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

SPEM Description of ADELFE Process

M. Cossentino, V. Seidita

Rapporto Tecnico N:
RT-ICAR-PA-05-07 **Luglio 2005**

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*
– Sede di Napoli, Via P. Castellino 111, 80131 Napoli, URL: *www.na.icar.cnr.it*
– Sede 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 ADELFE Process

M. Cossentino[1], Seidita[2]

Rapporto Tecnico N.: **RT-ICAR-PA-05-07**

Data: **Luglio 2005**

---

[1] Istituto di Calcolo e Reti ad Alte Prestazioni, ICAR-CNR, Sede di Palermo Viale delle Scienze edificio 11 90128 Palermo

[2] Università degli Studi di Palermo Dipartimento di Ingegneria Informatica Viale delle Scienze 90128 Palermo

Indice

1 ADELFE: an overview

The adaptive multi-agent software systems can be developed through different methodologies, the purpose of this document is to explain the ADELFE (**A**telier de **DÉ**veloppement de **L**ogiciels à **F**onctionnalité **E**mergente) methodology.

A system designed through ADELFE has the following characteristics[8][9][13]:

- it is founded on the AMAS (Adaptive Multi-Agent System) theory [7] and on object-oriented methodologies;
- it follows the Rational Unified Process (RUP);
- it uses the UML and the AUML notations;
- the designed agents are only cooperative;
- every agent has a specific function that contributes to the realization of the final objective.

The ADELFE methodology is shown in Figure 1.

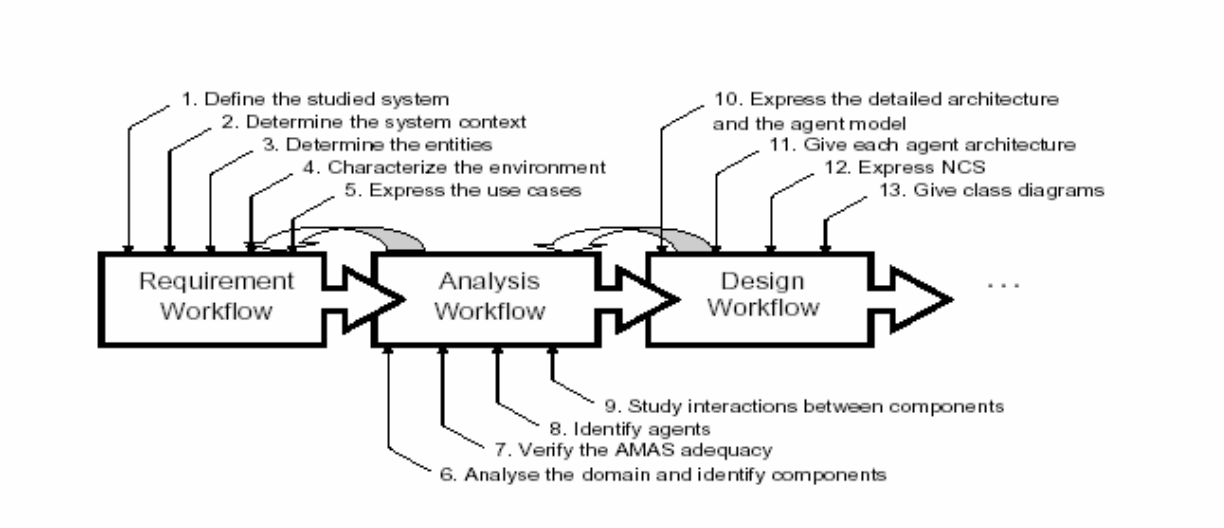

Figure 1 The ADELFE methodology

2 The ADELFE Process

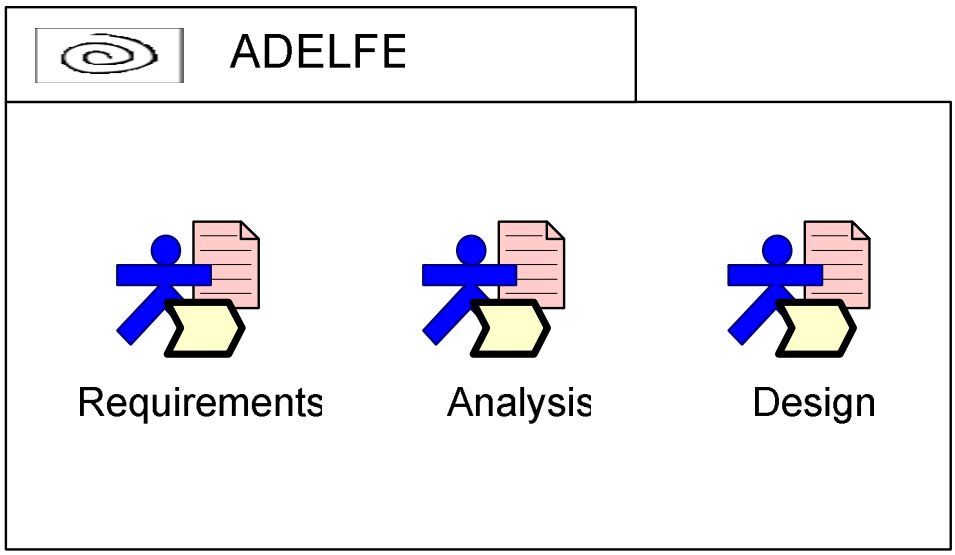

Figure 2 The disciplines of the ADELFE Process

The design process is composed of three disciplines (see Figure2): requirements, analysis and design [8].

In Figure 3 the ADELFE process in terms of the composing phases and their workproducts is shown :

- Requirements phase produces the Environment Model;
- Analysis phase produces the Identification of the Agent Model;
- Design phase produces the Agent and the NCS Models.

The **requirements phase** is fundamental in software engineering. During this phase it is necessary to give an Environment Model (see Figure 3) that, in the AMAS theory, will serve as base for the process of adaptation. This process begins with the interactions between the system and the environment. The environment model includes the following activities: actors determination, context definition and environment characterization.

In the **analysis phase**, the previously defined entities are analyzed in order to specify which will be agents. An agent is an entity having the capability to evolve during an unexpected situation, showing new behaviors and skills.

Two activities are added to the classical RUP: the agent identification (see Figure 3) and the adequacy at the AMAS theory.

The **design phase** aims to define the agents architecture describing their behaviors; the result of this activity adds two models to the RUP: the Agent Model and the Non Cooperative Situations (NCS) Model (see Figure 3).

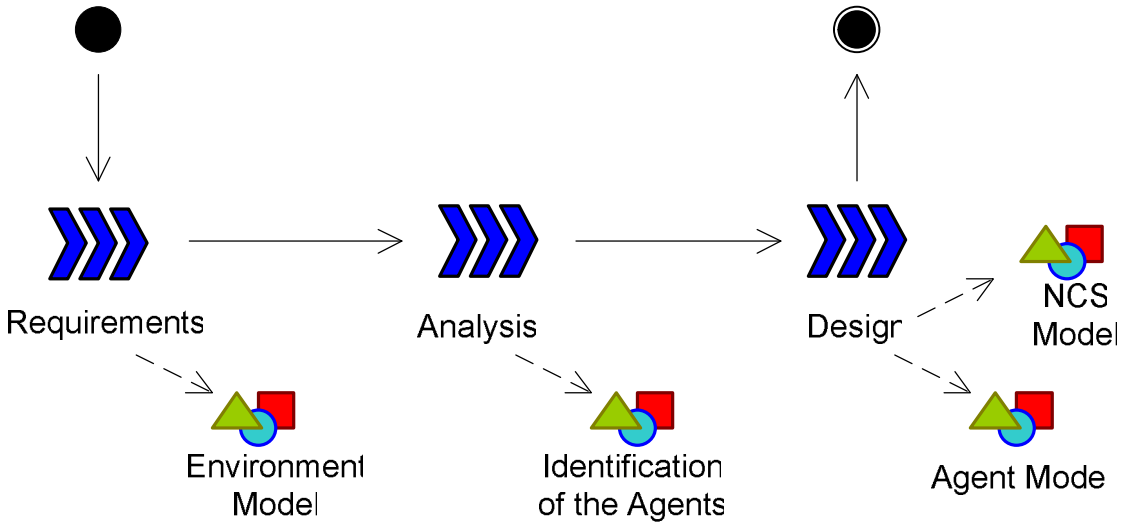

Figure 3 The complete Adelfe process

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

3 MAS Model in ADELFE

The MAS meta-model of ADELFE [9][10][11](Figure 4) represents the cooperative agent features. Its lifecycle is the classic perceive-decide-act.

The Non Cooperative Situations are cooperation failures, which can be solved through cooperation rules. There are different kinds of NCS, such as Incomprehension (an agent does not understand a perceived signal), Ambiguity (it has several contradictory interpretations for a perceived signal), Incompetence (it cannot satisfy the request of another one), Unproductiveness (it receives an already known piece of information or some information that leads to no reasoning for it), Concurrency (several agents want to access an exclusive resource), Conflict (several agents want to realise the same activity) or Uselessness (an agent may make an action that is not beneficial, according to its beliefs, to other agents).

When an agent detect a NCS, it does all it is able to do to solve it and to stay cooperative for others.

Each agent has a world representation concerning other agents and the environment that surrounds it, by this representation the agent can specify its behaviour.

The multi-agent system is used when the representation may evolve.

An agent possesses:

- the ability to communicate with other agents or with its environment in two ways: direct (through messages exchange), indirect (through the environment), it can interacts with the environment receiving information through the perception and operating through the action;
- aptitudes, that are representations of agent reasoning on its knowledge;
- skills, that are specific information that help the agent to perform its function;
- characteristics, that are intrinsic or physical properties.

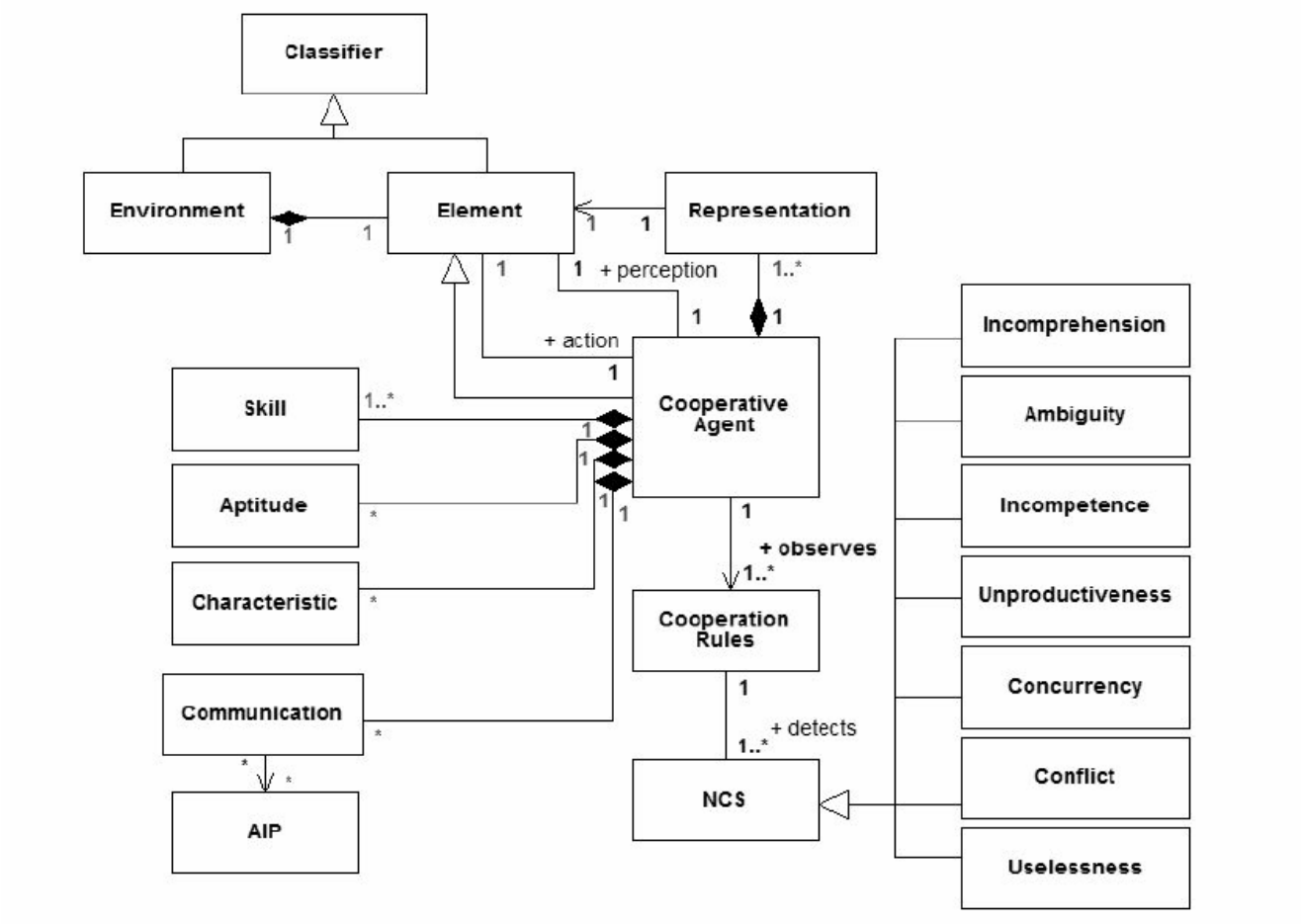

Figure 4 The Multi-Agent System Meta-model Adopted in ADELFE

4 Phases of the ADELFE Process

4.1 Requirements phase

The aims of this phase are [8][13]:

1. to define the system to be;
2. to transform this view in a use-case model;
3. to organize and to manage the requirements (functional or not) and their priorities.

