The ASPECS process

Massimo Cossentino, Vincent Hilaire, Nicolas Gaud, Stephane Galland, and Abderrafiaa Koukam

Abstract This chapter introduces an agent-oriented software process for engineering complex systems called ASPECS. ASPECS is based on a holonic organizational metamodel and provides a step-by-step guide from requirements to code allowing the modeling of a system with different levels of details using a set of refinement methods. This chapter introduces the ASPECS process using the documentation template provided by the IEEE FIPA DPDF Working Group. One of the founding principles of ASPECS is to combine both holonic structures and organizational approaches to ease the modeling and development of complex software applications. The target scope for the proposed approach can be found in complex systems and especially hierarchical complex systems. ASPECS is mainly suitable for open large-scale MAS. The main vocation of ASPECS is towards the development of holonic (as well as not-holonic) societies of software agents.

1 Introduction

ASPECS is a step-by-step requirement to code software process for engineering Complex Systems using Multiagent Systems and Holonic Multiagent Systems [5]. To deal with all aspects of complex systems, multiagent systems must deal with

Massimo Cossentino
ICAR Institute, National Research Council,
Palermo, Italy.
`http://www.pa.icar.cnr.it/cossentino`
e-mail: `cossentino@pa.icar.cnr.it`

Vincent Hilaire and Nicolas Gaud and Stephane Galland and Abderrafiaa Koukam
IRTES-SET, UTBM, UPR EA 7274,
90 010 Belfort cedex, France
`http://www.multiagent.fr`
e-mail: `vincent.hilaire@utbm.fr`

multiple levels of abstractions and openness. These multiple levels are frequently not taken into account for most solutions [12].

The presence of multiple levels is emphasized in many works, such as Simon [15], who postulates that complex systems often (if not always) exhibit a hierarchical configuration[1]. The idea is that the architecture of a complex system can be explained and understood using hierarchical organization structures as presented in [16]. Several metamodels and methodologies have been proposed for Multiagent systems [1]. However, most of them consider agents as atomic entities. There is no intuitive or natural way to deal with hierarchical organization structures. Considering agents as composed entities thus enables the modeling of nested hierarchies and proposes a solution to this problem.

Based on this statement, ASPECS exploits the concepts of holons: agents who may be composed of agents. ASPECS combines both holonic and organizational perspectives within a single metamodel allowing the modeling of a system at different levels of details using a set of refinement methods.

The authors of ASPECS tried to gather the advantages of organizational approaches as well as those of the holonic vision in modeling complex systems. The result is a set of organization-oriented abstractions that have been integrated into a complete methodological process. The target scope for the proposed approach can be found in complex systems and especially hierarchical complex systems. The main vocation of ASPECS is towards the development of societies of holonic (as well as not-holonic) multiagent systems. The interest reader could find informations about ASPECS on the ASPECS website http://www.aspecs.org and in [5, 4, 3]. A specific deployment platform has been developed [8] and may be accessed through the website http://www.janus-project.org.

1.1 Global process overview (Lifecycle)

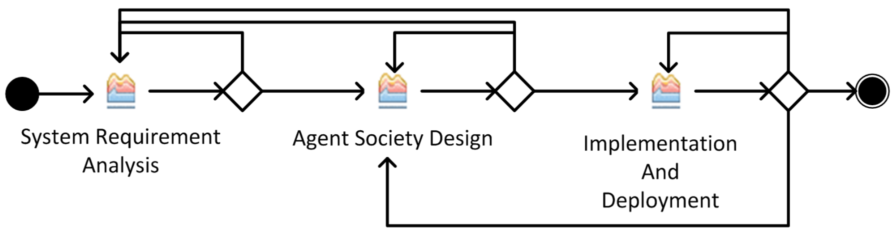


Fig. 1 The ASPECS process phases (and iterations)

The ASPECS life cycle consists of three phases organized in an iterative incremental process (see Figure 1). Each phase is briefly described below.

[1] Hierarchical here is meant as a “loose” hierarchy as presented by Simon.

