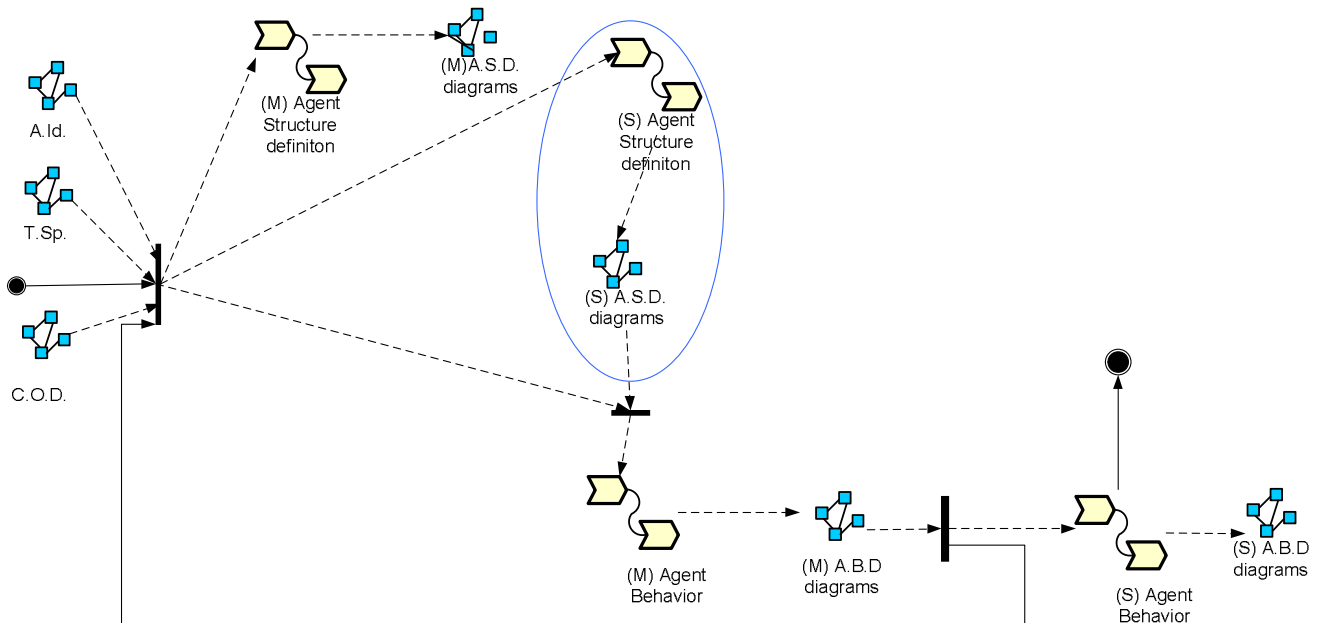


Fig.2 The Agent Implementation phase

The fragment, we are describing (blue oval), aims to represent each agent’s interior structure. The UML Model of this portion of process, SASD diagram, is designed following a standard UML notation.

The process that is to be performed in order to obtain the result is represented in fig. 3 as a SPEM diagram