The designer has to define the function of the studied system and to model its environment. This phase involves five process roles and seven work products (see Figure 5).

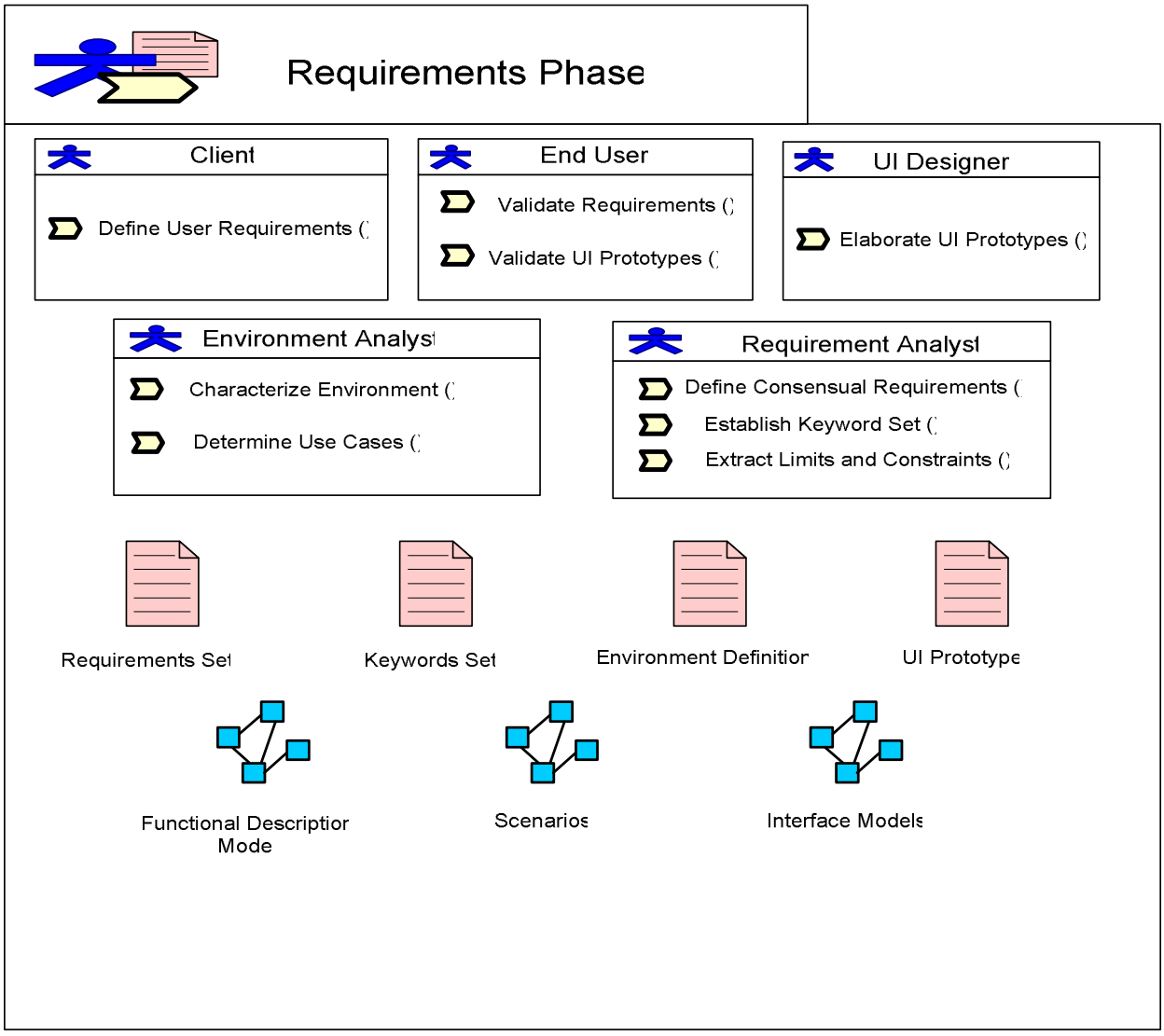

Figure 5. The Requirements phase represented as a SPEM discipline

The process to be performed in this phase is described in the following Figure 6. It is composed of five sub-phases level work definitions (Requirements Description, Keywords Identification, Environment Description, Use Cases Description and UI Prototypes Identification) and several related work products (mainly UML models and text documents).

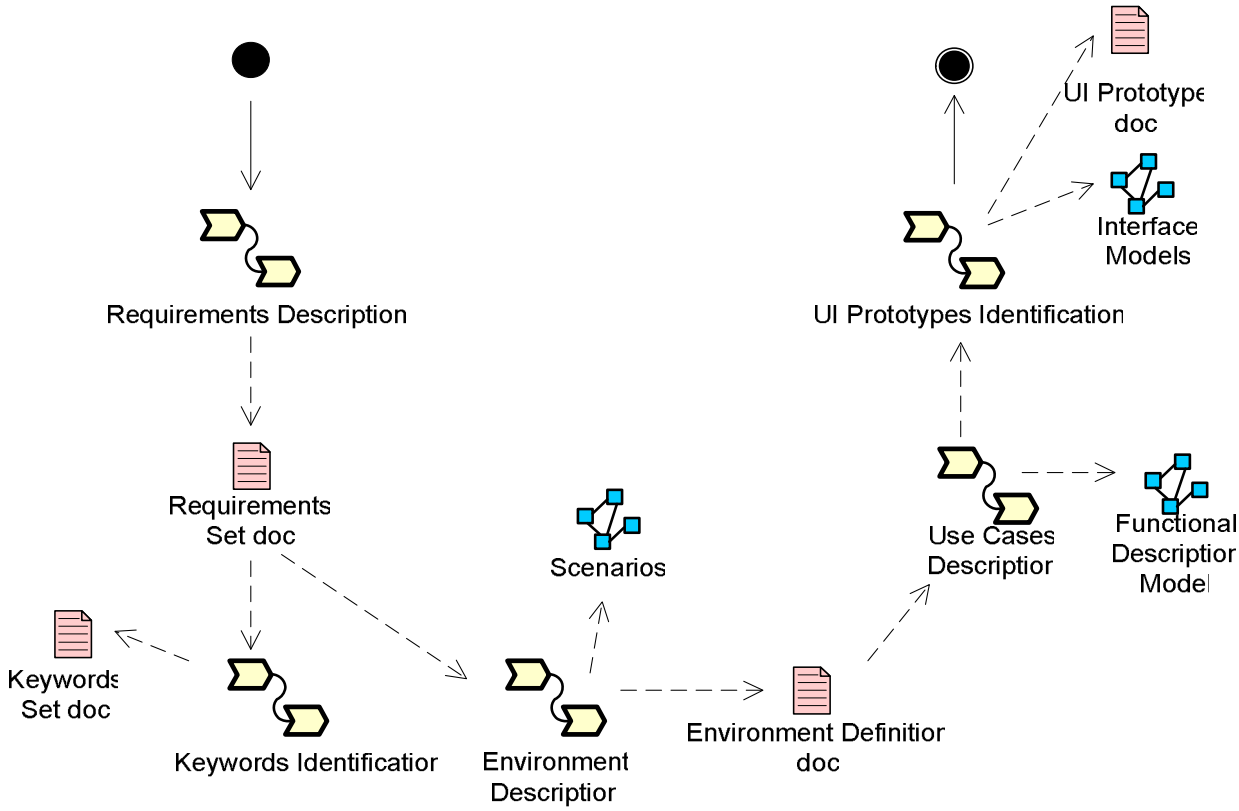

Figure 6. The Requirement phase described in terms of work definitions and work products

The five sub-phases are composed of several activities as described in Figure 7. 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.

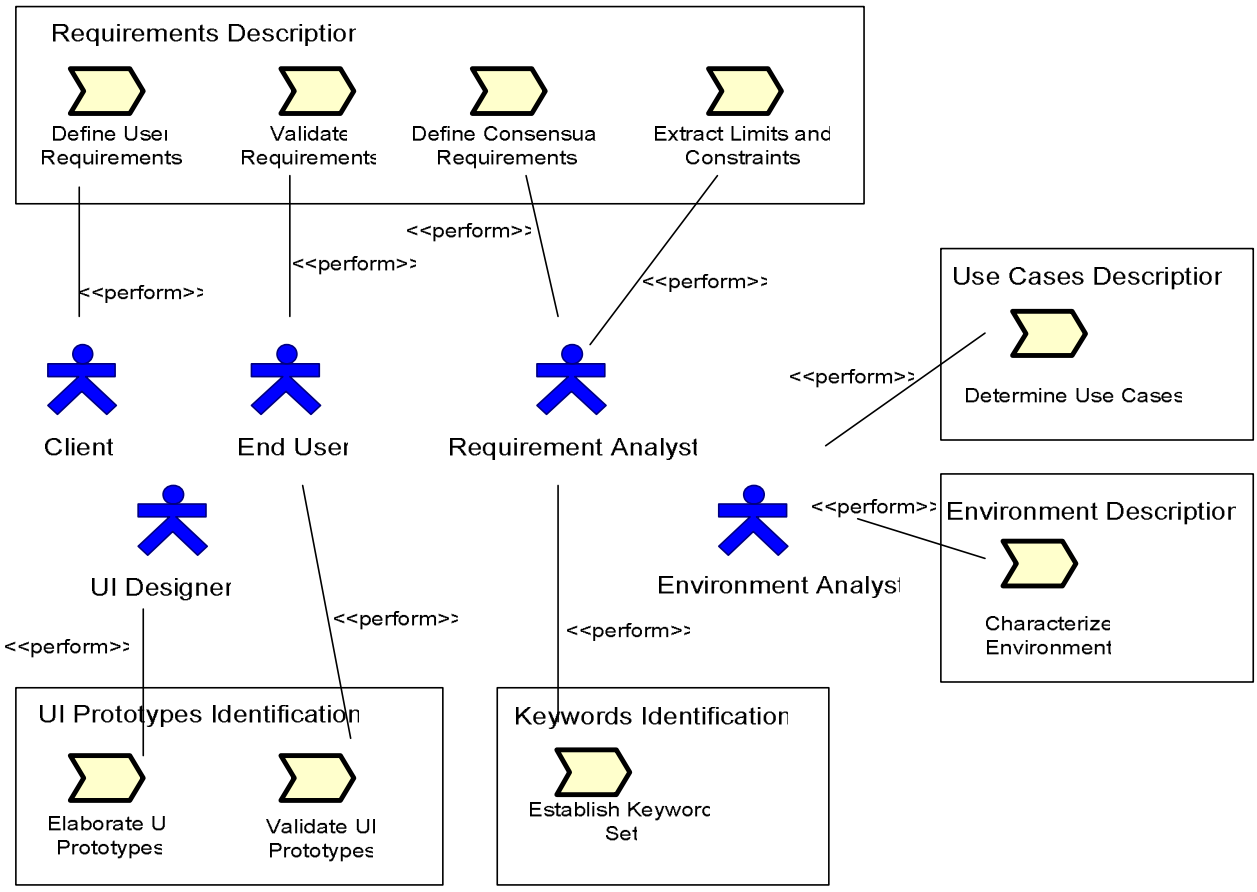

Figure 7. The Requirement phase activities clustered in sub-phase level work definitions

Here is a summary of this phase activities:

Work Definition	Activity	Activity Description	Roles involved
Requirements Description	Define User Requirements	User Requirements are used to define system requirements	Client (perform)
Requirements Description	Validate Requirements	Requirements are checked and approved	End User (perform)
Requirements Description	Define Consensual Requirements	The consensual requirements is added	Requirement Analyst (perform)
Requirements Description	Extract Limits and Constraints	Limits and constraints that the system must satisfy are defined	Requirement Analyst (perform)

Keywords Identification	Establish Keyword Set	The main concepts used to describe the system are identified	Requirement Analyst (perform)
Environment Description	Characterize Environment	System environment is described	Environment Analyst (perform)
Use Cases Description	Determine Use Cases	System functionalities are identified	Environment Analyst (perform)
UI Prototypes Identification	Elaborate UI Prototypes	The interfaces through which the user will interact with the system are defined	UI Designer (perform)
UI Prototypes Identification	Validate UI Prototypes	The interfaces are tested	End User (perform)

4.1.1 Process roles involved

Five roles are involved in the Requirements phase [13]: the Client, the End User, the Requirement Analyst, the Environment Analyst and the UI Designer. They are described in the following sub-sections.