The **System Requirements** phase aims at identifying a hierarchy of organizations, whose global behavior may fulfill the system requirements under the chosen perspective. It starts with a Domain Requirements Description activity where requirements are identified by using classical techniques such as use cases. Domain knowledge and vocabulary associated to the problem domain are then collected and explicitly described in the Problem Ontology Description activity. Then, requirements are associated to newly defined organizations. Each organization will therefore be responsible for exhibiting a behavior that fulfills the requirements it is responsible for. This activity is called Organization Identification, and it produces an initial hierarchy of organizations that will later be extended and updated, with further iterations, in order to obtain the global organization hierarchy representing the system structure and behavior. The behavior of each organization is realized by a set of interacting roles whose goals consist in contributing to the fulfillment of (a part of) the requirements of the organization within which they are defined. In order to design modular and reusable organization models, roles are specified without making any assumptions about the structure of the agent who may play them. To meet this objective, the concept of capacity has been introduced. A capacity is an abstract description of a know-how, i.e. a competence of a role. Each role requires certain skills to define its behavior, and these skills are modeled by means of a capacity. Besides, an entity that wants to play a role has to be able to provide a concrete realization for all the capacities required by the role. Finally, the last step of the system requirements phase: the capacity identification activity, aims at determining the capacities required by each role.

The second phase is the **Agent Society Design** phase that aims at designing a society of agents whose global behavior can provide an effective solution to the problem described in the previous phase and to satisfy associated requirements. The objective is to provide a model in terms of social interactions and dependencies between entities (holons and agents). Previously identified elements such as ontology, roles and interactions, are now refined from the social point of view (interactions, dependencies, constraints, etc.). At the end of this design phase, the hierarchical organization structure is mapped into a holarchy (hierarchy of holons) in charge of realizing the expected behaviors. Each of the previously identified organizations is instantiated in form of groups. Corresponding roles are then associated to holons or agents. This last activity also aims at describing the various rules that govern the decision-making process performed inside composed holons as well as the holons' dynamics in the system (creation of a new holon, recruitment of members, etc.). All of these elements are finally merged to obtain the complete set of holons involved in the solution.

The third and last phase, namely **Implementation and Deployment**, firstly, aims at implementing the agent-oriented solution designed in the previous phase by deploying it to the chosen implementation platform, in our case, JANUS. Secondly, it aims at detailing how to deploy the application over various computational nodes (JANUS kernels in our experiments). Based on JANUS, the implementation phase details activities that allow the description of the solution architecture and the production of associated source code and tests. It also deals with the solution reusability by encouraging the adoption of patterns. The code reuse activity aims at integrating the code of these patterns and adapting the source code of previous applications inside the new one. It is worth to note that although we will refer to a JANUS-based implementation, systems developed by using other platforms can be designed as well with the described process. This phase ends with the description of the deployment configuration; it also details how the previously developed application will be concretely deployed; this includes studying distribution aspects, holons physical location(s) and their relationships with external devices and resources. This activity also describes how to perform the integration of parts of the application that have been designed and developed by using other modeling approaches (i.e. object-oriented ones) with parts designed with ASPECS.

1.2 Metamodel

ASPECS has been built by adopting the Model Driven Architecture (MDA) [11], and thus we defined three levels of models each referring to a different metamodel. We also label the three metamodels "domains" thus maintaining the link with the PASSI metamodel that was one of our inspiration sources. The three domains we define are:

Problem Domain. It provides the organizational description of the problem independently of a specific solution. The concepts introduced in this domain are mainly used during the analysis phase and at the beginning of the design phase.

Agency Domain. It introduces agent-related concepts and provides a description of the holonic, multiagent solution resulting from a refinement of the Problem Domain elements.

Solution Domain. It is related to the implementation of the solution on a specific platform. This domain is thus dependent on a particular implementation and deployment platform. In our case, this part of the process is based upon the JANUS platform that we specifically designed to ease the implementation of holonic and organizational models. A complete description of the JANUS platform would take too much space to be dealt by this paper, and therefore, we prefer to present only the most significant JANUS issues. The interested reader can find more details in [8] and on the JANUS website[2].

The following sub-sections detail the three domain metamodels and some fundamental concepts within them. A complete description of all the elements reported in the metamodels is present on the ASPECS website and will not be reported here because of space concerns.

[2] JANUS: `http://www.janus-project.org/`

1.2.1 Problem Domain

The Problem Domain metamodel (see Fig. 2) includes elements that are used to catch the problem requirements and perform their initial analysis: Requirements (both functional and non-functional) are related to the organization that fulfills them. An organization is composed of Roles, which are interacting within scenarios while executing their Role plans. An organization has a context that is described in terms of an ontology. Roles participate in the achievement of their organization goals by means of their Capacities. Definitions of MAS metamodel elements may be found in Table 1, on the ASPECS website, and in [5].