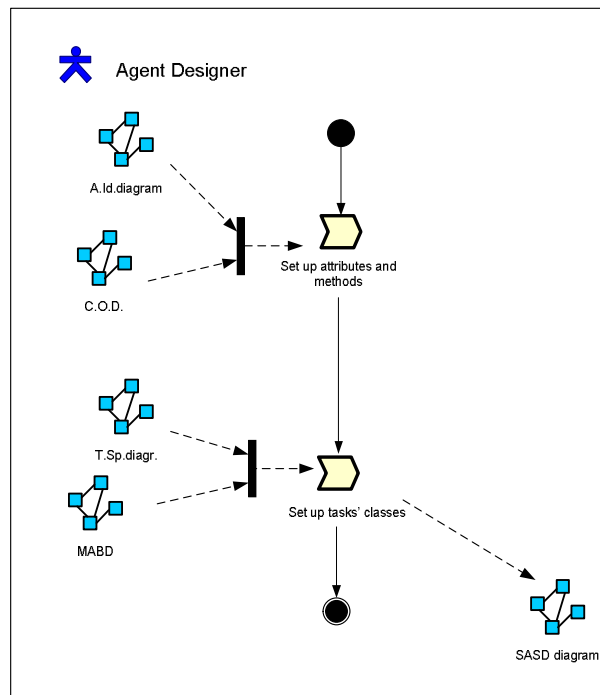


Fig.3. Single-Agent Structure Definition fragment-Procedural aspect

11.3 Notation

11.4 Single-Agent Structure Definition

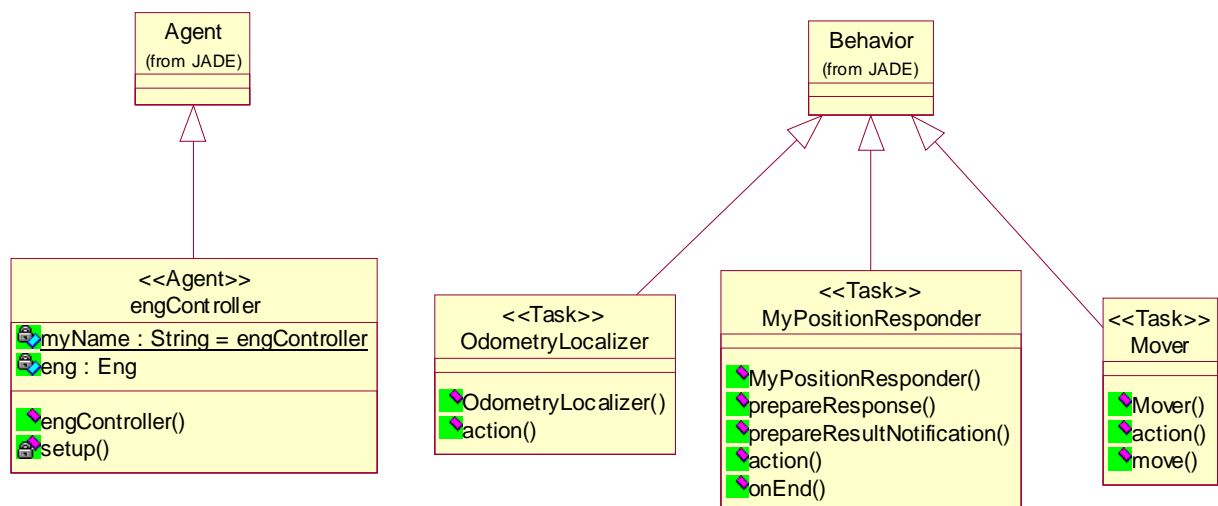


Fig.4. The SASD diagram

One different class diagram is drawn for each agent. This diagram describes the structure of the agent and all of its tasks.

Each class represents the agent or one of its task. The agent base class and the tasks are obtained specializing the base agent and task classes of the implementation platform (FIPA-OS in the figure above).

Attributes and methods are the elements that will constitute the real (code) implementation of the system. It is possible to automatically produce code from this diagram with many commercial tools.

11.5 Relation with MAS meta-model

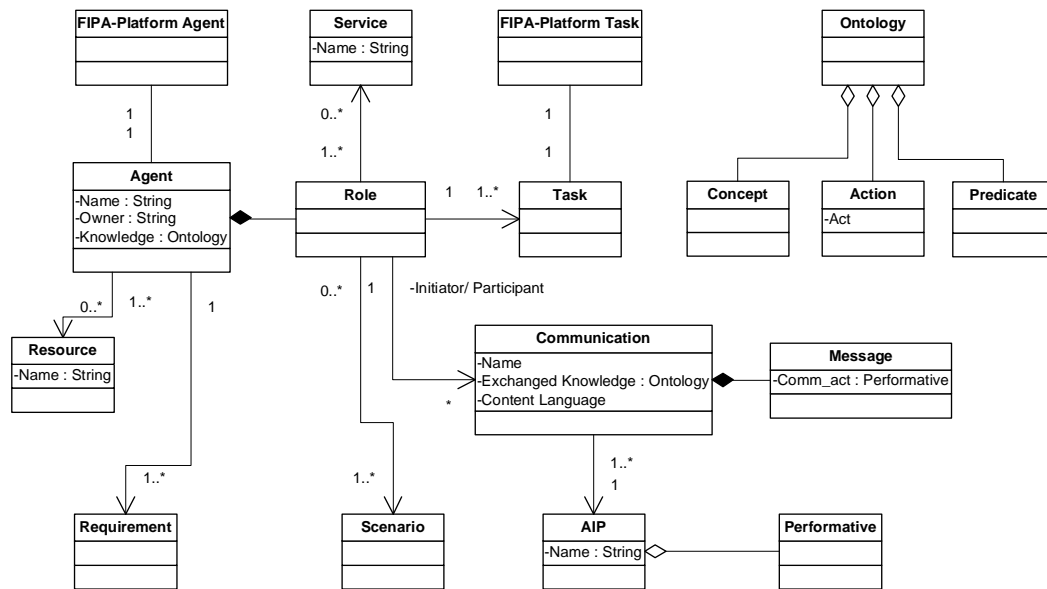


Fig.5. The MAS meta-model adopted in PASSI

This fragment refers to the MAS meta-model adopted in PASSI and contributes to define and describe a set of concepts in relation with it :agent and tasks .

