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

SPEM description of the PASSI process rel. 0.3.7

Massimo Cossentino, Luca Sabatucci, Valeria Seidita

RT-ICAR-20-03	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

SPEM description of the PASSI process rel. 0.3.7

Massimo Cossentino[1], Luca Sabatucci[1], Valeria Seidita[1]

Rapporto Tecnico N.: **RT-ICAR-20-03**	***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).

Index

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.

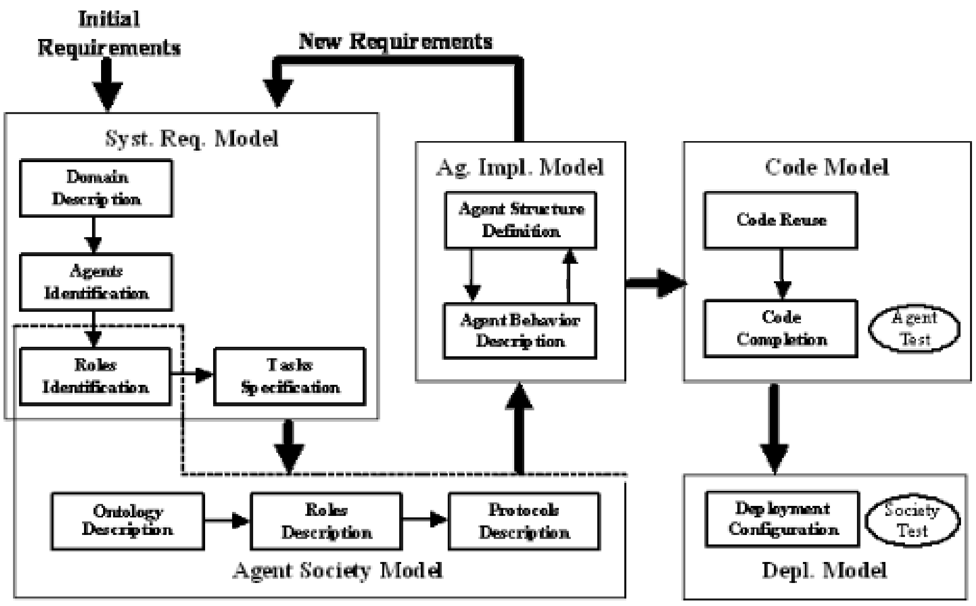


Fig.1: The PASSI methodology

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 first considering its whole process and disciplines and then detailing each of the different phases and sub-phases.

2 The PASSI Process

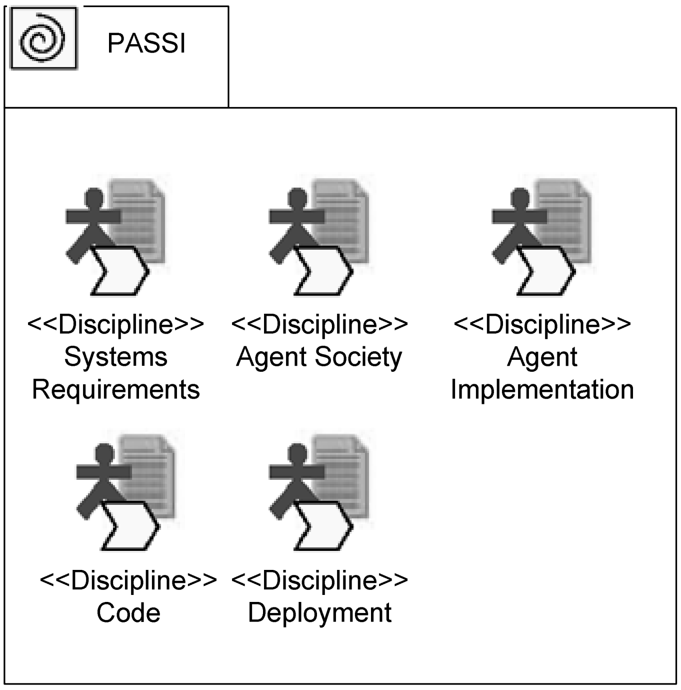
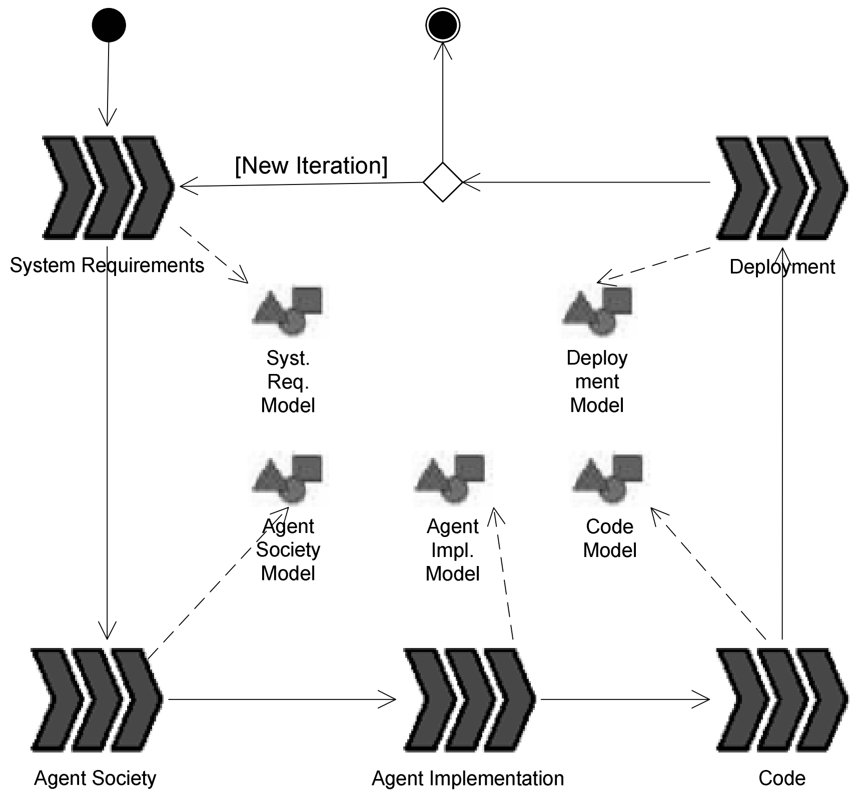

PASSI
<<Discipline>> Systems Requirements
<<Discipline>> Agent Society
<<Discipline>> Agent Implementation
<<Discipline>> Code
<<Discipline>> Deployment

Figure 1. The disciplines of the PASSI process

PASSI includes five disciplines:

- 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.
- Test: The verification of the single agent's behavior with regards to the original requirements of the system and the validation of efficiency of cooperation among agents at the society level.

We can draw a direct correspondence between the above cited disciplines and the five phases that characterize the procedural representation of PASSI.



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 disciplines).
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 session.

3 Phases of the PASSI Process

3.1 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)

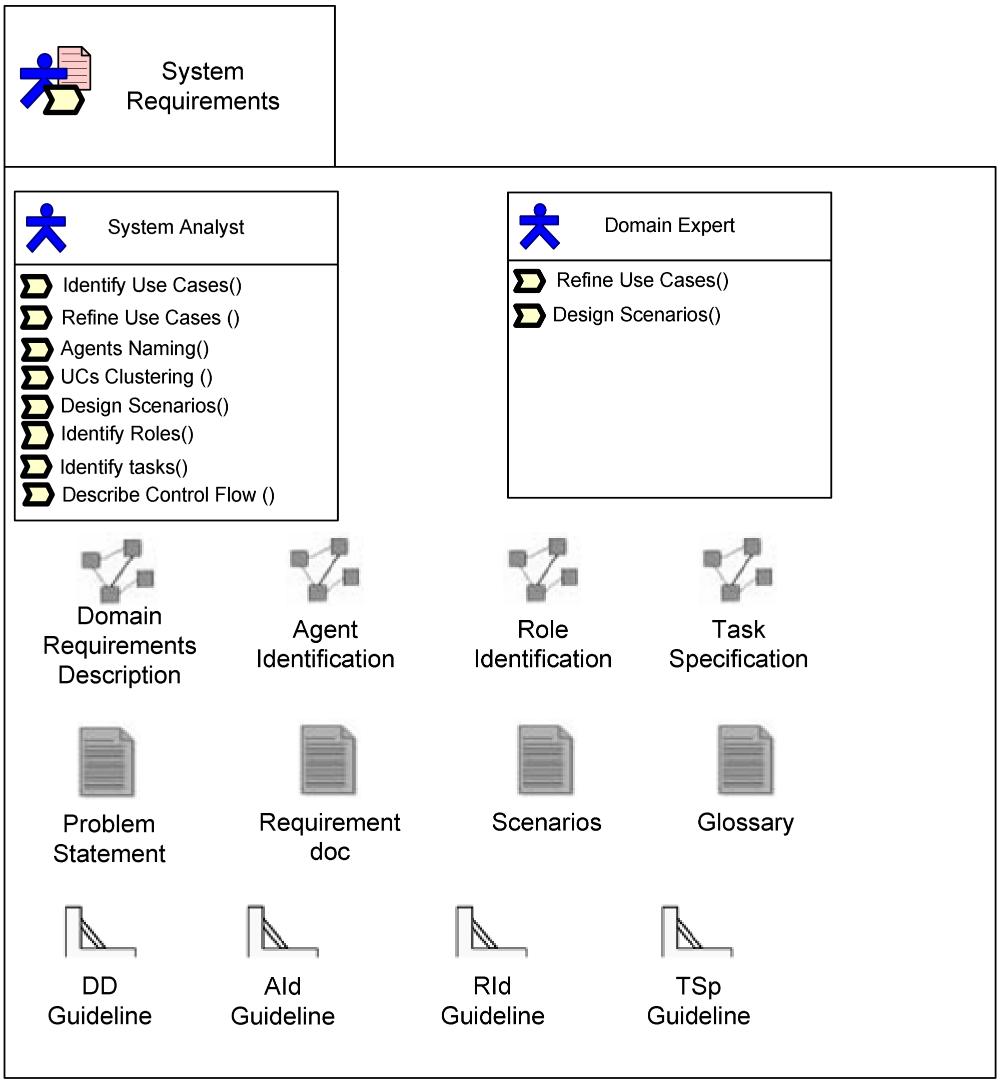


Figure 2. Exploitation of the System Requirements phase represented as a SPEM discipline

The process to be performed in this phase is described in the following Figure 3. It is composed of four sub-phase level work definitions (Domain Requirement Description, Agents Identification, Roles Identification and Task Specification) and several related work products (mainly UML models and text documents)

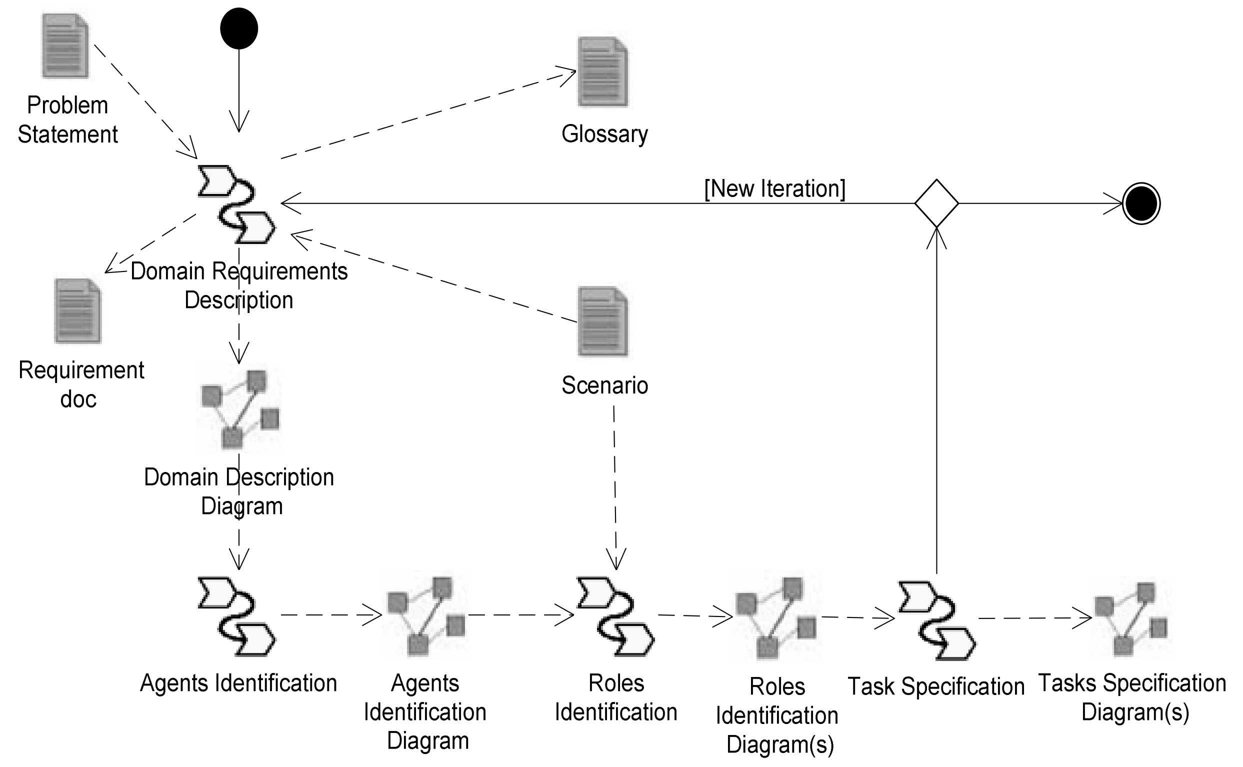

Problem Statement
Glossary
[New Iteration]
Domain Requirements Description
Requirement doc
Scenario
Domain Description Diagram
Agents Identification
Agents Identification Diagram
Roles Identification
Roles Identification Diagram(s)
Task Specification
Tasks Specification Diagram(s)

Figure 3. The System Requirement phase described in terms of work definitions (sub-phases) and work products

The four sub-phases are composed of several activities as described in Figure 4. This figure also describes the actors (process roles) involved in this portion of the process. The process flow will be described in following sub-sections.

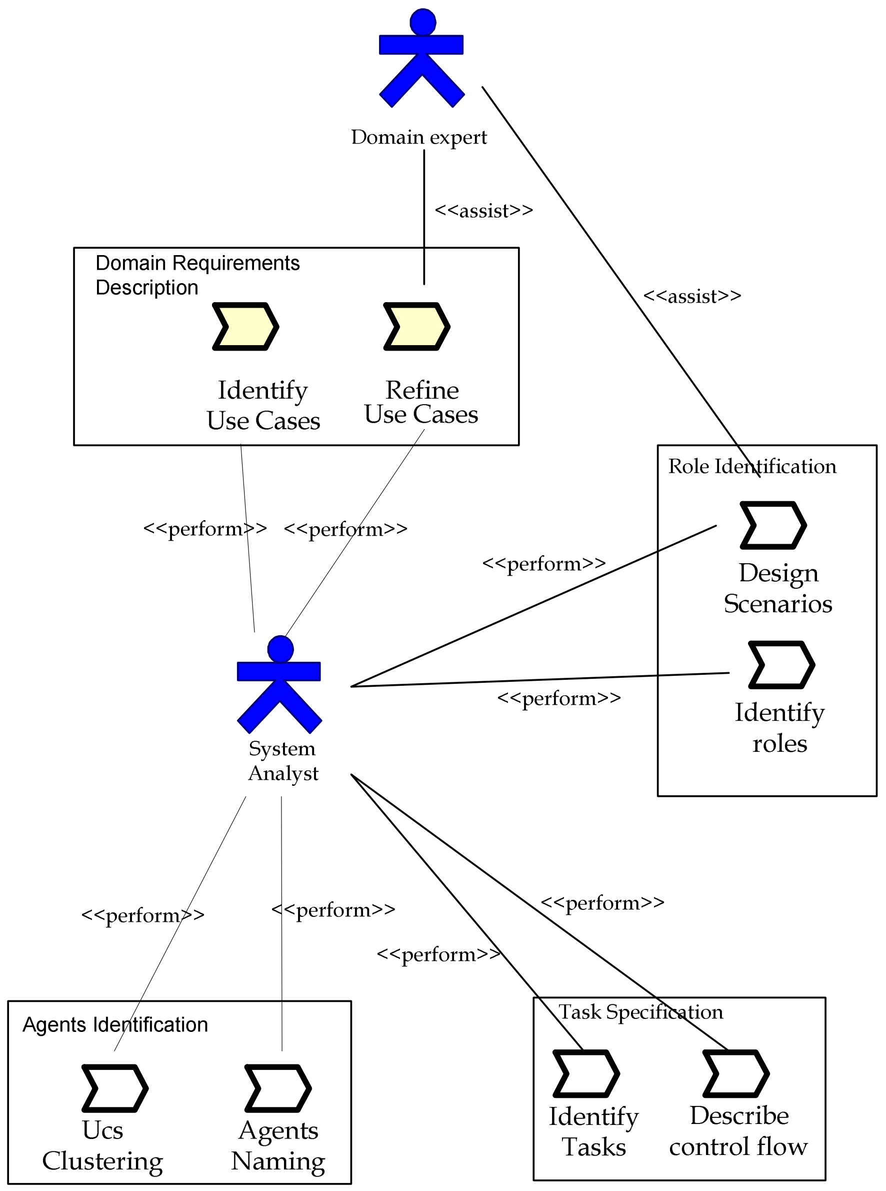


Figure 4. The Domain Requirements Description activities clustered in sub-phase level work definitions

Here is a summary of this phase activities:

Phase	Activity	Activity Description	Roles involved
Domain Description	Identify Use Cases	Use cases are used to represent system requirements	System Analyst (perform)
Domain Description	Refine Use Cases	Use cases are refined with the help of a Domain Expert	System Analyst (perform) Domain Expert (assist)
Agents Identification	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 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)
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 in designed in form of sequence diagram thus depicting the details of agents interactions	System Analyst (perform) Domain Expert (assist)
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	

3.1.1 Process roles involved

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

3.1.1.1 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.

