

ADELFE 2.0

Noelie Bonjean, Wafa Mefteh Mejri, Marie-Pierre Gleizes, Christine Maurel, Frédéric Migeon

▶ **To cite this version:**

Noelie Bonjean, Wafa Mefteh Mejri, Marie-Pierre Gleizes, Christine Maurel, Frédéric Migeon. ADELFE 2.0. Cossentino, Massimo; Hilaire, Vincent; Molesini, Ambra; Seidita, Valeria. Handbook on Agent-Oriented Design Processes, Springer, pp.19-64, 2013, 978-3-642-39975-6. 10.1007/978-3-642-39975-6_3 . hal-03792675

HAL Id: hal-03792675
https://hal.science/hal-03792675

Submitted on 3 Oct 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

ADELFE 2.0

N. Bonjean, W. Mefteh, M.P. Gleizes, C. Maurel and F. Migeon

ADELFE is a French acronym that means "Toolkit for Designing Software with Emergent Functionalities" ("Atelier de DEveloppement de Logiciels à Fonctionnalité Emergente" in french). ADELFE methodology is dedicated to applications characterized by openness and the need of the system adaptation to an environment. Its main goal is to help and guide any designer during the development of an Adaptive Multi-Agent System (AMAS). An AMAS is characterized by the following points: it is plunged into an environment and composed of interdependent agents, each agent carries out a partial function and the agents organization during runtime makes the system realizing an emergent function. Actually, an agent is locally cooperative, i.e. it is able to recognize cooperation failures called "Non Cooperative Situations" (NCS, which could be related to exceptions in classical programs) and to treat them.

ADELFE includes five Work Definitions that were initially inspired from the Rational Unified Process (RUP) and gathers twenty one activities, producing or refining twelve work products. These products are aggregating modelling diagrams or structured or free text. ADELFE, which is a Model-Driven (model-centered) development method, is not hardly dependent of Domain Specific Modelling Languages (DSML) but currently the recommendation is to use UML2 for general activities and to use AMASML (AMAS Modelling Language) and SpeADL (Species-based Modelling Language) for specific activities appearing in Analysis, Design or Implementation phases.

1 Introduction

ADELFE is a French acronym that means "Toolkit for Designing Software with Emergent Functionalities" ("Atelier de DEveloppement de Logiciels à Fonctionnalité Emergente" in french). The main goal of ADELFE is to help and guide any designer during the development of an Adaptive Multi-Agent System (AMAS). Adaptive software is used in situations in which the requirements are incompletely expressed, the environment is unpredictable or the system is open. In these cases, designers cannot implement a global control on the system and cannot list all sit-

Institut de Recherche en Informatique de Toulouse (IRIT), Toulouse, France
e-mail: Firstname.Name@irit.fr

uations that the system encounters. In these situations, ADELFE guarantees that the software is developed according to the AMAS theory. This theory, based on self-organizing multi-agent systems, enables to build systems in which agents only pursue a local goal while trying to keep cooperative relations with its neighbours agents. An AMAS is characterized by the following points: it is plunged into an environment and composed of interdependent agents, each agent carries out a partial function and the agents organization during runtime makes the system realizing an emergent function.

In the following, the ADELFE process is described by initially considering its whole process and then its five phases, which gather twenty one activities, producing or refining twelve work products.

Since 2003, ADELFE has been used in many academic and industrial projects, for a total of more than twenty AMAS produced. Recently, ADELFE has been lightly modified in order to integrate last research results on Multi-Agent Oriented Software Engineering and practical usages related by the industrial ADELFE partners. This chapter presents this up-to-date version of the method.

As it can be seen in the next sections, ADELFE is composed of several activities dedicated to Adaptive Multi-Agent-Systems. Considering this, it contains junction activities where the designer has to control if his problem requires a MAS solution or, even more, an AMAS solution. In the negative case, the design should be continued with a traditional process or a process dedicated to the problem characteristics. This documentation does not describe such activities and focus only on the process parts dedicated to AMAS development.

Finally, it is important to notice that ADELFE is a Model-Driven (model-centered) Development method which is still in progress. Works on fragmentation, simulation, formal methods, AMAS patterns are feeding the method every year in order to improve the development of complex systems based on AMAS. As the maturity of these topics is not sufficient to be included in this chapter, we left their presentation in research papers.

Relevant references about the ADELFE process and the ADELFE extensions are the following:

- Bernon, C.; Gleizes, M.-P.; Peyruqueou, S.; Picard, G.; ADELFE, a Methodology for Adaptive Multi-Agent Systems Engineering International Workshop on Engineering Societies in the Agents World (ESAW), Madrid, Spain, 16/09/2003-17/09/2003, Springer-Verlag, 2003, 156-169
- Bernon, C.; Camps, V.; Gleizes, M.-P.; Picard, G. Designing Agents' Behaviours within the Framework of ADELFE Methodology International Workshop on Engineering Societies in the Agents World (ESAW), Imperial College London, 29/10/2003-31/10/2003, Springer-Verlag, 2003, 311-327
- Rougemaille, S.; Arcangeli, J.-P.; Gleizes, M.-P.; Migeon, F. ADELFE Design, AMAS-ML in Action International Workshop on Engineering Societies in the Agents World (ESAW), Saint-Etienne, 24/09/2008-26/09/2008, Springer-Verlag, 2008

- Bernon, C.; Gleizes, M.-P.; Picard, G., Enhancing Self-Organising Emergent Systems Design with Simulation International Workshop on Engineering Societies in the Agents World (ESAW), Dublin, 06/09/2006-08/09/2006, Springer-Verlag, 2007, 4457, 284-299
- Lemouzy, S.; Bernon, C.; Gleizes, M.-P. Living Design: Simulation for Self-Designing Agents European Workshop on Multi-Agent Systems (EUMAS), Hammamet, 13/12/07-14/12/07, Ecole Nationale des Sciences de l'Informatique (ENSI, Tunisie), 2007
- Mefteh, W.; Migeon, F.; Gleizes, M.-P.; Gargouri, F. Simulation Based Design International Conference on Information Technology and e-Services, Sousse, Tunisie, 2012
- Bonjean, N.; Gleizes, M.-P.; Maurel, C.; Migeon, F., Forward Self-Combined Method Fragments. Workshop on Agent Oriented Software Engineering (AOSE 2012), Valencia, Spain, 04/06/2012-08/06/2012, Jorg Muller, Massimo Cossentino (Eds.), IFAAMAS, p. 65-74, juin 2012.

1.1 The ADELFE Process Lifecycle

ADELFE includes five Work Definitions (WD1..5) (see Fig.1):

- Preliminary Requirements (WD1): this phase represents a consensus description of specifications between customers, users and designers on what must be and what must give the system, its limitations and constraints.
- Final Requirements (WD2): in this work definition, the system achieved with the preliminary requirements is transformed in a use cases model, and the requirements (functional or not) and their priorities are organized and managed.
- Analysis (WD3): the analysis begins with a study or analysis of the domain. Then, identification and definition of agents are processed. The analysis phase defines an understanding view of the system, its structure in terms of components and identifies if the AMAS theory is required.
- Design (WD4): this phase details the system architecture in terms of modules, subsystems, objects and agents. These activities are important from a multi-agent point of view that a recursive characterization of multi-agent system is achieved at this point.
- Implementation (WD5): implementation of the framework and agent behaviours is produced in this work definition.



Fig. 1 The ADELFE Process Phases

It is important to notice that ADELFE is not a simple waterfall process but includes loops and increments that are not described on the graphical representation of the Fig. 1. For example, at the end of each phase, a validation step is executed which requires a possible transition towards an activity previously passed.

Each phase produces at least one document that is aggregated from modelling diagrams or from structured or free text. ADELFE is not hardly dependent of Domain Specific Modelling Languages (DSML) but currently the recommendation is to use UML2 for general activities and to use AMAS Modelling Language (AMASML) and Species-bAsed moDelling Language (SpeADL) for specific activities appearing in Analysis, Design or Implementation phases[1].

1.2 The ADELFE MAS Metamodel

1.2.1 Definition of MAS Metamodel Elements

The ADELFE method is composed of various tools based on model-driven development. To support model transformation and DSML editors an ADELFE metamodel has been defined. However, this metamodel is too much based on natural language (precise and complex) to support a designer guidance.

The ADELFE MMM is organised according to the five phases composing the process. We give in the following a short description of the main elements of this metamodel that is presented in five diagrams to simplify the layout and the discussion.

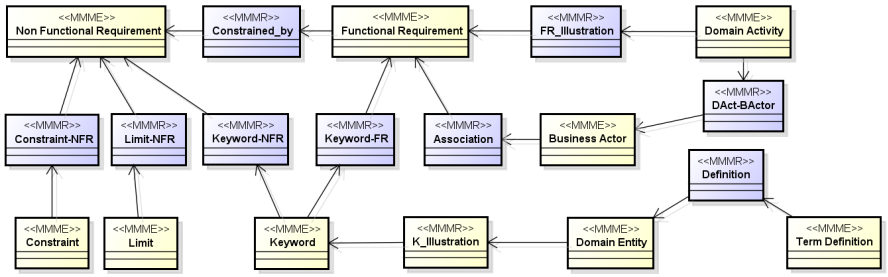


Fig. 2 The Preliminary Requirements Phase MAS Metamodel

The Preliminary Requirements Phase (see Fig.2) focuses on acquiring information about the client and his needs. Therefore, a consensual description of the problem is made in terms of functional and non-functional requirements, keywords, lim-

[1] See SMAC Team website (http://www.irit.fr/-Equipe-SMAC) for more information on these DSML.

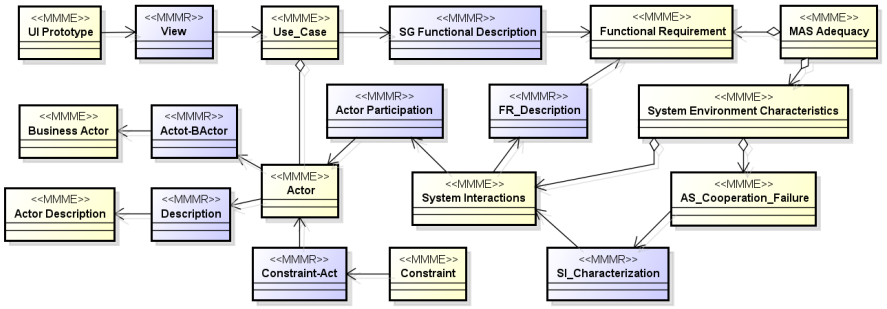


Fig. 3 The Final Requirements Phase MAS Metamodel

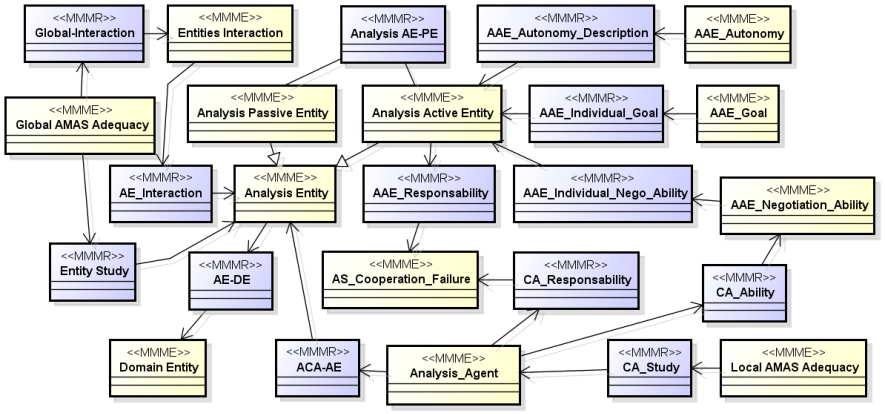


Fig. 4 The Analysis Phase MAS Metamodel

its and constraints of the system. A business model with business concepts and business activities is defined to complement the documentation.

In the Final Requirements Phase (see Fig.3), the objective is to validate the requirements and to detail the need through a description of the actors, a use case model, and scenarios descriptions. The end of the phase is dedicated to the characterization of the system environment and the identification of cooperation failures among the system interactions. This leads to a conclusion on the MAS adequacy to treat the problem of the client.