4.1.1.1 Client

He defines user requirements during the Requirements Description work definition. User Requirements are used to define system requirements.

4.1.1.2 End User

He is responsible of :

1. Validate Requirements during the Requirements Description work definition. Requirements are checked and approved.
2. Validate UI Prototypes during the UI Prototypes work definition. The interfaces are tested.

4.1.1.3 Requirement Analyst

He is responsible of :

1. Define Consensual Requirements during the Requirements Description work definition. The consensual requirements are added.
2. Extract limits and constraints during the Requirements Description work definition. Limits and constraints that the system must satisfy are defined.
3. Establish Keyword Set during the Keywords Identification work definition. The main concepts used to describe the system are identified.

4.1.1.4 Environment Analyst

He is responsible of:

1. Determine Use Case during the Use Case Description work definition. System functionalities are identified.
2. Characterize Environment during the Environment Description work definition. System environment is described.

4.1.1.5 UI Designer

He elaborates UI Prototypes during the UI Prototypes Identification work definition. The interfaces through which the user will interact with the system are defined.

4.1.2 Fragments extracted from Requirements Phase

4.1.2.1 Requirements Description fragment

4.1.2.1.1 Portion of process

This fragment concerns the description of the system and the environment in which the system will operate, therefore it consists in finding the appropriate system for the end-users.

In this fragment, you must define limits and constraints of the system you want to build (your application) [12][13].

The process that is to be performed in order to obtain the result is represented in Figure 8 as a SPEM diagram:

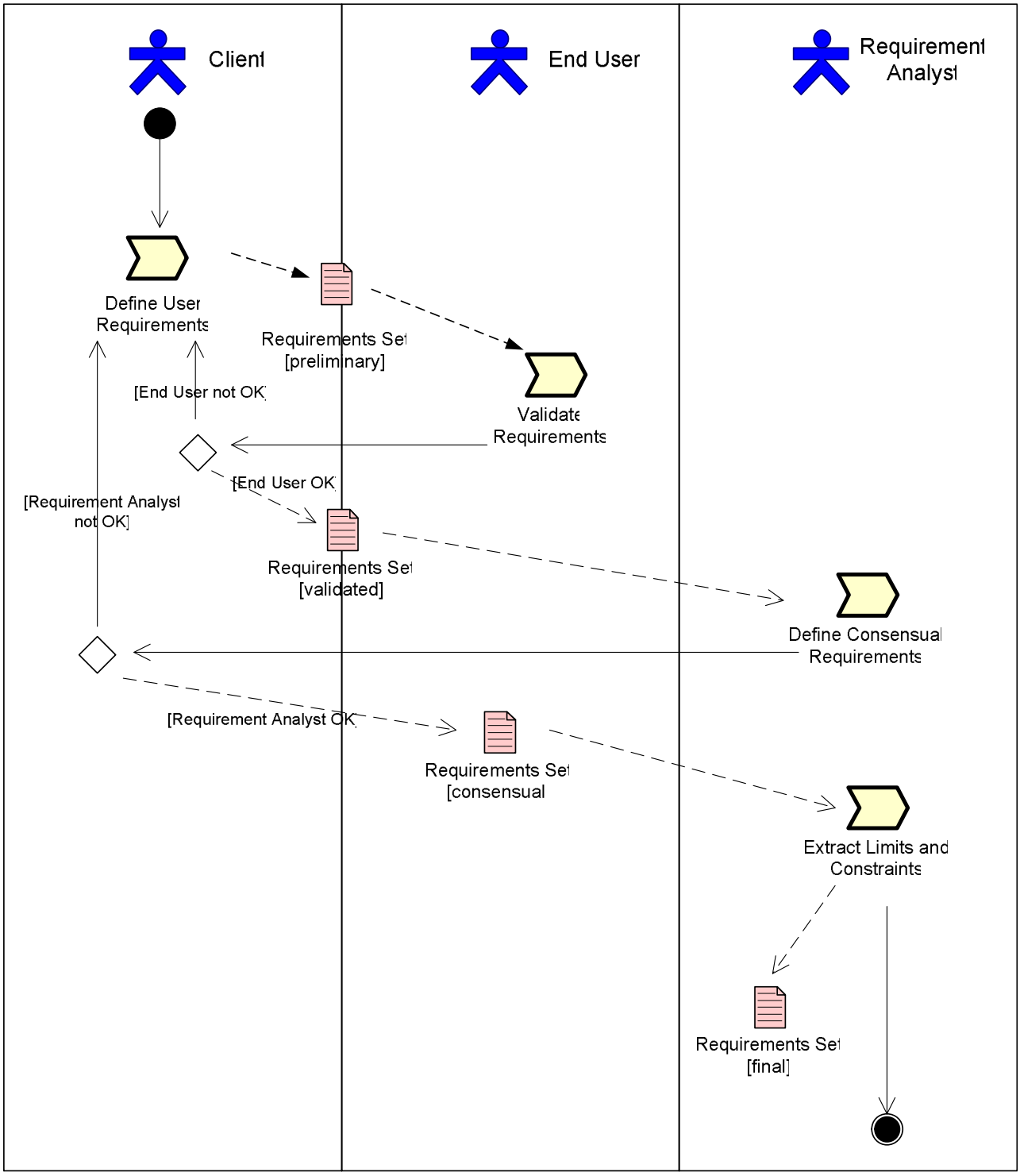

Figure 8. Requirements Description

4.1.2.1.2 Deliverables

This information must be written in the Requirements Set (final) document.

4.1.2.1.3 Guideline(s)

End-users, clients, analysts and designers have to list the potential requirements. The context in which the system will be deployed must be understood. The functional and non-functional requirements must be established.

The Requirements Set document must be checked, approved and updated with consensual requirements.

Limits and constraints can be found in the expression of non functional requirements and in the definition of the context in which the system will be deployed.

4.1.2.1.4 Input/Output

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

INPUT	TO BE DESIGNED	OUTPUT
	User Requirements	Requirements Description (final)

4.1.2.2 Keywords Identification fragment

4.1.2.2.1 Portion of process

This fragment aims to identify the main concepts used in the description of the system [12][13].

The process that is to be performed in order to obtain the result is represented in Figure 9 as a SPEM diagram:

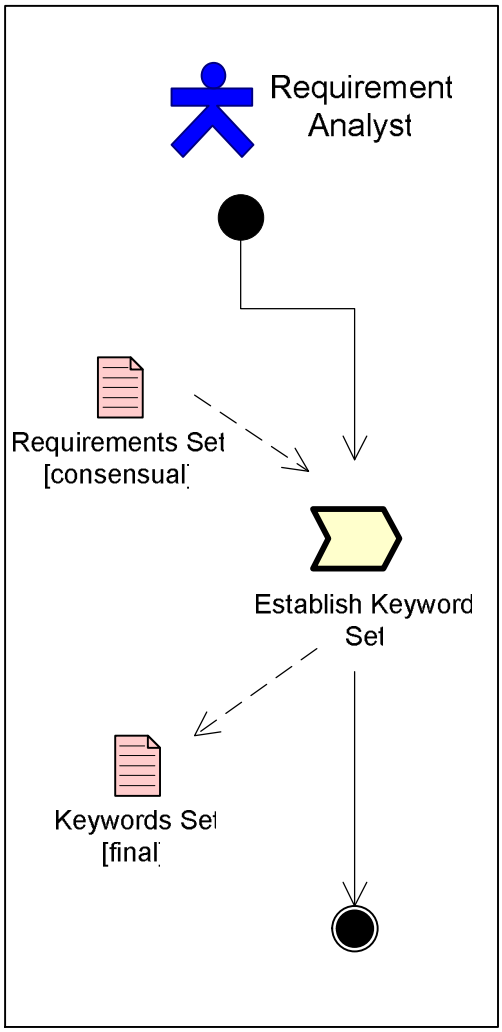

Figure 9. Keywords Identification

4.1.2.2.2 Deliverables

For each keyword, a definition will be given in the Keywords Set (final) document

4.1.2.2.3 Preconditions

context Keywords Identification::Establish Keyword Set **pre**:

existence of Requirements Set (consensual) document

4.1.2.2.4 Guideline(s)

You have to list the main concepts used to describe the application and its domain (the system and its environment).

4.1.2.2.5 Dependency relationships with other fragments

This fragment depend on the Requirements Description fragment since the keywords are extracted from the Requirement Set document.

4.1.2.2.6 Input/Output

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

INPUT	TO BE DESIGNED	OUTPUT
Requirements Description (consensual)	Keywords	Keywords Set (final)

4.1.2.3 Environment Description fragment

4.1.2.3.1 Portion of process

The main objective of this fragment is to define the system environment [12][13].

The process that is to be performed in order to obtain the result is represented in Figure 10 as a SPEM diagram:

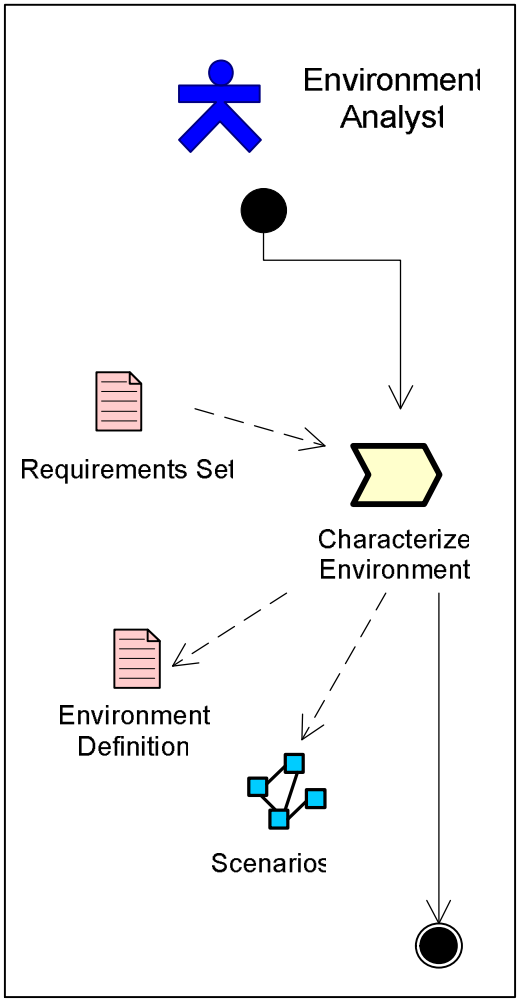

Figure 10. Environment Description

4.1.2.3.2 Deliverables

This information must be written in the Environment Definition document and the potential situations are represented (Scenarios) through UML diagrams.

4.1.2.3.3 Preconditions

context Environment Description::Characterize Environment **pre**:

existence of Requirements Set (final) document

4.1.2.3.4 Relationships with the MAS Meta-model

This fragment refers to the MAS meta-model adopted in ADELFE and contributes to define and describe the concept of environment in relation with it (Figure 11).

The following figure describes the relationship of the fragment with respect to the MAS model:

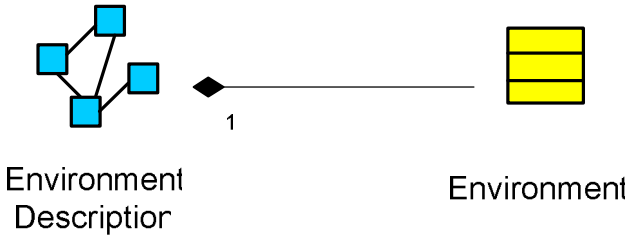

Figure 11. MAS Metamodel concept

4.1.2.3.5 Guideline(s)

This fragment is characterized by three phases: to determine entities, to define context and to characterize environment.

4.1.2.3.6 Dependency relationships with other fragments

This fragment depends of the Requirements Description fragment since the environment is described using the Requirements Set document.

4.1.2.3.7 Input/Output

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

INPUT	TO BE DESIGNED	OUTPUT
Requirements Description	Environment	Environment Definition, Scenarios

4.1.2.4 Use Cases Description fragment

4.1.2.4.1 Portion of process

The purpose of this fragment is to identify the system functionalities that it must provide [12][13].