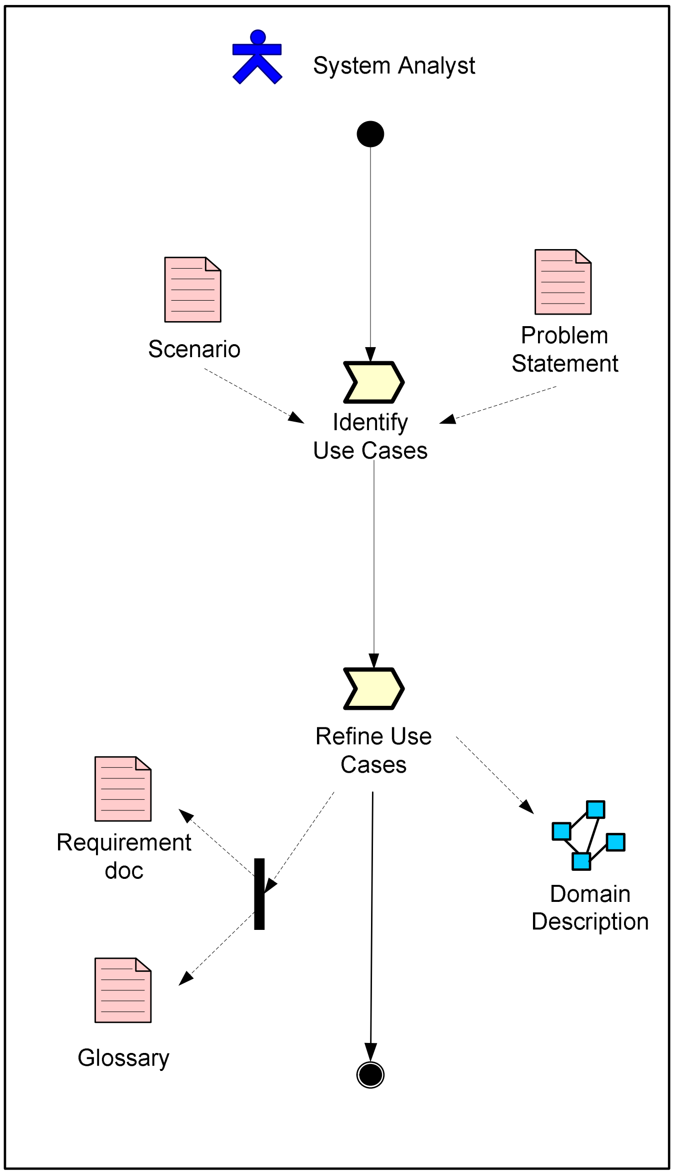
3. Use cases clustering during the AId sub-phase. The System Analyst analyzes the use case diagrams resulting from the previous phase and attempts their clustering in a set of packages.
4. Naming agents during the AId sub-phase. After grouping the use cases in a convenient set of packages, the last activity 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 in 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.

3.1.1.2 Domain Expert

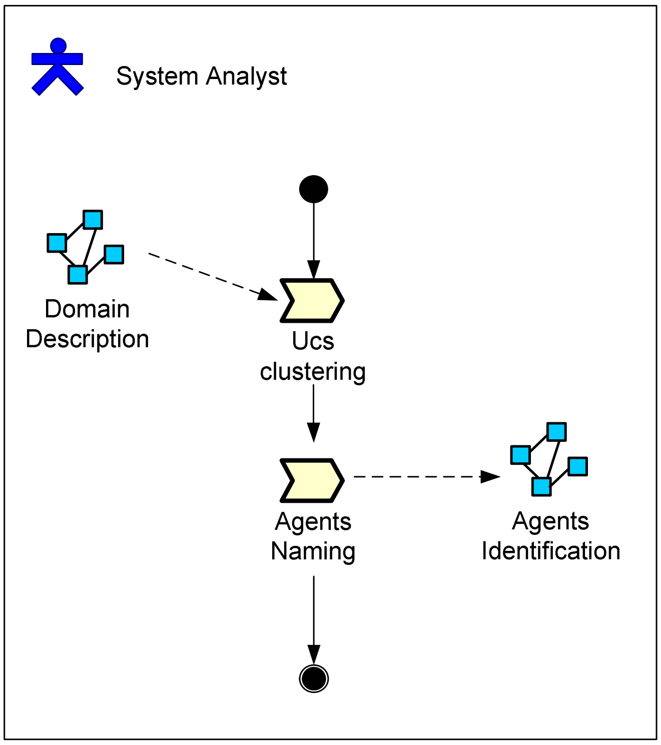
He supports the system analyst during the description of the domain requirements

3.1.2 Details on System Requirements sub-phases

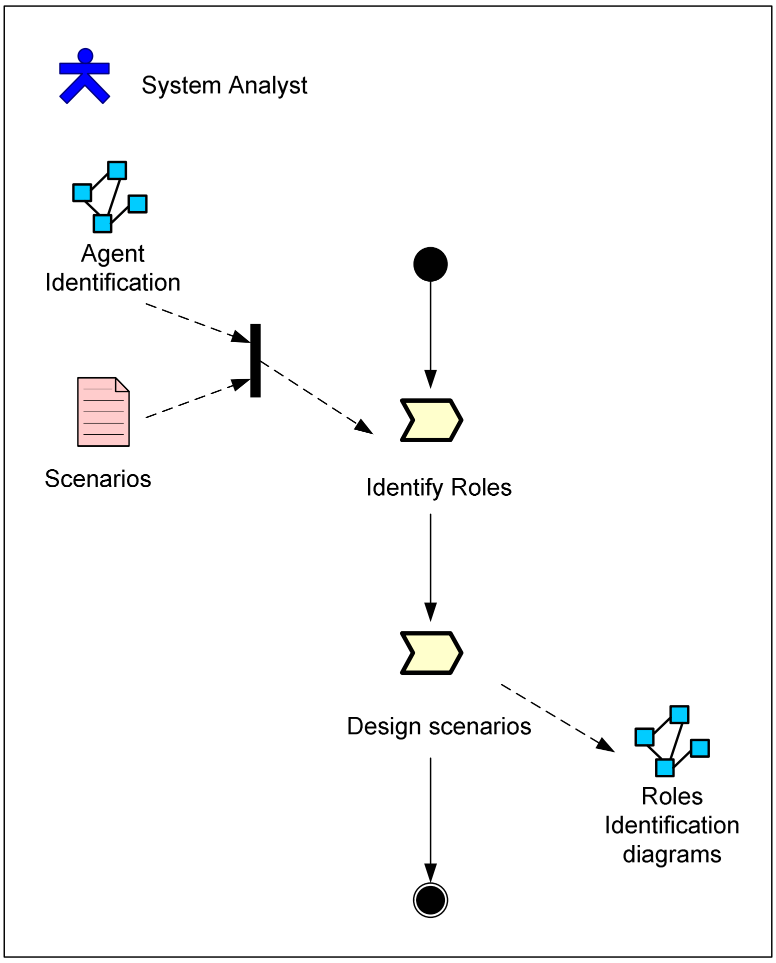
3.1.2.1 Domain Requirements Description



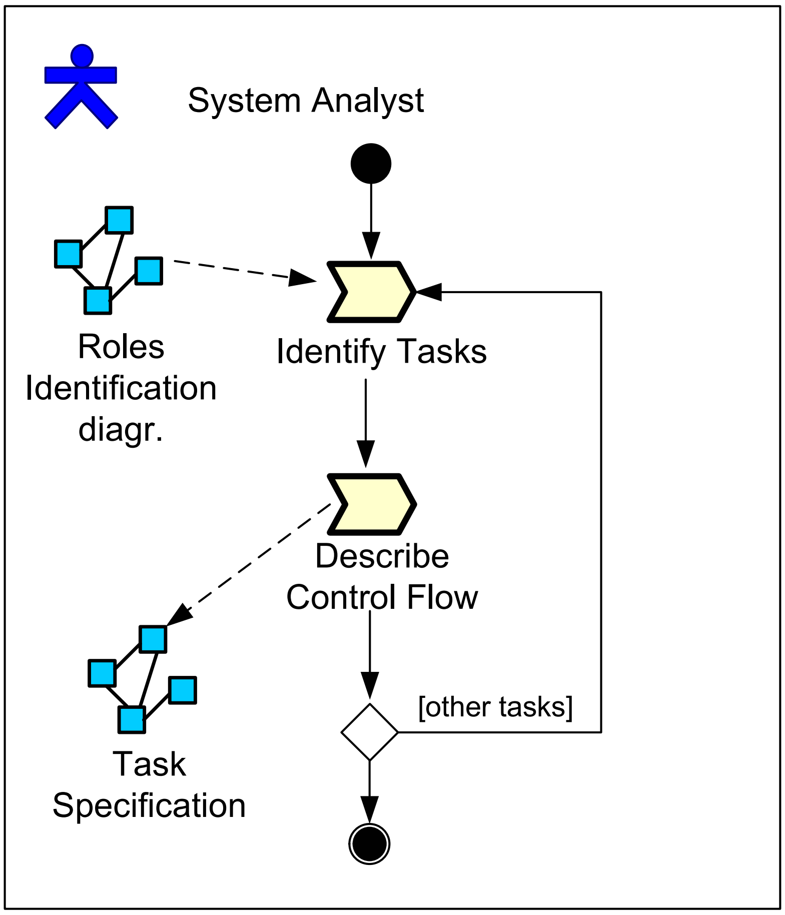
3.1.2.2 Agent Identification



3.1.2.3 Roles Identification



3.1.2.4 Task Specification



3.1.3 Work Products

The System Requirements Model includes four text documents and UML models. Their relationships with the MAS meta-model (see sub-section 5) are described in following

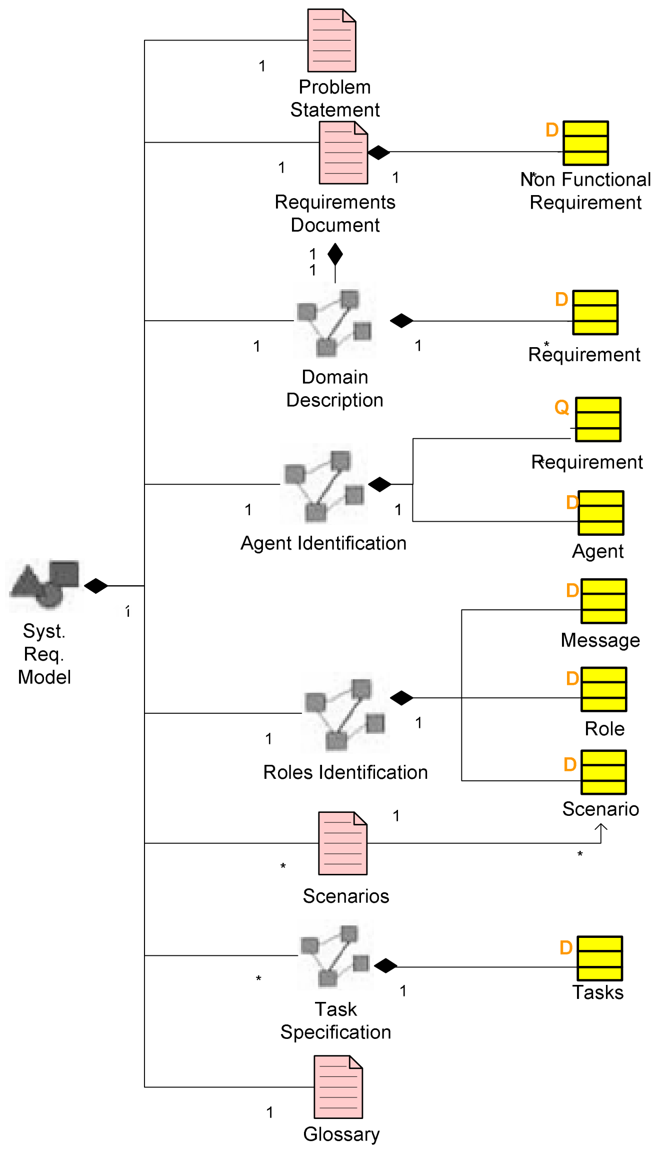


Figure 5. The System Requirements Model documents structure

This diagram represents the System Requirement model in terms of UML models and text documents. Each of these reports one or more elements from the PASSI MAS meta-model (see section 5); each MAS meta-model element is represented using an UML class icon (yellow filled) and in each of the documents such elements can be *defined* (D label near the element symbol, this means that the element is introduced for the first time in the design in this artefact), *refined* (R label, this means that an already defined element is detailed or someway updated) or simply *quoted* (Q label, this means that the elements has been already defined and is reported in this artefact only to complete its structure but no work has to be done on it).

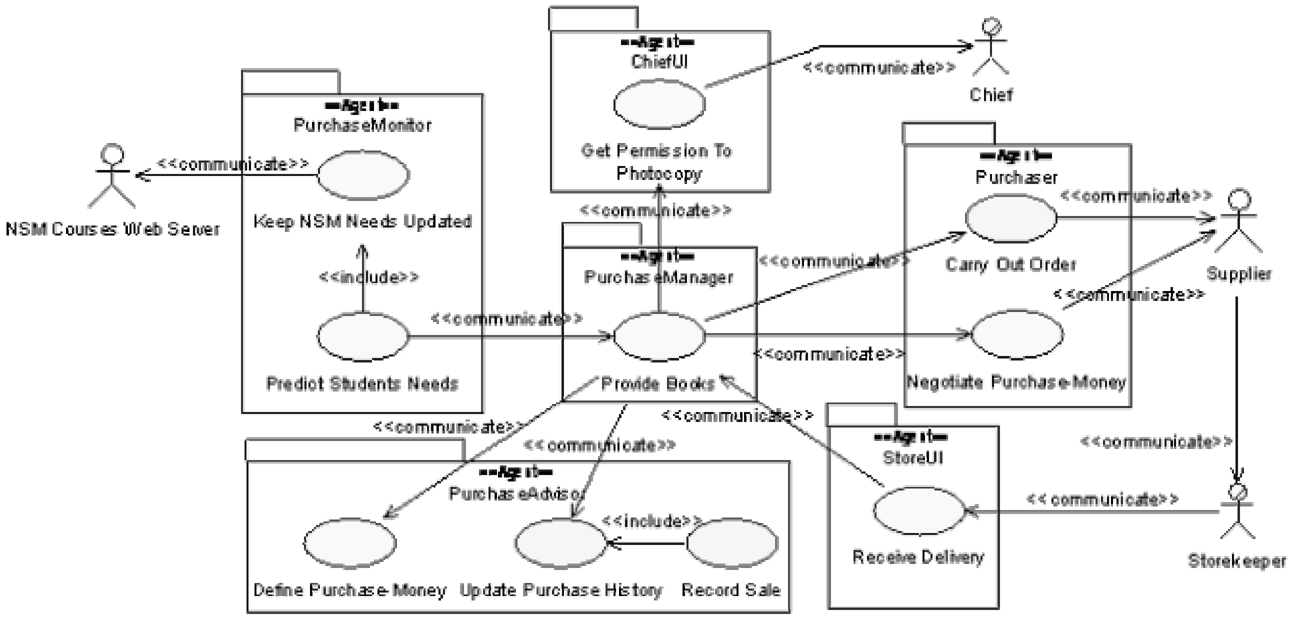
3.1.3.1 UML Models

The System Requirements discipline includes four UML Models whose aim and notation are described in the following sub-sections.

3.1.3.1.1 *Domain Description*

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

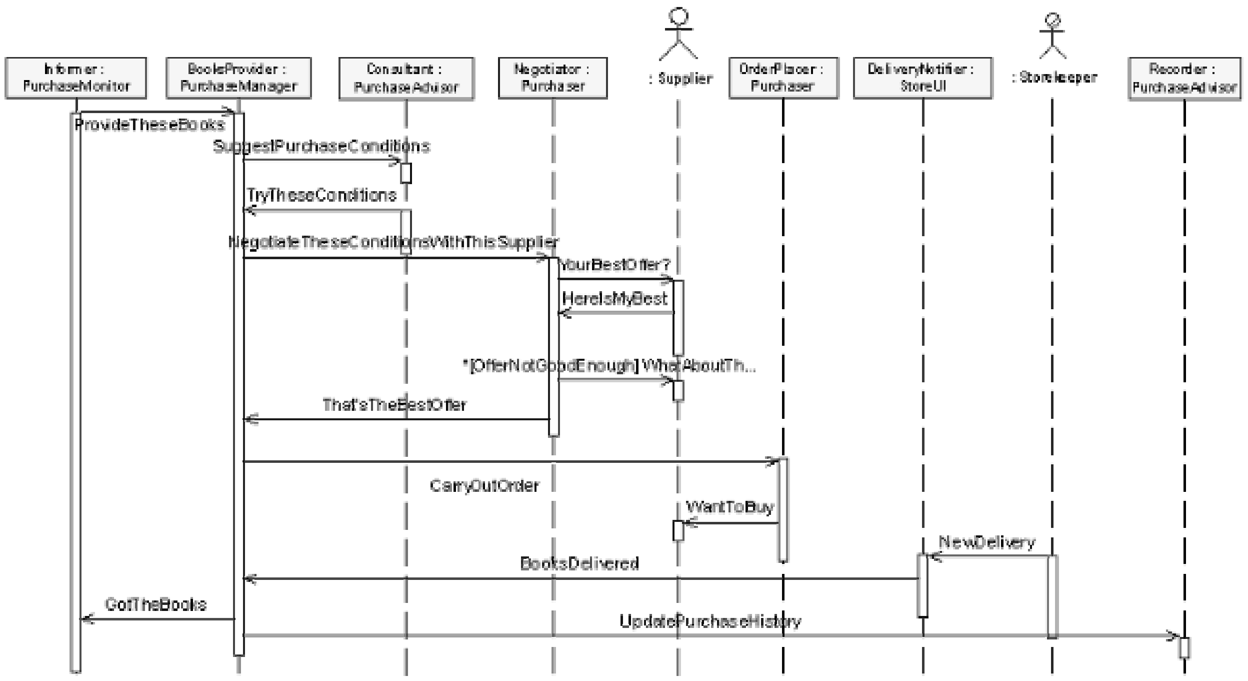
3.1.3.1.2 *Agent Identification*



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.

