This document presents a very preliminary work concerning the fragments of the ADELFE methodology.

The web site describing the ADELFE methodology is http://www.irit.fr/ADELFE but ADELFE process is also briefly described in this document.

In a first step, we have chosen to develop five fragments that are linked to three activities of the ADELFE process because these activities are specifically related to the kind of problems and agents which ADELFE deals with. We think that they are more representative than other ones.

The resulting fragments are numbered depending on the activity they are coming from.

ADELFE's FRAGMENTS – V1

1. Fragment Definition (FIPA)

A fragment is a portion of the development process, composed as follows:

1. A portion of process (what is to be done, in what order), defined with a SPEM diagram
2. One or more deliverables (artifacts like (A)UML/UML diagrams, text documents and so on). The result of the work could also be some kind of product/artifact that is not be delivered to anyone outside the development process. It includes a reference to a recommended notation/language/structure to be used to represent AUML/UML/other diagrams if they are part of the deliverables (as it is common)
3. Some preconditions (they are a kind of constraint because it is not possible to start the process specified in the fragment without the required input data or without verifying the required guard condition)
4. A list of concepts (related to the MAS meta-model) to be defined (designed) or refined during the specified process fragment.
5. Guideline(s) that illustrates how to apply the fragment and best practices related to that
6. A glossary of terms used in the fragment (in order to avoid misunderstandings if the fragment is reused in a context that is different from the original one)
7. Composition guidelines – A description of the context/ problem that is behind the methodology from which the specific fragment is extracted.
8. Aspects of fragment. Textual description of specific issues like for example: platform to be used, application area…
9. Dependency relationships useful to assemble fragments

It should be noted that not all of these elements are mandatory. Some of them (for instance, notation, guideline or inputs) could be not applicable or not necessary for some specific fragment.

The fragment refers to a MAS meta-model and its aim is to contribute in increasing the definition of the instance of this meta-model that will solve the problem the designer is facing.

2. ADELFE Process

2.1. Preliminary requirements

- Activity 1: Define User Requirements. Its objective is to produce a preliminary version of a document in which requirements are expressed, named Requirements Set.
- Activity 2: Validate User Requirements. This activity aims to validate, by the End User, the last document.
- Activity 3: Define Consensual Requirements. This activity aims to regroup in the Requirements Set document the requirements expressed by both the End User and the