During the Analysis Phase (see Fig.4), the entities are characterized as passive or active and their interactions are described. The work product obtained enables an AMAS analyst to conclude on the adequacy (or not) of AMAS to deal with the problem. If the result is positive, all the interactions between the entities are described and cooperation failures are identified. From this information, agents (according to the definition of agent in ADELFE, see section 2.3) are identified and local AMAS adequacy is studied to conclude the phase.

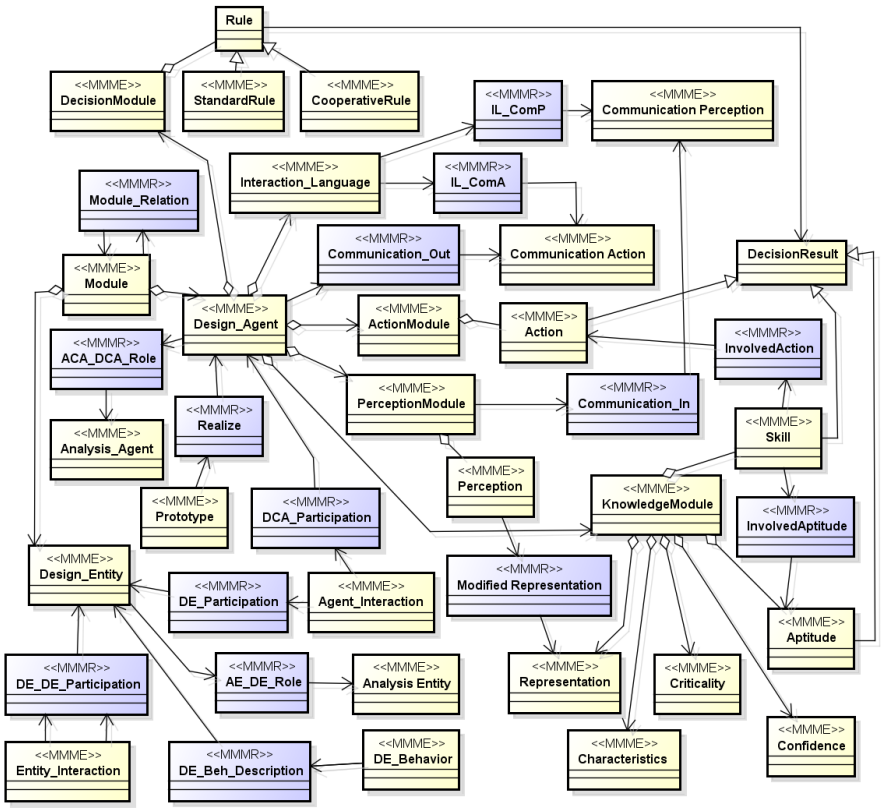


Fig. 5 The Design Phase MAS Metamodel

The heaviest phase of ADELFE is the Design Phase (see Fig.5) which consists in defining the multi-agent oriented architecture of the solution. It starts with the definition of a module view. Then, all the communication acts are defined in order to define precisely the entity interactions and agent interactions that will be useful to complete the module view and to define a component-connector view of the agents with their neighbourhood (agents, active and passive entities). The definition of the structure and behaviour of the agents is made in two steps which lead to the definition of the knowledge module, the action module, the perception module and the decision module. These two steps concern respectively the nominal behaviour of the agent, which enables the agent to reach its goal, and the cooperative behaviour which enables the agent to self-adapt to abnormal situations. Finally, a prototype is defined to validate the result.

Currently the last phase of ADELFE, the Implementation Phase (see Fig.6) focuses on the definition of the component-oriented architecture that will support the design. It is mainly composed of automated activities for model or code generation. However, in this document, we deliberately describe the process as manual oper-

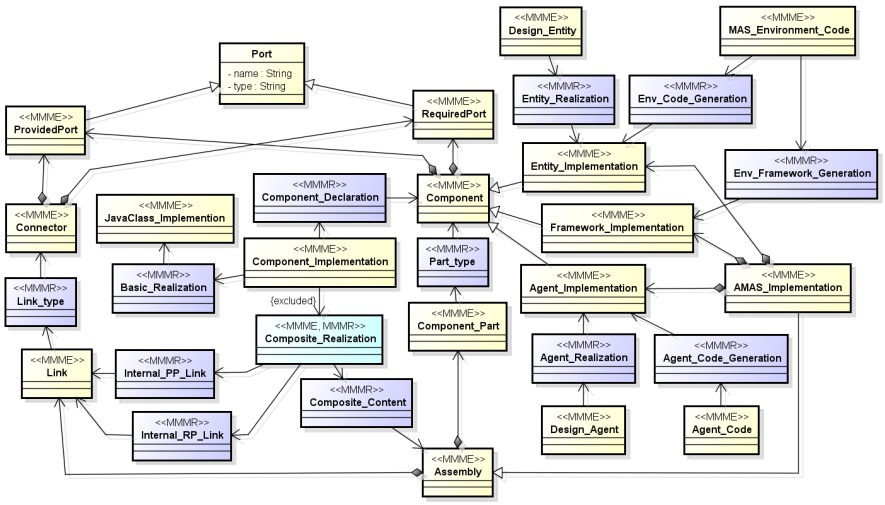


Fig. 6 The Implementation Phase MAS Metamodel

ations in order to give more details on the activities and in order to simplify the metamodel. The implementation of AMAS is not dependent of any MAS platform. On the contrary, it is recommended to produced a dedicated framework in order to gain in software quality. Actually, this framework is defined in terms of components (with provided and required ports, composite components, assembling) which are specified and implemented. First, the implementation focuses on the components defining entities, action and perception modules. Finally, decision module is implemented with standard behaviour rules and cooperative behaviour rules.
Like in every process, unit testing, integration testing and functional testing are taken into account but not described here.

1.2.2 Definition of MAS Metamodel Elements

The table below gives a set of concepts definitions related to AMAS theory and ADELFE method. It covers the entire MAS Metamodel used during Requirements, Analysis and Design.

Concept	**Definition**	**Referred Concepts**	**Domain**
(Software) System	A (software) system is the term describing the software to be produced, the application to be designed. Anything outside the system is called system environment.	System Environment	Requirements

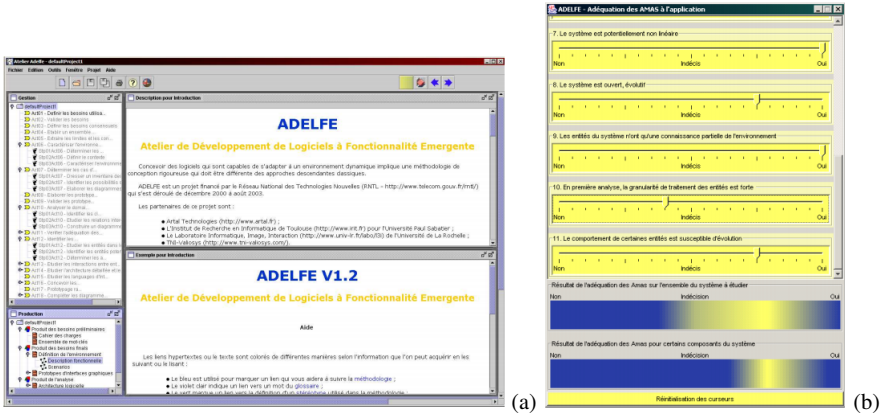
System Environment	It is the entire environment into which the software system is plunged and which is not under design. In UML, the term Actor (primary or secondary) is often used to characterize the environment. Of course, the frontier between environment and software system will use software entities that will deal with the interactions between environment and system.	Software System	Requirements
MAS Environment	The MAS environment is composed of all software elements that surround the MAS and which are not agents. The elements of the environment are called MAS entities (active or passive) and all have at minimum an interaction that is defined by means of sensors or effectors.	Passive Entity, Active Entity	Analysis
Agent Environment	The environment of an agent is the union of its neighborhoods during its life. It represents all the knowledge it has on other agents and on the MAS environment.	Agent Neighborhood, Agent	Analysis
Agent Neighborhood	The neighborhood of an agent is a part of the agent's environment at a particular time.	Agent Environment, Agent	Analysis
Multi-Agent System (MAS)	A MAS is the set of elements, called agents, which are not part of the environment. In a MAS, no agent can be isolated, that is to say without any link with another system component. Therefore, it can be considered as the set of agents that communicate (directly or via the environment) to achieve a common goal.	Goal, Agent	Analysis
Goal	The goal is an objective set by the designer to an agent or to the entire system.	Agent, (Software) System	Analysis
Passive Entity	In the MAS environment, passive entities are related to resources or to data. This implies that they have no autonomy and that a state transition can only be the result of an interaction with another system component. Moreover, a passive entity is unable to send or receive messages.	MAS Environment	Analysis
Active Entity	Unlike a passive entity, an active entity is given behavioural autonomy, allowing it to change state without necessarily interacting with another entity. An active entity can send messages, possibly proactively and receive messages.	Passive Entity	Analysis
(Cooperative) Agent	In ADELFE, an element of a MAS (i.e. an element that is not part of the MAS environment) is an agent. This agent is characterized by a cooperative attitude.	MAS, Cooperative Attitude	Analysis

Cooperation	It is a behavioural principle for an entity that avoids being placed in a situation of misunderstanding, ambiguity, incompetence, unproductiveness, conflict, concurrence or uselessness. This principle is applied to agents designed with ADELFE, but can also describe the behaviour of the software system as a whole. The definition of cooperation goes beyond the simple sharing of resources or collaboration. This cooperation includes all behaviours that allow the agent to prevent and to resolve conflicts that occur during system execution.	Agent, Software System	Analysis
Cooperative Attitude	An agent has a cooperative attitude when its activity tends to give priority to anticipate and solve all the Non Cooperative Situations (NCS) it might encounter with its environment. This implies the following properties: (i) *Sincerity:* If an agent knows a proposition p is true, it cannot say anything different to others. (ii) *Compassion:* An agent temporarily leaves its individual goal to help another agent in greater difficulty (temporary change of goal). (iii) *Reciprocity*: An agent knows that it has a cooperative attitude, like all other agents have.	Cooperative Attitude, Non Cooperative Situations, Agent Environment, Agent	Analysis
Communication Acts	A communication act is a mean implemented by an agent to interact with an agent and / or its environment. A speech act (e.g. FIPA ACL) is a communication act.	Agent, Agent Environment	Analysis
Behaviour	The behaviour of an agent is a life cycle consisting of the sequence: (i) perception of the environment (including communication aspects), (ii) decision that allows it to identify the state in which it lies and actions to be performed, (iii) execution of decided actions. The life cycle starts when the agent is created and completes when the agent dies. Agent behaviour can be represented as an automaton whose states are the situations that the agent can identify and transitions are actions it decides to execute.	Agent Environment, Agent	Design
Nominal Behaviour	The nominal behaviour is the part from the behaviour which enables the agent to reach its goal when it is in cooperative situation.	Goal, Agent, Behaviour	Design

Cooperative Behaviour	The cooperative behaviour of an agent enables it to detect the set of states identifying NCS and to describe the repairing actions to return to a cooperative situation or anticipatory attempt to avoid NCSs. In addition, an agent tries to help the most critical agent in its neighbourhood. In certain conditions, it spontaneously communicates information to agents that it thinks the information will be useful. Such a cooperative behaviour can be divided in three distinct steps: (i) tuning which consists in the modification of parameter values for parameters that influence the behaviour of the agent; (ii) reorganization which consists in the modification of the acquaintances of the agent that will lead to the reorganization of the system to make the resulting global function properly; (iii) evolution which consists in creation or suicide of agents.	Goal, Agent, Behaviour, NCS	Design
Criticality	For an agent, criticality represents the degree of non-satisfaction of its own goal. It enables an agent to determine the relative difficulty of agents in its neighbourhood. Evaluation methods and calculation of the criticality are specific to each type of agent.	Goal, Agent, Neighbourhood	Design
Behaviour Confidence	The behaviour confidence of an agent is an internal measure that provides information on the reliability of the decision on actions intended.	Agent, Behaviour	Design
Skills	The skills of an agent are capabilities in a domain that enables an agent to perform actions to achieve its goal.	Agent, Goal	Design
Characteristics	A characteristic is an intrinsic property of the agent. It can be visible or not and it can be modified by the agent or other agents.	Agent	Design
Aptitudes	The aptitudes of an agent are generic capabilities which are independent of its competence domain.	Agent	Design
Representations	Representations of an agent are the image that the agent has of its environment and itself, that is to say all of its perceptions and beliefs. They can be updated by means of its perceptions.	Agent, Agent Environment	Design
Interaction Language	The interaction language is a set of tools required by the agent to communicate with other agents. This communication can be done through messages (direct) or via the environment (indirect communication).	Agent	Design

1.3 Guidelines and Techniques

ADELFE is based on object-oriented methodology, inspired from the Rational Unified Process (RUP). Some steps have been added in the classical workflow fitting with adaptive MAS.
ADELFE is based on UML2 notation with the complementary use of DSML AMASML (AMAS Modelling Language) and SpeADL (Species-based Architecture Description Language), a design methodology, several model-driven tools and a library of components that can be used to facilitate the application development.
Guidance of the development process is supported by AdelfeToolkit (Fig.7 (a)). The general idea is to help the designer to follow the process, with descriptions, examples and presents a summary of works and artefacts already performed and remaining.
As mentioned in A11 and A13 activities of ADELFE process, the analyst must



(a) (b)

Fig. 7 The Tool for Monitoring the ADELFE Process: AdelfeToolkit (a); The Tool of the AMAS Adequacy (b)

verify that the problem needs or not AMAS. For this, a tool (Fig. 7 (b)) is provided to answer the questions at the macro-level (eight criteria) and the micro-level (three criteria). Answers to questions are graded from 0 to 20.