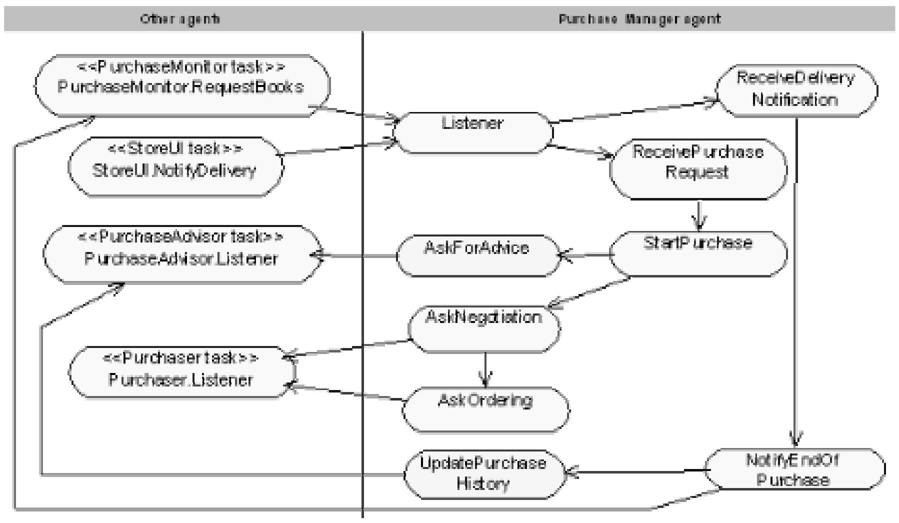
3.1.3.1.3 *Roles Identification*

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 symbolize roles (collection of tasks performed by agent in pursuing a sub-goal).


Informer : PurchaseMonitor
BookProvider : PurchaseManager
Consultant : PurchaseAdvisor
Negotiator : Purchaser
: Supplier
OrderPlacer : Purchaser
DeliveryNotifier : StoreUI
: Storekeeper
Recorder : PurchaseAdvisor
ProvideTheseBooks
SuggestPurchaseConditions
TryTheseConditions
NegotiateTheseConditionsWithThisSupplier
YourBestOffer?
HereIsMyBest
*[OfferNotGoodEnough] WhatAboutTh...
That'sTheBestOffer
CarryOutOrder
WantToBuy
NewDelivery
BooksDelivered
GotTheBooks
UpdatePurchaseHistory

3.1.3.1.4 *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).

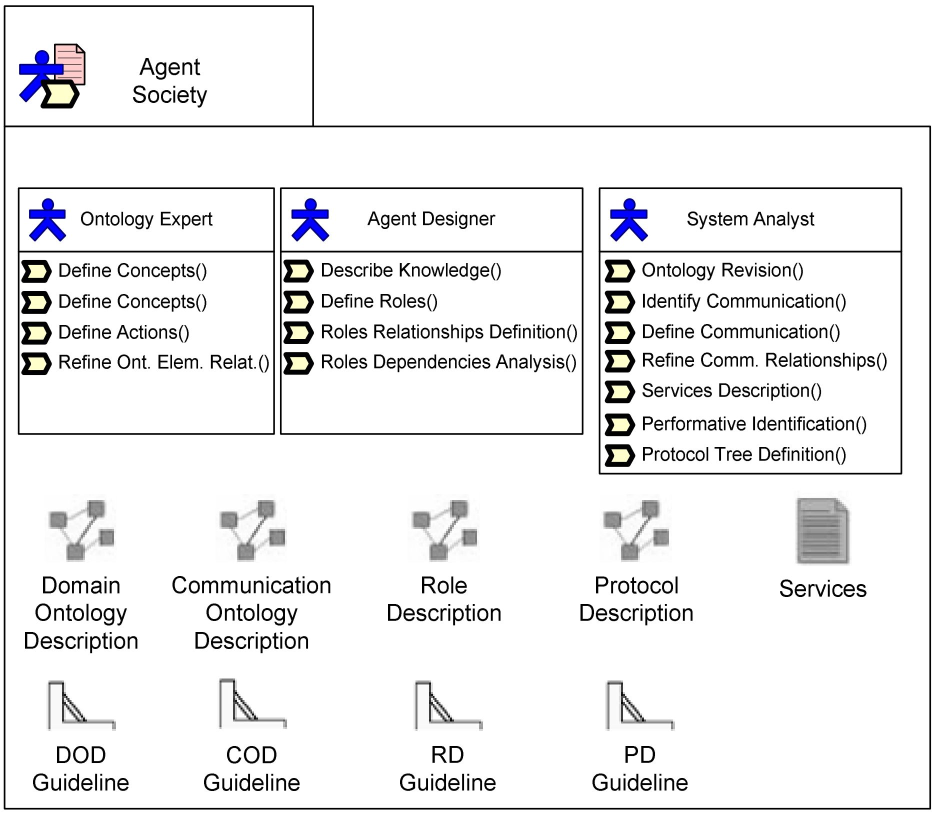

Other agents
Purchase Manager agent
<<PurchaseMonitor task>> PurchaseMonitor.RequestBooks
<<StoreUI task>> StoreUI.NotifyDelivery
<<PurchaseAdvisor task>> PurchaseAdvisor.Listener
<<Purchaser task>> Purchaser.Listener
Listener
ReceiveDelivery Notification
ReceivePurchase Request
StartPurchase
AskForAdvice
AskNegotiation
AskOrdering
UpdatePurchase History
NotifyEndOf Purchase

3.1.3.2 *Text Documents*

Name	Description
Problem Statement	A description of the problem to be solved with the system. It is complemented by the Scenario document
Scenarios	Textual description of the scenarios in which the system to be developed is involved.
Requirements Document	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
Glossary	A glossary of terms

3.2 The Agent Society phase



The Agent Society discipline involves three different process roles, five work products (four UML models and one document) and four guidance (one for each UML model)

The process to be performed in this phase is described in the following Figure 6. It is composed of four sub-phase level work definitions (Domain Ontology Description, Communication Ontology Description, Roles Description, and Protocol Description) and several related work products (UML models and text documents)

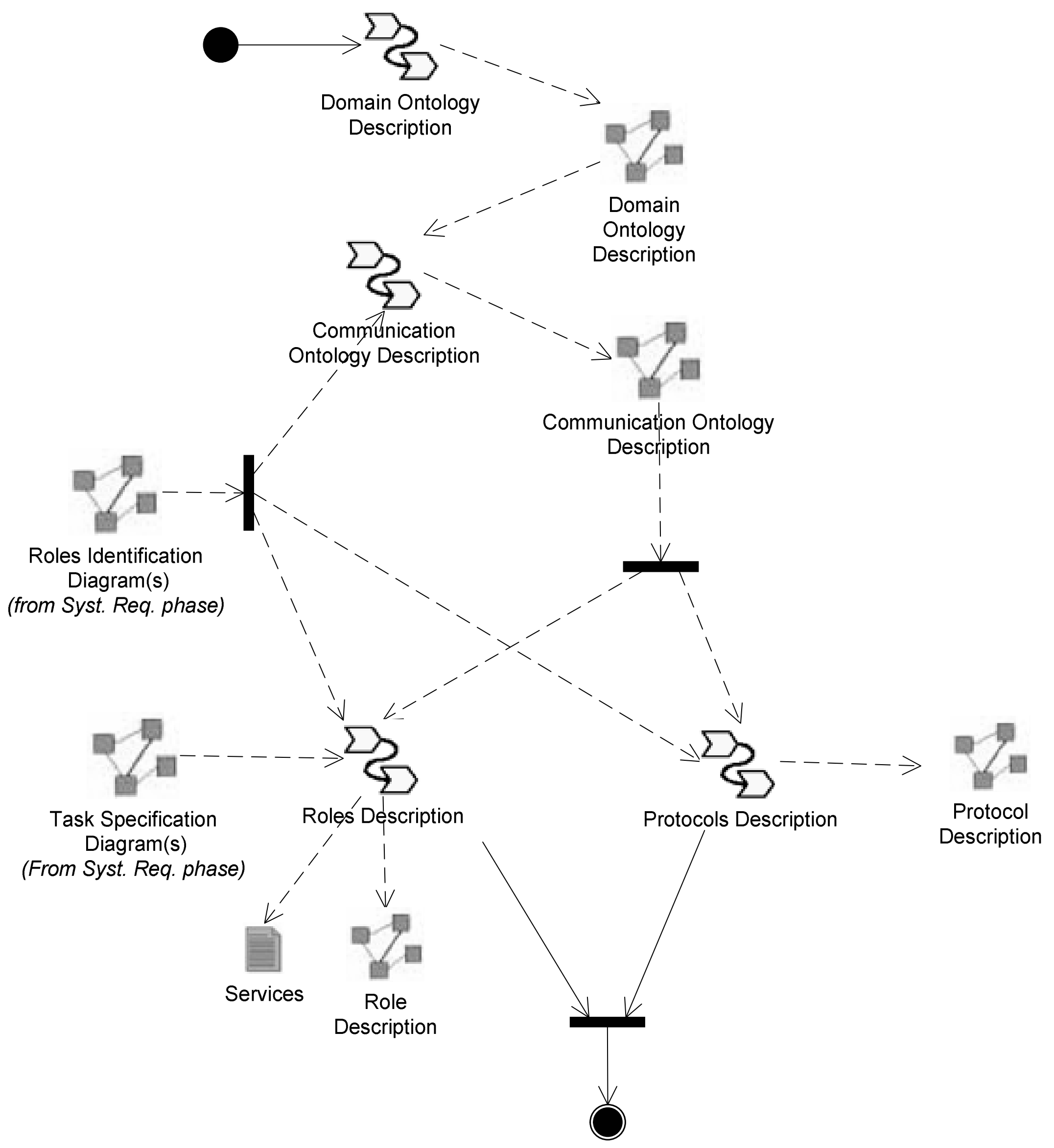

Domain Ontology Description
Domain Ontology Description
Communication Ontology Description
Communication Ontology Description
Roles Identification Diagram(s) *(from Syst. Req. phase)*
Task Specification Diagram(s) *(From Syst. Req. phase)*
Roles Description
Protocols Description
Protocol Description
Services
Role Description

Figure 6. The Agent Society phase described in terms of work definitions (sub-phases) and work products

The four sub-phases are composed of several activities as described in Figure 4. This figure also describes the actors (process roles) involved in this portion of the process. The process flow will be described in following sub-sections.

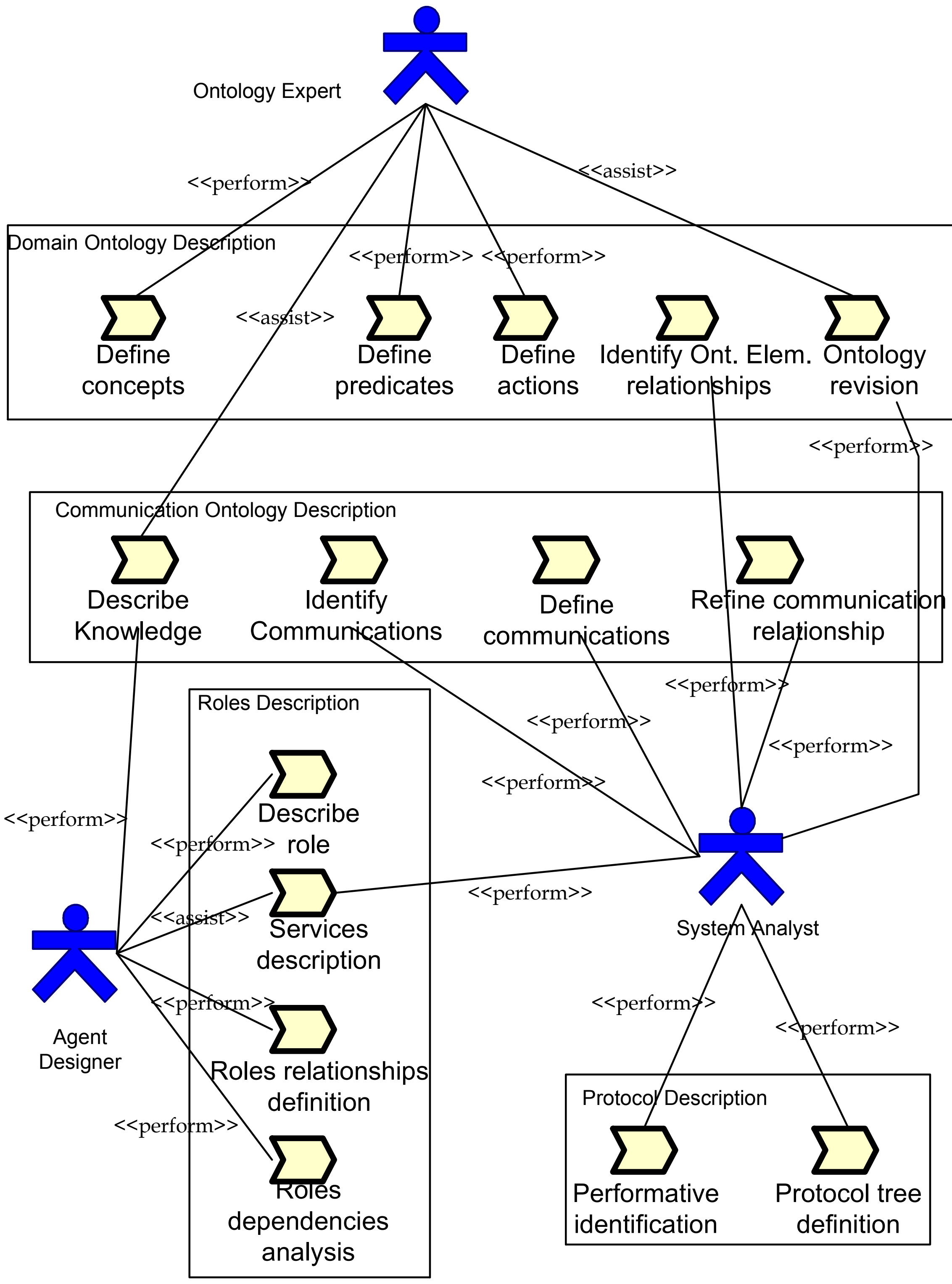


Figure 7. The Agent Society activities clustered in sub-phases level work definitions

Here is a summary of this phase activities:

Phase	Activity	Activity Description	Roles involved
Domain Ontology Description	Define Concepts	During this activity the ontology expert perform: the identification of the	Ontology Expert (perform)

		concepts describing the system domain	
Domain Ontology Description	Define Predicates		Ontology Expert (perform)
Domain Ontology Description	Define Actions		Ontology Expert (perform)
Domain Ontology Description	Identify Ont. Elem. Relationships		Ontology Expert (perform)
Domain Ontology Description	Ontology Revision		System Analyst (perform), Ontology Expert (assist)
Communication Ontology Description	Describe knowledge		Agent Designer (perform) Ontology Expert (assist)
Communication Ontology Description	Identify Communications		System Analyst (perform)
Communication Ontology Description	Define Communications		System Analyst (perform)
Communication Ontology Description	Refine Communication Relationships		System Analyst (perform)
Roles Description	Describe Roles	It consists in the description of roles' classes arranged in packages, each package representing one agent, whose behaviour is represented by tasks in the operation compartment.	Agent Designer (perform)
Roles Description	Roles Relationships Definition	The definition of association between roles (communications, dependencies, role changes). Communications derive from COD, changes of role from RID, dependencies from both COD and RID.	Agent Designer (perform)
Roles Description	Collaboration Rules Definition	In consists in the introduction of the agent society rules and the analysis of the collaborations an agent	Agent Designer (perform)

		needs. These depend on the society model that the design wants to introduce. These rules prevent some roles to provide services to other roles if they do not satisfy the required condition (actually reported in the Services description document). Goals of roles are defined in this activity too, and describe the objective of the specific role (when this is not involved in providing some service or sharing resources)	
Roles Description	Services Description	The description of service dependencies between two roles (service, resource, soft service and soft resource). A resource dependency is seen as a resource providing service and modelled in the Services description document.	System Analyst (perform) Agent Designer (assist)
Protocol Description	Performative Identification		System Analyst (perform)
Protocol Description	Protocol Tree Definition		System Analyst (perform)

3.2.1 Process roles involved

3.2.1.1 *Ontology Expert*

He is responsible of :

1. Concepts definition. It consists in the identification of the concepts describing the system domain.
2. Predicates definition. The identification of predicates, the assertions relating concepts to the system domain.
3. Actions definition. The identification of activities that agent may perform.
4. Ontology element relationships refinement. It consist in relating the previous three elements with classical ontological relationships (inheritance, aggregation, association)

3.2.1.2 Agent Designer

He is responsible of :

1. Roles description. It consists in the description of roles' classes arranged in packages, each package representing one agent, whose behaviour is represented by tasks in the operation compartment.
2. Roles relationships definition. The definition of association between roles (communications, dependencies, role changes). Communications derive from COD, changes of role from RID, dependencies from both COD and RID
3. Collaboration Rules Definition. In consists in the introduction of the agent society rules and the analysis of the collaborations an agent needs. These depend on the society model that the design wants to introduce. These rules prevent some roles to provide services to other roles if they do not satisfy the required condition (actually reported in the Services description document).
4. Knowledge description. It consists in listing agents specifying their knowledge represented as attributes.