The following figure describes the structure of the different work products, in the fragment, and their composition with respect to the MAS model.

Here the symbol:



represents an element of the MAS model .

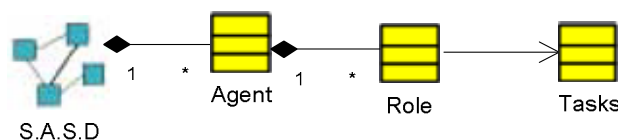


Fig. 7. MAS Metamodel concepts

11.6 Input/Output

Input, output and element to be designed in the fragment are detailed in the following table

Input	To Be Designed	Output
A.Id. diagram , COD diagram	Agent (attributes and methods)	SASD diagram
MABD, Task Sp. diagram	Tasks	SASD diagram

11.7 Glossary

Multi-Agent Structure Definition 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.

Task – It is a logical unit of individual or interactive behaviour. An agent uses tasks to execute its plan(s). Each task is an entity that aims to reach a sub-goal (for example dealing with a communication or executing some transformations on a specific resource).

12 Single-Agent Behaviour Description (MABD)

Version: January 11, 2004

12.1 Introduction

The PASSI process is composed of five different phases: System Requirements, Agent Society, Agent Implementation, Code and Deployment.

Each phase produces a document that is usually composed aggregating the UML models and work products of the work definitions that are inside each phase .

We will define a method fragment Single-Agent Structure Description, extracted from PASSI methodology whose process is completely represented in the following figure

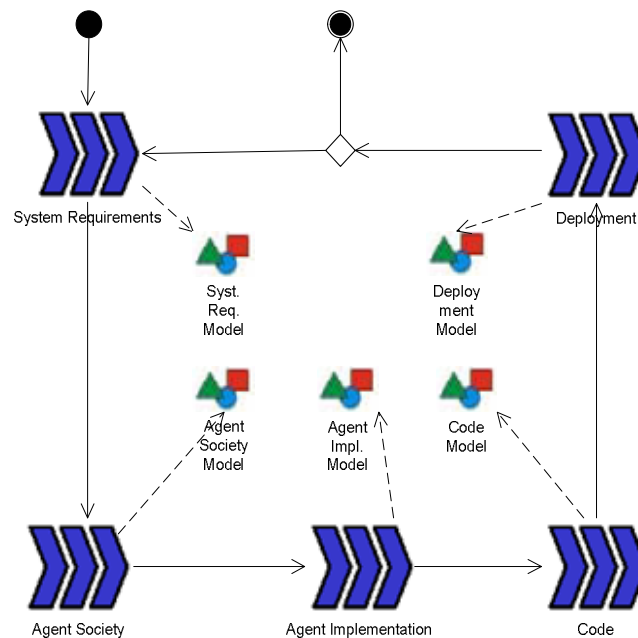


Fig. 1 The complete PASSI process

12.2 Fragment Definition

Consider the PASSI process (Fig. 1) and the phase “Agent Implementation” with its outcome “Agent Implementation Model”, the order of activities performed in this fragment is showed in the following SPEM diagram