2 Phases of the ADELFE Process

ADELFE defines its five phases from the RUP definition. They are respectively dedicated to Preliminary Requirements, Final Requirements, Analysis, Design and Implementation.

2.1 Preliminary Requirements Phase (WD1)

The Preliminary Requirements Phase involves traditional software development stakeholders which are assigned classical activities. The goal of this phase is to obtain a precise and consensual description of the problem as well as the client's business. No specific modelling language is used. The process flow at the level of activities is reported in Fig. 8 and Fig. 9 depicts this phase according to documents, roles and work products involved.

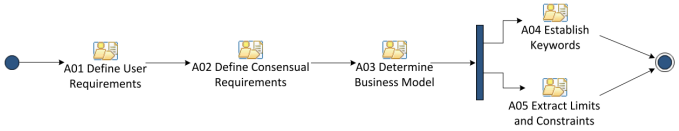


Fig. 8 The Preliminary Requirements Phase Flow of Activities

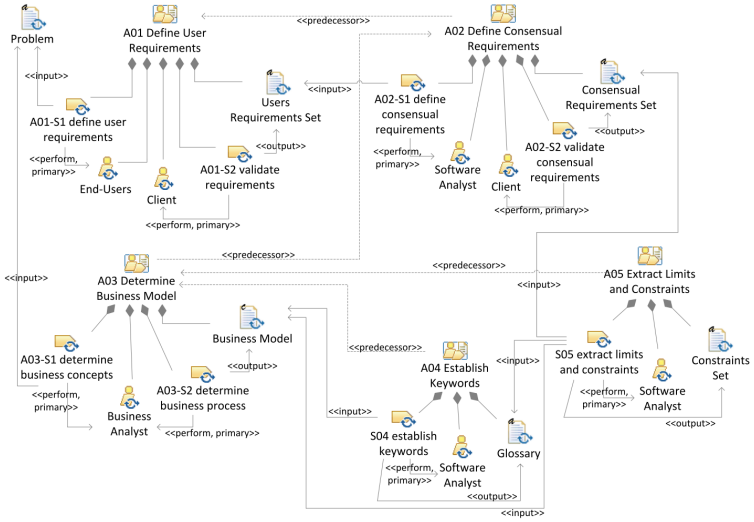


Fig. 9 The Preliminary Requirements Phase Described in terms of Activities and Work Products

2.1.1 Process Roles

Four roles are involved in the Preliminary Requirements Phase: the End-users, the Client, the Software analyst and the Business analyst.

- **Final User:** he is responsible of functional and non-functional requirements list during the *Define Users Requirements* activity. These requirements are used to define the system and its environment.

- **Client:** his main role is to validate product documents drawn up by others expert. He is responsible of the set of requirements approval during the *Define Users Requirements* activity and during the *Define Consensual Requirements* activity.

- **Software Analyst:** actually the software analyst gives a definition for the main concepts used to describe the system and its environment. He is responsible of: consensual requirements list during the *Define Consensual Requirements* activity, keywords during the *Establish Keywords* activity, and limits and constraints of the system during the *Extract Limits and Constraints* activity.

- **Business Analyst:** he is responsible of the business model during the *Determine Business Model* activity. He defines the business concept and the relationships between them. He also describes formally what are the business activities, what are the products provided or required and who are the responsible persons of these activities.

2.1.2 Activities Details

The flow of activities inside this phase is reported in Fig. 8 and are detailed in the following.

2.1.2.1 A01 : Define Users Requirements

This first activity concerns the description of the system and its environment in which it will be deployed. This activity consists in defining what to build or what is the most appropriate system for end-users. End-users and clients have to list, check and approve the requirements. The context in which the system will be deployed must be understood. The functional and non-functional requirements must be established. The flow of tasks inside this activity is reported in Fig. 10 and the tasks are detailed in the following table.

A01: Define Users Requirements		
Tasks	**Tasks Descriptions**	**Roles Involved**
S1: define users' requirements	End-users list the functional and non-functional requirements.	End-users
S2: validate users' requirements	The client has to check and approve the set of requirements. If this document is not validated, the requirements have to be improved; the previous step is made again.	Client

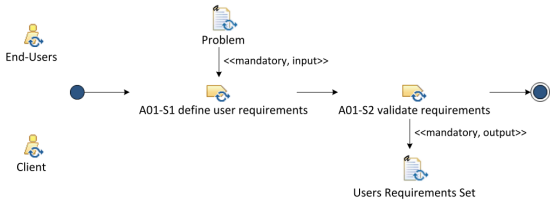


Fig. 10 Flow of Tasks of the *Define Users Requirements* Activity

2.1.2.2 A02: Define Consensual Requirements

This activity consists in defining what conditions or capabilities the system has to conform. The consensual requirements set is defined by the software analyst. The flow of tasks inside this activity is reported in Fig. 11 and the tasks are detailed in the following table.

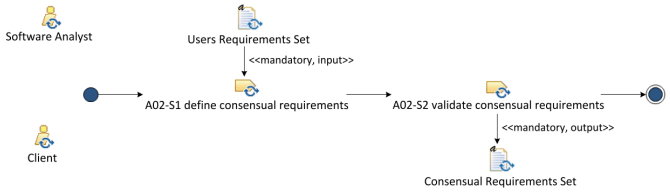


Fig. 11 Flow of Tasks of the *Define Consensual Requirements* Activity

A02: Define Consensual Requirements		
Tasks	**Tasks Descriptions**	**Roles Involved**
S1: define consensual requirements	The software analyst defines the requirements set with the consensual requirements.	Software Analyst
S2: validate consensual requirements	If there is no agreement on the Requirements Set document, a backtrack must be performed to study again the previous step	Client

2.1.2.3 A03: Determine Business Model

This activity provides an overview on the problem, the related concepts and the activities linked to. The flow of tasks inside this activity is reported in Fig. 12 and the tasks are detailed in the following table.

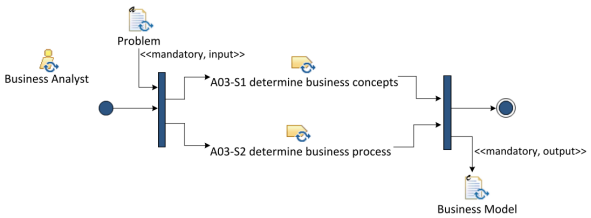


Fig. 12 Flow of Tasks of the *Determine Business Model* Activity

A03: Determine Business Model		
Tasks	**Tasks Descriptions**	**Roles Involved**
S1: determine business concepts	This step enables to understand the static and dynamic structure of the system.	Business Analyst
S2: determine business process	In this step, the sequence of actors' actions that achieve the goal of the system is determined.	Business Analyst

2.1.2.4 A04: Establish Keywords

The main concepts used to describe the application and its business model are listed. This activity, carried out by the software analyst, is composed of one task giving the definition of each keywords. These definitions will be stored in the glossary. The flow this activity is reported in Fig. 13.

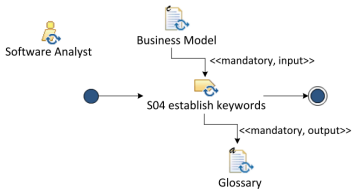


Fig. 13 Flow of Tasks of the *Establish Keywords* Activity

2.1.2.5 A05: Extract Limits and Constraints

In this activity, the limits and constraints of the system are defined by a software analyst. They can be found in the expression of non functional requirements and in the definition of the context in which the system will be deployed. This information will be defined mainly from the consensual requirements set documents. The flow of this activity is reported in Fig. 14.

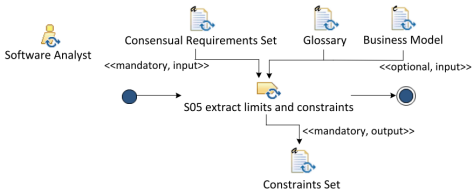


Fig. 14 Flow of Tasks of the *Extract Limits and Constraints* Activity

2.1.3 Work Products

The Preliminary Requirements Phase generates five work products (text document including textual description and/or diagrams). Their relationships with the MAS meta-model elements are described in the Fig. 15.

2.1.3.1 Work Products Kind

Name	Description	Work Product Kind
Users Requirements Set	Textual description of the functional and non-functional requirements	Free Text
Consensual Requirement Set	Textual description composed of consensual requirements	Free Text
Business Model	A document composed of: 1) a diagram modelling the domain-specific data structure; 2) a diagram showing the workflow of activities performed by the business actors	Composite (Structural and Behavioural)
Glossary	A glossary of terms	Free Text
Constraints Set	Textual description composed of the limits and constraints of the system	Free Text

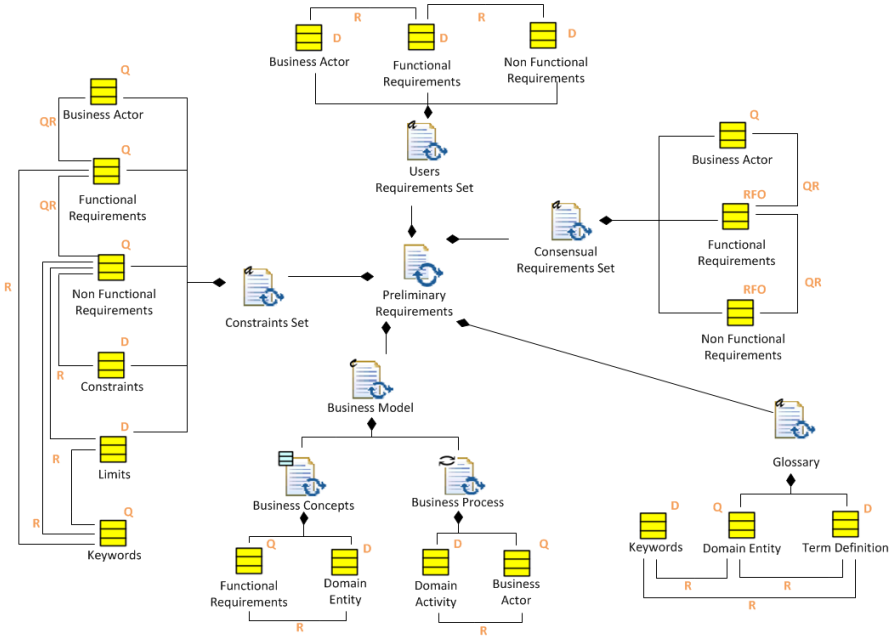


Fig. 15 The Preliminary Requirements Documents Structure

2.1.3.2 Example: Conference Management Study

The description of the system have been already detailed in the introductory chapter. As the requirements are common to all methods and because ADELFE is not really dedicated to preliminary requirements, only the business model is shown in this phase. Fig. 16 shows the business process on the left and the business concepts on the right.