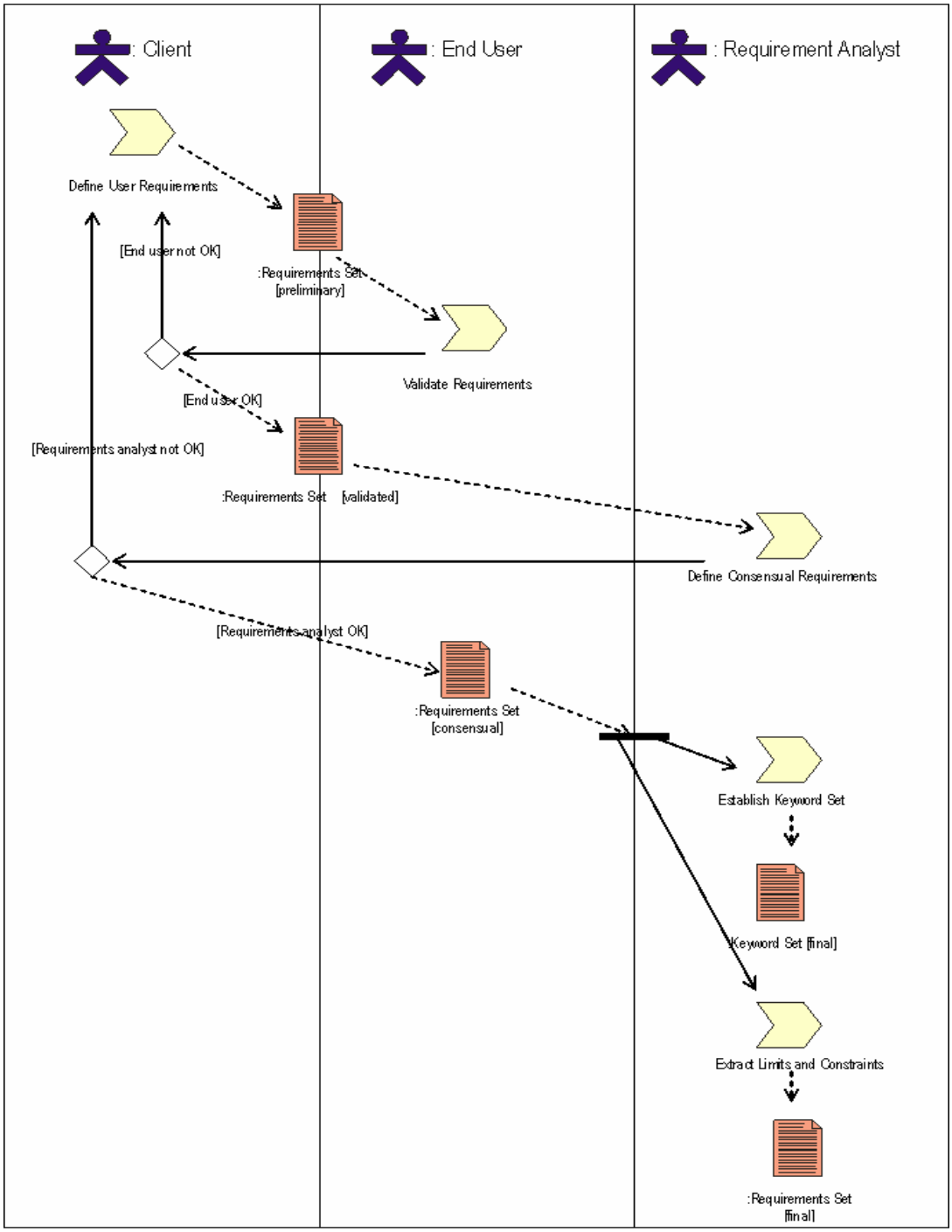

Figure 1. Preliminary Requirements Work Definition

Analyst.

- Activity 4: Establish the Keyword Set. From the Requirements Set document, the Analyst can extract keywords and list them in the Keyword Set document.
- Activity 5: Extract Limits and Constraints. The Analyst has to define limits for the system to be, in terms of operating system, languages, technology and so on. These constraints are added to the Requirements Set document.

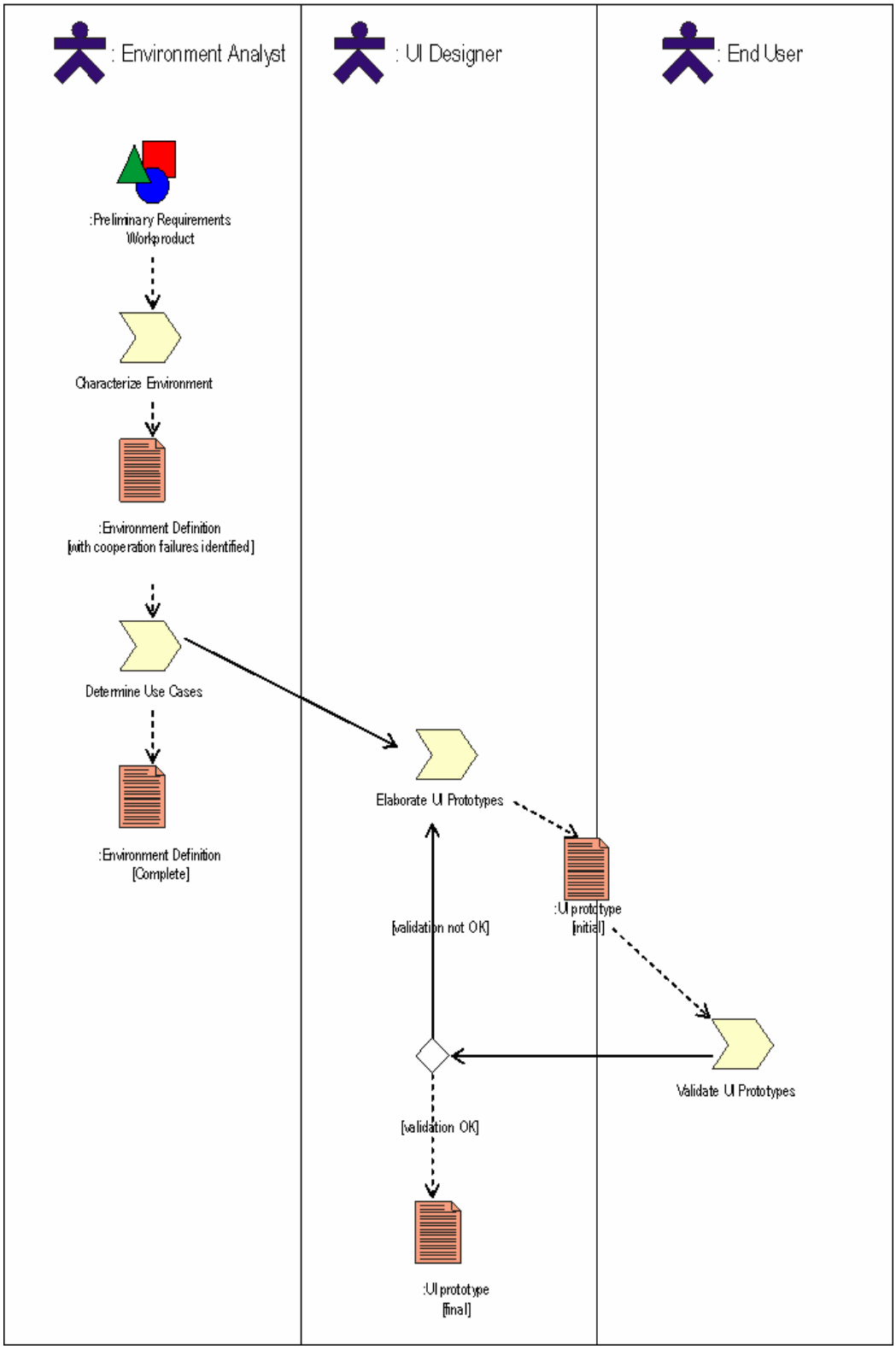

Figure 2. Final Requirements Work Definition

2.2. Final requirements

- Activity 6: Characterize the Environment. It consists in reasoning about the nature of the environment. The analyst has to characterize the environment by using a specific vocabulary and to check the input or output interfaces of the system-to-be in which cooperation failure may appear, according to the AMAS theory. This activity produces an initial Environment Definition document.
- Activity 7: Determine Use Cases. This activity, by using the use case UML notation, aims to clarify the functionalities the system-to-be has to provide. It results on the production of the