The process that is to be performed in order to obtain the result is represented in Figure 12 as a SPEM diagram:

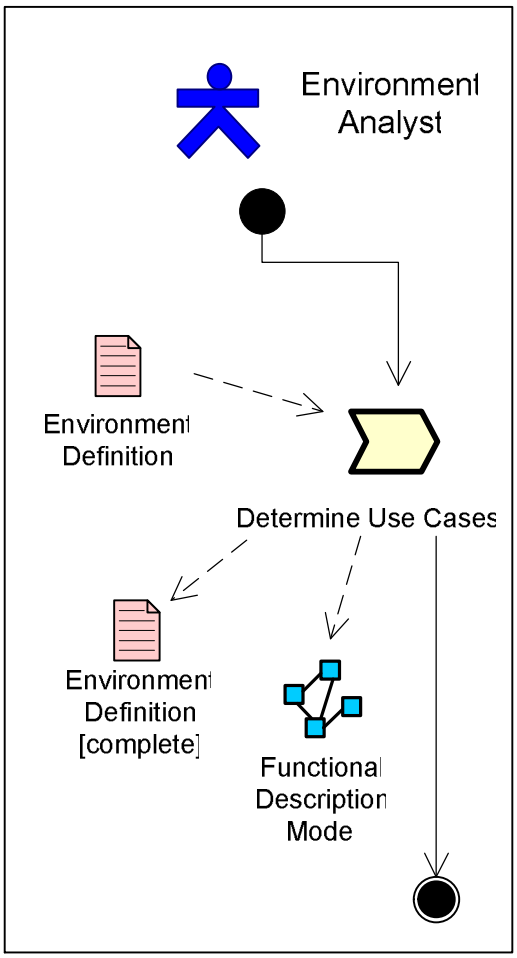

Figure 12. Use Cases Description

4.1.2.4.2 Deliverables

The functionalities of the system are represented through use case diagrams in the Functional Description Model. This information complete the Environment Definition document.

A use case is detailed using a textual description and specific sequence diagrams. To manage a possible exception can be inserted a special box in the use case.

4.1.2.4.3 Preconditions

context Use Cases Description::Determine Use Cases **pre**:

existence of Environment Definition document

4.1.2.4.4 Guideline(s)

This fragment is characterized by three phases: to draw up an inventory of the use cases, to identify cooperation failures, to elaborate sequence diagrams.

4.1.2.4.5 Aspects of Fragment

Use Cases are expressed using UML diagrams.

4.1.2.4.6 Dependency relationships with other fragments

This fragment depends of the Environment Description fragment since the use cases represent the system functionalities described using the Environment Definition document.

4.1.2.4.7 Input/Output

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

INPUT	TO BE DESIGNED	OUTPUT
Environment Definition	Use Cases	Environment Definition (complete), Functional Description Model

4.1.2.5 UI Prototypes Identification fragment

4.1.2.5.1 Portion of process

This fragment aims to define and to test the interfaces (GUIs) that allow at the user to interact with the system verifying adequacy of it [12][13].

The process that is to be performed in order to obtain the result is represented in Figure 13 as a SPEM diagram:

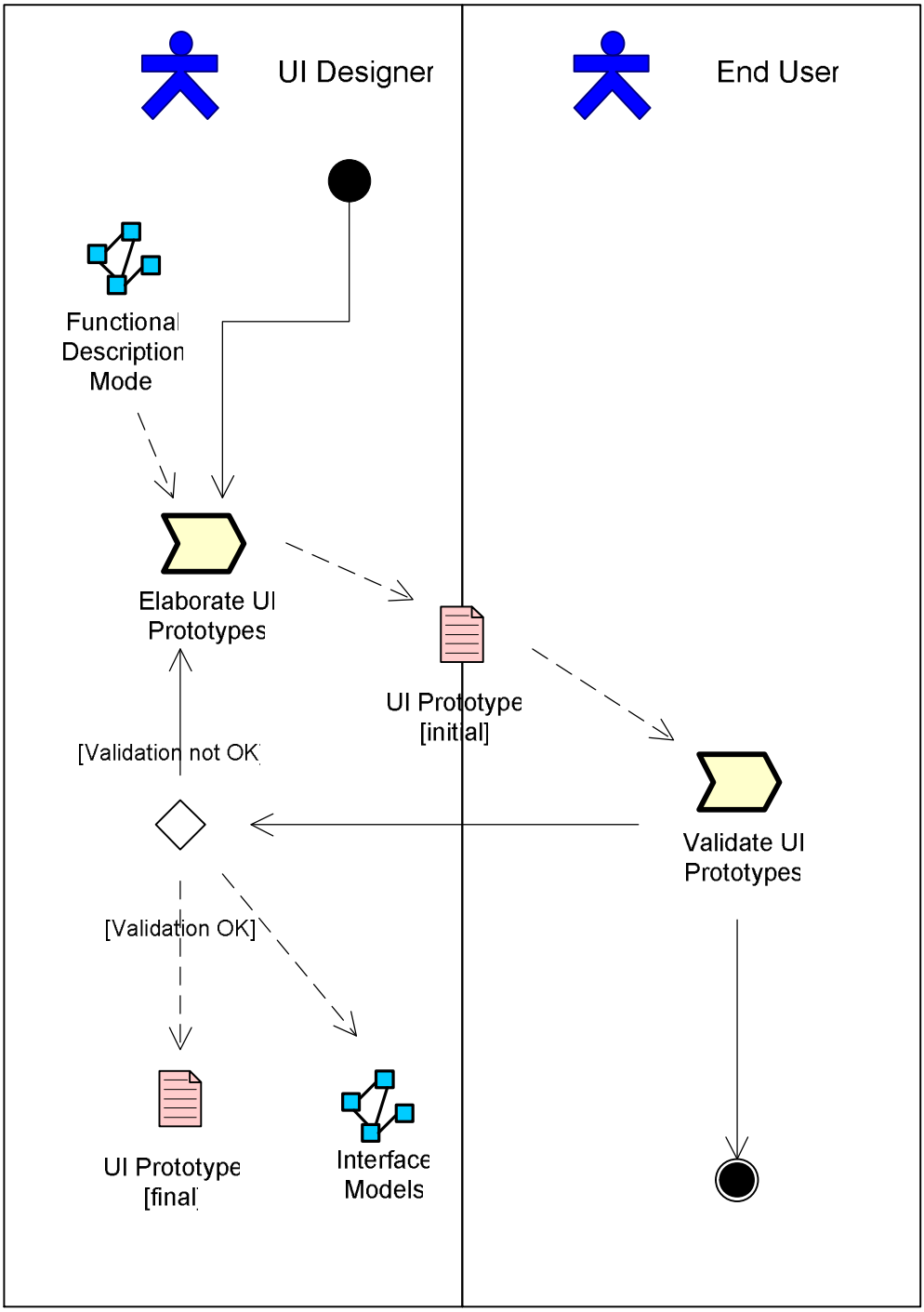

Figure 13. UI Prototypes Identification

4.1.2.5.2 Deliverables

The GUIs must be described in the UI Prototype (final) document and on the Interface Models represented through UML diagrams.

4.1.2.5.3 Preconditions

context UI Prototypes Identification::Elaborate UI Prototypes **pre**:

existence of Functional Description Model

context UI Prototypes Identification::Validate UI Prototypes **pre**:

existence of UI Prototype (initial) document

4.1.2.5.4 Aspects of Fragment

A possible means to describe the GUIs is to use the basic tool provided by OpenTool.

4.1.2.5.5 Dependency relationships with other fragments

This fragment depends of the Use Cases Description fragment since the UI must be defined knowing the system functionalities described in the Functional Description model.

4.1.2.5.6 Input/Output

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

INPUT	TO BE DESIGNED	OUTPUT
Functional Description	UI Prototypes	UI Prototypes (final), Interface Models

4.2 Analysis phase

The aims of this phase are [12][13]:

1. to identify the agents;
2. to verify the AMAS Adequacy

This phase involves two process roles and five work products (see Figure14).

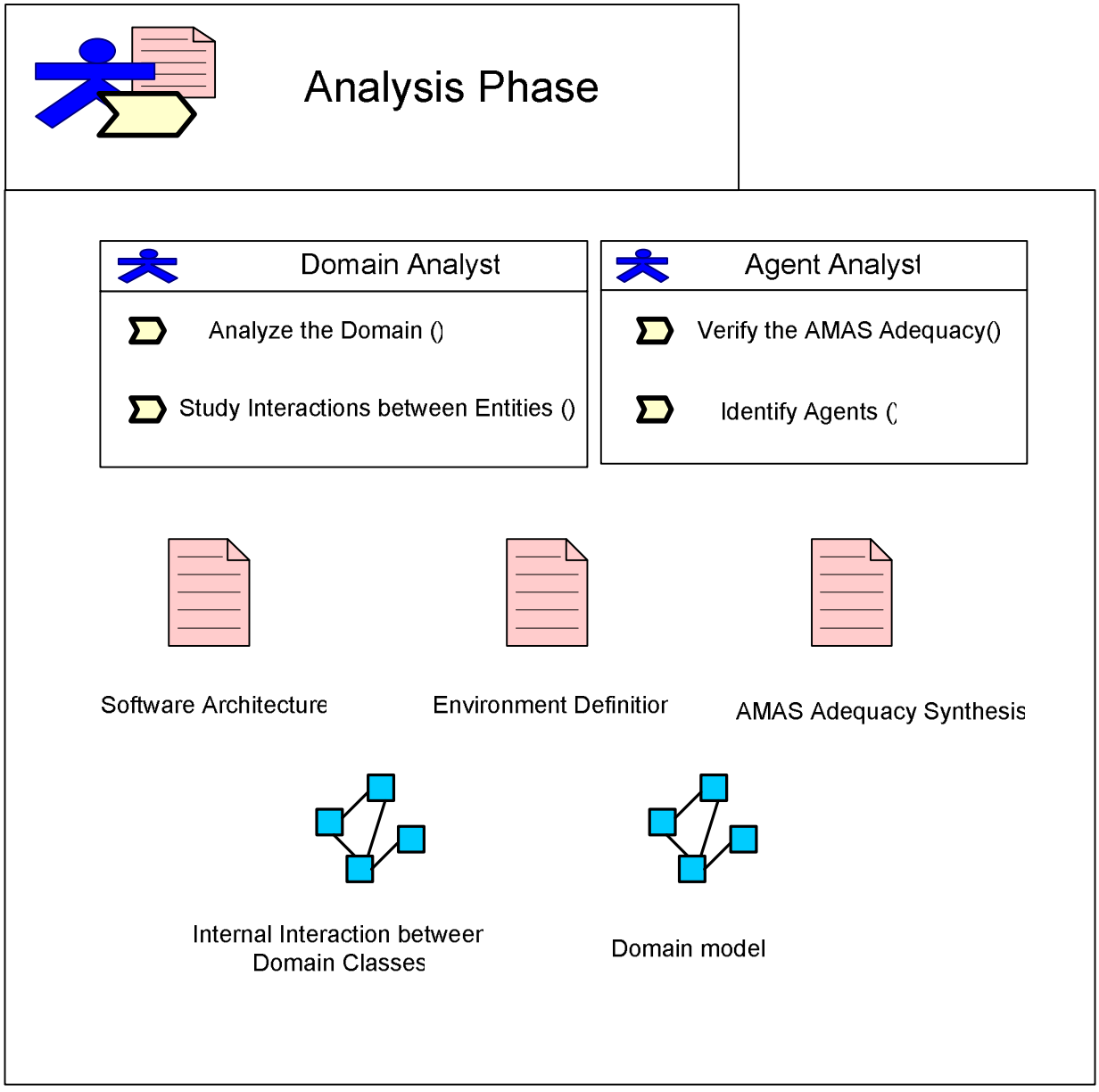

Figure 14 The Analysis phase represented as a SPEM discipline

The process to be performed in this phase is described in the following Figure 15. It is composed of four sub-phases level work definitions (Domain Description, Adequacy Verification, Agents Identification and Interaction among entities Identification) and several related work products (mainly UML models and text documents).

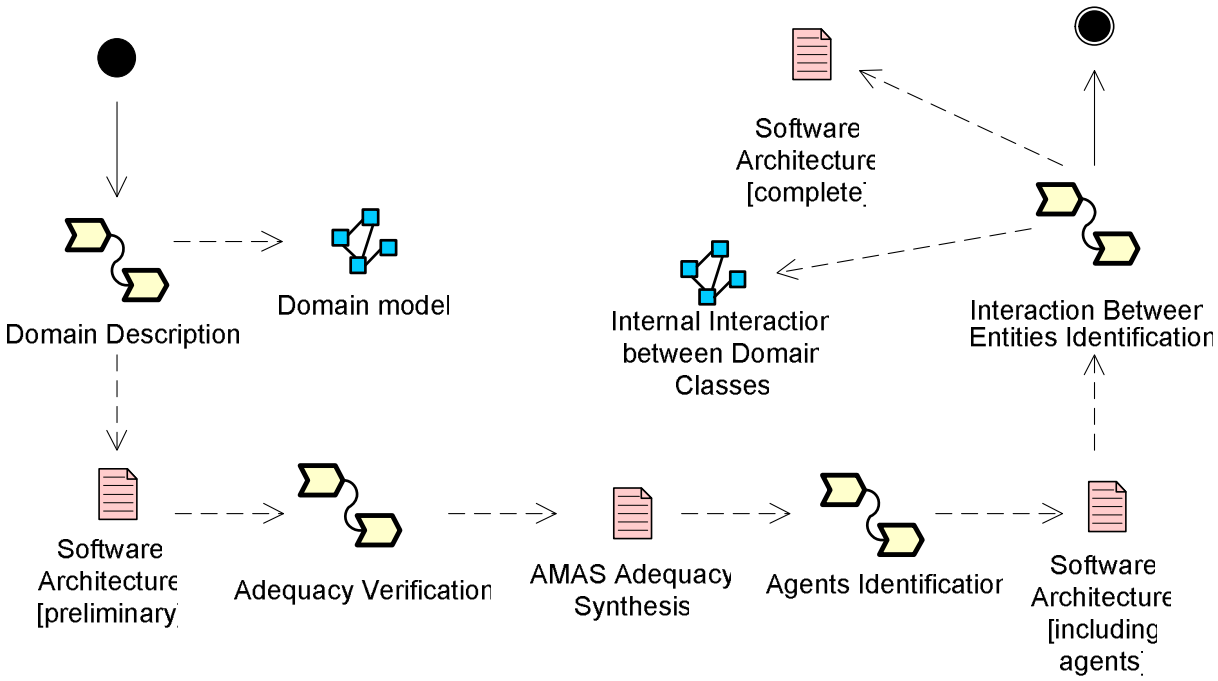

Figure 15.The Analysis phase described in terms of work definitions and work products

The four sub-phases are composed of several activities as described in Figure16. 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.