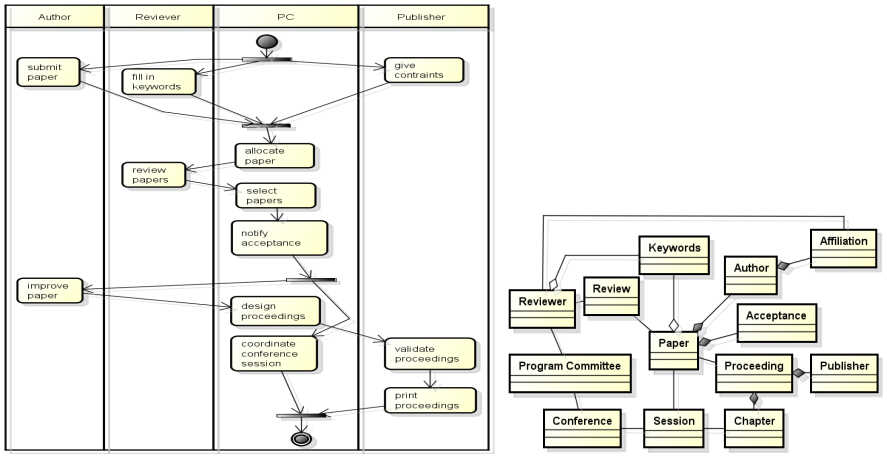


Fig. 16 Business Process (left); Business Concepts (right)

2.2 Final Requirements Phase (WD2)

The Final Requirements Phase is a classical requirement-oriented phase where the Business Analyst gives a detailed description of the system environment. It also embeds MAS-oriented tasks. The analysis, done by a MAS specialist, must add sufficient details to the description of the system environment in order to conclude if a MAS approach is needed to solve the problem with gains. The process flow at the level of activities is reported in Fig. 17 and Fig. 18 depicts this phase according to documents, roles and work products involved.



Fig. 17 The Final Requirements Phase Flow of Activities

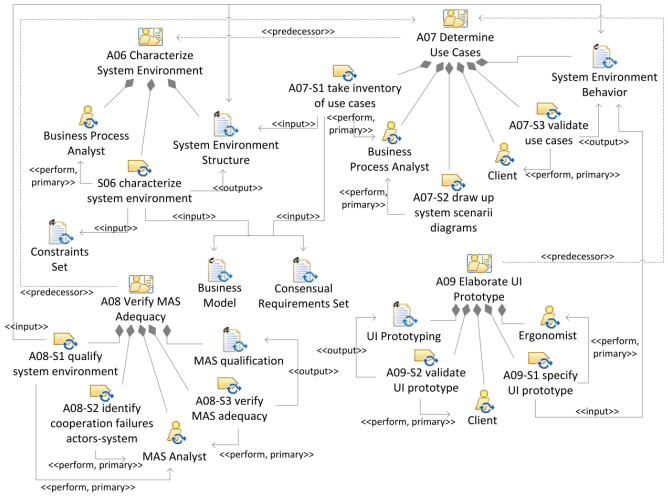


Fig. 18 The Preliminary Requirements Phase Described in terms of Activities and Work Products

2.2.1 Process Roles

Four roles are involved in the Final Requirements Phase: the Business Process Analyst, the Client, the MAS Analyst and the Ergonomist.

- **Business Process Analyst:** he is responsible of use cases identification during the *Determine Use Case* activity drawing diagrams that represent the interactions between actors and the system.

- **Client:** his main role is to validate product documents drawn up by others expert. He is responsible of approving the use cases defined during the *Determine Use Case* activity. Besides, he agrees the UI prototype during the *Elaborate UI Prototype* activity.

- **MAS Analyst:** he is responsible of verifying the MAS adequacy during the *Verify MAS Adequacy* activity. It consists in (1) the characterization of the system environment according to Russel and Norvig definition, (2) the identification of the possible ”bad” interactions between the actors and the system, (3) the analy-

sis of the previous results to justify the MAS use.

- **Ergonomist:** he is responsible of graphic user interfaces prototype during the *Elaborate UI Prototype* activity. He understands the interactions among humans and the system, and designs a prototype which optimizes human well-being and overall system performance

2.2.2 Activities Details

The flow of activities inside this phase is reported in Fig. 17 and is detailed in the following.

2.2.2.1 A06: Characterize System Environment

The main objective of this activity is to define the system environment in the system environment description document. This activity enables to identify and to describe briefly actors interacting with the system. The possible encountered constraints are also explained. The flow of tasks inside this activity is reported in Fig. 19.

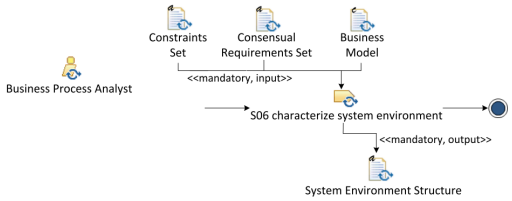


Fig. 19 Flow of Tasks of the *Characterize System Environment* Activity

2.2.2.2 A07: Determine Use Cases

The main objective of this activity is to clarify the different functionalities that the studied system must provide. The flow of tasks inside this activity is reported in Fig. 20 and the tasks are detailed in the following table.

A07: Determine Use Cases		
Tasks	**Tasks Descriptions**	**Roles Involved**
S1: take inventory of use cases	A set of steps defining interactions between an actor and a system are listed.	Business Process Analyst

S2: draw up system scenarii diagrams	This step explicits the system behaviour from users' point of view. The interactions between the actors and the system are drawn.	Business Process Analyst
S3: validate use cases	Approval of the System Environment Description document by the client. If the use cases have to be improved, the two previous steps have to be made again.	Client

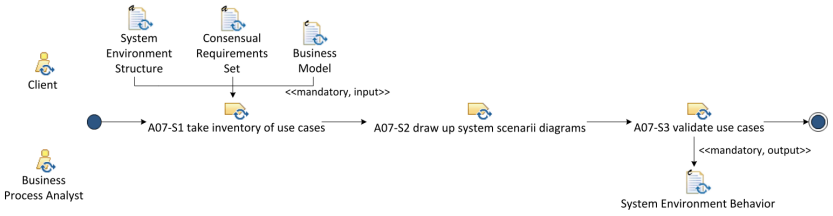


Fig. 20 Flow of Tasks of the *Determine Use Cases* Activity

2.2.2.3 A08: Verify MAS Adequacy

In this activity, one must verify that a Multi-Agent System (MAS) approach is needed to realize the system to be built. The question to answer is "a traditional (Object-Oriented) approach sufficient to solve the problem or has the problem some characteristics which implies MAS approach for the solving?". The flow of tasks inside this activity is reported in Fig. 21 and the tasks are detailed in the following table.

A08: Verify MAS Adequacy		
Tasks	**Tasks Descriptions**	**Roles Involved**
S1: qualify system environment	During this step, the MAS analyst characterises the system environment according to the Russel and Norvig definition.	MAS Analyst
S2: identify cooperation failures actors-system	The aim of this step is to show the inadequate interactions that may occur between the actors and the system.	MAS Analyst
S3: verify MAS adequacy	This step verifies the MAS adequacy by analyzing the results obtained during the two previous steps.	MAS Analyst

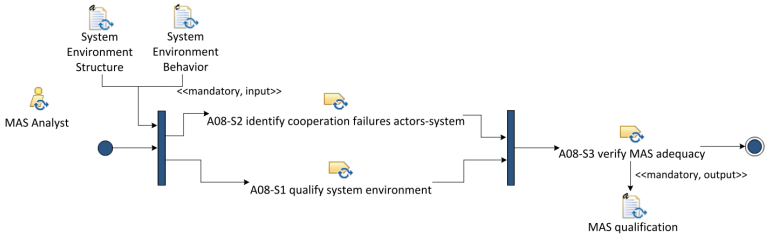


Fig. 21 Flow of Tasks of the *Verify MAS Adequacy* Activity

2.2.2.4 A09: Elaborate UI Prototypes

The GUIs described in the UI Prototype document have to be defined, judged and validated from functional or non functional (ergonomic, design...) points of view. The flow of tasks inside this activity is reported in Fig. 22 and the tasks are detailed in the following table.

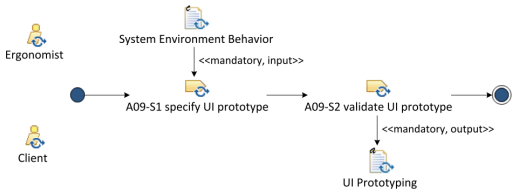


Fig. 22 Flow of Tasks of the *Elaborate UI Prototypes* Activity

A09: Elaborate UI Prototypes		
Tasks	**Tasks Descriptions**	**Roles Involved**
S1: specify UI prototypes	In this step, the interfaces supplying all needed functionalities are specified.	Ergonomist
S2: validate UI prototypes	The UI are used and assessed from functional and non-functional point of view.	Client

2.2.3 Work Products

The Final Requirements Phase generates four work products (text document including textual description and/or diagrams). Their relationships with the MAS metamodel elements are described in the Fig. 23.

2.2.3.1 Work Products Kind

Name	**Description**	**Work Product Kind**
System Environment Structure	A textual description describing the actors which interact with the system and the possible constraints. Moreover, this document contains a brief text of actors description.	Free Text

System Environment Behaviour	A document composed of: 1) a use case diagram representing actors and the functionalities assigned to them; 2) a structured text description of the actors; 3) diagrams representing the interactions between the actors and the system	Composite (Structural and Behavioural and Free text)
MAS Qualification	A text document composed by the description of the environment according the Russel and Norvig definition, the description of ”bad” interaction between actors and system and the justification of an implementation that using a MAS is needed	Free Text
UI Prototype	This document is composed by the GUIs description through which the user(s) interact with the system and the links between the GUIs.	Free Text

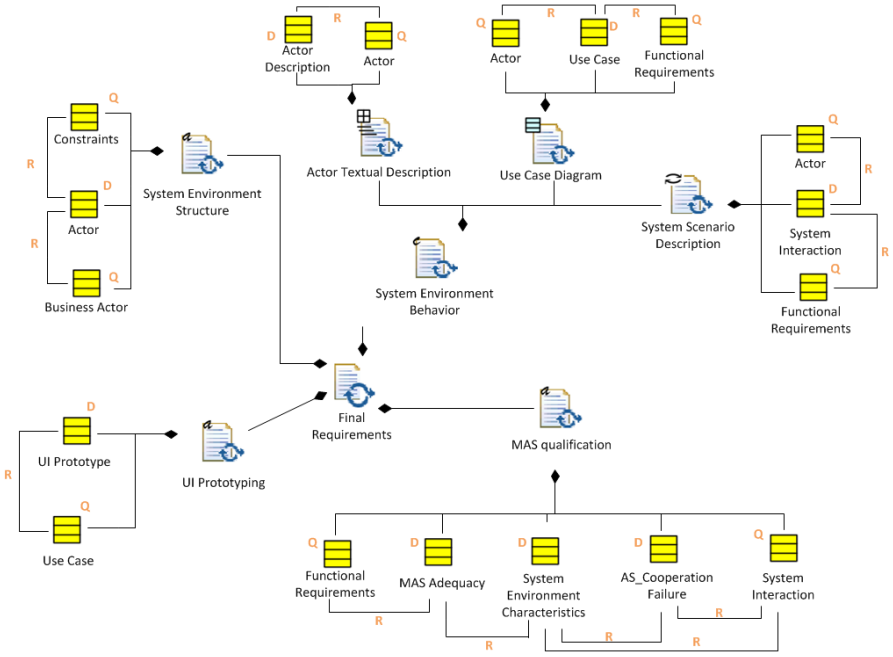


Fig. 23 The Final Requirements Documents Structure

2.2.3.2 Example: Conference Management Study

From the business model and the requirements previously established, four actors are defined. For each of them, the functionalities are detailed and depicted in a use cases diagram (see Fig.24).