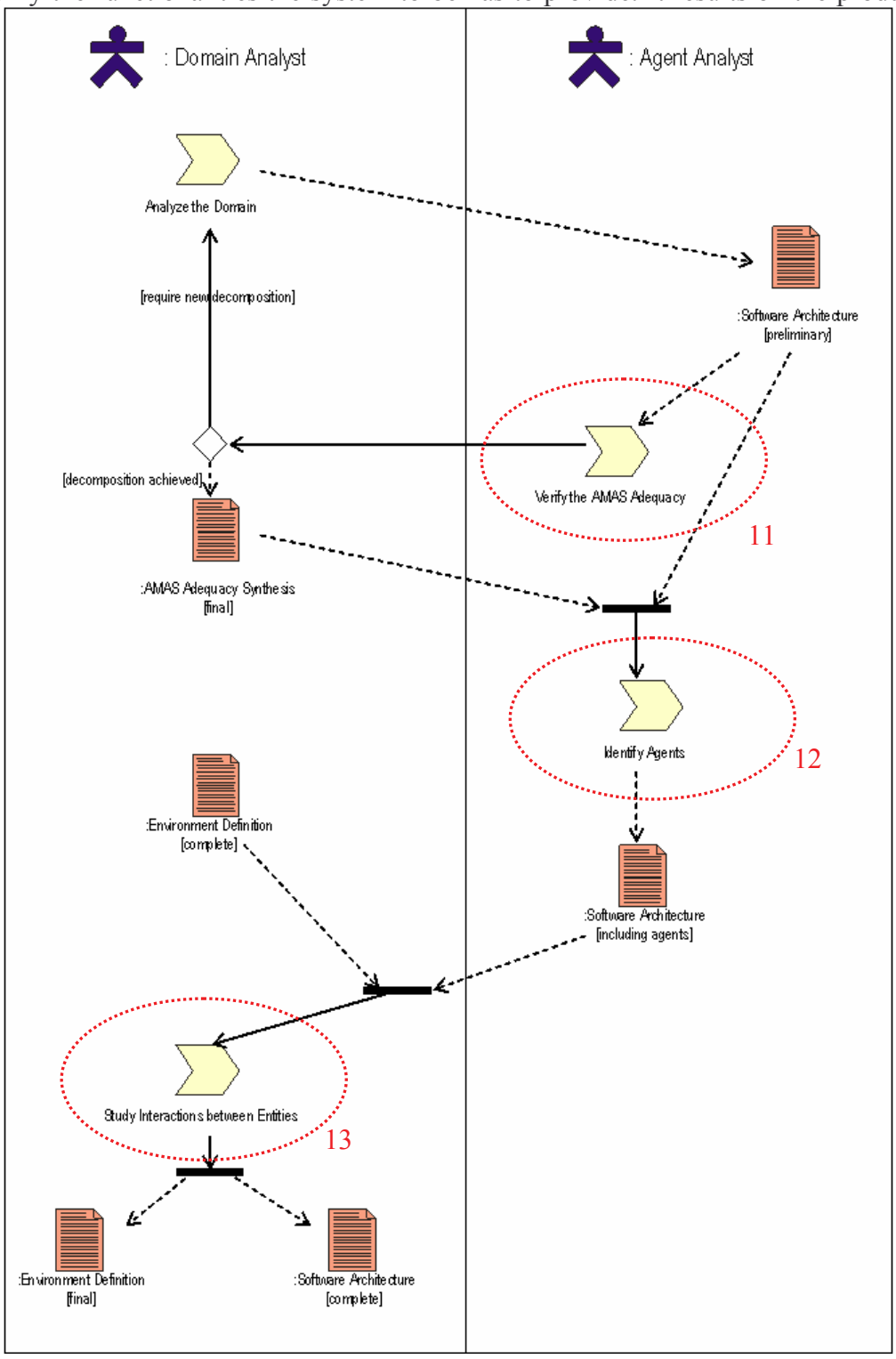

Figure 3. Analysis Work Definition

Functional Description model that is added to the Environment Description document.

- Activity 8: Elaborate UI Prototypes. This classical software engineering activity produces UI prototypes for each previously defined use cases.
- Activity 9: Validate UI Prototypes. This activity aims to validate the last work product by the End User.