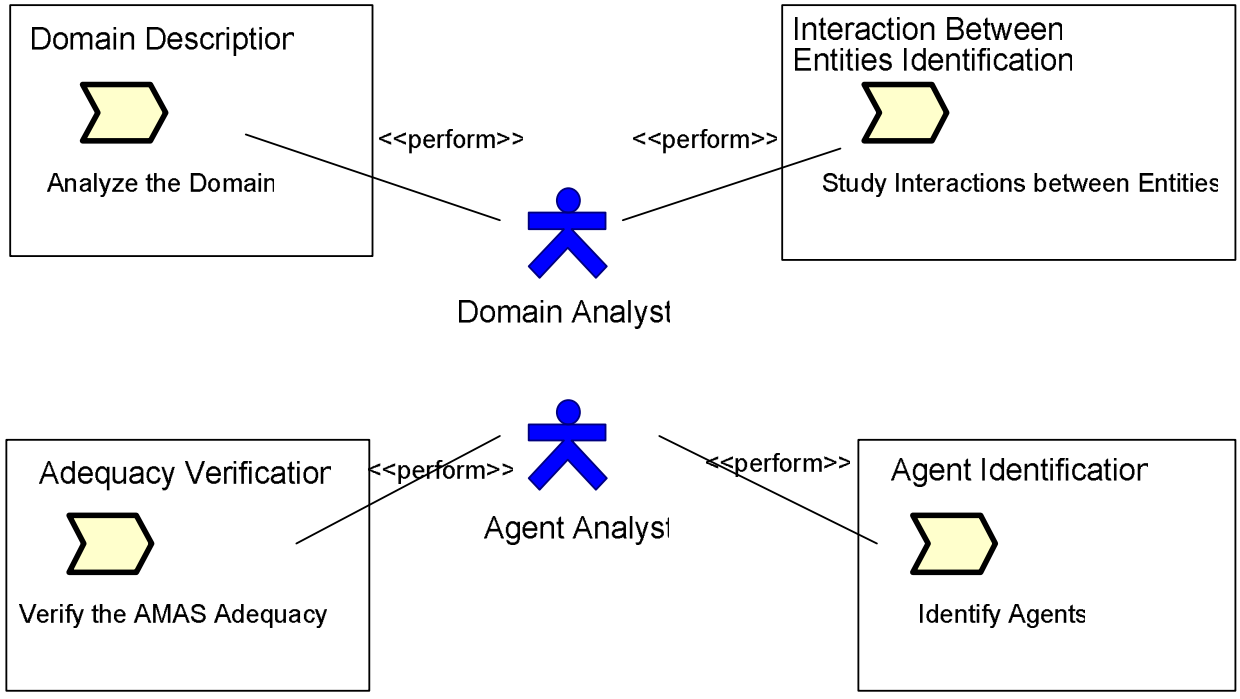

Figure 16.The Analysis phase activities clustered in sub-phase level work definitions

Here is a summary of this phase activities:

Work Definition	Activity	Activity Description	Roles involved
Domain Description	Analyze the Domain	The domain analysis aims to determine the entities of the system	Domain Analyst (perform)
Adequacy Verification	Verify the AMAS Adequacy	The Adequacy Verification checks if an AMAS is necessary to build the system	Agent Analyst (perform)
Agent Identification	Identify Agents	The Agent Identification establishes what between entities are agents	Agent Analyst (perform)
Interaction Between Entities Identification	Study Interactions Between Entities	The interactions between entities are defined	Domain Analyst (perform)

4.2.1 Process roles involved

Two roles are involved in the Analysis phase [13]: the Domain Analyst and the Agent Analyst. They are described in the following sub-sections.

4.2.1.1 Domain Analyst

He is responsible of :

1. Analyze the Domain during the Domain Description work definition. The domain analysis aims to determine the entities of the system.
2. Study Interactions Between Entities during the Interaction Between Entities Identification work definition. The interactions between entities are defined.

4.2.1.2 Agent Analyst

He is responsible of :

1. Verify the AMAS Adequacy during the Adequacy Verification work definition. The Adequacy Verification checks if an AMAS is necessary to build the system.
2. Identify Agents during the Agent Identification work definition. The Agent Identification establishes which entities are agents.

4.2.2 Fragments extracted from Analysis Phase

4.2.2.1 Domain Description fragment

4.2.2.1.1 Portion of process

This fragment aims to individualize entities, searching them in the use cases by defining scenarios. This activity is performed through the Requirements Set and Keywords Set documents [12][13].

The process that is to be performed in order to obtain the result is represented in Figure 17 as a SPEM diagram:

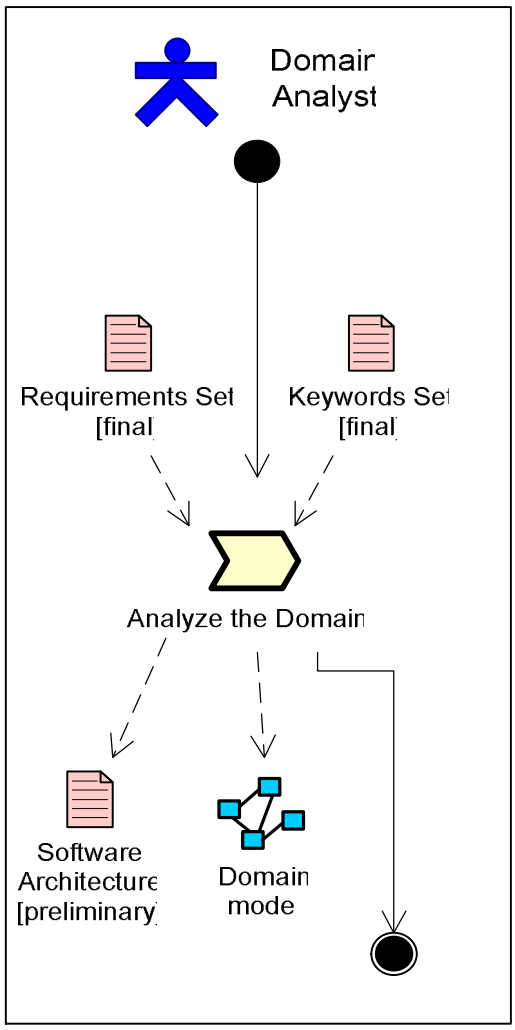

Figure 17. Domain Description

4.2.2.1.2 Deliverables

The output will be a set of entities that will compose a preliminary class diagram (Domain Model) through UML notation and a Software Architecture (preliminary) document.

4.2.2.1.3 Preconditions

context Domain Description::Analyze the Domain **pre**:

existence of Requirements Set (final) document

context Domain Description::Analyze the Domain **pre**:

existence of Keywords Set (final) document

4.2.2.1.4 Guideline(s)

The realization of activity described in this fragment happens through three phases:

1. to identify classes;
2. to study interclass relationships;
3. to construct the preliminary class diagram.

4.2.2.1.5 Dependency relationships with other fragments

This fragment depends of two fragments, Requirement Description and Keywords Identification since the domain is described using the Requirements Set and the Keywords Set documents.

4.2.2.1.6 Input/Output

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

INPUT	TO BE DESIGNED	OUTPUT
Requirements Description (final), Keywords (final)	Entities	Software Architecture (preliminary), Domain Model

4.2.2.2 Adequacy Verification fragment

4.2.2.2.1 Portion of process

This fragment aims to verify that the system has the necessity of one or more Adaptative Multi-Agent System (AMAS) to build the wanted system [12][13].

The process that is to be performed in order to obtain the result is represented in Figure 18 as a SPEM diagram:

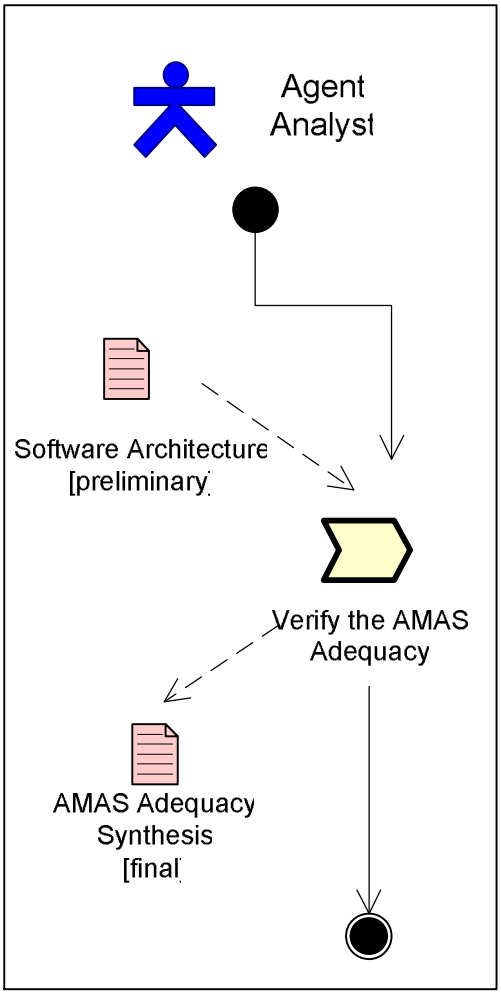

Figure 18. Adequacy Verification

4.2.2.2.2 Deliverables

The document in which the conclusions will be written on this verification is the AMAS Adequacy Synthesis (final) document.

4.2.2.2.3 Preconditions

context Adequacy Verification::Verify the AMAS Adequacy **pre**:

existence of Software Architecture (preliminary) document

4.2.2.2.4 Guideline(s)

The verification happens both to local that to global level.

To global level means to answer to the question: is it required an AMAS to implement the system?

To local level means to understand if there are agents to implement as AMAS or no.

4.2.2.2.5 Aspects of Fragment

The AMAS adequacy graphical tool can be used for the adequacy verification since it helps to answer to the questions on the global level and on that local.

4.2.2.2.6 Dependency relationships with other fragments

This fragment depends of the Domain Description fragment since the AMAS Adequacy is verified using the Software Architecture (preliminary) document.

4.2.2.2.7 Input/Output

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

INPUT	TO BE DESIGNED	OUTPUT
Software Architecture (preliminary)	AMAS Adequacy	AMAS Adequacy Synthesis (final)

4.2.2.3 Agent Identification fragment

4.2.2.3.1 Portion of process

The purpose of this fragment is the determination of the agents recognized between entities first defined. The considered agents are those that allow the construction of an AMAS system [12][13].

The process that is to be performed in order to obtain the result is represented in Figure 19 as a SPEM diagram:

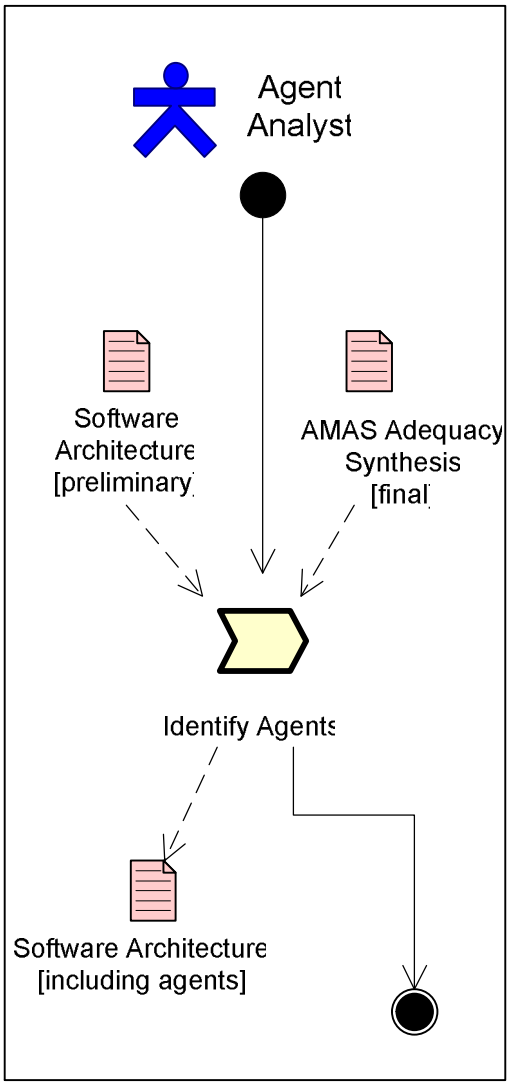

Figure 19. Agent Identification

4.2.2.3.2 Deliverables

This activity is included in the Software Architecture (including agents) document.