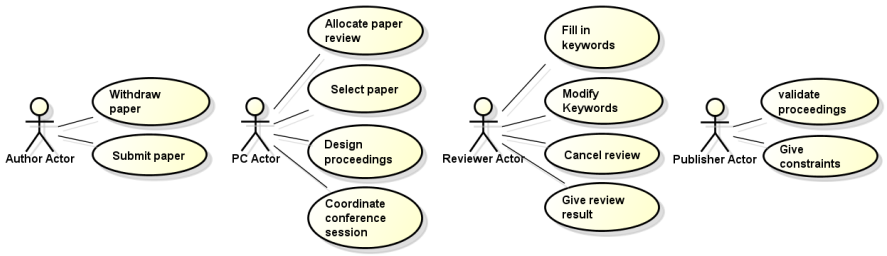


Fig. 24 Use Cases Diagram

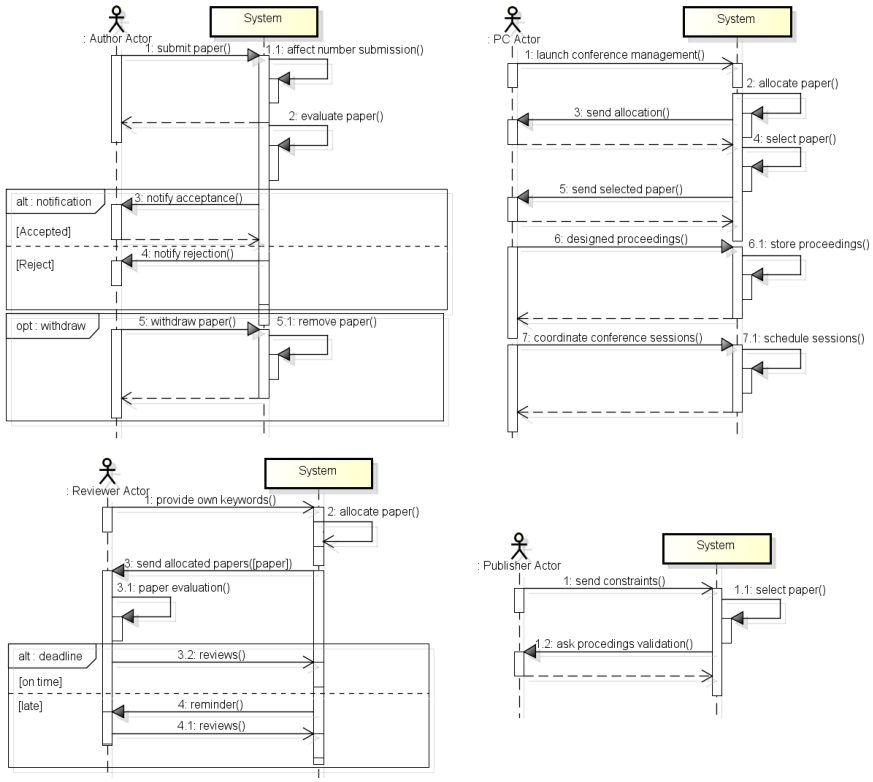


Fig. 25 System Sequence Diagrams

Moreover the interaction between the system and the actors are studied and shows in Fig. 25. From these diagrams, the following cooperative failure have been identified:

- at least one paper is not allocated
- not enough paper are accepted for the conference
- the chair committee disagrees with the paper allocation
- a reviewer does not have paper to review
- the reviewer disagrees with the paper allocation

Besides, according to Russel and Norvig 's definition, the environment of the system is described as

- *inaccessible* because knowing all about the environment (papers, keywords, ...) is difficult;
- *discrete* because the number of distinct percepts and actions is limited;
- *non-deterministic* because the actions have multiple unpredictable outcomes;
- *dynamic* because the state of the environment depends upon actions of the system that is within this environment.

2.3 Analysis Phase (WD3)

The Analysis phase aims at identifying the system structure and justifying the AMAS adequacy. This phase is composed of four activities enabling to analyse the domain characteristics, determine the agents and validate an AMAS approach at the global and local level. The process flow at the level of activities is reported in Fig. 26 and Fig. 27 depicts this phase according to documents, roles and work products involved.



Fig. 26 The Analysis Phase Flow of Activities

2.3.1 Process Roles

Two roles are involved in the Analysis Phase: the MAS Analyst and AMAS Analyst.

- **MAS Analyst:** he is responsible of detailing the MAS Environment in *Analysis Domain Characteristics* activity. It consists in (1) the identification of what are the entities which are active and the ones which are not (passive), (2) the identification of the interactions between the entities. He is also responsible in *Identify*

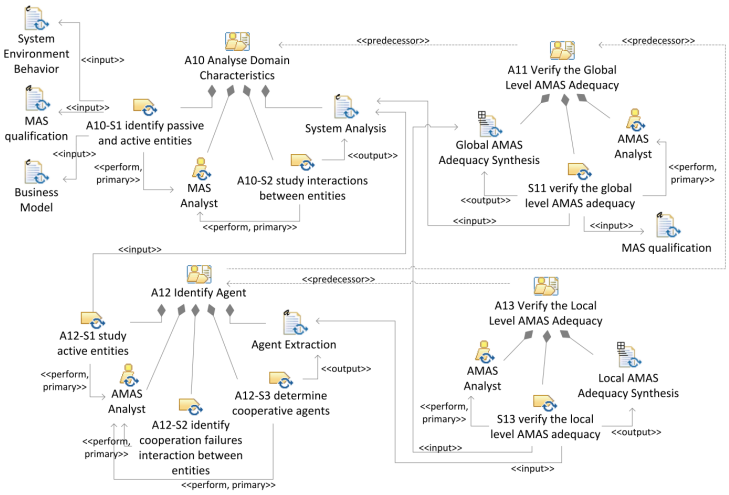


Fig. 27 The Analysis Phase Described in terms of Activities and Work P roducts

agent of the step which consists in defining autonomy, goal and negotiation abilities of active entities.

- **AMAS Analyst:** he is responsible of every activity dealing with the specificities of AMAS principles. They can be found in *Verify the global level AMAS adequacy* activity, in *Identify agent* activity and in *Verify the local level AMAS adequacy* activity. The identification of (cooperative) agents needs to determine the cooperation failures that can occur between entities and then to define the agents regarding the results of previous steps.

2.3.2 Activity Details

The flow of activities inside this phase is reported in Fig.26 and are detailed in the following.

2.3.2.1 A10 : Analyse Domain Characteristics

The main goal of this activity is to analyse the Business Domain and the System Environment Description in order to detail the entities of the domain and their interactions. The flow of tasks inside this activity is reported in Fig. 28 and the tasks are detailed in the following table.

A10: Analyse Domain Characteristics		
Tasks	**Tasks Descriptions**	**Roles Involved**
S1: identify passive and active entities	The MAS analyst splits the system into passive and active entities.	MAS Analyst
S2: study interactions between entities	This step shows the interactions between entities.	MAS Analyst

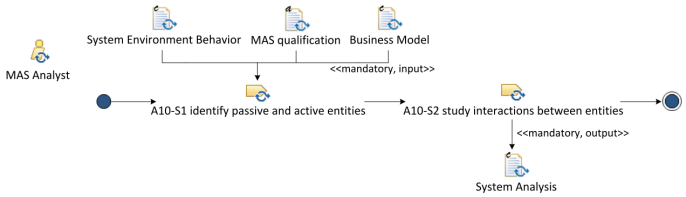


Fig. 28 Flow of Tasks of the *Analyse Domain Characteristics* Activity

2.3.2.2 A11: Verify the Global Level AMAS Adequacy

In this activity, the AMAS analyst must verify that an Adaptive Multi-Agent System (AMAS) approach is needed to realize the system to be built. For example, having a system which is able to adapt itself is sometimes completely useless if the algorithm required to solve the task is already known, if the task is not complex or if the system is closed and nothing unexpected can occur. In this activity, the adequacy at the global level is studied to answer the question ”is an AMAS required to implement the system?”. This is done throw several simple questions related the global level. The flow of this activity is reported in Fig. 29.

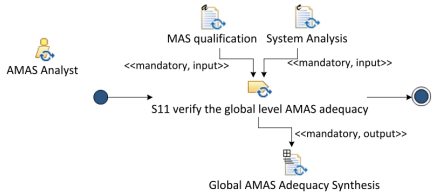


Fig. 29 Flow of Tasks of the *Verify the Global Level AMAS Adequacy* Activity

2.3.2.3 A12: Identify Agent

This activity aims at finding what will be considered as agents in the desired system. These agents are defined among the previously defined entities. The flow of tasks inside this activity is reported in Fig. 30 and the tasks are detailed in the following table.

A12: Identify Agent		
Tasks	**Tasks Descriptions**	**Roles Involved**
S1: study active entities	For each previously defined active entity, its autonomy, its goal and its negotiation abilities are studied.	MAS Analyst
S2: identify cooperation failures interaction between entities	During its interactions with other entities, an entity can encounter failures to respect the protocol or failures in the content of the interaction (misunderstanding...). This step extracts this kind of interactions.	AMAS Analyst
S3: determine cooperative agents	The entities pertaining to the previous step are considered as agents. In addition, the AMAS system diagram is drawn.	AMAS Analyst

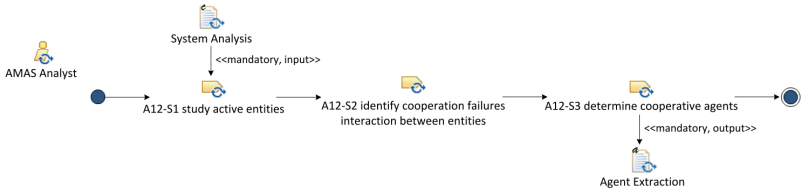


Fig. 30 Flow of Tasks of the *Identify Agent* Activity

2.3.2.4 A13: Verify the Local Level AMAS Adequacy

In this activity, the AMAS adequacy is studied at the local level in order to determine if some agents are needed to be implemented as AMAS i.e. if a certain kind of decomposition or recursion is required during the building of the system. The flow of this activity is reported in Fig. 31.

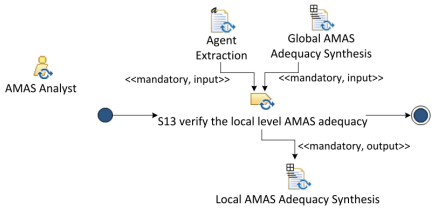


Fig. 31 Flow of Tasks of the *Verify the Local Level AMAS Adequacy* Activity

2.3.3 Work Products

The Analysis Phase generates four work products (text document including textual description and/or diagrams). Their relationships with the MAS meta-model elements are described in the Fig. 32.