2.3. Analysis

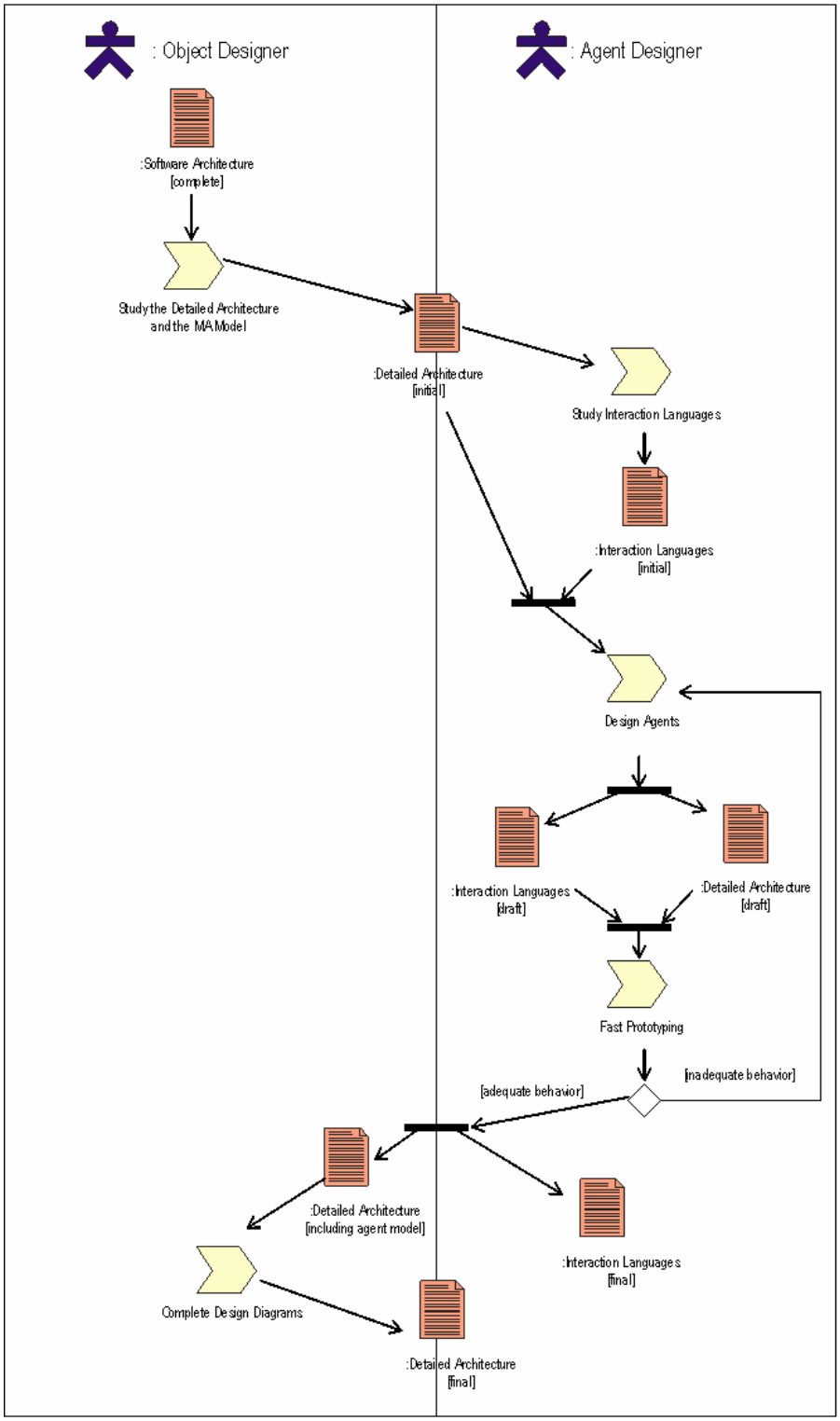

Figure 4. Design Work Definition

- Activity 10: Analyze the Domain. In order to model a static view of the system, by using actors and classes, the Domain Analyst has to index identified entities in the Software Architecture document.
- Activity 11: Verify the AMAS Adequacy. During this activity, the Agent Analyst has to use the AMAS Adequacy Tool to decide if the AMAS technology is necessary to design the system-to-be. This tool may also guide the Agent Analyst to detect a possible recursive decomposition of the system. The results of this analysis appeared in the AMAS Adequacy Synthesis document.
- Activity 12: Identify Agents. The Agent Analyst has to examine previously identified entities—during Activity 6—in their context, use case or sequence diagrams, to decide if some of them may be represented as cooperative agents in the system—in the sense of the AMAS theory—in terms of their interactions or the possibility to be in interaction with cooperation failures. This activity enhances the Software Architecture document.
- Activity 13: Study Interaction between Entities. The Domain Analyst has to reason on relations between active and passive entities, between active entities, and between agents. This activity produces models: sequence diagrams and AIP protocol diagrams. It aims to finalize the Software Architecture and the Environment Description documents.

2.4. Design

- Activity 14: Study the detailed Architecture and the Multi-Agent Model. The aim of this activity is to define the different components (packages, classes, etc.) that appear in the system, and to possibly re-use pre-existing components (i.e. design-patterns) in order to design the architecture of the system and to produce the Detailed Architecture document.
- Activity 15: Study Interaction Languages. Agents generally interact by using specific protocols. This activity aims to define the different interaction protocols agents may use by using AIP model. This notation has been integrated to the OpenTool software. This activity produces an initial Interaction Languages document, and its models.
- Activity 16: Design Agents. For each previously identified agent, the Agent Designer has to specify its architecture, i.e. specify its skills, its aptitudes, its interaction language(s), its world representations and its non-cooperative situations. So, the Detailed Architecture and Interaction Languages documents are enhanced and then completed.
- Activity 17: Fast Prototyping. This activity enables the Agent Designer to test the agents' behaviours by simulating instances of agents in the OpenTool platform. During this activity, the Agent Designer may modify agents' components by backtracking to the previous activity while agents' behaviours are not correct.
- Activity 18: Enhance Design Models. This last design activity aims to complete previous specifications models to close the design work definition and to design dynamical behaviours of the different entities appearing in the system by using state-chart diagrams.