4.2.2.3.3 Preconditions

context Agent Identification::Identify Agents **pre**:

existence of Software Architecture (preliminary) document

context Agent Identification::Identify Agents **pre**:

existence of AMAS Adequacy Synthesis (final) document

4.2.2.3.4 Relationships with the MAS Meta-model

This fragment refers to the MAS meta-model adopted in ADELFE and contributes to define and describe the concept of agent in relation with it (Figure 20).

The following figure describes the relationship of the fragment with respect to the MAS model:

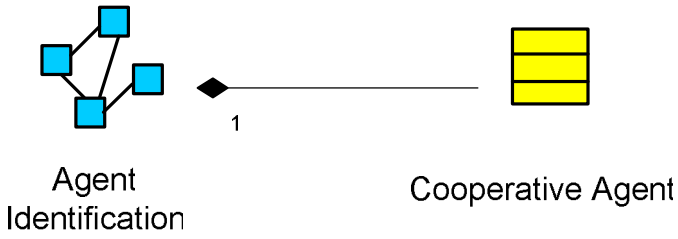

Figure 20. MAS Metamodel concept

4.2.2.3.5 Guideline(s)

This fragment is characterized by three phases: to study entities in the domain context, to identify the potentially cooperative entities and to determine agents.

4.2.2.3.6 Dependency relationships with other fragments

This fragment depends of three fragments:

1. Environment Description, because the entities are identified in this fragment;
2. Domain Description, because the agent identification use the Software Architecture (preliminary) document;
3. Adequacy Verification, because the agent identification use the AMAS Adequacy Synthesis document.

4.2.2.3.7 Input/Output

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

INPUT	TO BE DESIGNED	OUTPUT
Software Architecture (preliminary), AMAS Adequacy Synthesis (final)	Agents	Software Architecture (including agents)

4.2.2.4 Interaction Between Entities Identification fragment

4.2.2.4.1 Portion of process

This fragment aims to establish the relationships between the entities previously defined [12][13].

The process that is to be performed in order to obtain the result is represented in Figure 21 as a SPEM diagram:

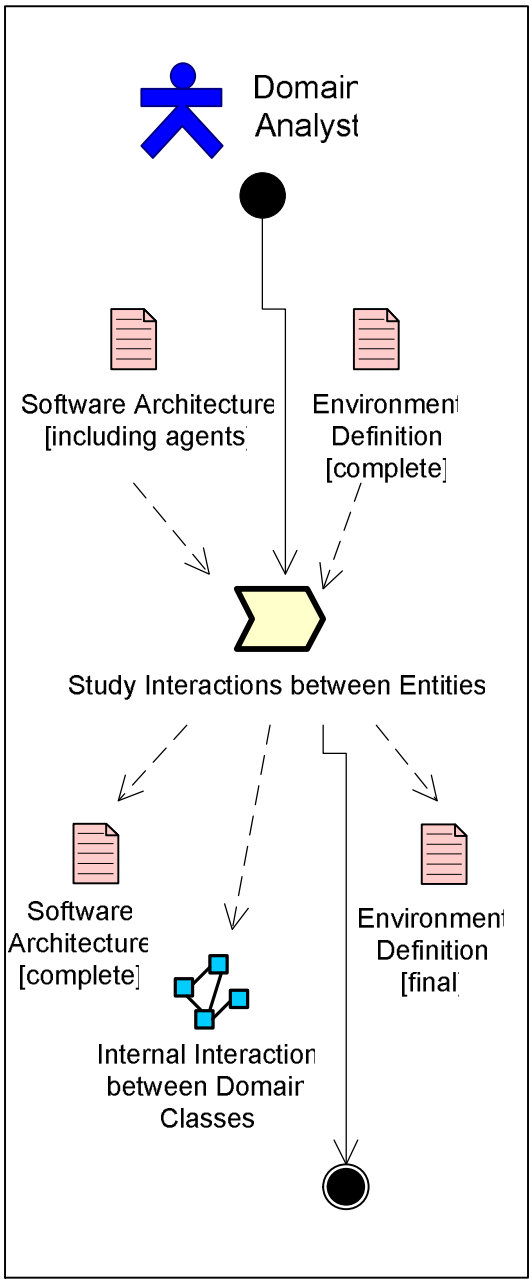

Figure 21. Interaction Between Entities Identification

4.2.2.4.2 Deliverables

The relationships found will be inserted in the Environment Definition (final) document to update it and in the Software Architecture document to complete it. This information will be inserted, also, in the Internal Interaction between Domain Classes UML diagram.

4.2.2.4.3 Preconditions

context Interaction Between Entities Identification::Study Interactions between Entities **pre**: existence of Software Architecture (including agents) document

context Interaction Between Entities Identification::Study Interactions between Entities **pre**: existence of Environment Definition (complete) document

4.2.2.4.4 Guideline(s)

The interactions to consider are of three types: active-passive entities relationships, active entities relationships and agents relationships.

4.2.2.4.5 Aspects of Fragment

The interactions can be expressed using UML and AUML diagrams.

4.2.2.4.6 Dependency relationships with other fragments

This fragment depends of two fragments:

1. Environment Description, because the entities are identified in this fragment;
2. Agent Identification, because the interactions between entities are identified using the Software Architecture (including agents) document.

4.2.2.4.7 Input/Output

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

INPUT	TO BE DESIGNED	OUTPUT
Software Architecture (including agents), Environment Definition (complete)	Interactions Between Entities	Software Architecture (complete), Environment Definition (final), Internal Interaction between Domain Classes

4.3 Design phase

The aims of this phase are [12][13]:

1. to define the system architecture identifying the software components;
2. to define the interaction languages;
3. to design the agents.

This phase involves two process roles, four work products (see Figure 22).

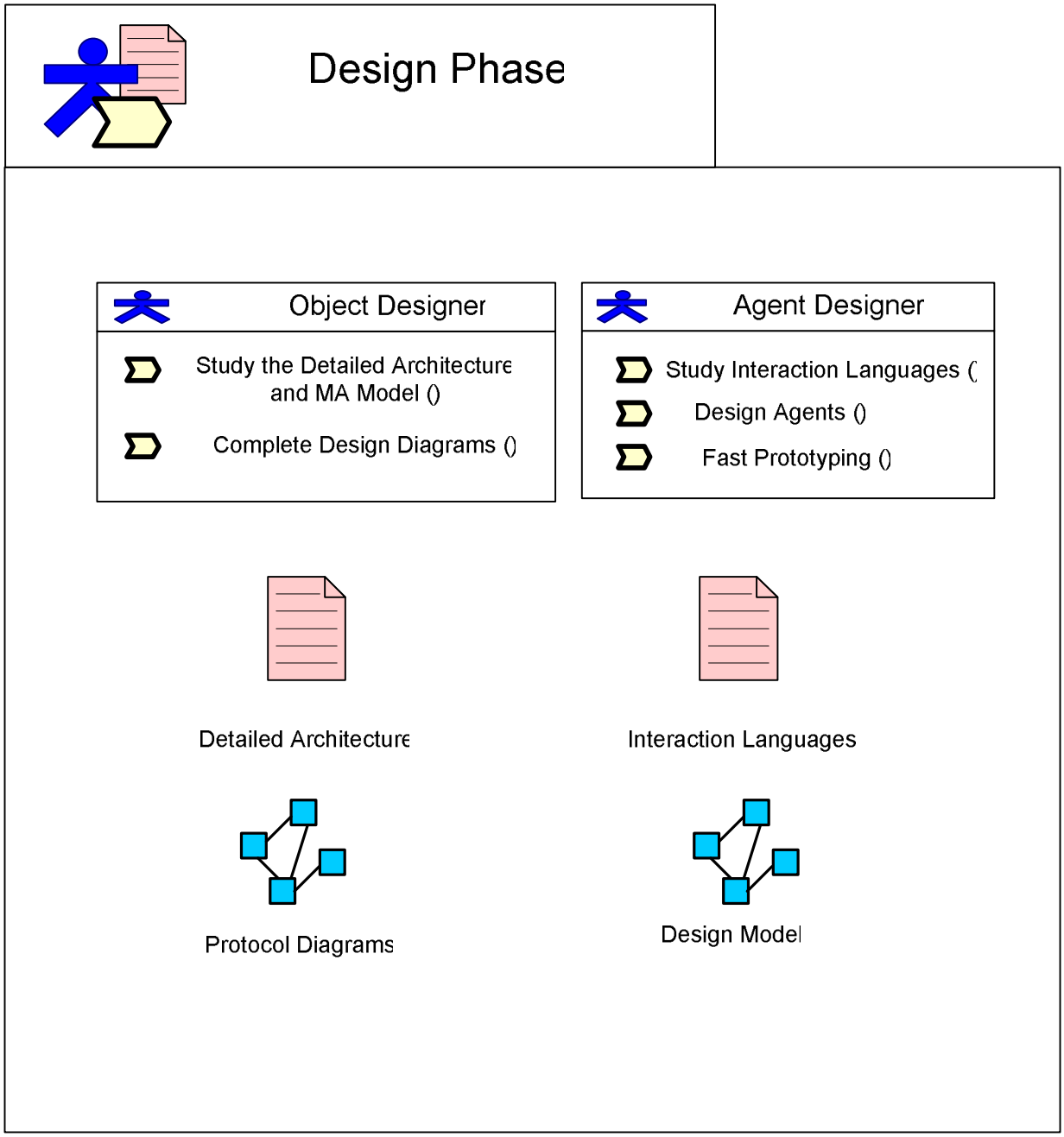

Figure 22. The Design phase represented as a SPEM discipline

The process to be performed in this phase is described in the following Figure 23. It is composed of three sub-phases level work definitions (Architecture Definition, Agents Specification and Architecture Refinement) and several related work products (mainly UML models and text documents).

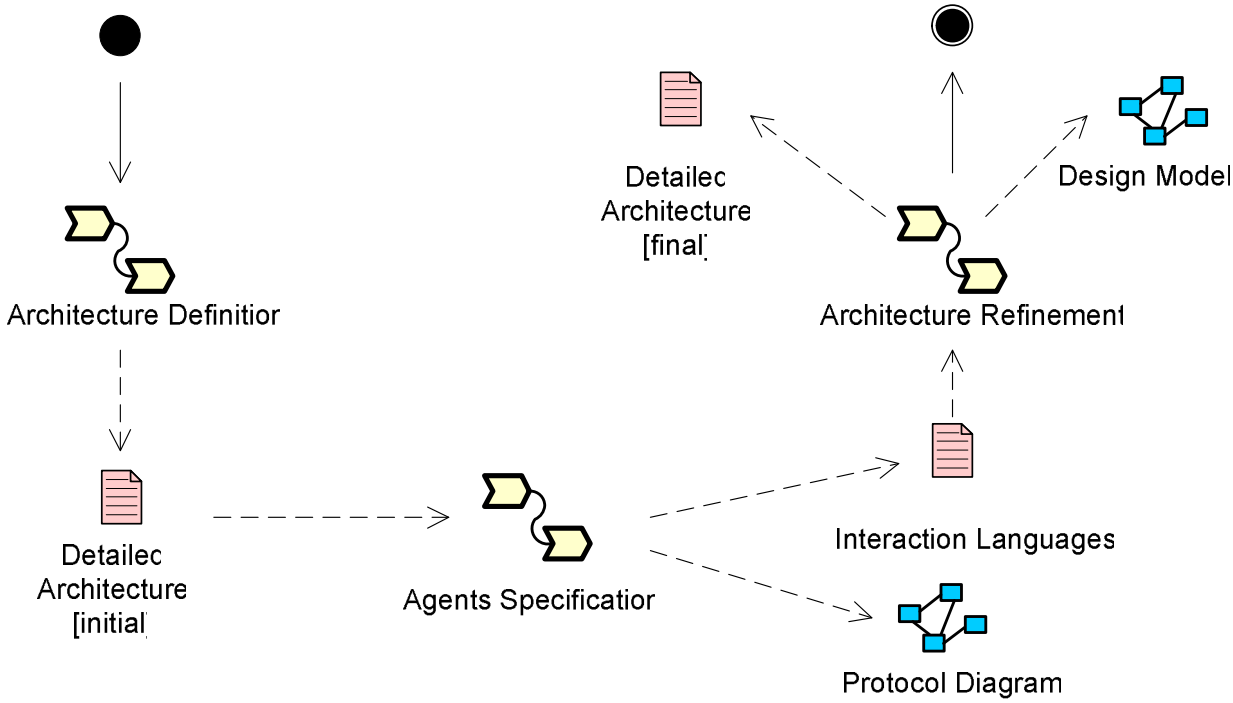

Figure 1. The Design phase described in terms of work definitions and work products

The three sub-phases are composed of several activities as described in Figure 24. 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.