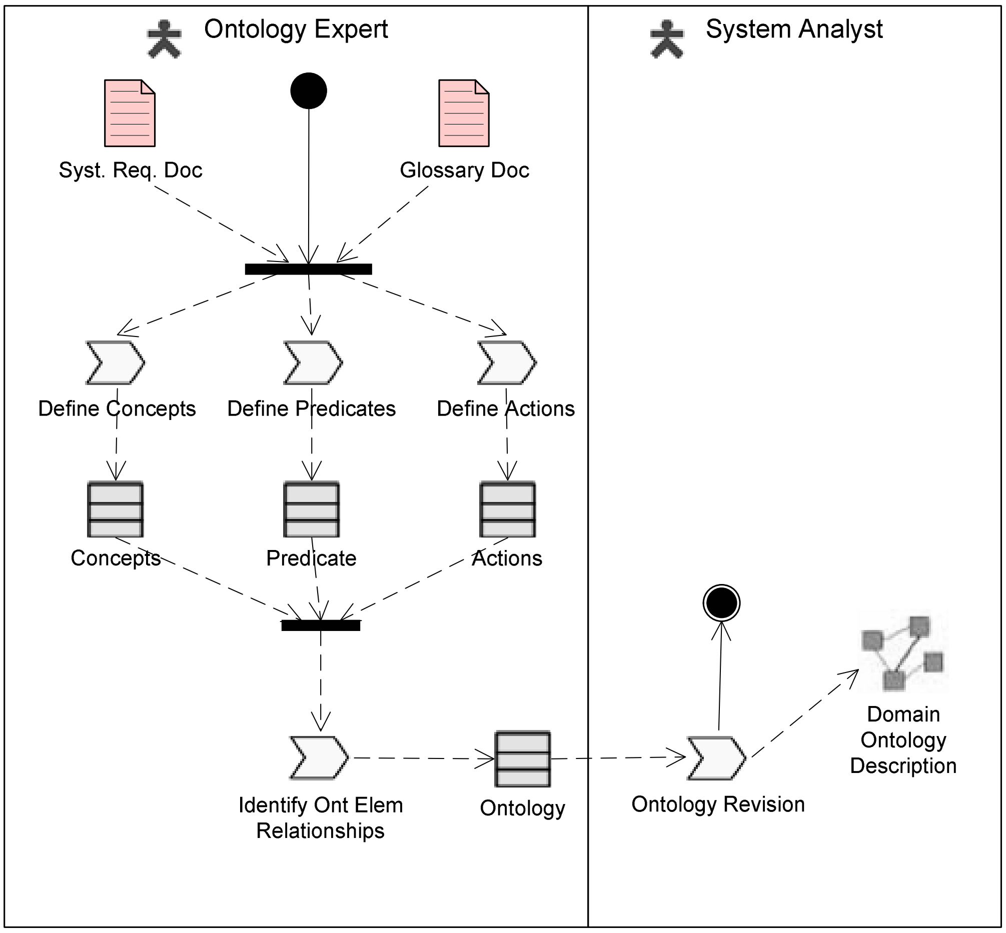
3.2.1.3 System Analyst

He is responsible of :

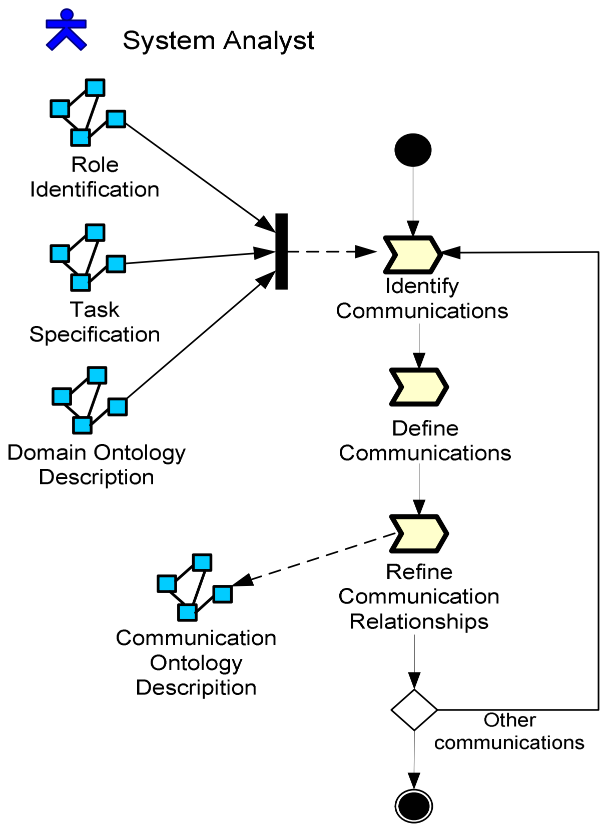
1. Ontology revision. The revision of ontological elements in order to define the pieces of knowledge of each agent and their communication ontology.
2. Communications identification. It consists in introducing an association for each communication between two agents, looking at exchanged messages in the scenario.
3. Communications definition. The description of agents communication in term of ontology , content language and interaction protocol.
4. Communication relationships refinement. The identification of association classes in order to link each communication to the three fundamental element of communication itself (ontology , language and protocol).
5. Services description. The description of service dependencies between two roles (service, resource, soft service and soft resource)
6. Performative identification
7. Protocol tree definition

3.2.2 Details on Agent Society sub-phases

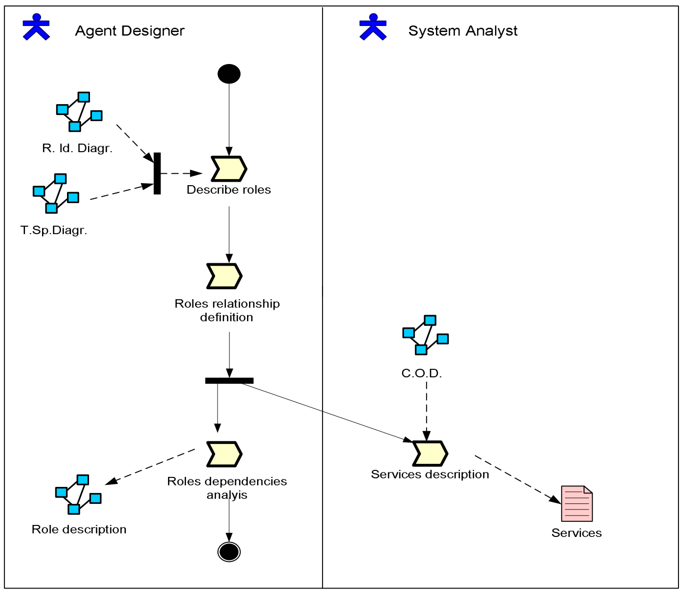
3.2.2.1 Domain Ontology Description



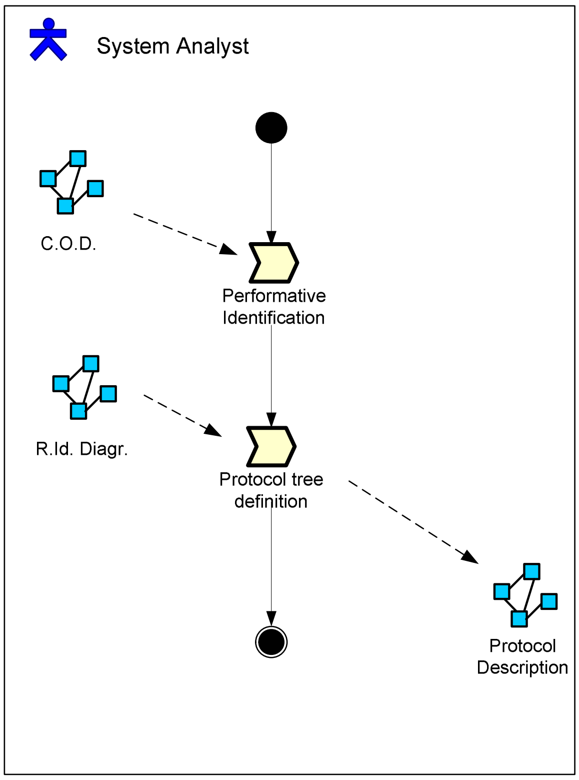
3.2.2.2 Communication Ontology Description



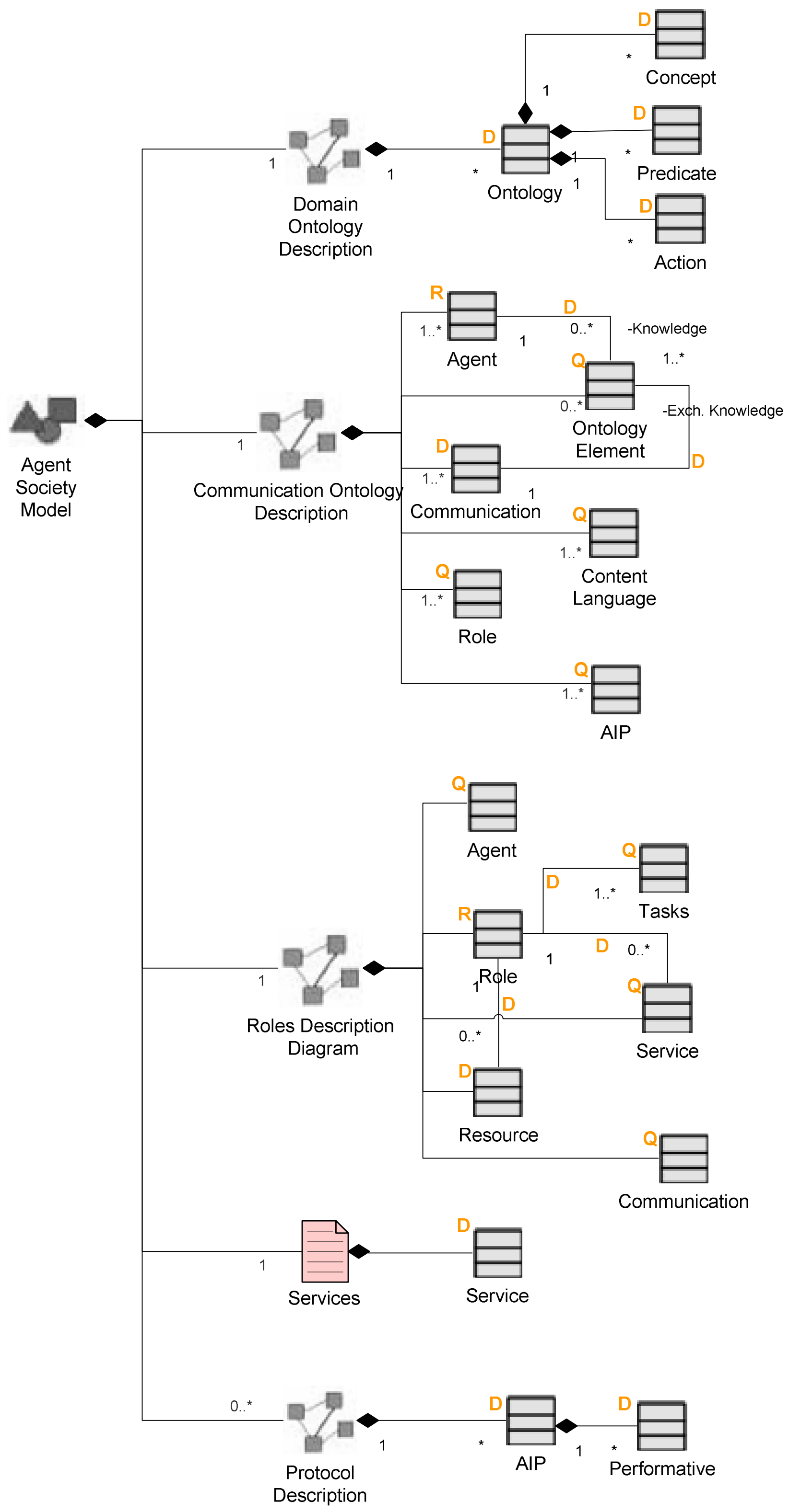
3.2.2.3 Roles Description



3.2.2.4 Protocols Description


System Analyst
C.O.D.
Performative Identification
R.Id. Diagr.
Protocol tree definition
Protocol Description

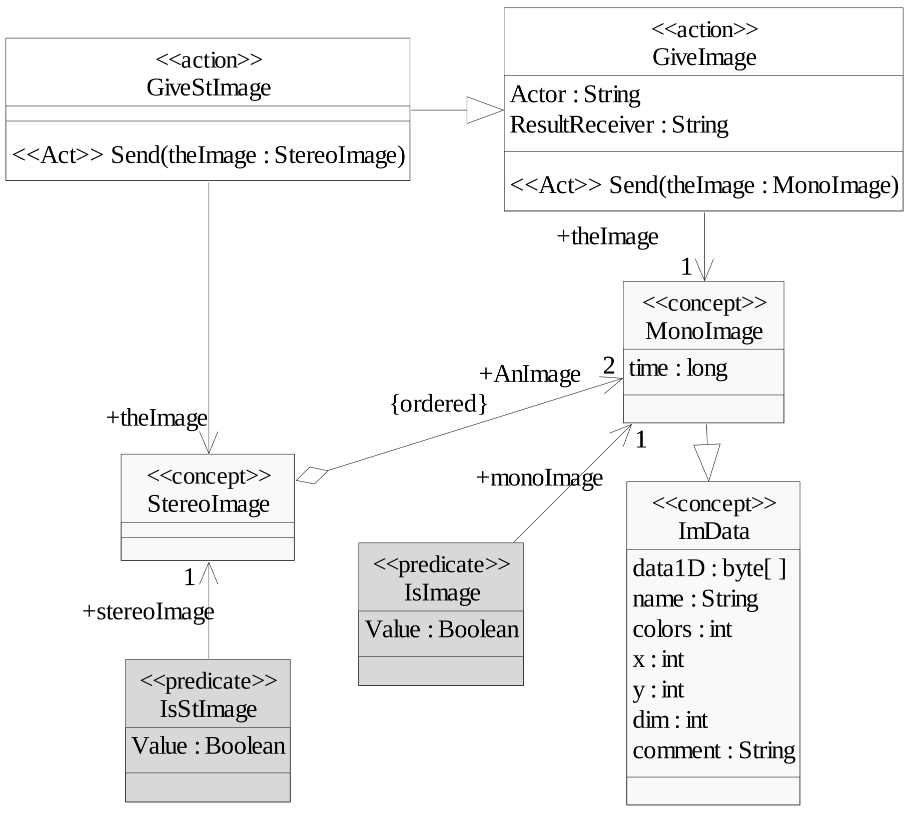
3.2.3 Work Products Structure



3.2.3.1 UML Models

The Agent Society discipline includes four UML Models whose aim and notation are described in the following sub-sections.

3.2.3.1.1 Domain Ontology Description



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*).