3. MAS Meta-model for ADELFE

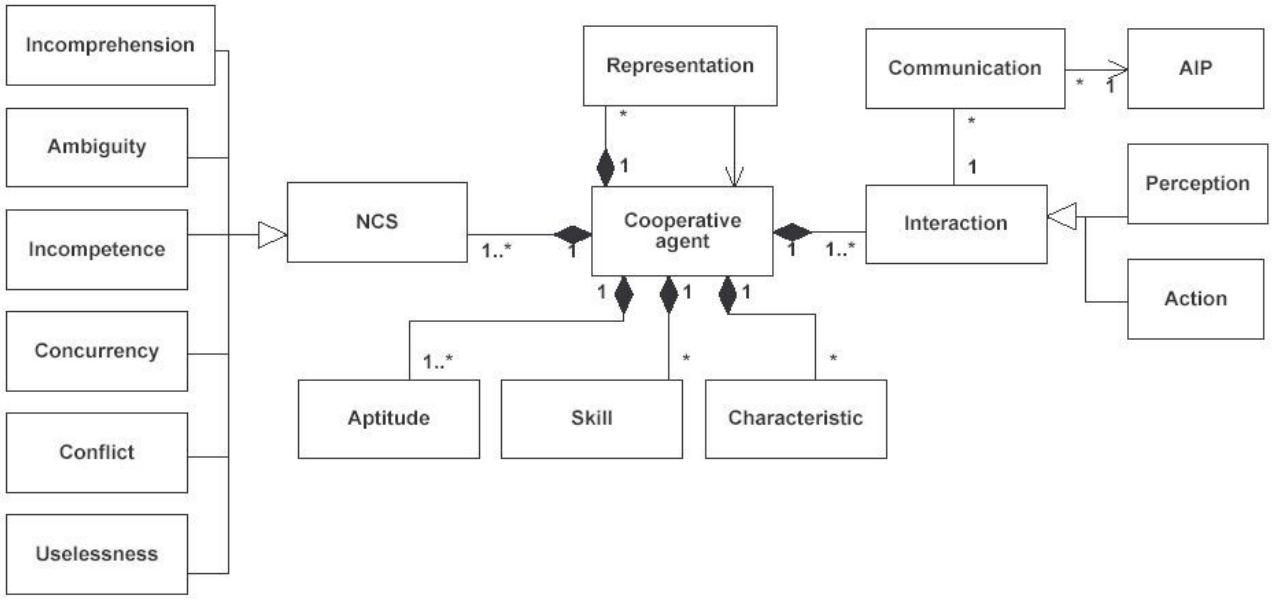

Figure 5. MAS Meta-model for ADELFE

4. ADELFE's Fragments

Based on a SPEM description, a fragment could be a phase or a work definition, an activity or a step.

In ADELFE, we think that a work definition is a too general point of view and a too complex one to become a step. Furthermore, the unit of decomposition that has been chosen in the ADELFE's process is the activity which can be decomposed into steps.

Therefore, the degree of cohesion and dependency between the different steps composing an activity must be studied in order to determine the fragment granularity.

Steps belonging to a same activity will be considered as separate fragments if they are loosely related to one another and could be applied independently. For instance, in the activity 13, the study of relationships between entities and between agents can be separately done and then each step will be viewed as a fragment. On the contrary, the steps of the activity 11 have no meaning if they are separated because this activity is purely linked to ADELFE and therefore couldn't be separately reused in another methodology.

A fragment could be then:

- A simple activity,
- A complex activity in which steps are so strongly dependent that thay cannot be separated and must be applied in a given predefined order,
- A step belonging to an activity which will not then appear as a fragment.

4.1. Fragment 11 - Verify the AMAS Adequacy

4.1.1. Portion of Process

This fragment corresponds to the portion numbered 11 on the "Analysis Work Definition SPEM specification" shown in figure 3.

This fragment is an activity composed of two steps that cannot be dissociated. The Agent Analyst has to answer several given questions concerning first, the global level of the system he has to design and second, the local level i.e. the entity level.

4.1.2. Deliverables

This fragment produces a textual deliverable called "AMAS Adequacy Synthesis". This text is made of the different answers given by the analyst when using the AMAS Adequacy tool as well as the results given by this tool.

4.1.3. Preconditions

This fragment is intended to define if an adaptive system is necessary to build the multi-agent system suited to the needs of the designer. It can be used to define which kind of MAS is useful.

In order to apply it, requirements have to be done. More especially, the analyst has characterized the environment and he produced a textual functional description that has been enclosed in the "Environment Definition" document. The domain analyst has identified the entities of the system in the "Software Architecture" document.

Input - Precondition	To be designed	Output
Environment definition (text)	Answers to questions	AMAS Adequacy Synthesis (text)
Software Architecture (UML)	-	-

4.1.4. Relationships with the MAS Meta-model

None.

4.1.5. Guideline(s)

Sometimes, AMAS programming is completely useless. For example, if the algorithm required to solve a task is already known, if the task is not complex or if the system is closed and nothing unexpected can occur, having an adaptive system is useless.

So, in this fragment, the adequacy of AMAS must be studied at two levels: at the global level to answer to the question "is an AMAS required to implement the system?" and at the local one to try to determine if some agents are needed to be implemented like AMAS i.e. if a certain kind of decomposition is required during the building of the system.