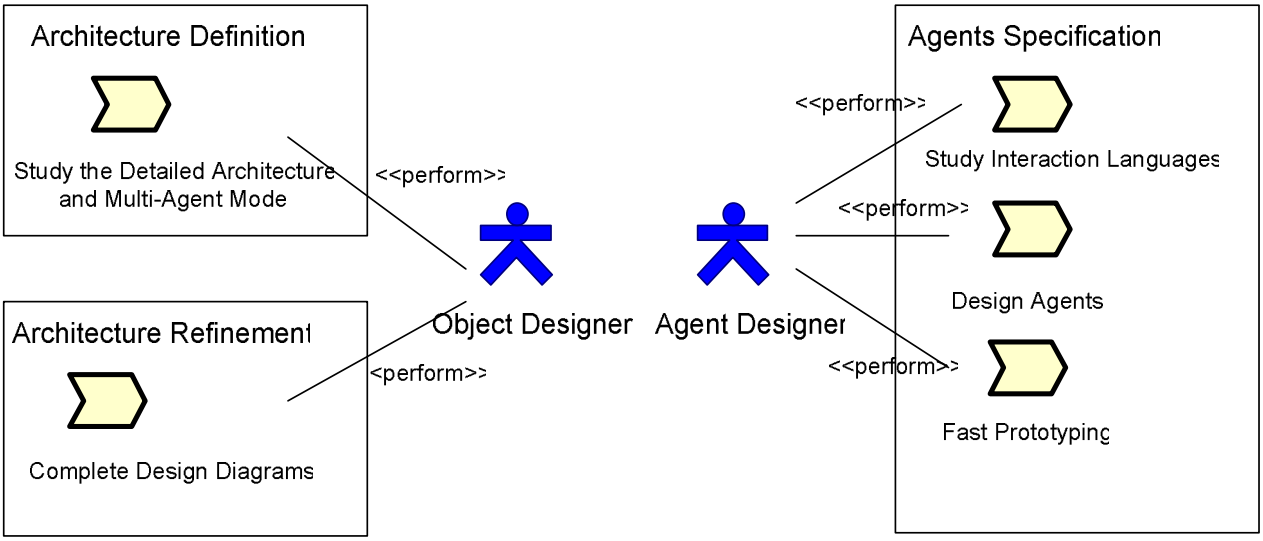

Figure 24. The Design phase activities clustered in sub-phase level work definitions

Here is a summary of this phase activities:

Work Definition	Activity	Activity Description	Roles involved
Architecture Definition	Study the Detailed Architecture and Multi-Agent Model	The software components are defined	Object Designer (perform)
Agent Specification	Study Interaction Languages	The agents interaction languages are defined	Agent Designer (perform)
Agent Specification	Design Agents	The agents behaviors are defined	Agent Designer (perform)
Agent Specification	Fast Prototyping	The agents behaviors are tested	Agent Designer (perform)
Architecture Refinement	Complete Design Diagrams	The system architecture and the project are completed	Object Designer (perform)

4.3.1 Process roles involved

Two roles are involved in the Design phase [13]: the Object Designer and the Agent Designer. They are described in the following sub-sections.

4.3.1.1 Object Designer

He is responsible of:

1. Study the Detailed Architecture and Multi-Agent Model during the Architecture Definition work definition. The software components are defined.
2. Complete Design Diagrams during the Architecture Refinement work definition. The system architecture and the project are completed.

4.3.1.2 Agent Designer

He is responsible of:

1. To study Interaction Languages during the Agent Specification work definition. The agents interaction languages are defined.
2. To design Agents during the Agent Specification work definition. The agents behaviors are defined.
3. Fast Prototyping during the Agent Specification work definition. The agents behaviors are tested.

4.3.2 Fragments extracted from Design Phase

4.3.2.1 Architecture Definition fragment

4.3.2.1.1 Portion of process

The main objective of this fragment is to define the system software components: packages, classes, objects and agents.

The Architecture is refined by using design patterns and re-usable components [12][13].

The process that is to be performed in order to obtain the result is represented in Figure 25 as a SPEM diagram:

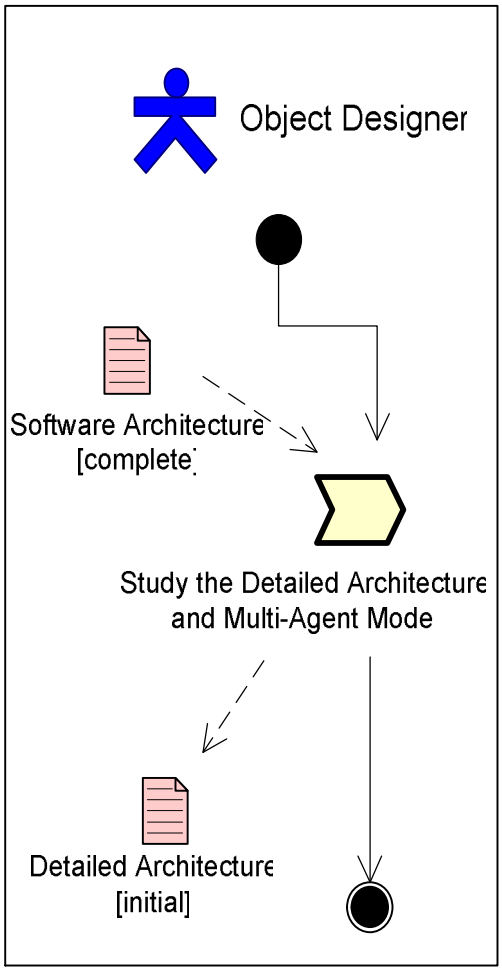

Figure 25. Architecture Definition

4.3.2.1.2 Deliverables

This fragment produce the Detailed Architecture (initial) document.

4.3.2.1.3 Preconditions

context Architecture Definition::Study the Detailed Architecture and Multi-Agent Model **pre**:

existence of Software Architecture (complete) document

4.3.2.1.4 Guideline(s)

This fragment is articulated in four phases: packages determination, classes determination, design patterns utilisation and component and class diagrams elaboration.

4.3.2.1.5 Dependency relationships with other fragments

This fragment depends on the Interaction Between Entities Identification fragment because it uses the Software Architecture (complete) document.

4.3.2.1.6 Input/Output

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

INPUT	TO BE DESIGNED	OUTPUT
Software Architecture (complete)	System software components	Detailed Architecture (initial)

4.3.2.2 Agent Specification fragment

4.3.2.2.1 Portion of process

This fragment, for every agent previously identified, aims to define its behavior: the skill, the aptitudes, an interaction language, a world representation, the Non Cooperative Situations. Then, the identified behaviors are tested [12][13].

The process that is to be performed in order to obtain the result is represented in Figure 26 as a SPEM diagram:

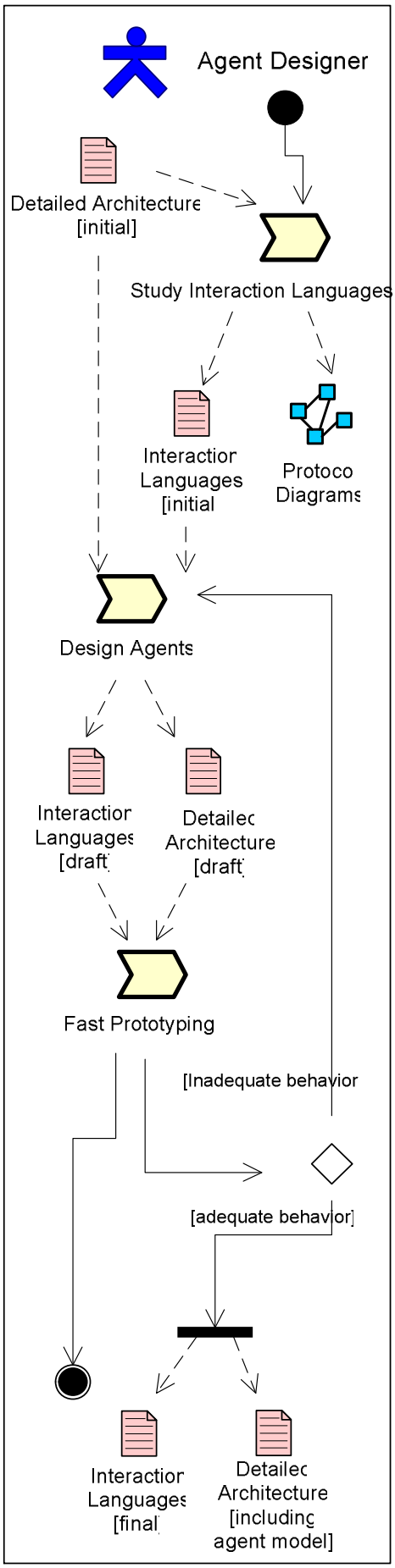

Figure 26. Agents Specification

4.3.2.2.2 Deliverables

The protocols that specify the interaction languages are represented through AUML diagrams and they are described in the Protocol Diagrams.

This step produces the Interaction Language (final) document and the Detailed Architecture document (including agent model).

4.3.2.2.3 Preconditions

context Agent Specification::Study Interaction Languages **pre**:

existence of Detailed Architecture (initial) document

context Agent Specification::Design Agents **pre**:

existence of Detailed Architecture (initial) document

context Agent Specification::Design Agents **pre**:

existence of Interaction Languages (initial) document

context Agent Specification::Fast Prototyping **pre**:

existence of Detailed Architecture (draft) document

context Agent Specification:: Fast Prototyping **pre**:

existence of Interaction Languages (draft) document

4.3.2.2.4 Relationships with the MAS Meta-model

This fragment refers to the MAS meta-model adopted in ADELFE and contributes to define and describe a set of concepts in relation with it: skills, aptitudes, characteristics, communication, Agent Interaction Protocol (AIP), representation, NCS.

The following figure 27 describes the relationship of the fragment with respect to the MAS model:

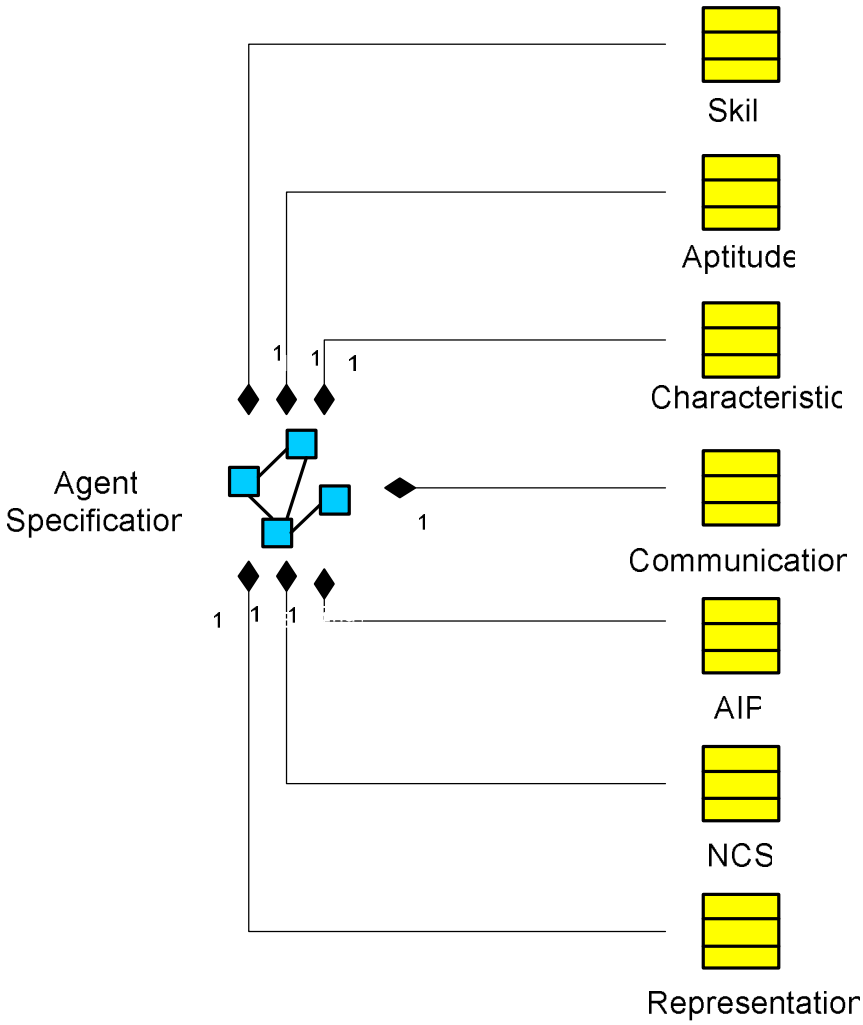

Figure 27. MAS Metamodel concept

4.3.2.2.5 Guideline(s)

This fragment is made up of two activities: to define behaviors (define skills, aptitudes and NCS, determine interaction languages and world representations), to test the behaviors.