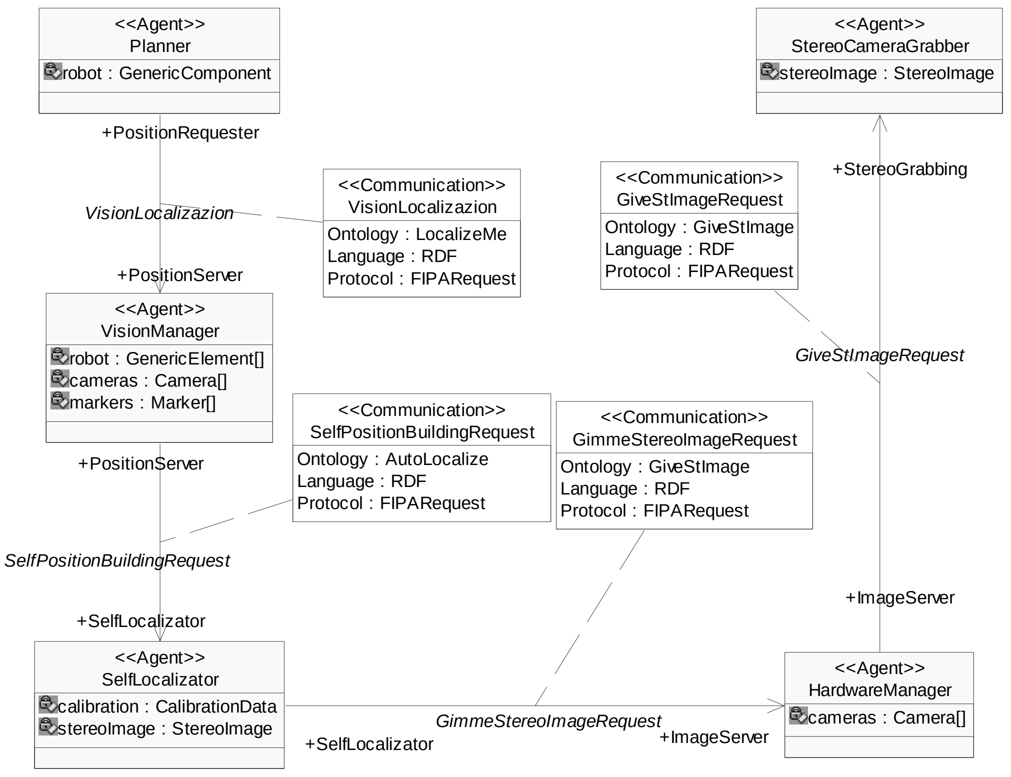
3.2.3.1.1.1 Dependencies

This diagram describes/relates/defines:

Element	From	Notes
Concepts	System Requirements doc, Glossary doc	
Actions		
Predicates		
Ontology elements Relationships		Generalization, aggregation,…

3.2.3.1.2 Communication Ontology Description

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



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 an 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.

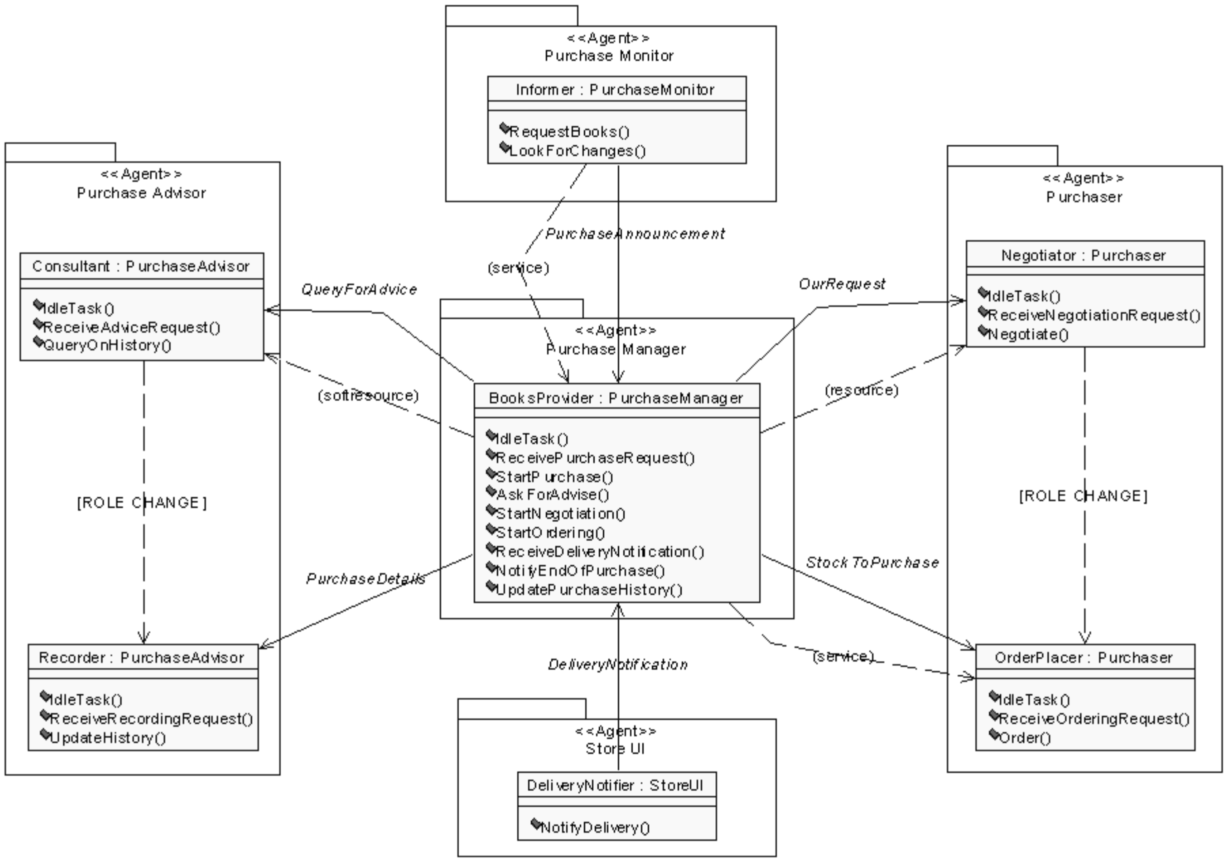
3.2.3.1.2.1 Dependencies and new definitions

This diagram describes/relates/defines the following elements of the MAS (see also the MAS model in section 5):

Element	From	Notes
Agents	R. Id. UML Model	
Tasks	Tasks Specification UML Model	Not shown in the diagram in this version of the interface/diagram but selected

		by the designer that relates a task to each incoming/outgoing communication of each agent
Agents Ontology (Knowledge)	D.O.D. UML Model	Pieces of ontology. Usually they are referred to the knowledge exchanged in the agents' communications
Agents' Roles	R. Id. UML Model	In the communications (reported as roles in the relationships among agents)
Communication Name	-	Defined here by the tool (syntax: <initiator agent name>-<participant agent name><serial number for communications among the same agents>), the designer can modify it
Communication Ontology	D.O.D. UML Model	Chosen by the Designer among the elements of the DOD
Communication Protocol	Roles Id. UML Model	Chosen by the designer considering the exchanged knowledge and the communication as described in the Roles Id. diagram
Communication Content Lang.	-	Chosen from the designer

3.2.3.1.3 Roles Description



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.

3.2.3.1.4 Protocols Definition

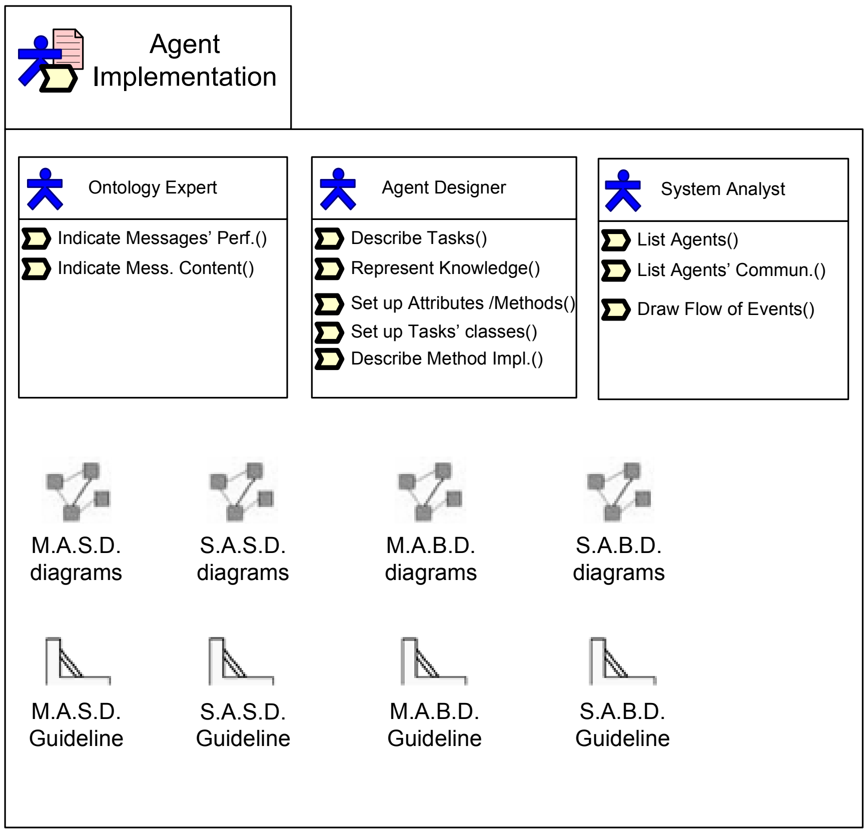
We use sequence diagram to describe interaction protocols like in AUML.

3.2.3.2 Text Documents

3.2.3.2.1 Services

A description of the services offered by the agents as referred in the Roles Description UML model (see 3.2.2.3)

3.3 The Agent Implementation phase



The Agent Implementation discipline involves the same roles of the preceding phase , four work products (UML models) and four guidance documents (one for each UML model)

The process to be performed in this phase is described in the following Figure 6. It is composed of four sub-phase level work definitions (Multi-Agent Structure Definition, Single-Agent Structure Definition, Multi-Agent Behaviour Description, Single-Agent Behaviour Description) and several related work products.

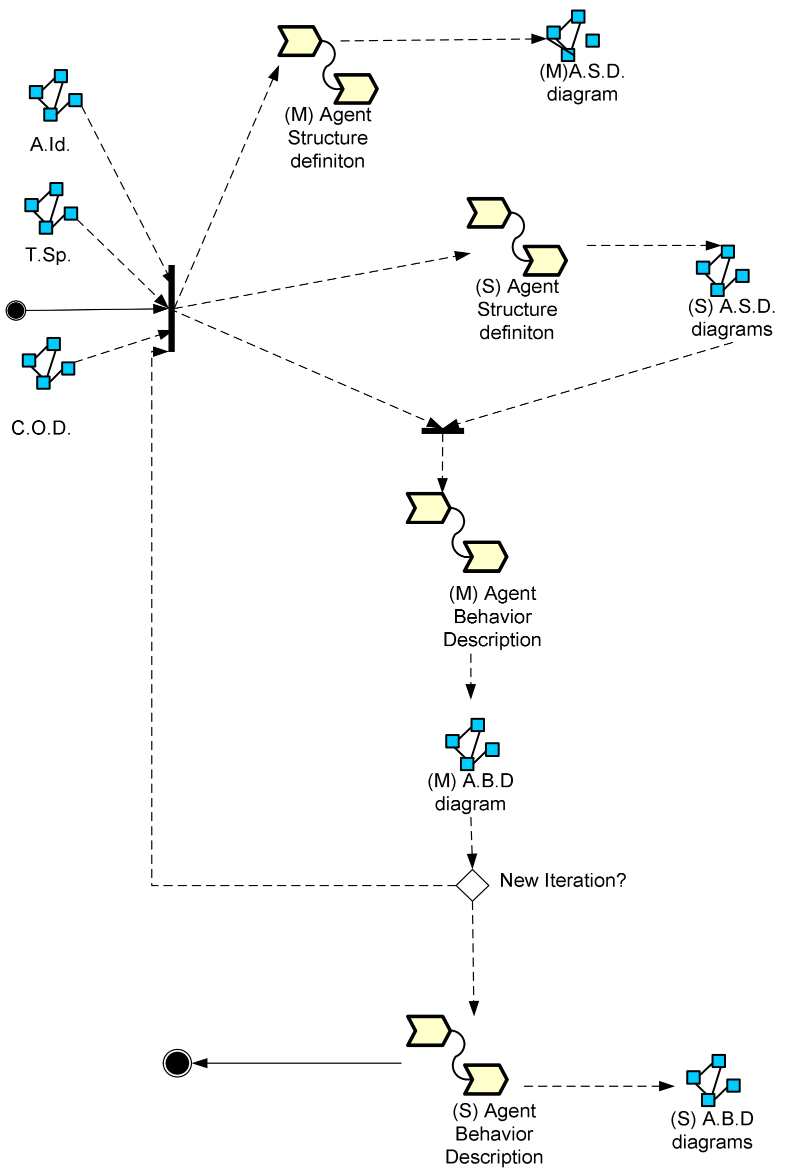

A.Id.
T.Sp.
C.O.D.
(M) Agent Structure definiton
(M)A.S.D. diagram
(S) Agent Structure definiton
(S) A.S.D. diagrams
(M) Agent Behavior Description
(M) A.B.D diagram
New Iteration?
(S) Agent Behavior Description
(S) A.B.D diagrams

Figure 8. The Agent Implementation phase described in terms of work definitions (sub-phases) and work products

The four sub-phases are composed of several activities as described in Figure 4. This figure also describes the actors (process roles) involved in this portion of the process. The process flow will be described in following sub-sections.

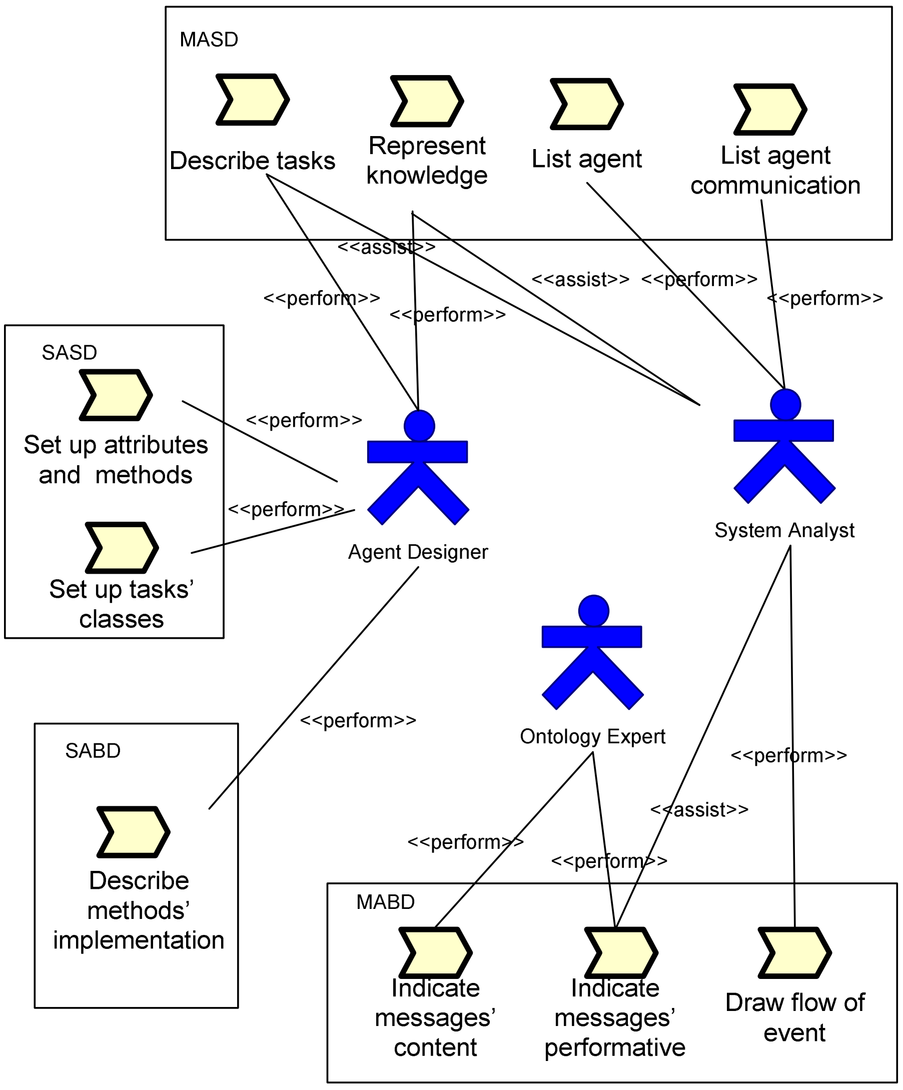


Figure 9. The Agent Implementation activities clustered in sub-phases level work definitions

Here is a summary of this phase activities:

Phase	Activity	Activity Description	Roles involved
M.A.S.D.	Describe Tasks		Agent Designer (perform) System Analyst (assist)
M.A.S.D.	Represent Knowledge		Agent Designer (perform) System Analyst (assist)
M.A.S.D.	List Agent		System Analyst (perform)
M.A.S.D.	List Agent Communication		System Analyst (perform)

S.A.S.D.	Set up Attributes and Methods	the constructor and the shutdown method required by FIPA_OS environment).	Agent Designer (perform),
S.A.S.D.	Set up Tasks' classes	methods required to deal with communication events.	Agent Designer (perform)
M.A.B.D.	Indicate Messages' Performative	.	Ontology Expert (perform) System Analyst (assist)
M.A.B.D.	Indicate Messages' Content		Ontology Expert (perform)
M.A.B.D.	Draw Flow of Events		System Analyst (perform)
S.A.B.D.	Describe Methods Implementation	most appropriate form.	Agent Designer (perform)

3.3.1 Process roles involved

3.3.1.1 Ontology Expert

He is responsible of :

1. Indicate messages' performative. It consists in the connection between methods of each task pointing out messages' performative as it is specified in the C.O.D. diagrams.
2. Indicate messages' content. The same activity of the previous step this time the messages' content as it is described in the D.O.D. diagrams.