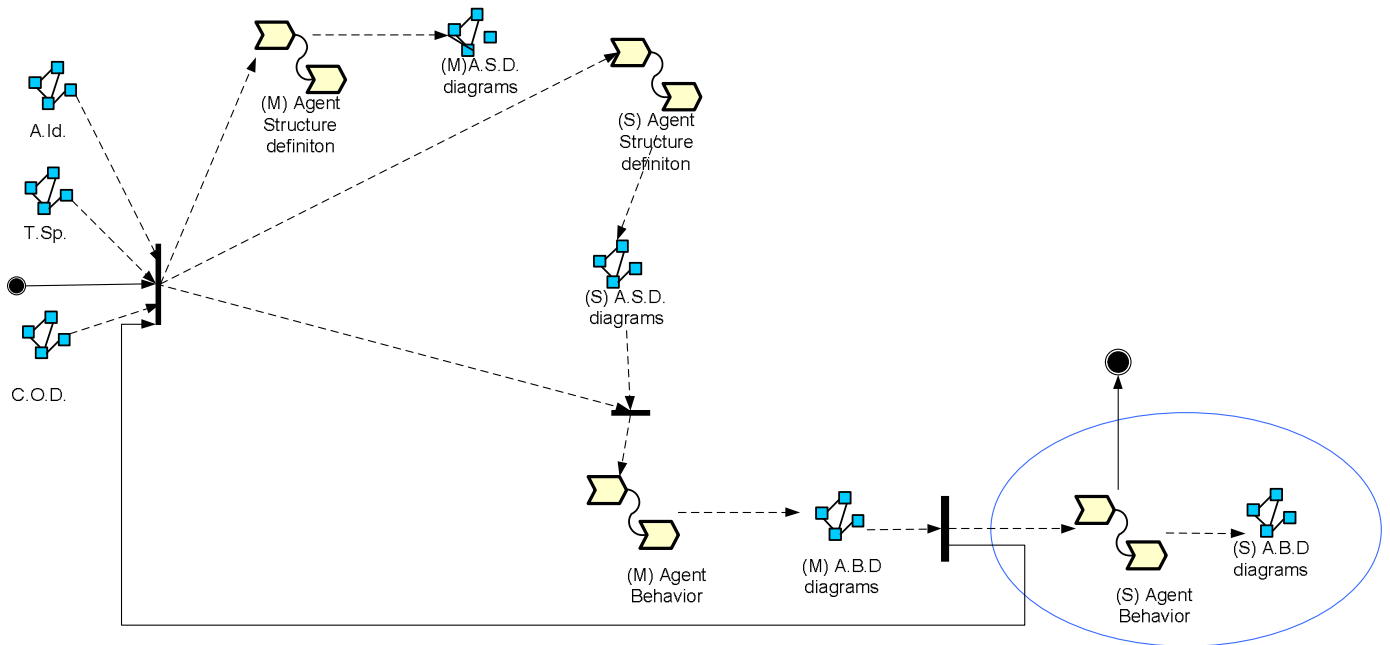


Fig.2 The Agent Implementation phase

The fragment, we are describing (blue oval), is quite a common one as it involves methods implementation, exactly the ones introduced in the [SASD diagrams](#). The UML Model of this portion of process, SABD diagram, is freely described in the most appropriate way (for example, using flow charts, state diagrams or semi-formal text descriptions).

The process that is to be performed in order to obtain the result is represented in fig. 3 as a SPEM diagram

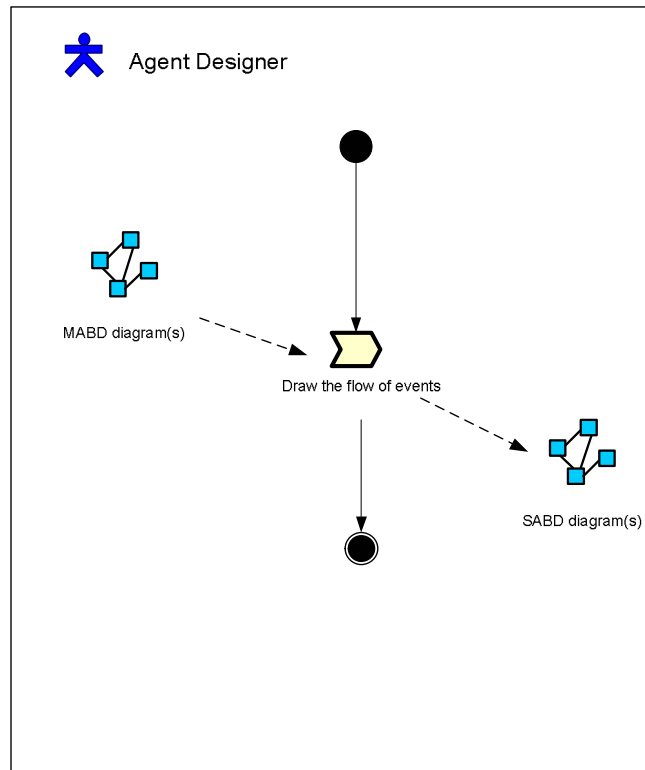


Fig.3. Single-Agent Behaviour Description fragment-Procedural aspect

12.3 Glossary

Multi-Agent Behaviour Description Fragment uses this list of model element:

Task – It is a logical unit of individual or interactive behaviour. An agent uses tasks to execute its plan(s). Each task is an entity that aims to reach a sub-goal (for example dealing with a communication or executing some transformations on a specific resource).

Communication – a communication is an interaction between two agents. Each communication is described in terms of: ontology (related to the part of knowledge exchanged by the agents), content language and interaction protocol.