The interaction languages may be determined by a set of classes or by a design pattern.

4.3.2.2.6 Aspects of Fragment

The interaction languages may be implemented by a specific agent communication tools for example ACL (implemented by FIPA).

To test the agents behaviors may be used the OpenTool simulation functionality, it creates the simulation environment (collaboration diagram), and then, with OTScript, it implements some methods to test.

4.3.2.2.7 Dependency relationships with other fragments

This fragment depend on two fragments:

1. Agent Identification, because the agents are defined in this fragment;
2. Architecture Definition, because the agent specification uses the Detailed Architecture (initial) document.

4.3.2.2.8 Input/Output

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

INPUT	TO BE DESIGNED	OUTPUT
Detailed Architecture (initial)	Interaction Languages	Interaction Languages (initial), Protocol Diagram
Detailed Architecture (initial), Interaction Languages (initial)	Agents Behaviors	Interaction Languages (draft), Detailed Architecture (draft)
Interaction Languages (draft), Detailed Architecture (draft)	Tested Agents Behaviors	Interaction Languages (final), Detailed Architecture (including agent model)

4.3.2.3 Architecture Refinement fragment

4.3.2.3.1 Portion of process

This fragment aims to complete the system architecture and the design activities [12][13].

The process that is to be performed in order to obtain the result is represented in Figure28 as a SPEM diagram:

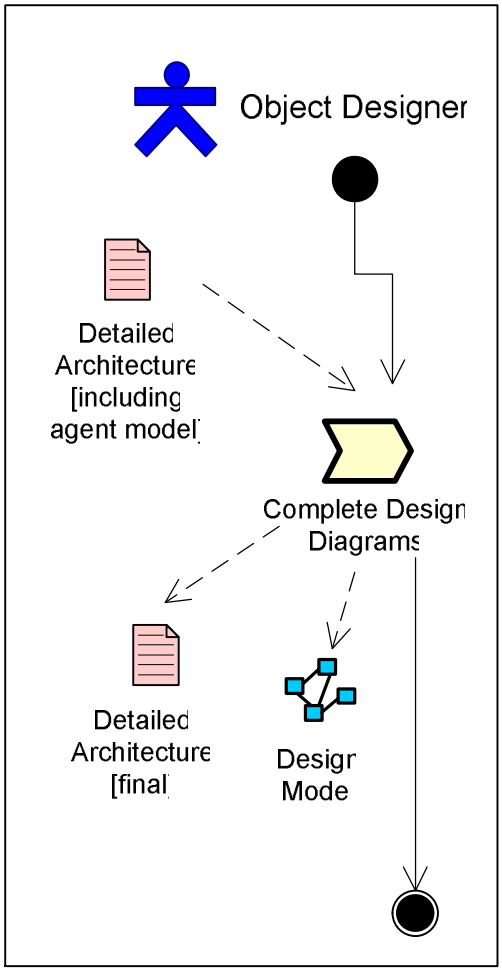

Figure 28. Architecture Refinement

4.3.2.3.2 Deliverables

The Architecture Document is completed and is produced the Design Model through UML diagrams.

4.3.2.3.3 Preconditions

context Architecture Refinement::Complete Design Diagrams **pre**:

existence of Detailed Architecture (including agent model) document

4.3.2.3.4 Guideline(s)

This activity is performed by two consecutive activities: to enhance design diagrams and to design dynamic behaviors.

4.3.2.3.5 Dependency relationships with other fragments

This fragment depends on the Agent Specification fragment because the architecture refinement uses the Detailed Architecture (including agent model) document.

4.3.2.3.6 Input/Output

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

INPUT	TO BE DESIGNED	OUTPUT
Detailed Architecture (including agent model)	Detailed Architecture	Detailed Architecture (final), Design Model

5 Glossary

Action - An action is a mean through which an agent acts in the environment. Only the agent can use one determined action since the actions are private.

Agent - Physical or virtual entity that it possesses the following abilities: to act in an environment, to communicate with other agents, to pursue an objective, to reproduce itself, to possess resources, to perceive his environment, to have one limited (or absent) environment representation.

Adaptive Multi-Agent System AMAS - The behavior is chosen while running to complete the task. It aims to get the better result.

Aptitude - Agent characteristic to reason about its knowledge and beliefs.

AUML - The Agent Unified Modeling Language expresses the interactions between the agents in a MAS, it is based on the UML notation.

Characteristics - Agent intrinsic or physical property.

Class Diagram - Static system structure, it is a set of classes, interfaces, collaborations and their relationship.

Collaboration Diagram - It describe both the static structure and dynamic behaviour of a system. It takes information from the use case, class and sequence diagrams, and it gets the interactions between the objects in terms of exchanged messages.

Consensual requirements - Condition in which end- user, designers and developers agree.

Cooperation - The cooperation is a social attitude of the agent that allows it to detect and to resolve Non Cooperative Situation (NCS), as incomprehension, ambiguity, etc.

Cooperation failure – NCS detection, it can occur when the cooperation protocol isn't respected or wrong interactions are occurred between the system and its environment.

Cooperative agent - A cooperative agent is an agent with a social and cooperative attitude. Its lifecycle is classical perceive-decide-action.

Design pattern - Problem solution occurred during the design phase. It has a format that indicates the objective, the motivation and the situation in which to apply it, the structure and the proposed solution.

Entity – An entity is a set of roles that the users play when they interact with use cases.
The entities can be of two types:

- Active entities that act autonomously;
- Passive entities that can only exchange data with the system, therefore, they are considered system resources.

Environment - The environment is everything that surrounds the agent in which this acts.

Functional requirements - The functional requirements are expressions of the possible actions and of the behaviors of the system.

Non functional requirements - The non functional requirements are expressions of the system properties and of the constraints on the functional requirement.

Goal - A goal is the objective that the system must to achieve and to maintain.

GUI - The GUIs are the interfaces that allow to the user to interact with the system.

Interaction - The languages of interaction allows to an agent to communicate with other agents or directly with its environment. The interaction can be classified in: perception and action.

Interaction language - The interaction language is a set of classes or of design pattern used by the agent to directly or indirectly communicate with other agents or with the environment.

Multi-Agent System (MAS) - A multi-agent system is based on the cooperation of autonomous entitities, said agents, able to withdraw informations from the surrounding

environment, to communicate to the other agents and to collaborate with them to achieve their own objectives;

Non Cooperative Situation - NCS are unusual situations in which the agent must face an unpredictable environment. There are different kinds of NCS: incomprehension, ambiguity, incompetence, unproductiveness, concurrency, conflict, uselessness.

Package - The package, in UML, is used for grouping the system elements.

Perception - Through the perception is possible to receive some information on the physical or social environment, therefore the designer must to endow the agents of perceptive ability.

Protocol Diagram - A protocol diagram describes the model of communication like a sequence of messages exchanged between the agents and the constraints contained in those communications.

Representation - A representation is used from the agent to determine its behaviour. The representation that evolve are represented through the multi-agent system.

Sequence Diagram - UML sequence diagram represents the behavior in terms of interactions. It also is used for illustrating the use cases representing the collaborations between entities from a temporal point of view.

Skill - The skill of an agent is referred to the knowledge that it has on the domain, allowing him to understand what are its functions.

State-chart diagram - UML State-chart diagram represents the behaviour of the classes when there are external stimuli, it provide a model of the of dynamic control from state to state.

System - A system offers to the final user a set of use cases. The system corresponds to the final software.

Use Case - A use case corresponds a set of scenarios to reach a final objective. A scenario is composed by a series of steps that describes the interaction between the system and the user.

World Representation - The world representation correspond to the description of physical and social environment of the agent. The agent must to access at the information of the world to act and to change it.

6 Artifacts Dependency Diagram

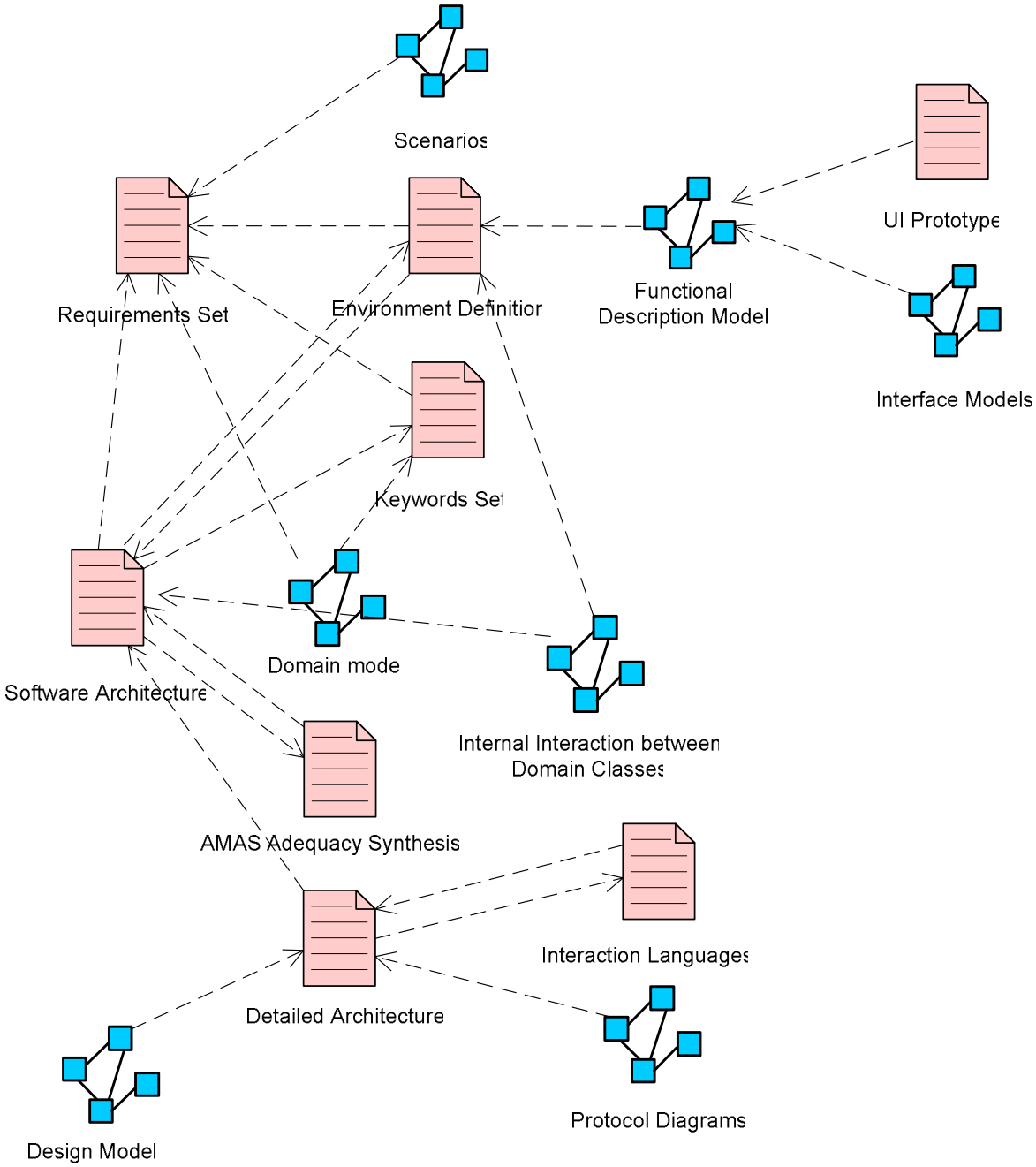

Figure 28. Dependencies Diagram

References

[1] Katia P. Sycara – Multyagent Systems.

[2] http://www.pa.icar.cnr.it/~cossentino/FIPAmeth/metamodel.htm

[3] Object Management Group – Software Process Engineering Metamodel Specification.

[4] Adriano Comai – RUP (Rational Unified Process) Caratteristiche, Punti di Forza, Limiti.

[5] Amund Tveit – A survey of Agent-Oriented Software Engineering.

[6] http://www.metamodel.com

[7] Pierre Glize – The AMAS Theory.

[8] Carole Bernon, Marie-Pierre Gleizes, Sylvain Peyruqueou, Gauthier Picard - ADELFE, a Methodology for Adaptive Multi-Agent Systems Engineering.

[9] Carole Bernon, Massimo Cossentino, Marie-Pierre Gleizes, Paola Turci, Franco Zambonelli - A Study of some Multi-Agent Meta-Models.

[10] Carole Bernon, Valérie Camps, Gauthier Picard, IRIT, Université Paul Sabatier, Toulouse, France – MAS Model in ADELFE.

[11] http://www.irit.fr/ADELFE

[12] IRIT/SMAC, Université Paul Sabatier, Toulouse, France – ADELFE's Fragments.

[13] Marie-Pierre Gleizes, Thierry Millan, Gauthier Picard – ADELFE: Using SPEM Notation to Unify Agent Engineering Processes and Methodology.