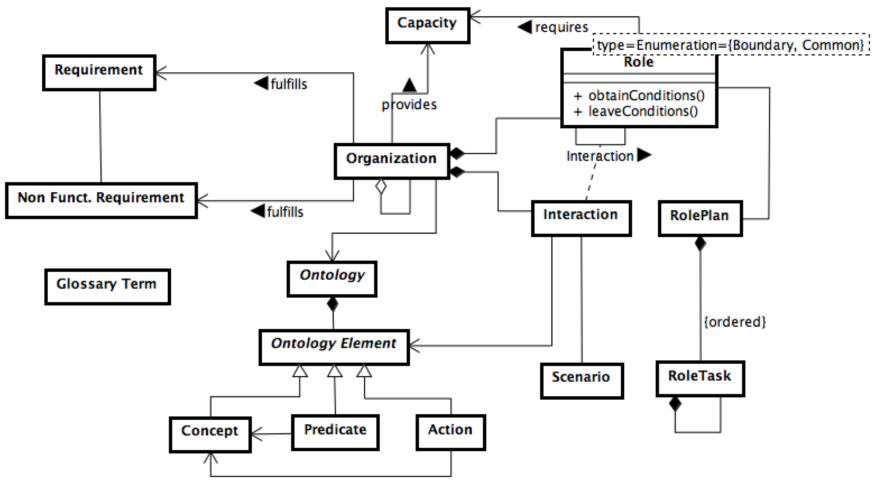


Fig. 2 The ASPECS Problem Domain MAS metamodel

1.2.2 Agency Domain

The Agency Domain metamodel (see Fig. 3) includes the elements that are used to define an agent-oriented solution for the problem analyzed in the previous stage. By adopting an organizational approach, the solution will be mainly composed of the necessary social structures designed in a multi-perspective way. Definitions of the MAS metamodel elements may be found in Table 2, further details on the ASPECS website, and in [5].

Concept	**Definition**
Ontology	An explicit specification of a conceptualization of a knowledge domain [10]. An ontology is composed of abstract ontology elements having three possible concrete types: **Concept**, **Predicate** or **Action**.
Concept	A category, an abstraction that shortens and summarizes a variety/multiplicity of objects by generalizing common identifiable properties.
Predicate	Assertions on concepts properties.
Action	A change realized by an entity that modifies one or more properties of one or more concepts.
Organization	An organization is defined by a collection of roles that take part in systematic institutionalized patterns of interactions with other roles in a common context. This context consists in shared knowledge and social rules/norms, social feelings, and is defined according to an ontology. The aim of an organization is to fulfill some requirements.
Role	An expected behavior (a set of role tasks ordered by a plan) and a set of rights and obligations in the organization context. The goal of each Role is to contribute to the fulfillment of (a part of) the requirements of the organization within which it is defined. A role can be instantiated either as a Common Role or Boundary Role. A Common Role is a role located inside the designed system and interacting with either Common or Boundary Roles. A Boundary Role is a role located at the boundary between the system and its outside, and it is responsible for interactions happening at this border (i.e. GUI, Database, etc.).
Interaction	A dynamic, not a priori known sequence of events (a specification of some occurrence that may potentially trigger effects on the system) exchanged among roles, or between roles and entities outside the agent system to be designed. Roles may react to the events according to theirs behaviors.
Capacity	A capacity is an abstract description of a know-how, i.e. a competence of a role. Each role requires certain skills to define its behavior, and these skills are modeled by means of a capacity. It may be considered as a specification of the pre- and post-conditions of a goal achievement.
Role Task	An activity that defines a part of a role behavior. A *Role Task* may be atomic or composed by a coordinated sequence of subordinate *Role Tasks*. The definition of these *Role Tasks* can be based on capacities, required by roles.
Role plan	The behavior of a *Role* is specified within a *Role plan*. It is the description of how to combine and order *Role Tasks* and interactions to fulfill a (part of a) requirement.
Scenario	Describes a sequence of role interactions, which fulfills a (part of) requirement.

Table 1 Definition of the problem domain concepts (from [5])

1.2.3 Solution Domain

The solution domain metamodel contains elements used for the implementation of the designed solution in the chosen platform. These elements are general enough to be applied to several existing platforms with minor or no changes but nonetheless, the most suitable choice is JANUS that directly inspired this portion of the metamodel.