3.3.1.2 System Analyst

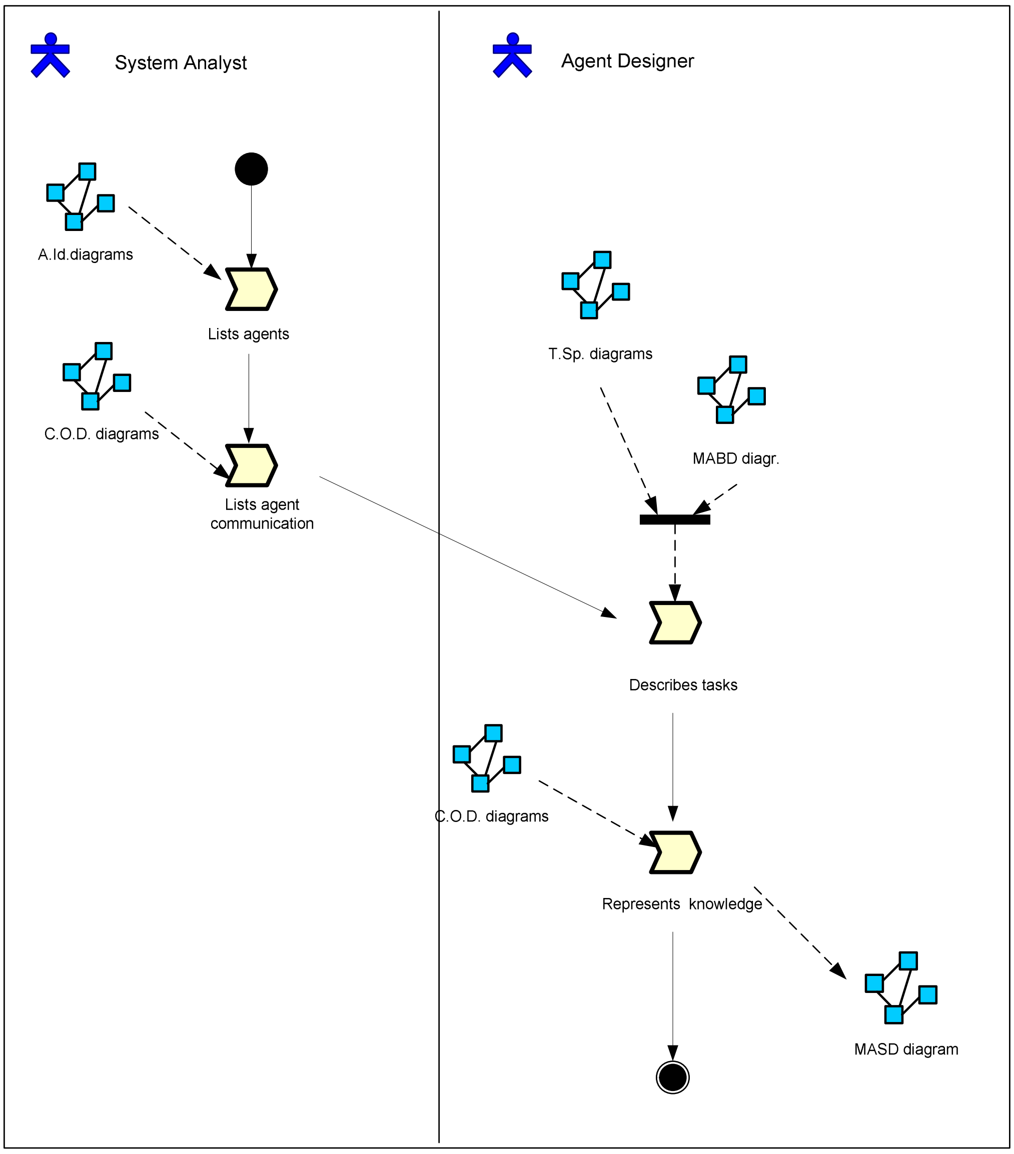
1. List agents. It consists in drawing a class diagram for the whole system , each class corresponding to an agent.
2. List agents' communication. It consists in joining two or more agent pointing out the communications they exchange.
3. Draw flow of event. The description of the flow of event between agents.

3.3.1.3 Agent Designer

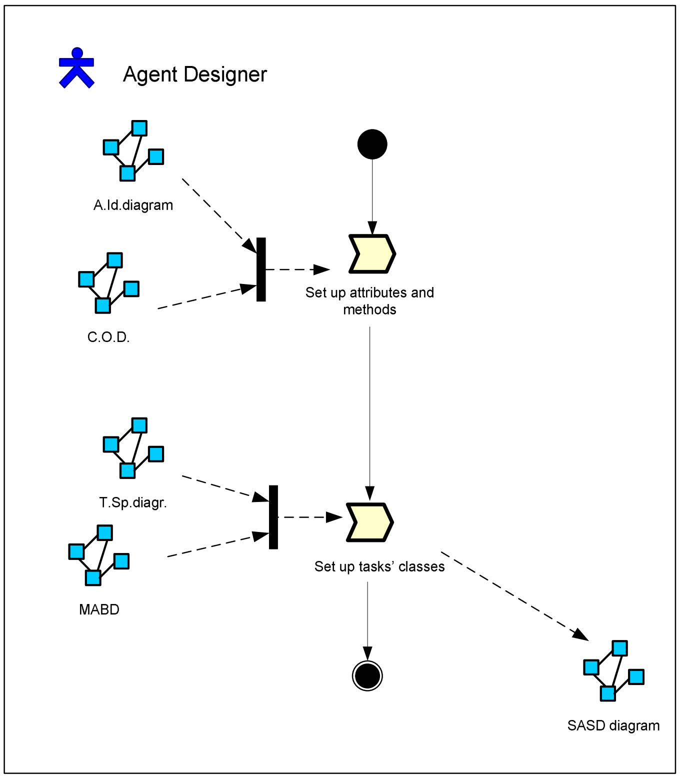
1. Describe tasks. The description of tasks for each agent.
2. Represent knowledge. The description of attributes for each class (agent) in order to represent a piece of knowledge on the domain.
3. Set up attributes and methods. It consists in representing each agent interior structure (e.g the constructor and the shutdown method required by FIPA_OS environment).
4. Set up tasks' classes. It consists in detailing each task (for each agent) pointing out methods required to deal with communication events.
5. Describe methods implementation. The description of methods implementation using the most appropriate form.

3.3.2 Details on Agent Implementation sub-phases

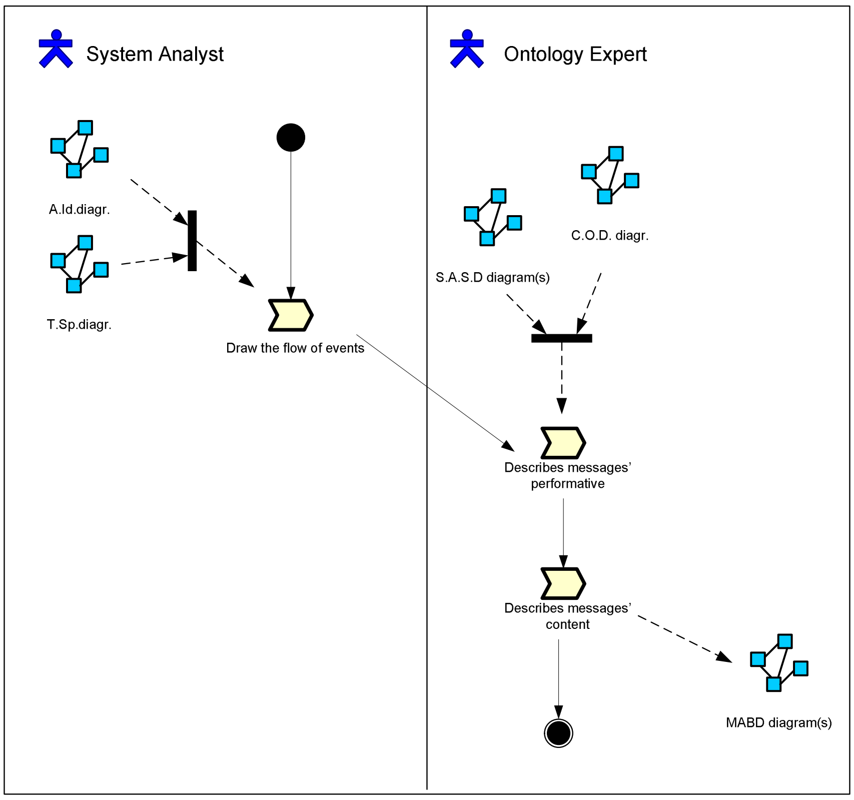
3.3.2.1 Multi Agent Structure Definition



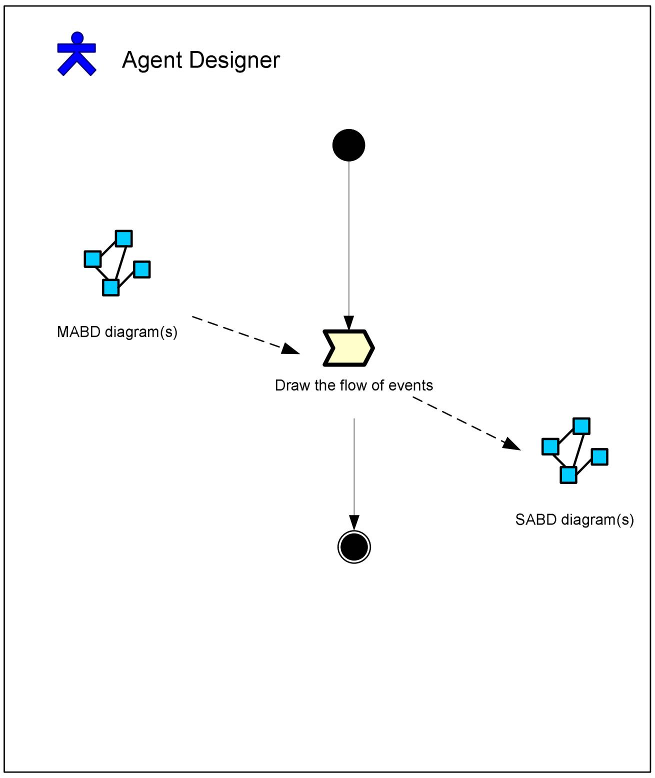
3.3.2.2 Single Agent Structure Definition



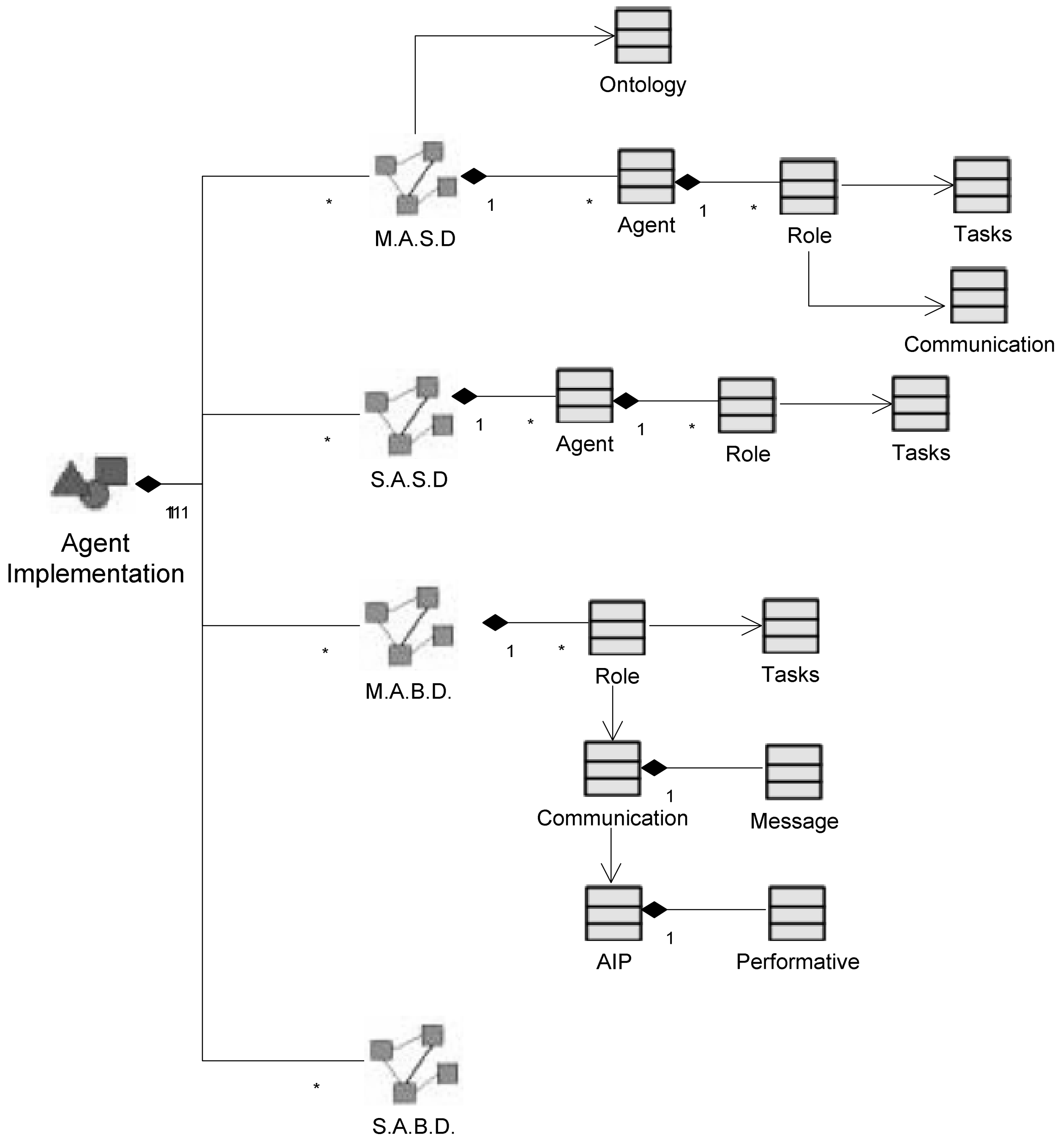
3.3.2.3 Multi Agent Behaviour Description



3.3.2.4 Single Agent Behaviour Description



3.3.3 Work Products Structure



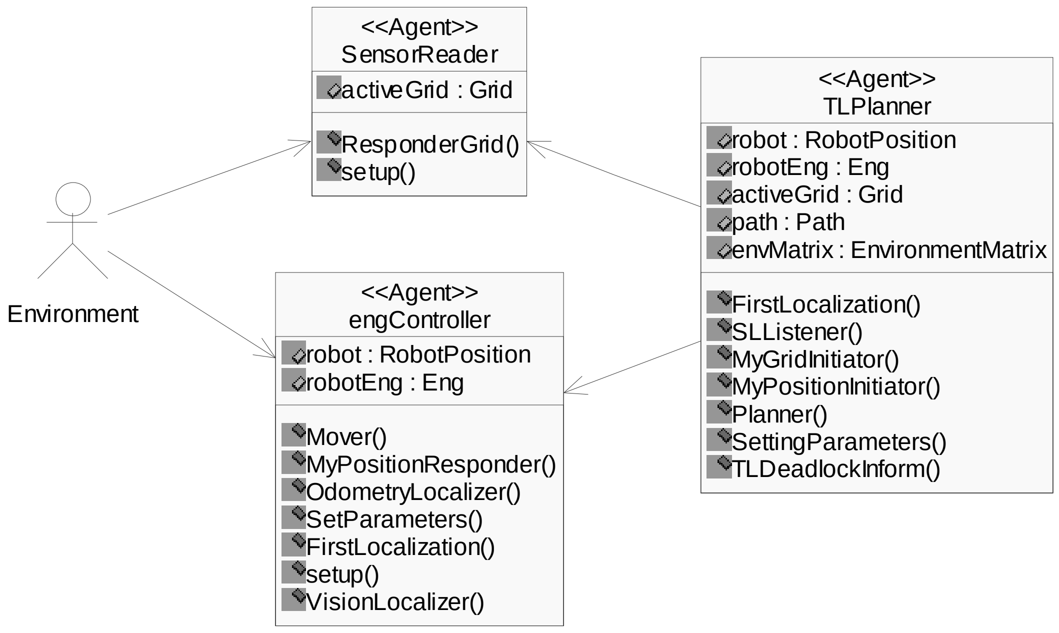
3.3.3.1 UML Models

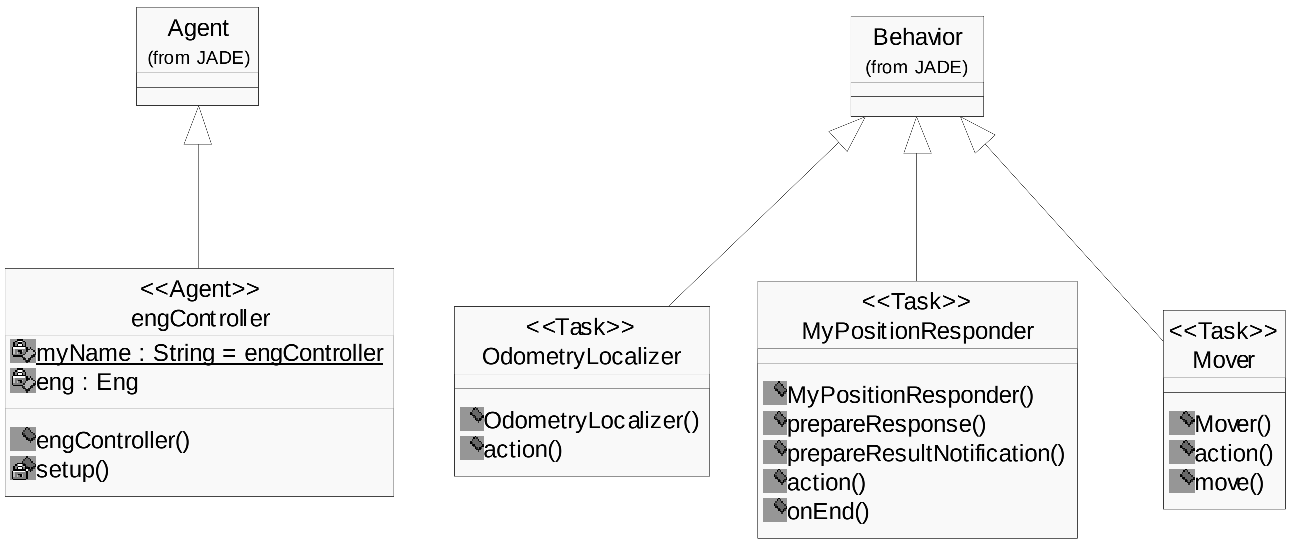
The Agent Implementation discipline includes three UML Models whose aim and notation are described in the following sub-sections.