The AMAS Adequacy graphical tool will be a help to answer to these two points. Firstly, a certain number of questions regarding the global level must be answered. Then other questions concerning the local level will be added. In the end, the two areas at the bottom of the graphical tool window will show the answers of ADELFE regarding the global level and the local one; by clicking on those areas, an interpretation for these results can be obtained.

4.1.6. Glossary

- Agent – An agent is a physical or virtual entity which can act in an environment, which can communicate directly or not with other agents, which is driven by a set of tendencies, which possesses resources of its own, which is capable of perceiving its environment, which has only a partial representation of this environment, which possesses skills and can offer services, whose behavior tends towards satisfying its objective, taking account of the resources and skills available to it and depending of its perception, its representation and the communication it receives. [Ferber 99]. Furthermore, the main properties of agents in AMAS are their autonomy and the fact that they are cooperative. A cooperative agent has a social attitude: it always tries to

act cooperatively. It must be able to detect when it is not cooperative and act to come back in a state it judges being cooperative from its own point of view.

- AMAS – An adaptive multi-agent system is a multi-agent system which is able to autonomously change its behaviour while running. It does this to adjust its behaviour to its dynamic environment in order to be functionally adequate. Such a system is characterised by the following points: (1) it is plunged into an environment, (2) it has a function to achieve and (3) it is composed of interacting agents.
- Entity – An entity is an actor in the UML sense. It is a set of coherent roles which the users of use cases play when they interact with use cases. In ADELFE, two types of entities will be used: "active" entities which may behave autonomously and "passive" ones which can be considered as resources by the system.
- Environment – The environment of an agent refers to all that is external to the agent. One can distinguish the social environment (the agents it knows) from the physical environment (the material resources that can be perceived by the agent or by its own effectors). In ADELFE, environment is characterized using terms provided in [Russel 95], it can then be accessible or inaccessible, continuous or discrete, deterministic or not and dynamic or static.

4.1.7. Composition Guidelines

ADELFE, the methodology from which this fragment is taken, is devoted to the design of AMAS (Adaptive MAS). The related AMAS theory says that for all functionally adequate system (realizing the desired function) there is at least a system having a cooperative internal medium which realizes an equivalent function. In other words, to design a system realizing the desired function, having a system formed by cooperative agents is sufficient; this cooperation directs the social attitude of these agents.

Applications concerned by this kind of programming are open and complex ones, applications for which there is no algorithm a priori known (for example, flood forecast, electronic commerce, ants simulation…).

4.1.8. Aspects of Fragment

The AMAS adequacy tool (provided with ADELFE methodology) must be used in this fragment.

4.1.9. Dependency relationships with other fragments

None.

4.2. Fragment 12: Identify Agents

4.2.1. Portion of process

This fragment corresponds to the portion numbered 12 on the "Analysis Work Definition SPEM specification" shown in figure 3.

This fragment is an activity composed of three steps that cannot be dissociated and consists in determining the entities (already identified during previous activities) that must be viewed as agents more especially as cooperative agents. These entities must be examined in their context (use case or sequence diagrams) to decide if some of them may be represented as cooperative agents in the system in terms of their interactions or the possibility to be in interaction with cooperation failures.

4.2.2. Deliverables

This activity refines the Domain Model which belongs to the "Software Architecture" document by determine classes that will be tagged with the "Cooperative Agent" stereotype.

4.2.3. Preconditions

Entities must be known and indexed (active, passive) before establishing if they are agents or not. This fragment then needs a preliminary Domain Model built during previous activities of the methodology.

The aim of this fragment is to identify cooperative agents, this kind of agents must be then useful and the system built must be an AMAS. So the result given by the AMAS Adequacy activity must be positive.

Characteristics of cooperative agents must be known to be studied.

4.2.4. Relationships with the MAS Meta-model

Final results are identification of the Cooperative Agents (see figure 5). These Cooperative Agents are composed of different concepts that do not intervene in this fragment.

4.2.5. Guideline(s)

The first step of this fragment consists in studying entities in the domain context. For each entity already defined, the Agent Analyst has to decide if it is autonomous, it has a local goal to pursue, it has to interact with some other entities and if it has a partial view of the environment or has some abilities of negotiation.

When interacting with other entities or with the environment, an entity can encounter failure to respect the protocol or failure in the content of the interaction (misunderstanding...). These failures can be the consequences of a dynamic environment, the openness of the system... These situations are called "Non Cooperative Situations".

The second step has then to identify the potentially cooperative entities. For each entity coming from the previous step, the Agent Analyst has to determine if it has to move in a dynamic environment, has to face up to cooperation failures, has to treat Non Cooperative Situations.

In the end, entities coming from these two previous steps can be considered as agents.