JANUS (see [8]) was designed to facilitate the transition between the design and implementation phases of holonic systems development processes. It is imple-

Concept	**Definition**
Communica-tion	An interaction between two or more roles where the content (language, ontology, and encoding) and the sequence of communication acts (protocol) are explicitly detailed. A *communication* is composed of messages expressing *communicative acts* [7, 6]. In a communication, participants are *Agent Roles*, and the knowledge exchanged between them is explicitly represented by a set of ontology elements.
Protocol	The sequence of expected message communicative acts; it represents a common pattern of communication, a high-level strategy that governs the exchange of information between *Agent Roles*.
Group	An instance in the Agency Domain of an *Organization* defined in the Problem Domain. It is used to model an aggregation of *Agent Roles* played by holons.
Agent Role	An instance of the Problem Domain *Role*. It is a behavior (expressed by a set of Agent Tasks), and it owns a set of rights and obligations in a specific group context. *Agent Roles* interact with each other by using communications within the context of the group they belong to. Several *Agent Roles* are usually aggregated in the *Autonomous Entity* that plays them. An *Agent Role* may be responsible for providing one of more services to the remaining part of the society.
Holonic Group	A group that is devoted to contain *holonic roles* and takes care of the holon internal decision-making process (composed-holon's government). Holonic roles are used to represent in an organizational way the notion of moderated group (see [9]). They describe the level of authority of a member inside the holon members' community and the degree of commitment of a member to its super-holon.
Agent Task	An *Agent Task* is a refinement of a Problem Domain *Role Task*. It is a portion of a role behavior, and it may be composed by other *Agent Tasks* or atomic *Agent Actions*. It may contribute to provide (a portion of) an *Agent Role*'s service.
Agent Action	The atomic composing unit of a behavior. An action takes a set of inputs and converts them into a set of outputs, though either or both sets may be empty. An example of the most basic *Agent Action* consists in invoking a capacity or a service requiring the same inputs.
Autonomous Entity	An abstract rational entity that adopts a decision in order to obtain the satisfaction of one or more of its own goals. An autonomous entity may play a set of *Agent Roles* within various groups. These roles interact with each other in the specific context provided by the entity itself. The entity context is given by the knowledge, the capacities owned by the entity itself. Roles share this context by the simple fact of being part of the same entity.
Agent	An autonomous entity that has specific individual goals and the intrinsic ability to realize some capacities.
Goal	A description of an objective to pursue and represents an abstraction of a projected state of affairs to obtain.
Individual Goal	A goal pursued by an individual agent that may be related to its personal desires or intentions. This agent will deliberate to determine a plan or a strategy to achieve its individual goals.
Collective Goal	A goal pursued by a community of individuals, which has the commitment of (a part of) the community members. Usually members commit to collective goals because achieving these goals contributes to the achievement of members' individual goals.
Service	It provides the result of the execution of a capacity thus accomplishing a set of functionalities on behalf of its owner: a role, a group, an agent or a holon. These functionalities can be effectively considered as the concrete implementation of various capacities. A role can thus publish some of its capacities and other members of the group can profit of them by means of a service exchange. Similarly a group, able to provide a collective capacity can share it with other groups by providing a service. A capacity is an internal aspect of an organization or an agent, while the service is designed to be shared between various organization or entities. To publish a capacity and thus allow other entities to benefit from it, a service is created.
Resource	The abstraction of an environmental entity. It may be manipulated by roles through specific capacities.

Table 2 Definition of the agency domain concepts extracted from [5]