Message - an individual unit of communication between two or more agents that point out the standard FIPA message format.

Performtaive – message's performative indicates the adopted FIPA Interaction Protocol.

13 Code Reuse Library

Version: December 9, 2003

13.1 Introduction

The PASSI process is composed of five different phases: System Requirements, Agent Society, Agent Implementation, Code and Deployment.

Each phase produces a document that is usually composed aggregating the UML models and work products of the work definitions that are inside each phase.

We will define a method fragment Code Reuse Library, extracted from PASSI methodology whose process is completely represented in the following figure

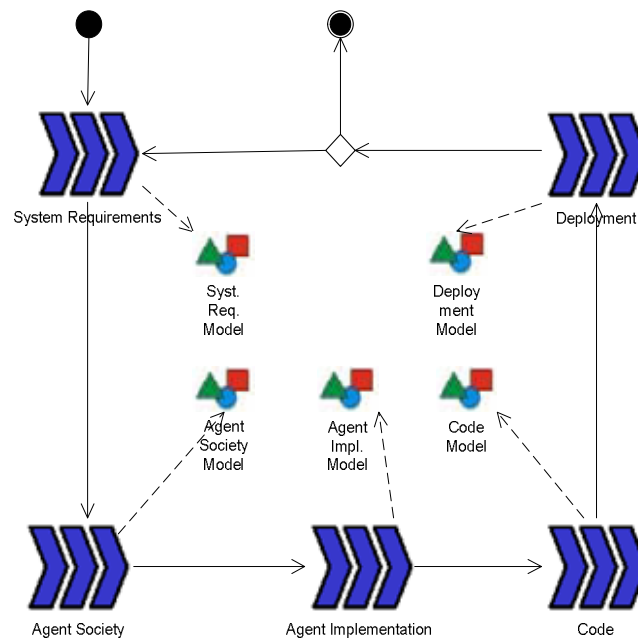


Fig. 1. The complete PASSI process

13.2 Fragment Definition

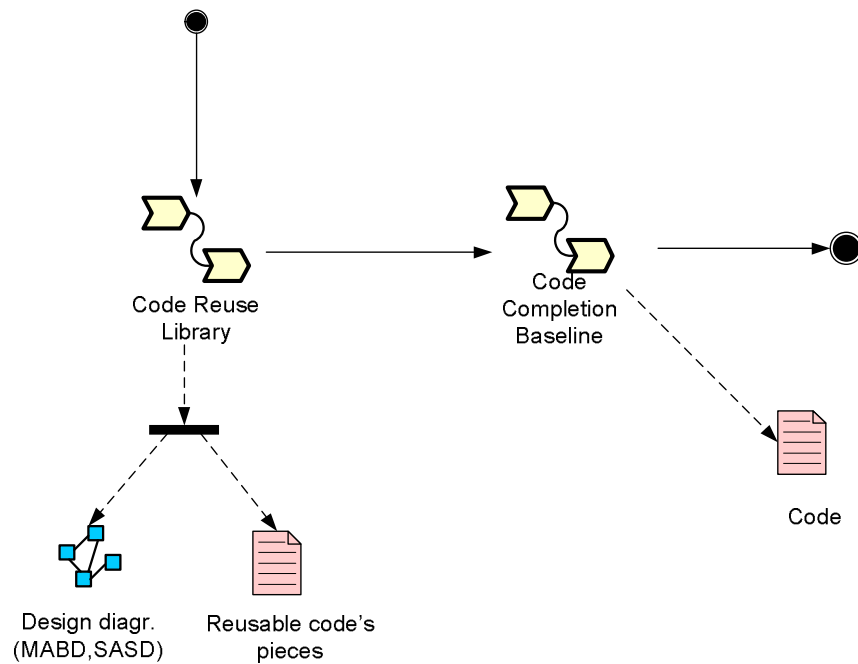


Fig.2. The Code phase

Let us consider the work definition “Code Reuse Library” whose aim is we try to reuse existing patterns of agents and tasks

The process that is to be performed in order to obtain the result is represented in fig. 3 as a SPEM diagram