4.2.6. Glossary

- Agent – An agent is a physical or virtual entity which can act in an environment, which can communicate directly or not with other agents, which is driven by a set of tendencies, which possesses resources of its own, which is capable of perceiving its environment, which has only a partial representation of this environment, which possesses skills and can offer services, whose behavior tends towards satisfying its objective, taking account of the resources and skills available to it and depending of its perception, its representation and the communication it receives. [Ferber 99]. Furthermore, the main properties of agents in AMAS are their autonomy and the fact that they are cooperative. A cooperative agent has a social attitude: it always tries to act cooperatively. It must be able to detect when it is not cooperative and act to come back in a state it judges being cooperative from its own point of view.
- AMAS – An adaptive multi-agent system is a multi-agent system which is able to autonomously change its behaviour while running. It does this to adjust its behaviour to its dynamic environment in order to be functionally adequate. Such a system is characterised by the following points: (1) it is plunged into an environment, (2) it has a function to achieve and (3) it is composed of interacting agents.
- Entity – An entity is an actor in the UML sense. It is a set of coherent roles which the users of use cases play when they interact with use cases. In ADELFE, two types of entities will be used: "active" entities which may behave autonomously and "passive" ones which can be considered as resources by the system.

- Autonomy – The autonomy of an agent can be expressed as following: An agent has its own life, independently of the existence of other agents, an agent is able to survive in dynamic environments without an external control and an agent takes internal decisions about its behaviour only considering the perceptions, knowledge and representations it possesses.
- Cooperation failure – A cooperation failure corresponds to the detection of a Non Cooperative Situation. Such a failure can be viewed as a cooperation protocol which is not obeyed or "bad" interactions that may occur between the system and its environment.
- "Cooperative Agent" stereotype – A cooperative agent is an agent that possesses a cooperative social attitude. As an object, a cooperative agent must have a run method which simulates its lifecycle which consists in having perceptions, taking decisions and then doing actions (perceive-decide-act). To ensure that this method does exist, an agent inherits from a super-class called CooperativeAgent.
- Dynamic environment – The environment of an agent refers to all that is external to the agent. One can distinguish the social environment (the agents its knows) from the physical environment (the material resources that can be perceived by the agent or by its own effectors). The state of a dynamic environment depends upon actions of the system that is within this environment but is also dependent on the actions of some other processes. So, changes cannot be predicted by system. For example, the Internet is a highly dynamic environment.
- Goal – A goal is a set of states of the world that an agent is committed to achieve/maintain. Therefore a goal is a situation, but not all situations are goals. A set of states of the world is generally not a goal unless there is an agent committed to achieve/maintain this set of states [Eurecom 00].
- Non Cooperative Situations – When the environment is unpredictable, or when the system is open, classical algorithms fail because the designer is unable to find an algorithm which is able to list all the existing possibilities. The aim of the AMAS technology is to design systems that do their best when a difficulty is encountered. In classical programs, these unexpected events can be processed as exceptions. In the AMAS theory context, these "exceptions" - expressing unusual situations that an agent may be faced with - are called "Non Cooperative Situations" (NCS). Different kinds of NCS exist, such as: incomprehension, ambiguity, incompetence, concurrency, conflict, uselessness.

4.2.7. Composition Guidelines

ADELFE, the methodology from which this fragment is taken, is devoted to the design of AMAS (Adaptive MAS). The related AMAS theory says that for all functionally adequate system (realizing the desired function) there is at least a system having a cooperative internal medium which realizes an equivalent function. In other words, to design a system realizing the desired function, having a system formed by cooperative agents is sufficient; this cooperation directs the social attitude of these agents.

Furthermore, in ADELFE, agents are not a priori known. They must be identified among the different entities composing the system. This is the aim of this fragment, to determine if some entities will stay 'simple' objects or become agents depending on cooperative agents' characteristics.

Applications concerned by this kind of programming are open and complex ones, applications for which there is no algorithm a priori known (for example, flood forecast, electronic commerce, ants simulation…).

4.2.8. Aspects of Fragment

This fragment is linked to a specific agent architecture: cooperative agents.

4.2.9. Dependency Relationships with Other Fragments

This fragment depends on the fragment that tells the analyst is the AMAS technology is useful to him: AMAS Adequacy fragment (that corresponds to the activity #11 of ADELFE).

4.3. Fragment 13-A: Study Interactions between Active and Passive Entities

In the activity #13 of ADELFE, the Domain Analyst has to reason on relations between active and passive entities, between active entities, and between agents.

The analyst may want to study these three kinds of relations independently so, each step of this activity can be viewed as a fragment to be reused in another methodology than ADELFE.

4.3.1. Portion of Process

This fragment provides from the Analysis Work Definition of ADELFE (see figure 3) and represents the first step of the activity #13.

Interactions between active and passive entities of the system must be expressed as scenarios.

4.3.2. Deliverables

This fragment produces scenarios showing how active and passive entities may interact. These scenarios are described using UML diagrams such as sequence diagrams or collaboration diagrams. This study can also change the software architecture if new entities are identified or if some may be refined or decomposed.

If these documents already exist, these deliverables will be used to refine the "Environment Definition" and "Software Architecture" documents coming from previous activities.

4.3.3. Preconditions

Entities have to be identified and indexed as being passive or active ones. In fact, most of the requirements have to be done.

4.3.4. Relationships with the MAS Meta-model

None. Active or passive entities do not appear in the ADELFE MAS meta-model.

4.3.5. Guideline(s)

In this fragment, the Domain Analyst has to express relations between active and passive entities using sequence diagrams or collaboration diagrams.