3.3.3.1.1 Agent Structure definition

A series of class diagram(s), subdivided into two views, the multi-agent and the single-agent one, are used to describe the internal structure (in terms of classes and methods) of agents involved in the process.

In multi-agent Structure Definition the attention is focused on general architecture of the system (agents their knowledge and their tasks).
In Single Agent Structure Definition the attention is focused on each agent and its internal structure (attributes and methods).
Moreover this phase influences and is influenced by the next phase , the Agent Behaviour Description.

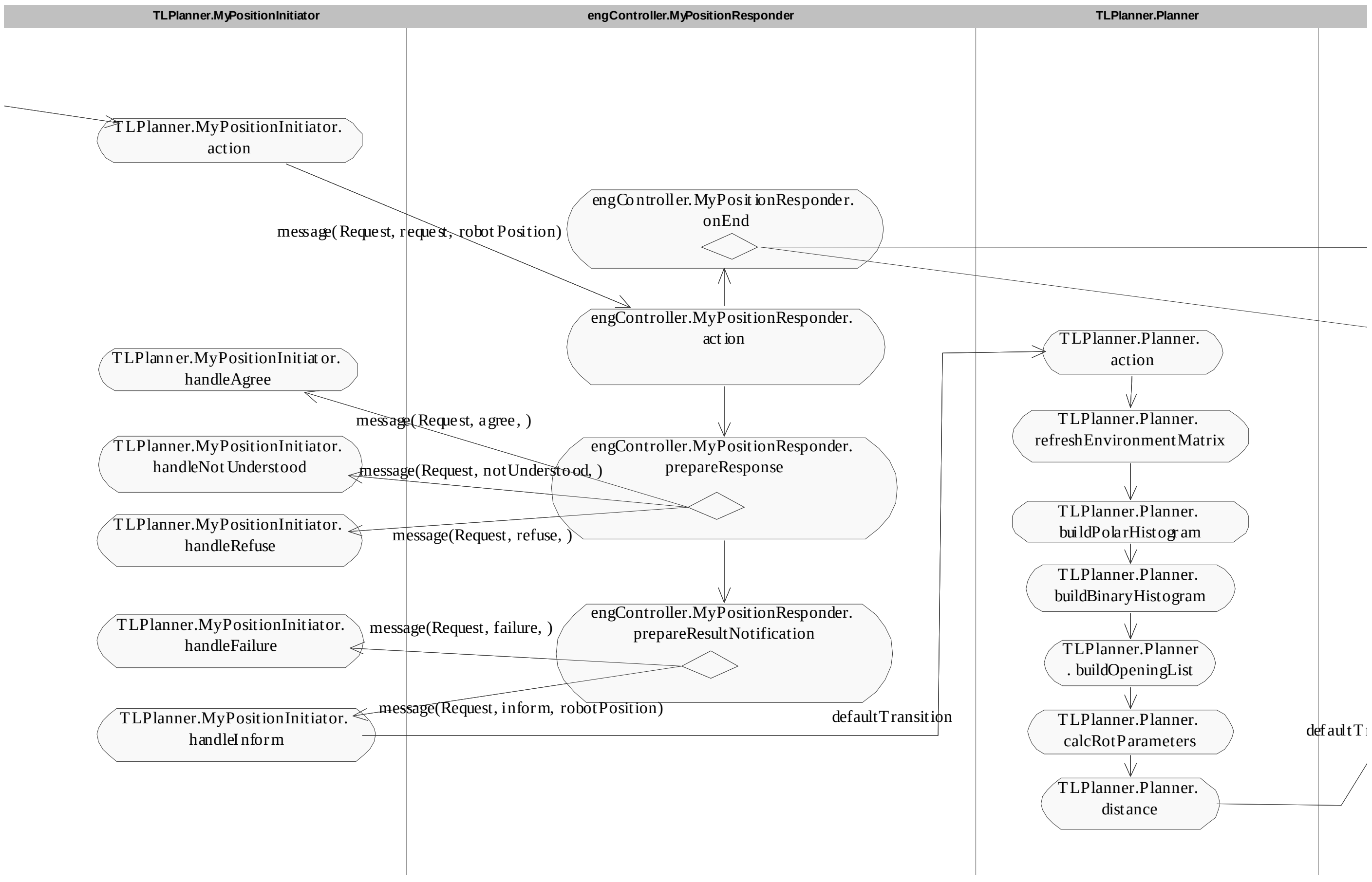




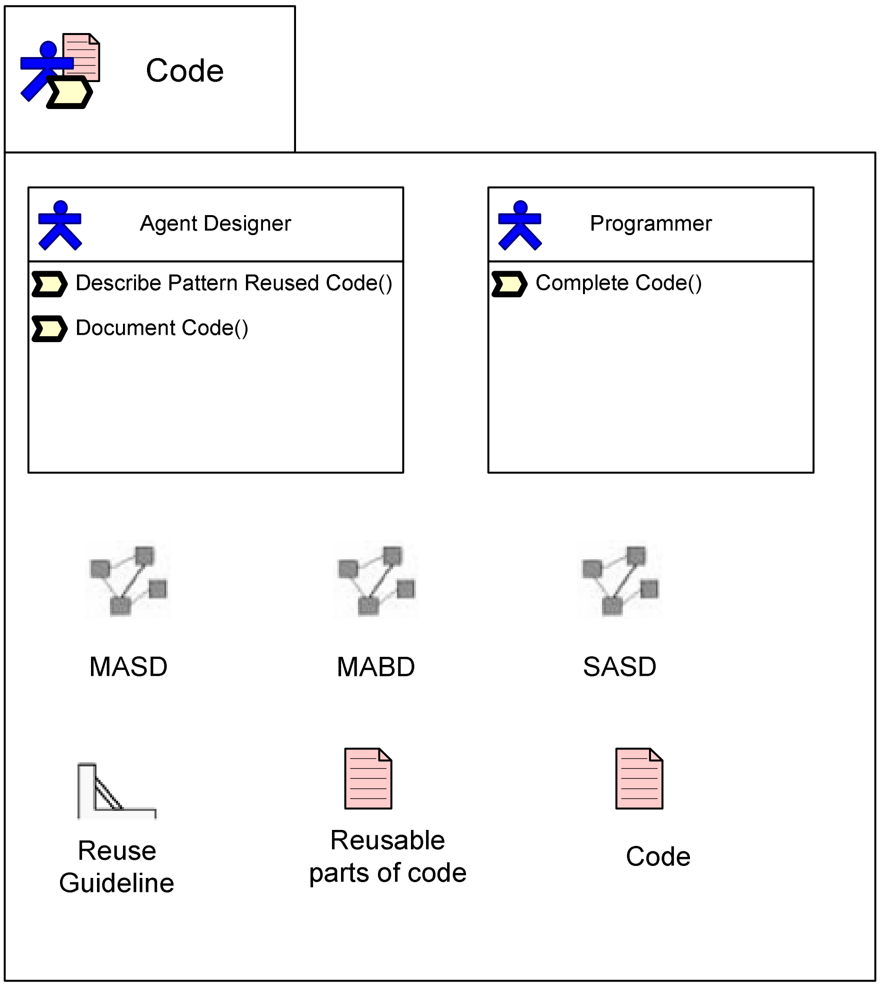
3.3.3.1.2 Agent Behaviour Description

Also in this phase we subdivide the diagrams into two views, the single and the multi one.
Activity diagrams or statecharts are used to describe the behaviour of individual agent, showing the flow of events between and within both the main agent classes and their inner classes, in terms of methods invocation and messages exchanged.

Besides in Single-Agent Behaviour Description we can use the representation we consider most appropriate , flow chart, state diagrams or semi-formal text descriptions).


TLPlanner.MyPositionInitiator
engController.MyPositionResponder
TLPlanner.Planner
TLPlanner.MyPositionInitiator. action
engController.MyPositionResponder. onEnd
message(Request, request, robotPosition)
engController.MyPositionResponder. action
TLPlanner.MyPositionInitiator. handleAgree
message(Request, agree,)
TLPlanner.MyPositionInitiator. handleNotUnderstood
message(Request, notUnderstood,)
engController.MyPositionResponder. prepareResponse
TLPlanner.MyPositionInitiator. handleRefuse
message(Request, refuse,)
TLPlanner.MyPositionInitiator. handleFailure
message(Request, failure,)
engController.MyPositionResponder. prepareResultNotification
TLPlanner.MyPositionInitiator. handleInform
message(Request, inform, robotPosition)
defaultTransition
TLPlanner.Planner. action
TLPlanner.Planner. refreshEnvironmentMatrix
TLPlanner.Planner. buildPolarHistogram
TLPlanner.Planner. buildBinaryHistogram
TLPlanner.Planner . buildOpeningList
TLPlanner.Planner. calcRotParameters
TLPlanner.Planner. distance

3.4 The Code phase



The Code discipline involves two different process roles, five work products (three UML models and two text documents) and one guidance.

The process to be performed in this phase is described in the following Figure 10. It is composed of two sub-phase level work definitions (Code Reuse and Code Completion) and several related work products (UML models and text documents)

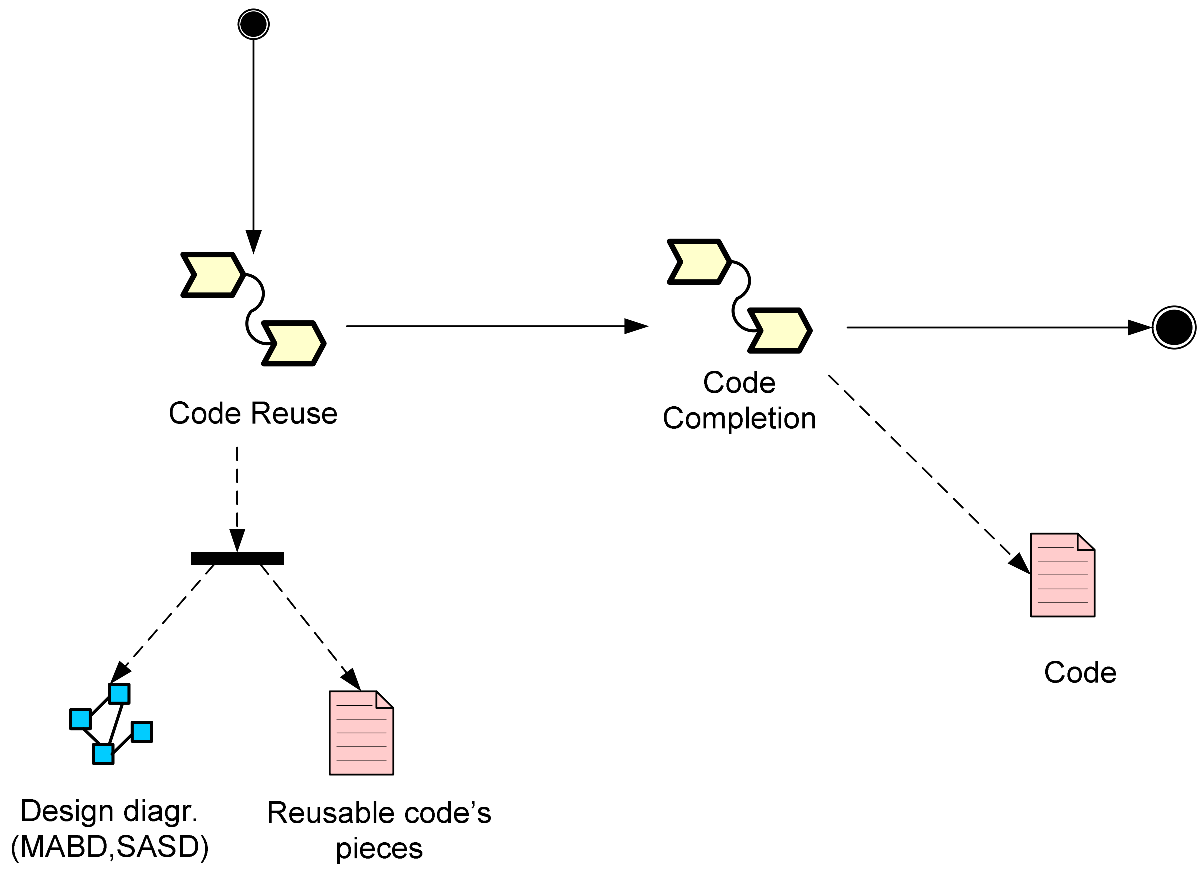


Figure 10. The Code phase described in terms of work definitions (sub-phases) and work products

The two sub-phases are composed of several activities as described in Figure 4. This figure also describes the actors (process roles) involved in this portion of the process. The process flow will be described in following sub-sections.

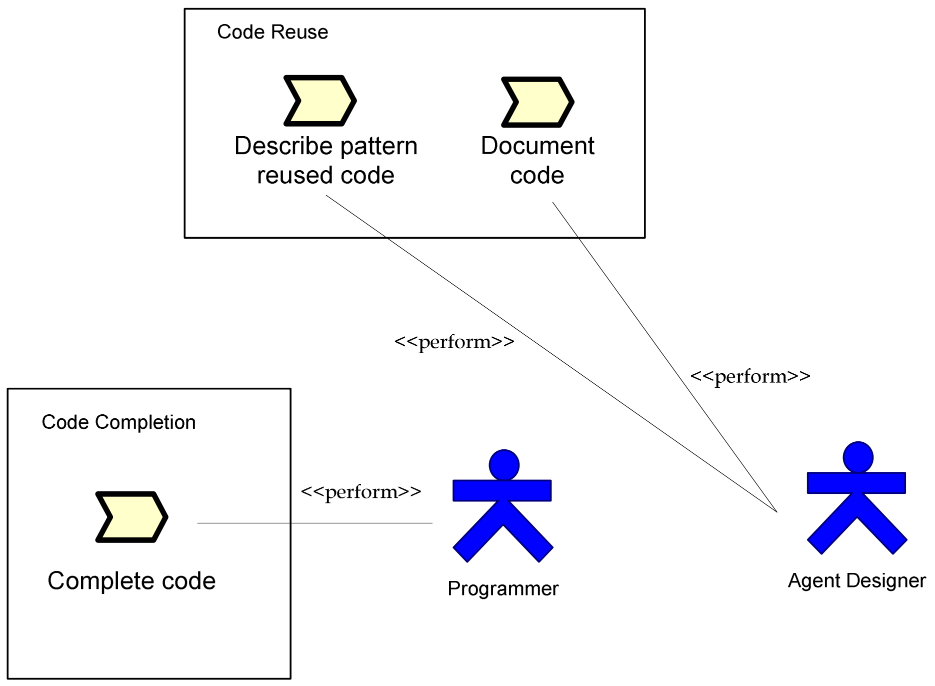


Figure 11. The Code activities clustered in sub-phase level work definitions

Here is a summary of this phase activities:

Phase	Activity	Activity Description	Roles involved
Code Reuse	Describe pattern reused code		Agent Designer (perform)
Code Reuse	Document code		Agent Designer (perform)
Code Completion	Complete code		Programmer (perform)

3.4.1 Process roles involved

Two roles are involved in this discipline: the Agent designer and the Programmer. They are described in the following subsection.

3.4.1.1 Agent Designer

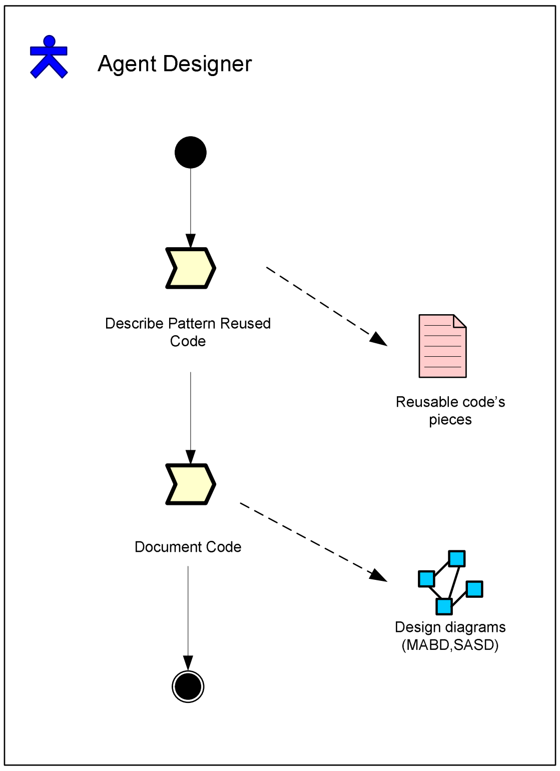
1. Describe pattern reused code. It consists in trying to reuse predefined patterns of agents and tasks.
2. Document code. The documentation of the previous step with MABD and SASD diagrams.