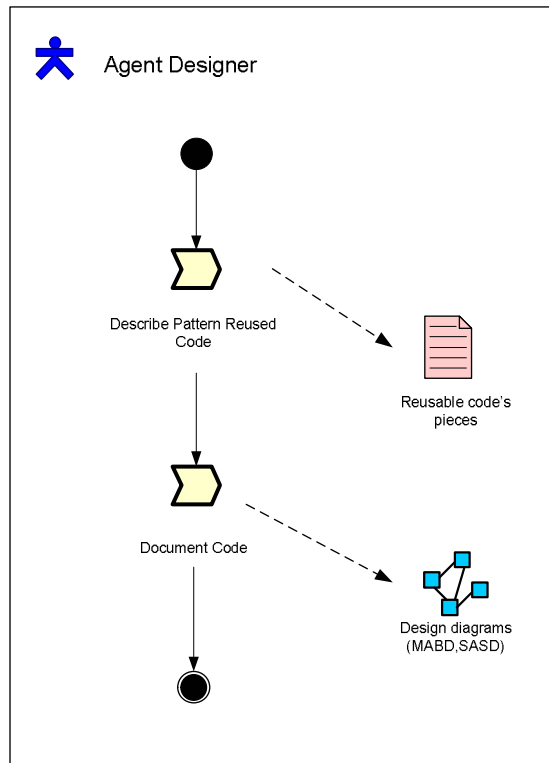


Fig.3. Code Reuse Library fragment-Procedural aspect

13.3 Notation

Code Reuse

The repository of patterns is described as reported below:

Name	The name of the pattern
Classification	<p>The classification of the pattern according to the following criteria and related categories:</p> <ul style="list-style-type: none"> • Application context: Action, Behavior, Component and Service pattern • Functionality: Access to local resource, Communication, Elaboration, Mobility
Intent	A description of what the pattern does and its rationale and intent
Motivation	A scenario that illustrates a design problem and how the agents and their tasks in the pattern solve the problem.
Pre-conditions	The initial situation in which the pattern can be applied.
Post-conditions	The consequences of the application of the pattern: what changes the pattern introduces into the system
Structure	A graphical representation of the structure of the agent and its tasks (usually done with a class diagram)
Participants	A description of the agents involved in the pattern and their roles
Collaborations	A (graphical) representation of the collaborations of the agents involved in the pattern (if any)
Implementation availability	Availability of the implementation code for the FIPA-OS/JADE platforms. Availability of the UML diagrams of the solution

	(XMI) for importing them in the existing system design
Implementation description	Comments on the most significant code fragments to illustrate the pattern implementation in the specific agent platforms
Implementation Code	FIPA-OS/JADE code of the solution
Related Patterns	Patterns that should be used in conjunction with this one

14 Code Completion Baseline

Version: December 9, 2003

14.1 Introduction

The PASSI process is composed of five different phases: System Requirements, Agent Society, Agent Implementation, Code and Deployment.

Each phase produces a document that is usually composed aggregating the UML models and work products of the work definitions that are inside each phase .

We will define a method fragment Code Completion Baseline, extracted from PASSI methodology whose process is completely represented in the following figure