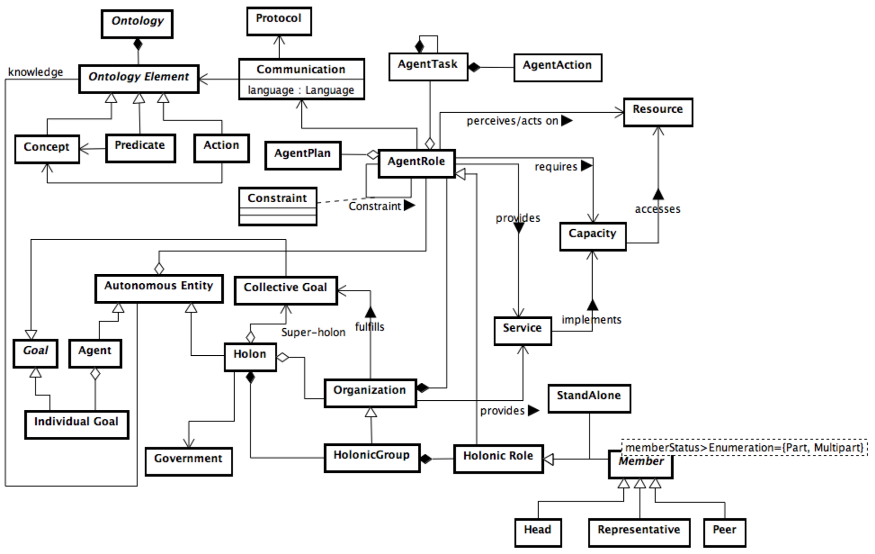


Fig. 3 The ASPECS Agency Domain MAS metamodel

mented in Java, and it supports the direct implementation of the five key concepts used in the design phase: organization, group, role, agent and capacity.

The definitions of the ASPECS solution domain MAS metamodel elements are given in Table 3, further details on the ASPECS website, and in [5].

2 Phases of the process

2.1 System Requirement Analysis

The process flow at the level of activities is reported in Fig. 4. The process flow inside each activity will be detailed in the following subsections (after the description of process roles). The System Requirement Analysis phase involves two different process roles, seven work products as described in the following Fig. **??** and **??**. The phase has been divided in two figures for clarity reasons. Each figure is complete in terms of activities and corresponding roles and workproducts. The second figure add the required predecessor reference(s) to activitie(s) presented in the first figure. The phase is composed of seven activities (i.e. Domain Requirements Description, Problem Ontology Identification, Organization Identification, Interaction and Role Identification, Scenario Description, Role Plan, and Capacity Identification), each of them composed of one or more tasks.

Concept	**Definition**
JOrganization	A JOrganization is defined by a set of roles and a set of constraints to instantiate these roles (e.g. maximal number of authorized instances).
JGroup	JGroup is the runtime context of interaction. It contains a set of roles and a set of Holons playing at least one of these roles. In addition to its characteristics and its personal knowledge, each agent/holon has mechanisms to manage the scheduling of its own roles. It can change dynamically its roles during the execution of the application (leave a role and request a new one). The life-cycle of a Janus agent is composed of three main phases: activation, life, termination. The life of an agent consists in consecutively executing its set of roles and capacities.
JRole	An implementation of the Agency Domain *Agent Role*.
JHolonicRole	Four JHolonicRoles are defined to describe the status of a member inside a super-holon: Head, Representative, Part and Multi-Part.
JHolon	A JHolon is primarily a roles and capacities container. The roles container provides the necessary means for the roles of a holon to interact in the internal interaction context of a holon. The local mechanism of interaction inside a holon is called influence and it is implemented using an event-based communication. Each role can register itself to inform its holon that it wishes to receive all the influences of a given type.
JCapacity Implementation	A capacity can be implemented in various ways, and each of these implementation is modeled by the notion of JCapacityImplementation.
JCapacity	The JCapacity concept is an interface between the holon and the roles it plays. The role requires some capacities to define its behavior, which can then be invoked in one of the tasks that make up the behavior of the role. The set of capacities required by a role are specified in the role access conditions.
JCommunication	To communicate, holons must belong to a common group and play a role in this group. If a group is distributed among several kernels, an instance of this group exists in each kernel, and all the instances have the same name. Communication in JANUS is based on the roles. Messages are delivered to agents according to their roles. This mode of interaction allows the implementation of communications between an emitter and several receivers (one-to-many communication). The address of the receiver agents is dynamically discovered according to the roles they play. When several agents play the same role within the same group, a mode of communication based on role and agents identifier may also be used to distinguish which role player will receive the message.

Table 3 Definition of the solution domain concepts extracted from [5]

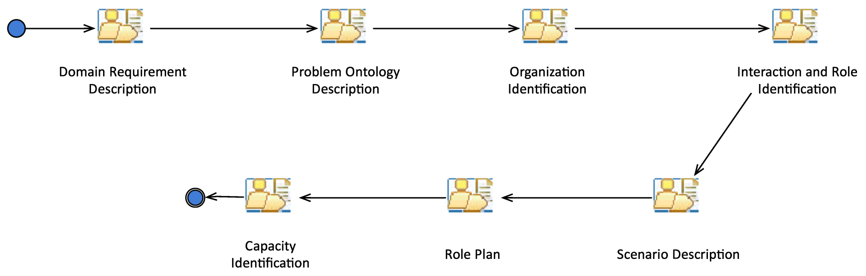

Domain Requirement Description
Problem Ontology Description
Organization Identification
Interaction and Role Identification
Capacity Identification
Role Plan
Scenario Description