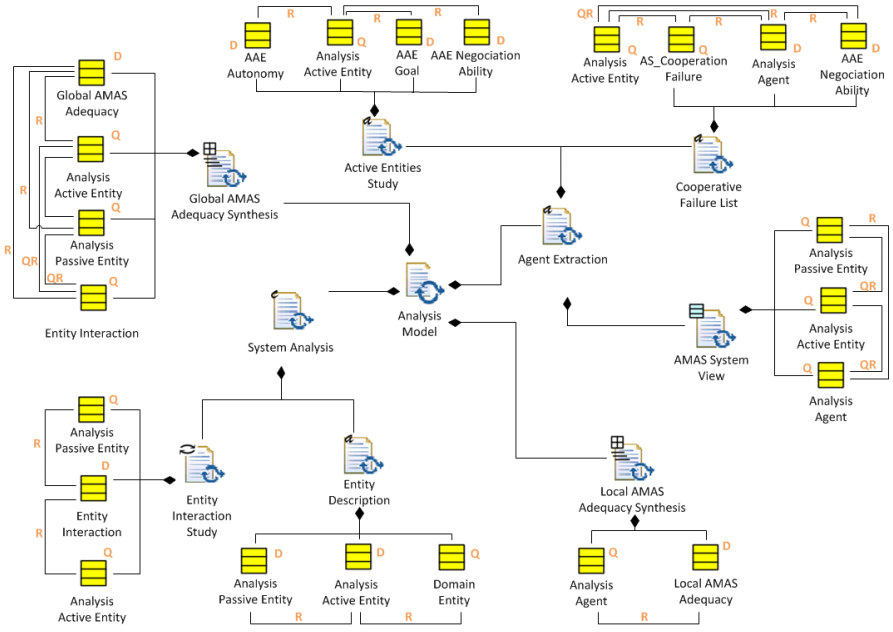


Fig. 32 The Analysis Documents Structure

2.3.3.1 Work Products Kind

Name	Description	Work Product Kind
System Analysis	A document composed of: 1) a textual description of the entities described as active or passive; 2) diagrams depicting the interactions between entities.	Composite (Free Text and Behavioural)
Global AMAS Adequacy Synthesis	This document stores the answers to the questions regarding the global level about an implementation using an AMAS.	Structured Text
Agent Extraction	This document supplements the System Analysis document with: 1) the definition of the goal, the study of autonomy and the negotiation abilities for each active entity; 2) the list of the cooperation failure interactions between entities or between entity and its environment; 3) the definition of the cooperative agent and the AMAS system diagram which represents them.	Composite (Free Text and Behavioural)
Local AMAS Adequacy Synthesis	This document completes the Global AMAS adequacy synthesis with the answers to the questions regarding the local level about an implementation using an AMAS.	Structured Text

2.3.3.2 Example: Conference Management Study

From the business concepts model, we define the active and passive entities. In our case, we define six active entities and the other concepts as passive entities. The entities system structure is depicted in Fig. 33. Broadly speaking, entities which are linked to users were considered as active because users may change their mind which implies a change in the state of the related entity. Moreover, papers and sessions are considered as active because they will need negotiation while finding a review or a session organization.

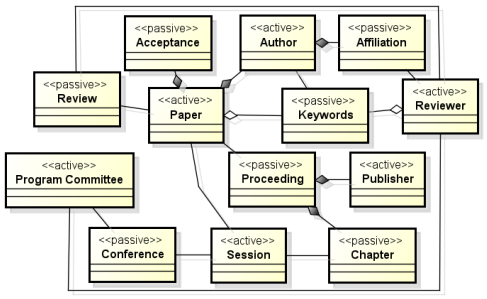


Fig. 33 Entities Structure System Diagram

The following step is the interactions study between entities shown Fig. 34 and 35. The started interactions and the other one shows the interaction after the paper notification are represented in Fig. 34.

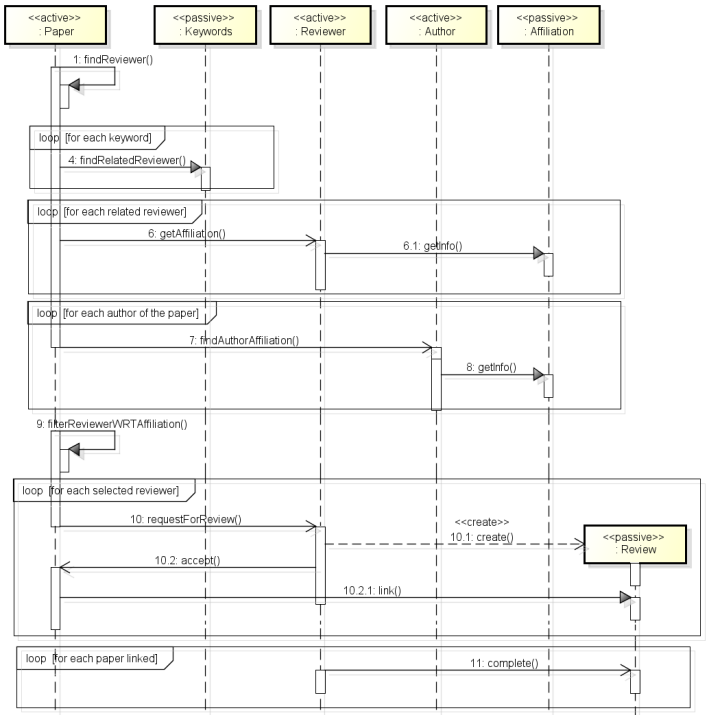


Fig. 34 Entities Interactions System Structure (beginning)

Verifying the AMAS adequacy consists in studying some specific features of AMAS with respect to the target application. The designer is provided with a tool which helps him to answer some questions. Here after are the 8 questions that are asked and some answers for our case study.

- *Is the global task incompletely specified? Is an algorithm a priori unknown?* YES: the CMS is precisely defined and some algorithms may be found to solve this kind of problem.
- *If several entities are required to solve the global task, do they need to act in a certain order?* YES: there are dependencies in the interactions needed between entities.
- *Is the solution generally obtained by repetitive tests, are different attempts required before finding a solution?* NO: no need for that.

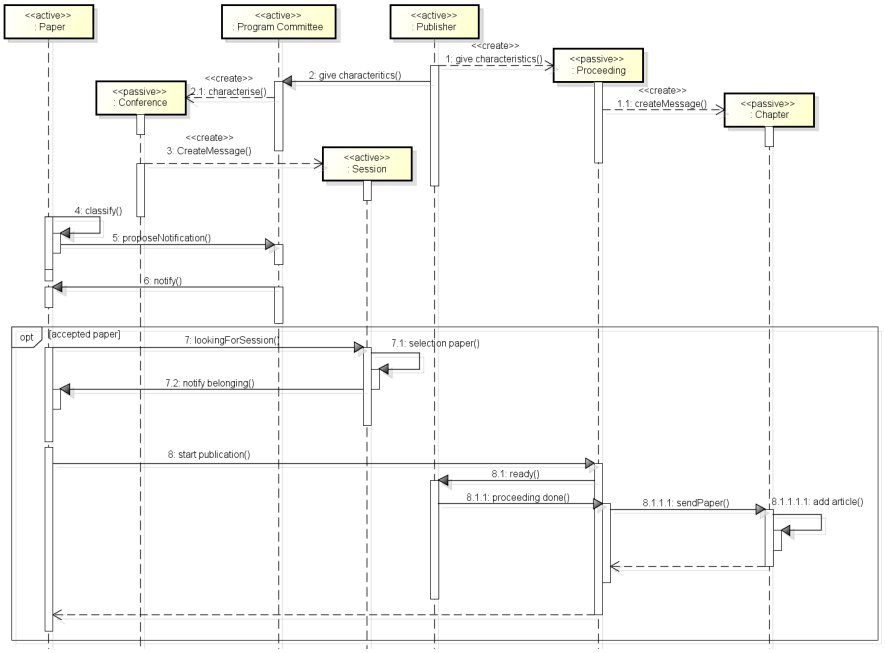


Fig. 35 Entities Interactions System Structure (ending)

- *Can the system environment evolve? Is it dynamic?* YES: it is highly dynamic. Authors, reviewers and papers may appear or disappear while a solution is calculated.
- *Is the system process functionally or physically distributed? Are several physically distributed entities needed to solve the global task? Or is a conceptual distribution needed?* YES/NO: people is physically distributed but there is no need for a distributed solving of the problem.
- *Does a great number of entities needed?* YES: depending on the conference but potentially, in conferences like AAMAS, there are a great number of entities.
- *Is the studied system non-linear?* RATHER YES: in the nominal case, it can be rather easy to find a linear decomposition of the problem, but with the openess described above, the great number of interactions may lead to a complex system.
- *Is the system evolutionary or open? Can new entities appear or disappear dynamically?* YES: it is highly dynamic. Authors, reviewers and papers may appear or disappear while a solution is calculated.

In the ”Identify Agents” activity, the active entities previously defined are studied. The following table depicts the autonomy, the goal and the negotiation abilities for each of them.

Active Entities	Autonomy	Goal	Negotiation Abilities
Paper	Take its own decision	Know its result: acceptance or rejection	Talk with other papers to choose reviewer and find its position in the acceptance
Reviewer	Act according the paper and users	Review papers	
Program Committee	Act according the users	Validate	
Publisher	Act according the users	Give constraints and proceedings	
Author	Act according the users	Deal with a paper	
Session	Select itself the right papers to insert for the conference management	Select paper to be full	Discuss with the others sessions to select papers

From the interactions between entities which have been previously identified, three other cooperative failure interactions are identified: (i) a paper does not find a review; (ii) several papers want rise to the same rank; (iii) sessions select the same paper. Two kinds of agent are therefore deduced: the paper agent and the session agent. The AMAS system is therefore represented in Fig 36.

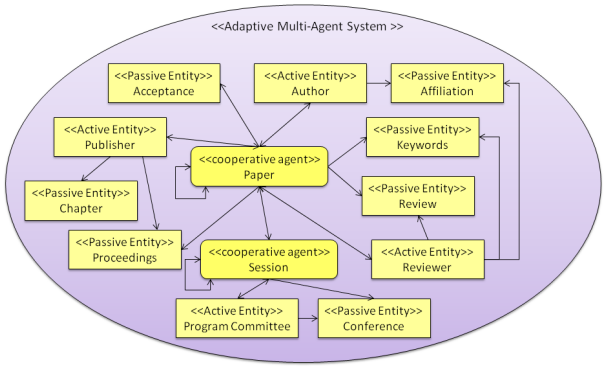


Fig. 36 AMAS System Diagram

2.4 Design Phase (WD4)

The Design Phase aims at providing a detailed architecture of the system. During this phase, the definition of a module view is defined, the communication acts are studied and the different behaviours are determined. The process flow at the level

of activities is reported in Fig. 37 and Fig. 38 depicts this phase according to documents, roles and work products involved.

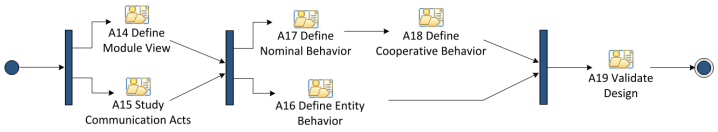


Fig. 37 The Design Phase Flow of Activities

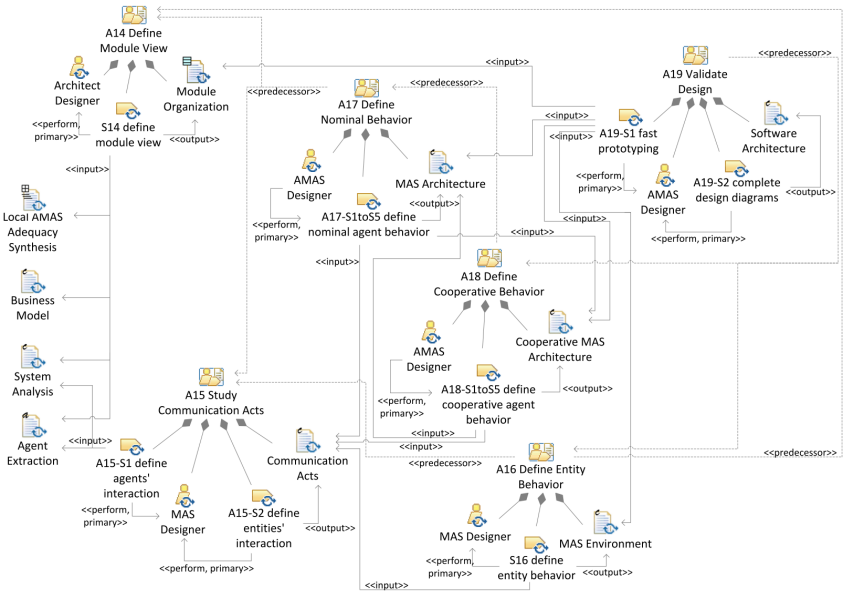


Fig. 38 The Design Phase Described in terms of Activities and Work Products

2.4.1 Process Roles

Three roles are involved in the Design Phase: the architectural designer, the MAS designer and the AMAS designer.

- **Architectural Designer:** he is responsible of module organization during the *Define Module View* activity. He defines the detailed architecture of the system in terms of modules.

- **MAS Designer:** he is responsible of communication acts during the *Study Communication Acts* activity and the definition of the entities behaviour during the *Define Entity Behaviour* activity. He defines how the entities and the agents interact together or with their own environment.