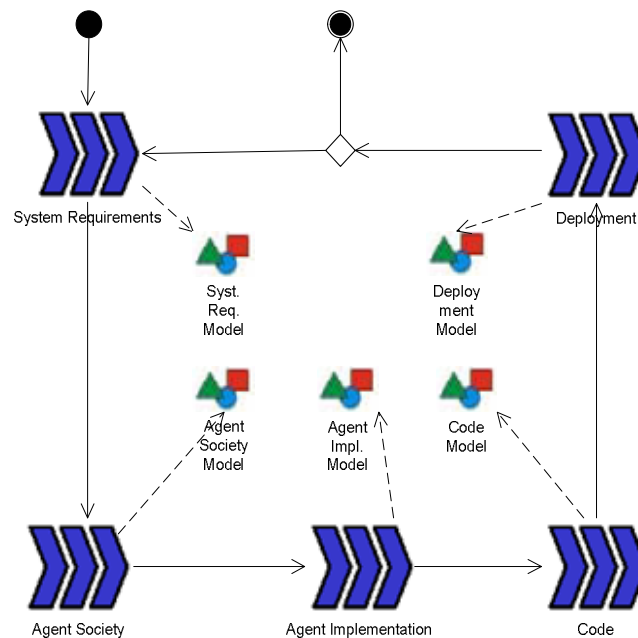


Fig. 1. The complete PASSI process

14.2 Fragment Definition

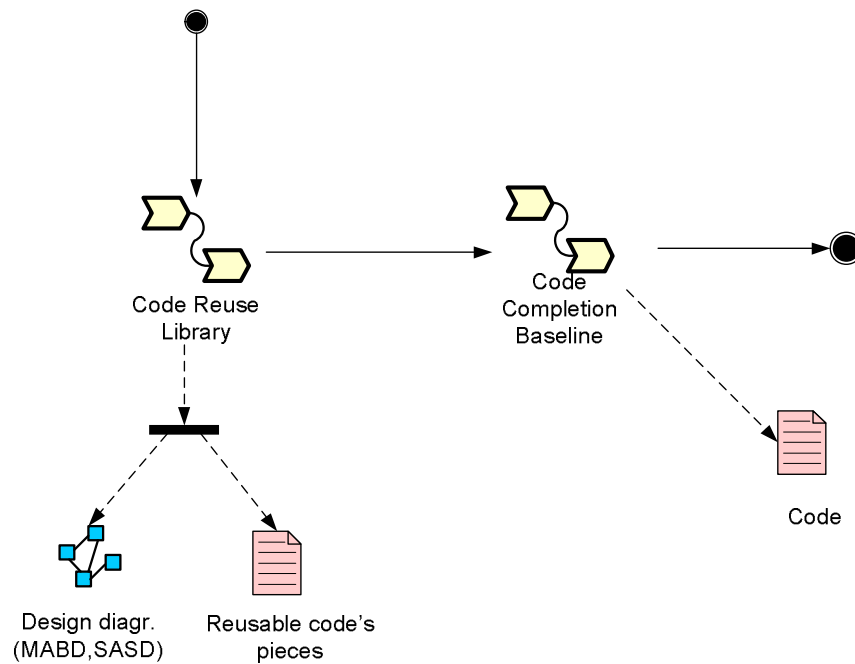


Fig.2. The Code phase

Let us consider the work definition “Code Reuse Library” whose aim is we try to reuse existing patterns of agents and tasks

The process that is to be performed in order to obtain the result is represented in fig. 3 as a SPEM diagram

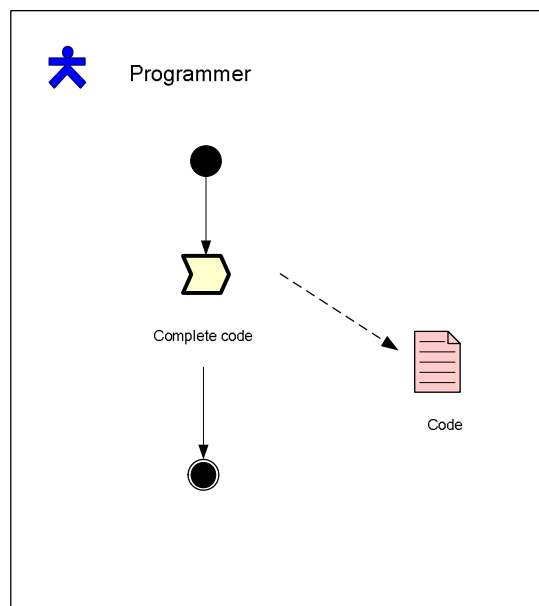


Fig.3. Code Completion Baseline fragment-Procedural aspect

This is rather a conventional phase. The programmer completes the code of the application starting from the design, the skeleton produced and the patterns reused.

15 Deployment Configuration

Version: January 11, 2004

15.1 Introduction

The PASSI process is composed of five different phases: System Requirements, Agent Society, Agent Implementation, Code and Deployment.

Each phase produces a document that is usually composed aggregating the UML models and work products of the work definitions that are inside each phase .

We will define a method fragment Deployment Configuration, extracted from PASSI methodology whose process is completely represented in the following figure

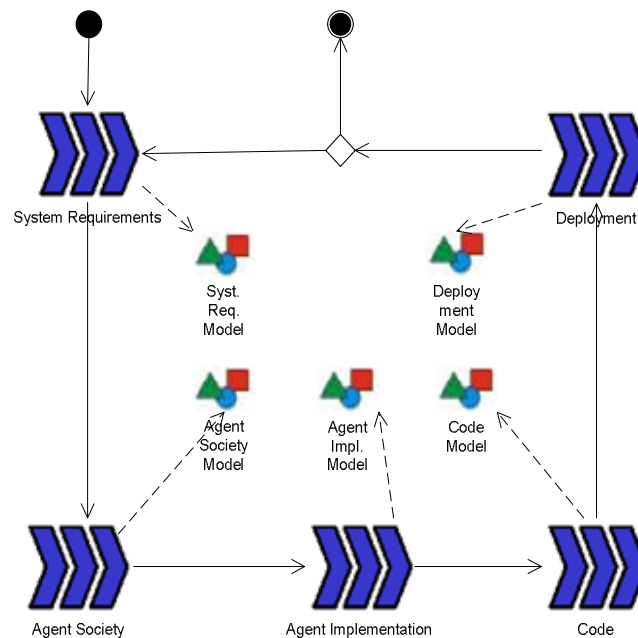


Fig. 1 The complete PASSI process

15.2 Fragment Definition

Consider the PASSI process (Fig. 1) and the phase “Agent Implementation” with its outcome “Agent Implementation Model”, the order of activities performed in this fragment is showed in the following SPEM diagram