3.4.1.2 *Programmer*

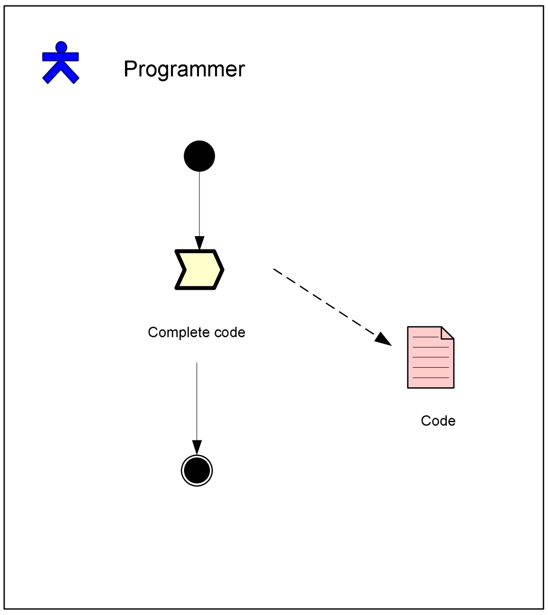
1. Complete code. The completion of methods' body yielded to this point by taking into account the design diagrams.

3.4.2 Details on Code sub-phases

3.4.2.1 *Code Reuse*



3.4.2.2 *Code Completion*



3.4.3 UML Models

The Code discipline includes two UML Models whose aim and notation are described in the following sub-sections.

3.4.3.1 Code Reuse

In this phase, we try to reuse existing patterns of agents and tasks looking at diagrams detailing the library of patterns.
Patterns are not only pieces of code but also pieces of design (of agents and tasks) that can be reused to implement new systems.
In this phase we have therefore produced a series of pieces of reusable code that are documented with their MABD and SASD diagrams. In the former we describe the behaviour of the pattern through the sequence of events and implemented methods whilst in the latter we have a structural description of the pattern in the form of a class or a group of classes (for example an agent main class together with its tasks).
We have found that in our applications and with our specific implementation environment (FIPA-OS and Jade), the most useful patterns are those that could be classified as *interaction* patterns. This is due to the specific structure of FIPA-compliant platforms that delegate a specific task for each specific communication. Each time an agent needs to use a protocol, the related pattern task can be easily reused and only the part of the code devoted to the information treatment necessitates modification.

The repository of patterns is described as reported below:

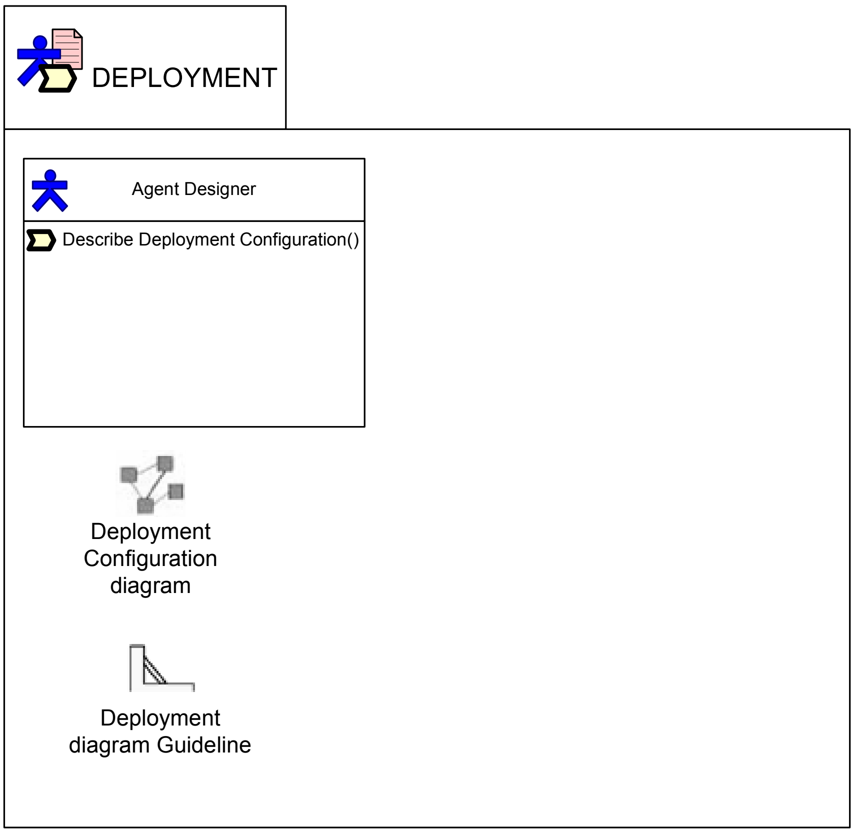
Name	The name of the pattern
Classification	The classification of the pattern according to the following criteria and related categories: • Application context: Action, Behaviour, Component and Service pattern

	• 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

3.4.3.2 Code Completion

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.

3.5 The Deployment phase


DEPLOYMENT
Agent Designer
Describe Deployment Configuration()
Deployment Configuration diagram
Deployment diagram Guideline

The Deployment discipline involves one process role, one workproduct (an UML model) and one guidance .

The process to be performed in this phase is described in the following Figure 12. It is composed of one sub-phase level work definition (Deployment Configuration) and one related work product (an UML model).

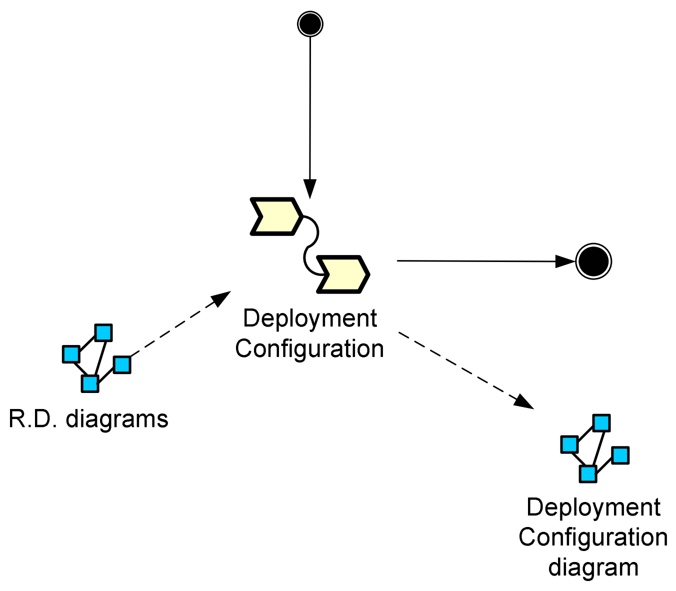


Figure 12. The Deployment phase described in terms of work definitions (sub-phases) and work products

The unique sub-phases is composed of one activity as described in Figure 13. This figure also describes the actor (process role) involved in this portion of the process. The process flow will be described in following sub-sections.

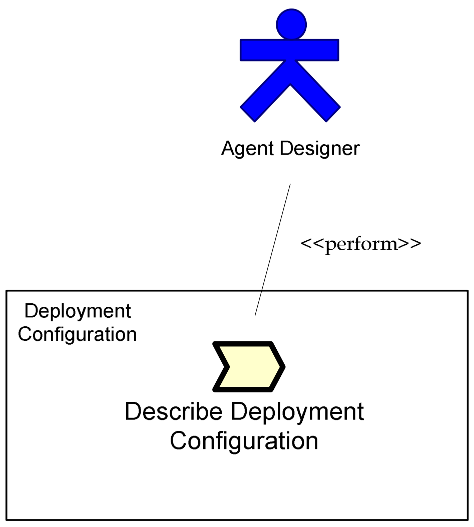


Figure 13. The Deployment activities clustered in sub-phase level work definitions

Here is a summary of this phase activities:

Phase	Activity	Activity Description	Roles involved
Deployment Configuration	Describe Deployment Configuration	It consists in the allocation of agents to the available process units, this includes describing hardware requirements for agents' deployment in the different computational units, defining conditions for agents' movements and requirements related to agents' communication paths.	Agent Designer (perform)

3.5.1 Process roles involved

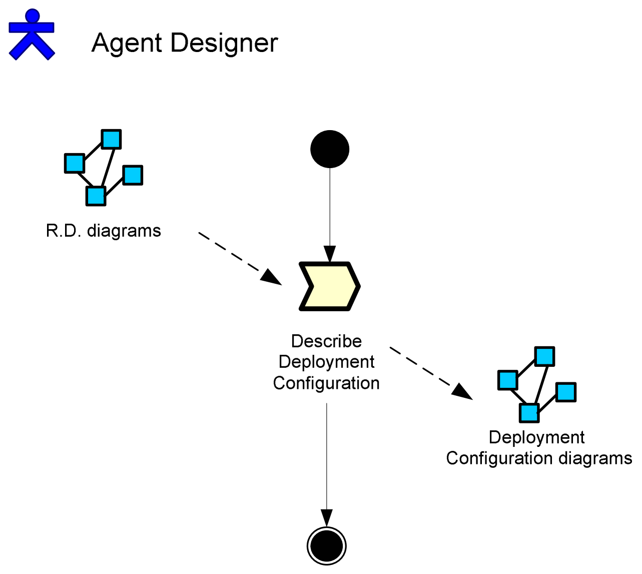
One role is involved in this discipline: the Agent designer that is described in the following subsection.

3.5.1.1 *Agent Designer*

It consists in the allocation of agents to the available process units, this includes describing hardware requirements for agents' deployment in the different computational units, defining conditions for agents' movements and requirements related to agents' communication paths.

3.5.2 Details on Deployment sub-phases

3.5.2.1 *Describe Deployment Configuration*


Agent Designer
R.D. diagrams
Describe Deployment Configuration
Deployment Configuration diagrams

3.5.3 Work Products

The Deployment Configuration model includes …

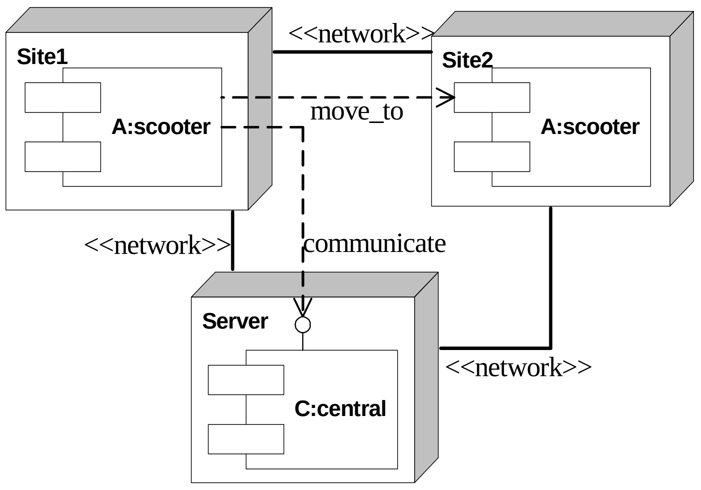
3.5.4 UML Models

The Deployment discipline includes one UML Models whose aim and notation are described in the following sub-sections.

3.5.4.1 *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

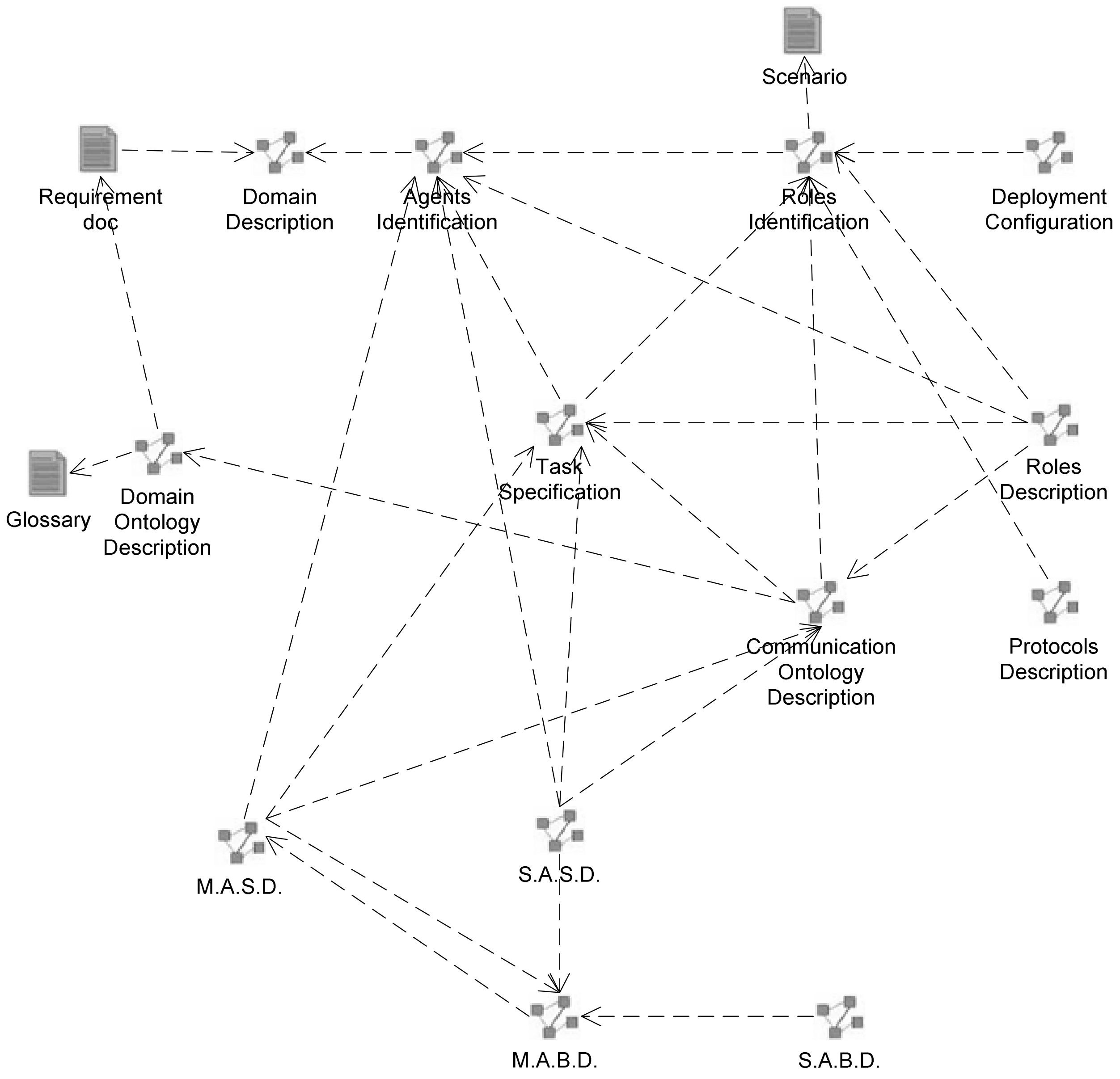
useful for representing processing units (by boxes), agents (by components) and the like. What is not supported by UML is the representation of the agent’s mobility, which we have done by means of a syntax extension consisting of a dashed line with a “move to” stereotype..



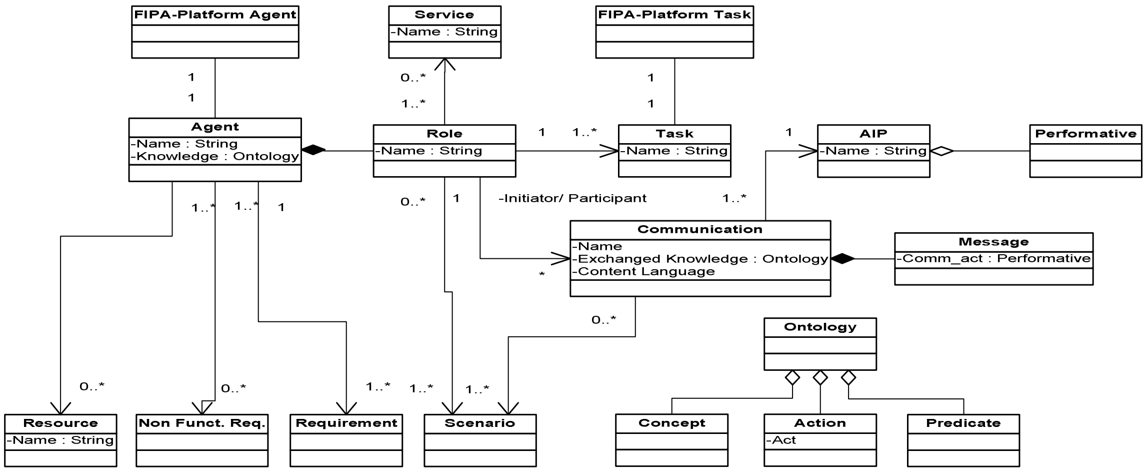
In this diagram is also possible to specify the hardware devices used by the agents (sensors and effectors) and the modes of communication among agents in different elaborating units.

4 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.



5 MAS Meta-Model


FIPA-Platform Agent
Service
-Name : String
FIPA-Platform Task
Agent
-Name : String
-Knowledge : Ontology
Role
-Name : String
Task
-Name : String
AIP
-Name : String
Performative
Communication
-Name
-Exchanged Knowledge : Ontology
-Content Language
Message
-Comm_act : Performative
-Initiator/ Participant
Ontology
Resource
-Name : String
Non Funct. Req.
Requirement
Scenario
Concept
Action
-Act
Predicate
1
1..*
0..*
*