- **AMAS Designer:** he is responsible of nominal behaviour of agents during the *Define Nominal Behaviour* activity, cooperative behaviour of agents in the *Define Cooperative Behaviour* activity and fast prototyping during the *Validate Design Phase* activity. Indeed, from the structure analysis and the communication acts previously detailed, He defines skills, aptitudes, an interaction language, a world representation, a criticality and the characteristics of an agent. He fulfils the agent behaviour by adding a cooperative attitude i.e. giving rules which enable to anticipate or detect and repair the non cooperative situations. For that, skills, aptitudes, an interaction language, a world representation, a criticality and the characteristics are filled out. Finally, he tests the behaviour of agents ie. the protocols, the methods and the general behaviour of agents.

2.4.2 Activity Details

The flow of activities inside this phase is reported in Fig.37 and are detailed in the following.

2.4.2.1 A14 : Define Module View

This activity shows how the architectural designer maps the key elements of the software to the modules. Their organisation and dependencies are defined. The following kinds of dependencies can be used: *use , allowToUse, include/decompose, CrossCut, EnvModel*). The flow of this activity is reported in Fig. 39 .

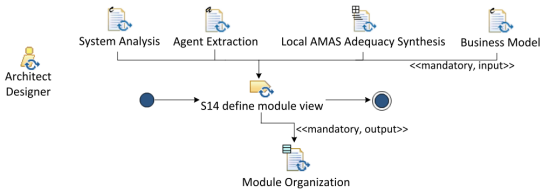


Fig. 39 Flow of Tasks of the *Define Module View* Activity

2.4.2.2 A15: Study Communication Acts

This activity aims at making clear interactions between the entities and/or the agents previously identified. The flow of tasks inside this activity is reported in Fig. 40 and the tasks are detailed in the following table.

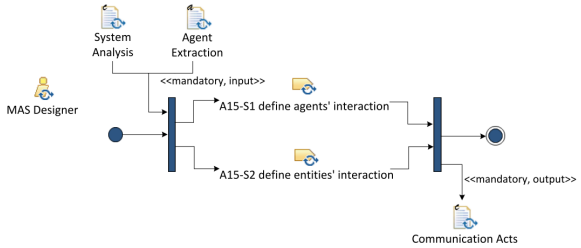


Fig. 40 Flow of Tasks of the *Study Communication Acts* Activity

A15: Study Communication Acts		
Tasks	**Tasks Descriptions**	**Roles Involved**
S1: define agents interaction	This step consists in defining the way in which an agent is going to interact with the others and its environment.	MAS Designer
S2: define entities interaction	This step consists in defining the way in which an entity is going to interact with others entities.	MAS Designer

2.4.2.3 A16: Define Entity Behavior

The aim of this activity is to define the entity behaviour. It can be illustrated by an inner state related to its current role. This activity is performed by MAS analyst. The flow of this activity is reported in Fig. 41.

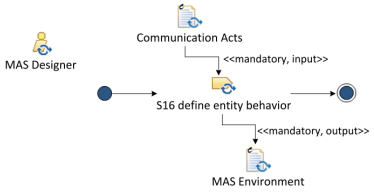


Fig. 41 Flow of Tasks of the *Define Entity Behaviour* Activity

2.4.2.4 A17: Define Nominal Behavior

The purpose of this activity is to define the nominal behaviour. The AMAS designer has to define skills, aptitudes, an interaction language, a world representation, a criticality which compose the nominal behaviour. Agents may also have physical characteristics such as weight, colour... which may be necessarily found during this activity. The structural diagrams of agent are drawn and the structural rules are described. In addition, an agent can be defined by an inner state related to its current role in the MAS organization. The flow of tasks inside this activity is reported in Fig. 42 and the tasks are detailed in the following table.

A17: Define Nominal Bhavior		
Tasks	**Tasks Descriptions**	**Roles Involved**
S1: define its skills	The knowledge about a domain allowing the agent to execute actions are defined.	AMAS Designer
S2: define its aptitudes	The aim of this activity is to determine the capabilities of an agent to reason on its knowledge about the domain or on its representation of the world.	AMAS Designer
S3: define its interaction language	This step consists in defining the way in which agents are going to interact. Actually, if agents interact to communicate, information exchanges between agents are described. Technically, these protocols are specified through protocol diagrams.	AMAS Designer
S4: define its world representation	The AMAS designer gives the way to describe the representations of an agent about others agents, itself and its environment.	AMAS Designer
S5: define criticality and confidence of agent behavior	This step determines the relative difficulty of agents in its neighbourhood and its internal measure that provides information on the reliability of the decision on actions intended.	AMAS Designer

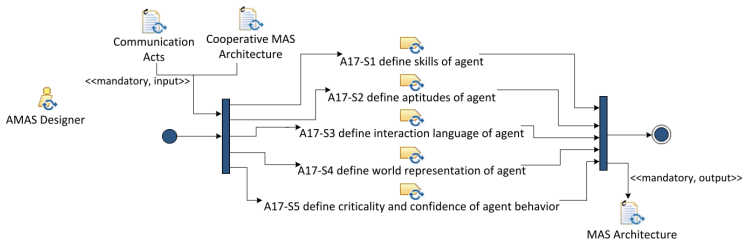


Fig. 42 Flow of Tasks of the *Define Nominal Behavior* Activity

2.4.2.5 A18: Define Cooperative Behavior

This activity is a key step. Indeed, the AMAS designer defines the cooperative behaviour by the allocation of cooperation rules. These rules enable an agent to have a cooperative attitude i.e. anticipate or detect and repair the non cooperative situations. During this activity, the structural diagram is completed by appropriated skills, representations, the attitudes or any other agent characteristic. The flow of tasks inside this activity is reported in Fig. 43 and the tasks are detailed in the following table.

A18: Define Cooperative Behaviour		
Tasks	**Tasks Descriptions**	**Roles Involved**
S1: define its skills	The knowledge about a domain allowing the agent to execute actions are defined.	AMAS Designer
S2: define its aptitudes	The aim of this activity is to determine the capabilities of an agent to reason on its knowledge about the domain or on its representation of the world.	AMAS Designer
S3: define its interaction language	This step consists in defining the way in which agents are going to interact. Actually, if agents interact to communicate,information exchanges between agents are described. Technically, these protocols are specified through protocol diagrams.	AMAS Designer
S4: define its world representation	The AMAS designer gives the way to describe the representations of an agent about others agents, itself and its environment.	AMAS Designer
S5: define criticality and confidence of agent behaviour	This step determines the relative difficulty of agents in its neighbourhood and its internal measure that provides information on the reliability of the decision on actions intended.	AMAS Designer

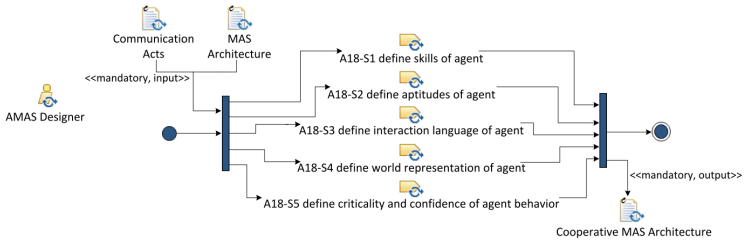


Fig. 43 Flow of Tasks of the *Define Cooperative Behavior* Activity

2.4.2.6 A19: Validate Design Phase

During this activity, the AMAS designer may test the behaviour of the agents. This test can lead to improve the agent's behaviour if it is not adequate. The flow of tasks

inside this activity is reported in Fig. 44 and the tasks are detailed in the following table.

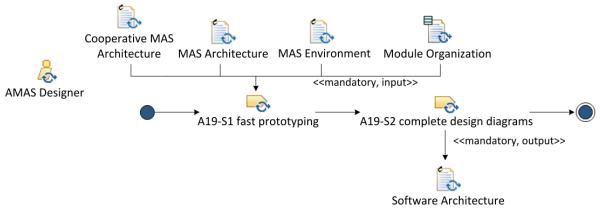


Fig. 44 Flow of Tasks of the *Validate Design Phase* Activity

A19: Validate Design Phase		
Tasks	**Tasks Descriptions**	**Roles Involved**
S1: fast prototyping	During this step, the agents' behaviour are tested. The prototype has to point out the possible lacks of an agent behaviour and of cooperative attitude.	AMAS Designer
S2: complete design diagrams	The aim of this step is to finalize the module organization and finish the design phase.	AMAS Designer

2.4.3 Work Products

The Design Phase generates six work products. Their relationships with the MAS meta-model elements are described in the Fig. 45.

2.4.3.1 Work Products Kind

Name	**Description**	**Work Product Kind**
Module Organization	This document depicts the organization and the dependencies of the key elements of the software.	Structural
Communication Acts	This document is composed of the specific textual description of the entity interactions and the agent interactions and the precise diagrams depicting this.	Composite (Free and Behavioural)
MAS Environment	This document contains the description of the entities behaviour. It is illustrated with inner state related to their current role.	Composite (Free and Behavioural)

MAS Architecture	This document is composed of the agent nominal behaviour description, illustrated with inner state related to their current role and depicted by structural diagram of agent. Skills, aptitudes, an interaction language, a world representation and a criticality define an cooperative agent behaviour. Moreover, it contains the physical characteristics of the agent and its structural rules.	Composite (Free and Structural and Behavioural)
Cooperative MAS Architecture	This document contains the elements of a cooperative agent behaviour enabling to anticipate or detect and repair the non cooperative situations. A cooperative agent behaviour is composed of skills, aptitudes, an interaction language, a world representation and a criticality.	Composite (Free and Structural and Behavioural)
Software Architecture	This document is composed of the fast prototyping of the agent behaviour and the refinement of the Software architecture (entities), Software architecture (nominal) and Software architecture (cooperative) document.	Composite (Free and Structural and Behavioural)

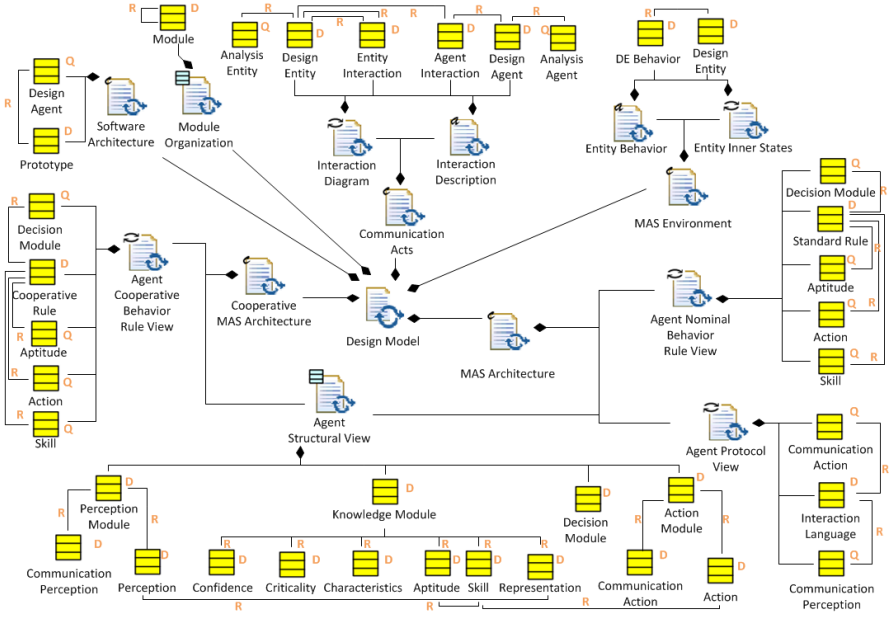


Fig. 45 The Design Documents Structure

2.4.3.2 Example: Conference Management Study

In this section where the phase settles the design of the software architecture of each agent and entity, we only describe the paper agent behaviour. Actually, its skills, its aptitudes, its interaction language and its representations is depicted in Fig.46 on the left part. The right part of Fig.46 represents an inner state related to the current state of the paper agent. The nominal behaviour of a paper agent starts in a *Submitted* state which corresponds to the creation of a paper agent. In this state, the paper agent is looking for reviews. It is in *IsReviewing* state when it finds all reviews it needs. It becomes *Reviewed* when all its reviews are complete. In order to reach the following state, *Accepted* or *Rejected*, the paper agent self-evaluates the reviews' results and changes its state. If the paper agent is *Accepted*, it informs the session of its state and become *IsImproving*. Finally, the paper agent is *Printed* when the proceedings are published.