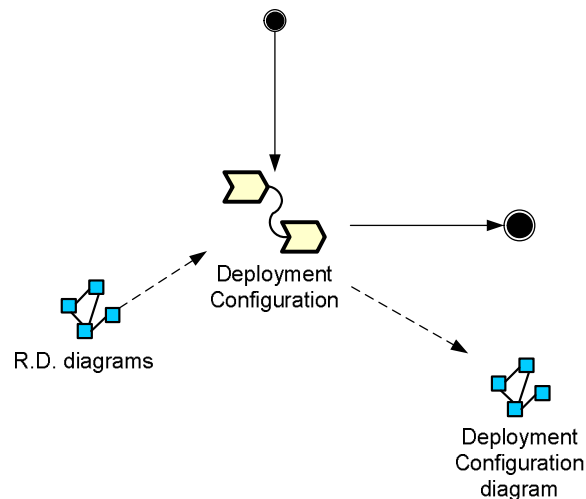


Fig.2 The Deployment Configuration phase

The fragment describes where the agents are located and which different elaborating units they need in order to communication with each other.

The process that is to be performed in order to obtain the result is represented in fig. 3 as a SPEM diagram

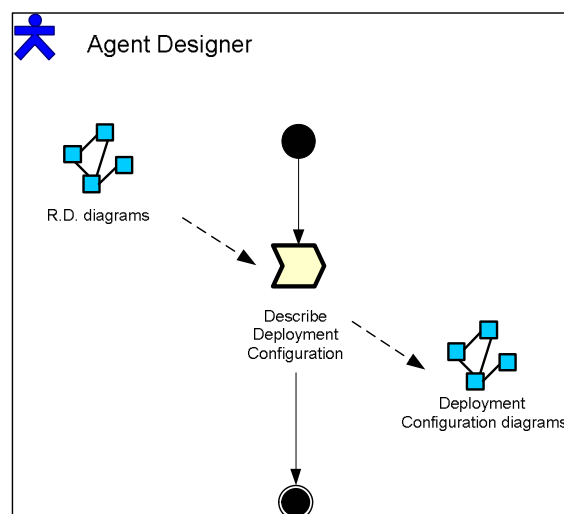


Fig.3. Deployment Configuration fragment-Procedural aspect

15.3 Notation

Deployment Configuration

This phase has been thought to comply with the requirements of detailing the agents' positions in distributed systems or more generally in mobile-agents' contexts.

The Deployment Configuration diagram illustrates the location of the agents (the processing units where they live), their movement and their communication support. The standard UML notation is useful for representing processing units (by boxes), agents (by components) and the like. What is not supported by UML is the representation of the agent's mobility, which we have done by means of a syntax extension consisting of a dashed line with a "move to" stereotype..

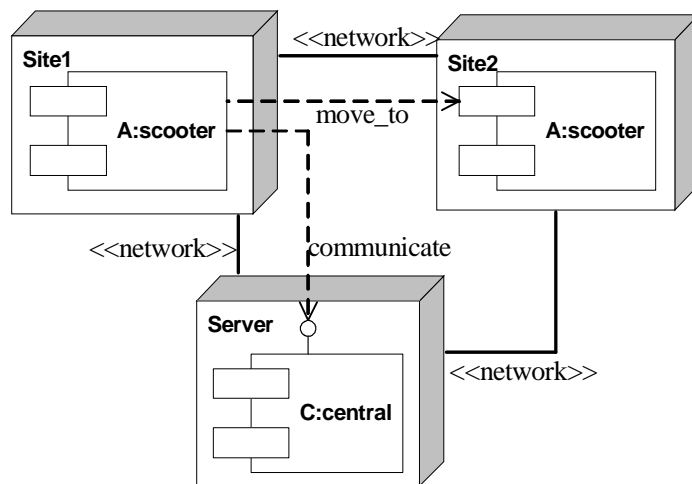


Fig.4. The Deployment Configuration diagram

In this diagram is also possible to specify the hardware devices used by the agents (sensors and effectors) and the modes of communication among agents in different elaborating units.