4.3.6. Glossary

- Collaboration diagram – A collaboration diagram describes interactions among objects in terms of sequenced messages. Collaboration diagrams represent a combination of information taken from class, sequence, and use case diagrams describing both the static structure and dynamic behaviour of a system.
- Entity – An entity is an actor in the UML sense. It is a set of coherent roles which the users of use cases play when they interact with use cases. In ADELFE, two types of entities will be used: "active" entities which may behave autonomously and "passive" ones which can be considered as resources by the system.
- Sequence diagram – A sequence diagram is a means to illustrate a use case by representing the collaborations between entities from a temporal point of view.

4.3.7. Composition Guidelines

In ADELFE, the methodology from which this fragment is taken, agents are not a priori known. Every entity in the system must then be studied to determine if it will be considered as an agent or as a 'simple' object. That is why during the requirements and domain analysis, entities must be identified and indexed as being passive or active depending on their degree of autonomy.

4.3.8. Aspects of Fragment

Scenarios are expressed using UML diagrams.

4.3.9. Dependency Relationships with Other Fragments

None.

4.4. Fragment 13-B: Study Interaction between Active Entities

In the activity #13 of ADELFE, the Domain Analyst has to reason on relations between active and passive entities, between active entities, and between agents.

The analyst may want to study these three kinds of relations independently so, each step of this activity can be viewed as a fragment to be reused in another methodology than ADELFE.

4.4.1. Portion of process

This fragment represents the second step of the activity #13 of ADELFE, situated in the Analysis Work Definition (see figure 3).

In this fragment, scenarios of interactions between active entities must be identified and expressed.

4.4.2. Deliverables

This fragment produces scenarios showing how active entities may interact. These scenarios are described using UML sequence diagrams that will enrich the "Environment Definition" document if it exist (that depends on the methodology built so far).

New active entities may appear and will be added to the "Software Architecture" document coming from previous activities, if it already exists.

4.4.3. Preconditions

Active entities have to be identified.

4.4.4. Relationships with the MAS Meta-model

None. Active entities do not belong to the ADELFE MAS meta-model except if they become agents. It is then in the fragment concerning the study of interactions between agents that a relation with the MAS meta-model will be found.

4.4.5. Guideline(s)

During this fragment, the Domain Analyst identifies relations between active entities and expresses them using UML sequence diagrams.

4.4.6. Glossary

- Entity – An entity is an actor in the UML sense. It is a set of coherent roles which the users of use cases play when they interact with use cases. In ADELFE, two types of entities will be used: "active" entities which may behave autonomously and "passive" ones which can be considered as resources by the system.

- Sequence diagram – A sequence diagram is a means to illustrate a use case by representing the collaborations between entities from a temporal point of view.

4.4.7. Composition Guidelines

In ADELFE, the methodology from which this fragment is taken, agents are not a priori known. Every entity in the system must then be studied to determine if it will be considered as an agent or as a 'simple' object. That is why during the requirements and domain analysis, entities must be identified and indexed as being passive or active depending on their degree of autonomy.

4.4.8. Aspects of Fragment

Scenarios are expressed using UML diagrams.

4.4.9. Dependency Relationships with Other Fragments

None.

4.5. Fragment 13-C: Study Interaction between Agents

In the activity #13 of ADELFE, the Domain Analyst has to reason on relations between active and passive entities, between active entities, and between agents.

The analyst may want to study these three kinds of relations independently so, each step of this activity can be viewed as a fragment to be reused in another methodology than ADELFE.

4.5.1. Portion of Process

This fragment corresponds to the last step in the activity #13 of ADELFE which belongs to the Analysis Work Definition (see figure 3).

Its aim is to identify and express relations between all the agents that are composing the system to build scenarios.

4.5.2. Deliverables

Scenarios enabling interactions between agents are expressed using (AUML) protocol diagrams that could enrich the "Environment Definition" document if it already exists.

4.5.3. Preconditions

Agents must have been identified.

4.5.4. Relationships with the MAS Meta-model

This fragment is related to the MAS meta-model of ADELFE, it concerns the Interaction part. Interactions enable an agent to communicate with others. In case of a direct communication through message exchanges, AIPs elaborated during this fragment can be used to express this kind of communication.

4.5.5. Guideline(s)

In this fragment, the Domain Analyst expresses relations between agents using (AUML) protocol diagrams. Relations between all the previously identified agents must be expressed. If agents seem to be too coarse-grained, if the algorithm of communication is then too complex, it is possible to refine the study of agents and come back to apply this fragment later.

4.5.6. Glossary

- Agent – An agent is a physical or virtual entity which can act in an environment, which can communicate directly or not with other agents, which is driven by a set of tendencies, which possesses resources of its own, which is capable of perceiving its environment, which has only a partial representation of this environment, which possesses skills and can offer services, whose behavior tends towards satisfying its objective, taking account of the resources and skills available to it and depending of its perception, its representation and the communication it receives. [Ferber 99]. Furthermore, the main properties of agents in AMAS are their autonomy and the fact that they are cooperative. A cooperative agent has a social attitude: it always tries to act cooperatively. It must be able to detect when it is not cooperative and act to come back in a state it judges being cooperative from its own point of view.
- Protocol diagram – An agent interaction protocol describes a communication pattern as an allowed sequence of messages between agents and the constraints on content of those messages [Odell 01].

4.5.7. Composition Guidelines

None.

4.5.8. Aspects of Fragment

Protocol diagrams are expressed using AUML.

These protocols could be designed using the OpenTool graphical tool linked with the ADELFE methodology.

4.5.9. Dependency Relationships with Other Fragments

None.