Moreover, this cooperative agent can meet some non cooperative situations, for a paper agent, two situations are detected: (i) a review can be linked to a limited number of papers, it can happen that several paper agents want the same review ; (ii) when the paper agents have to deal with the acceptance, the paper agents can be put in concurrence or competition.

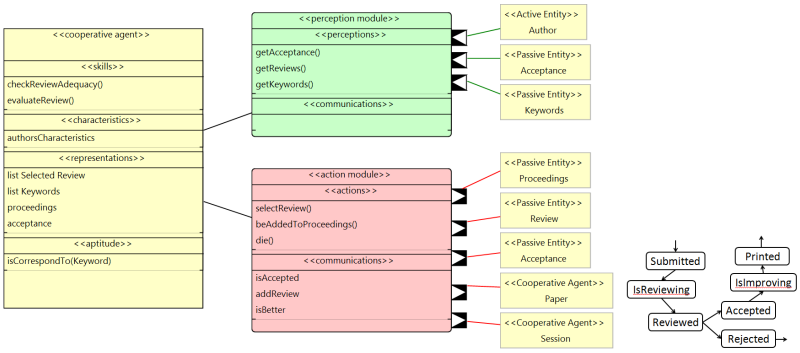


Fig. 46 Paper Agent Software Architecture: (left) Structural Diagram and (right) Inner State Related to its Behaviour

2.5 Implementation Phase (WD5)

The Implementation Phase aims at providing the wished system. Actually, the aspects of the detailed architecture are firstly described using SpeADL, then implemented using the Java programming language by relying on code generated from the ADL, and finally executed to give the wished system. The process flow at the

level of activities is reported in Fig. 47 and Fig. 48 depicts this phase according to documents, roles and work products involved.



Fig. 47 The Implementation Phase Flow of Activities

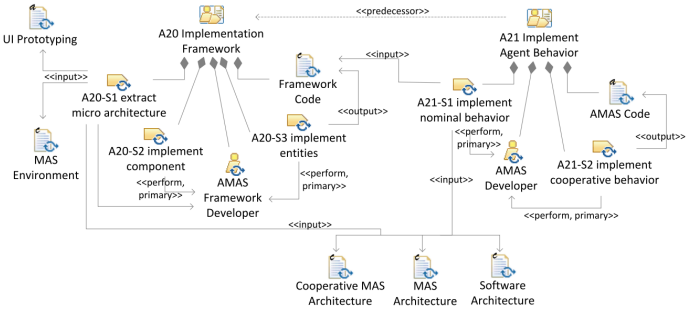


Fig. 48 The Implementation Phase Described in terms of Activities and Work Products

2.5.1 Process Roles

Two roles are involved in the Implementation Phase: the AMAS Framework Developer, and the AMAS Developer.

- **AMAS Framework Developer:** he is responsible of the description of the system architecture in the SpeAD (Species-based Architecture Description) model language during the *Implement Framework* activity and the implementation of everything that is not an agent. Actually, the AMAS framework developer implement the passive entities, the active entities and all programs required by the system such as a scheduler.

- **AMAS Developer:** he is responsible of the agent behaviour implementation during the Implement Agent Behaviour activity. He implements the nominal and cooperative behaviour according to the designed software architecture.

2.5.2 Activities Details

The flow of activities inside this phase is reported in Fig.47 and the tasks are detailed in the following table.

2.5.2.1 A20: Implement Framework

During this activity, the mechanisms are software components with provided and required services that can be composed together to form the architectures of the system. Entities and agents' architecture are therefore described in terms of components. The architecture is described using the textual architecture description language SpeADL (Species-based Architecture Description Language). Then the architectural elements which are not a cooperative agent are implemented. The flow of tasks inside this activity is reported in Fig. 49 and the tasks are detailed in the following table.

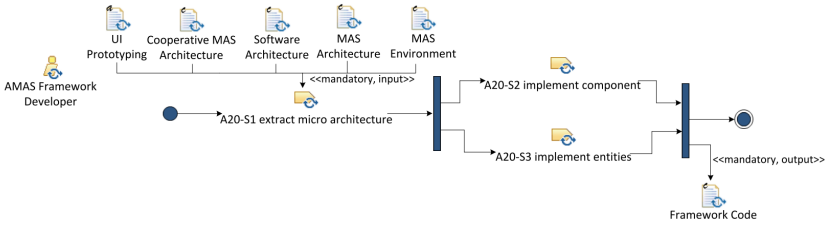


Fig. 49 Flow of Tasks of the *Implement Framework* Activity

A20: Implement Framework		
Tasks	**Tasks Descriptions**	**Roles Involved**
S1: extract micro architecture	The previously defined software architecture is translated in SpeAD model language.	AMAS Framework Developer
S2: implement component	The AMAS framework developer implements everything that is not related to the agent or entity behaviour in the system.	AMAS Framework Developer
S3: implement entities	The AMAS framework developer implements the active and passive entities behaviour.	AMAS Framework Developer

2.5.2.2 A21: Implement Agent Behaviour

During this activity, the behaviour of the cooperative agent are implemented. The flow of tasks inside this activity is reported in Fig. 50 and the tasks are detailed in the following table.

A21: Implement Agent Behaviour		
Tasks	**Tasks Descriptions**	**Roles Involved**
S1: implement nominal behaviour	The nominal behaviour of agents is implemented by working out the agents' process decision.	AMAS Developer
S2: implement cooperative behaviour	The cooperative behaviour of agents are implemented by working out the agents' process decision which enables to anticipate or detect and repair the non cooperative situations.	AMAS Developer

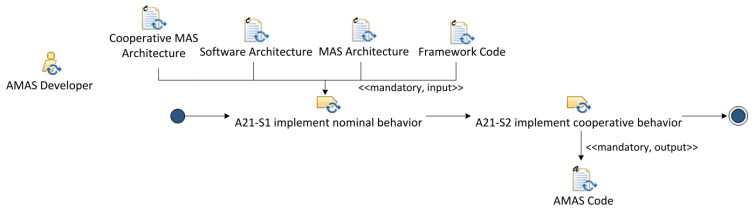


Fig. 50 Flow of Tasks of the *Implement Agent Behaviour* Activity

2.5.3 Work Products

The Implementation Phase generates two work products. Their relationships with the MAS meta-model elements are described in the Fig. 51.

2.5.3.1 Work Products Kind

Name	**Description**	**Work Product Kind**
Framework Code	This document is composed of: 1)a textual description of the architecture of the system, according the SpeADL language; 2) the implementation of all what is not agent.	Composite (Structured Text and Free Text)
AMAS code	This document is composed of the implementation of the cooperative agent behaviour (nominal behaviour and cooperative behaviour).	Composite (Structured Text and Free Text)

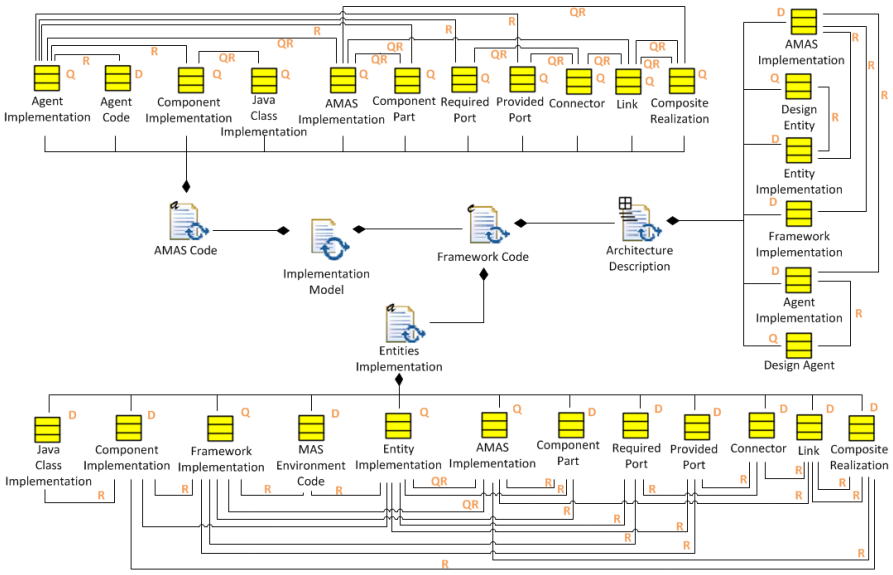


Fig. 51 The Implementation Documents Structure

2.5.3.2 Example: Conference Management Study

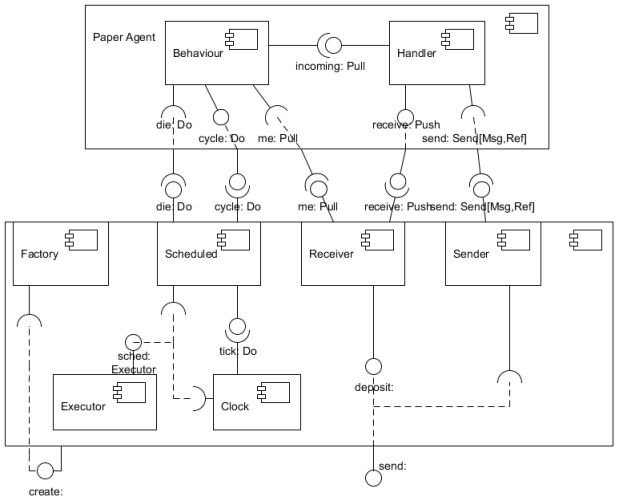


Fig. 52 Architectural Description of Paper Agent

Fig. 52 is a graphic description of a SpeADL. This architecture defined with SpeAD is made of components connected together with simple connectors. The components externally provide ports, for which they have an implementation, and require ports, that they can use in their implementation. Note that the description of components made with SpeADL will be then translated to Java.

3 Work Product Dependencies

Fig. 53 describes the dependencies among the different work products produced by the process. A dashed arrow is used to relate two of them if one is an input document to the other. Its direction points from the input document to the consumer one.

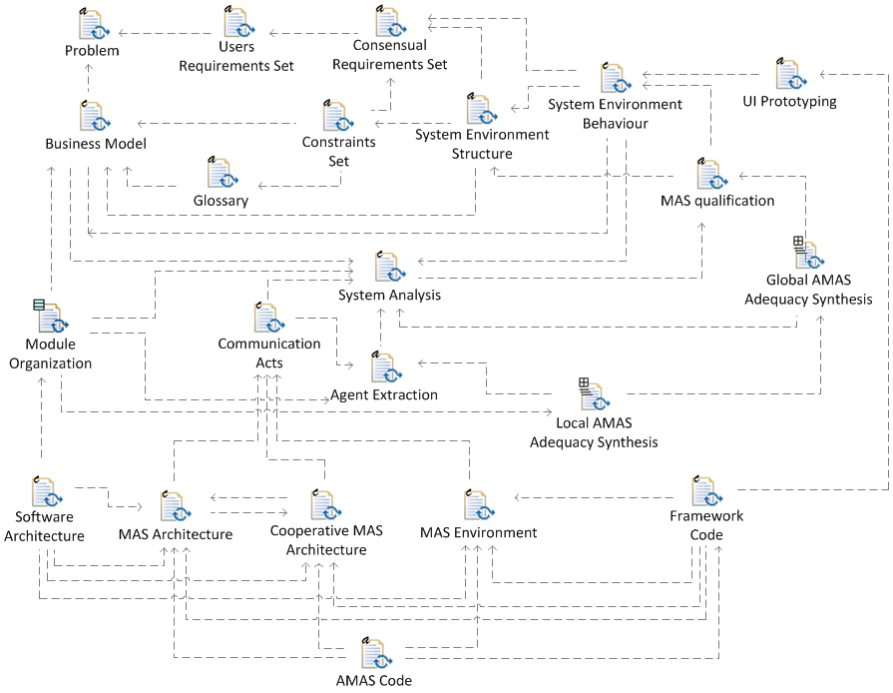


Fig. 53 The Work Products Dependencies