Fig. 4 The System Requirement Analysis Phase flow of activities

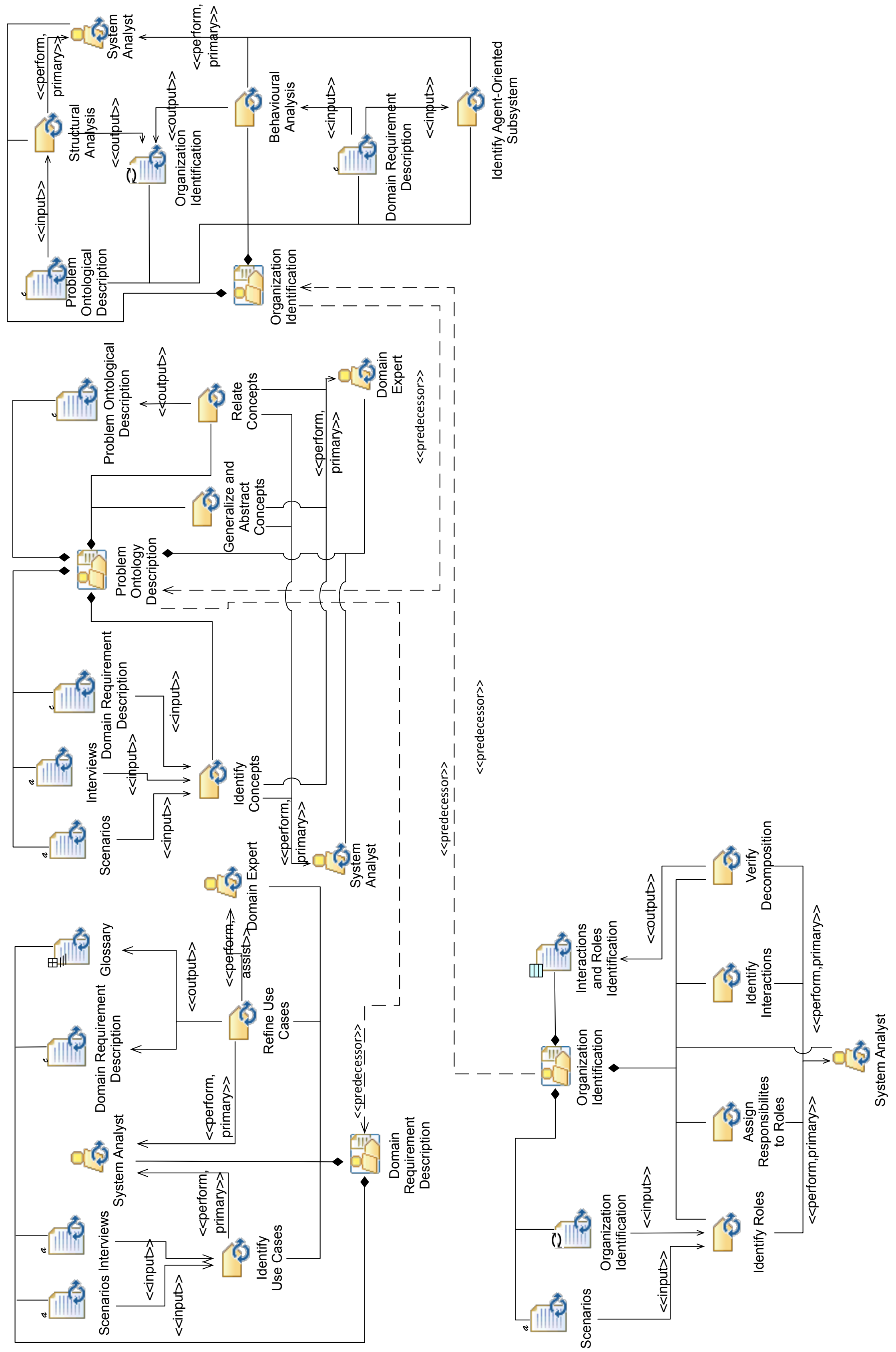


Fig. 5 The System Requirement Analysis Phase described in terms of Activities and Workproducts
I

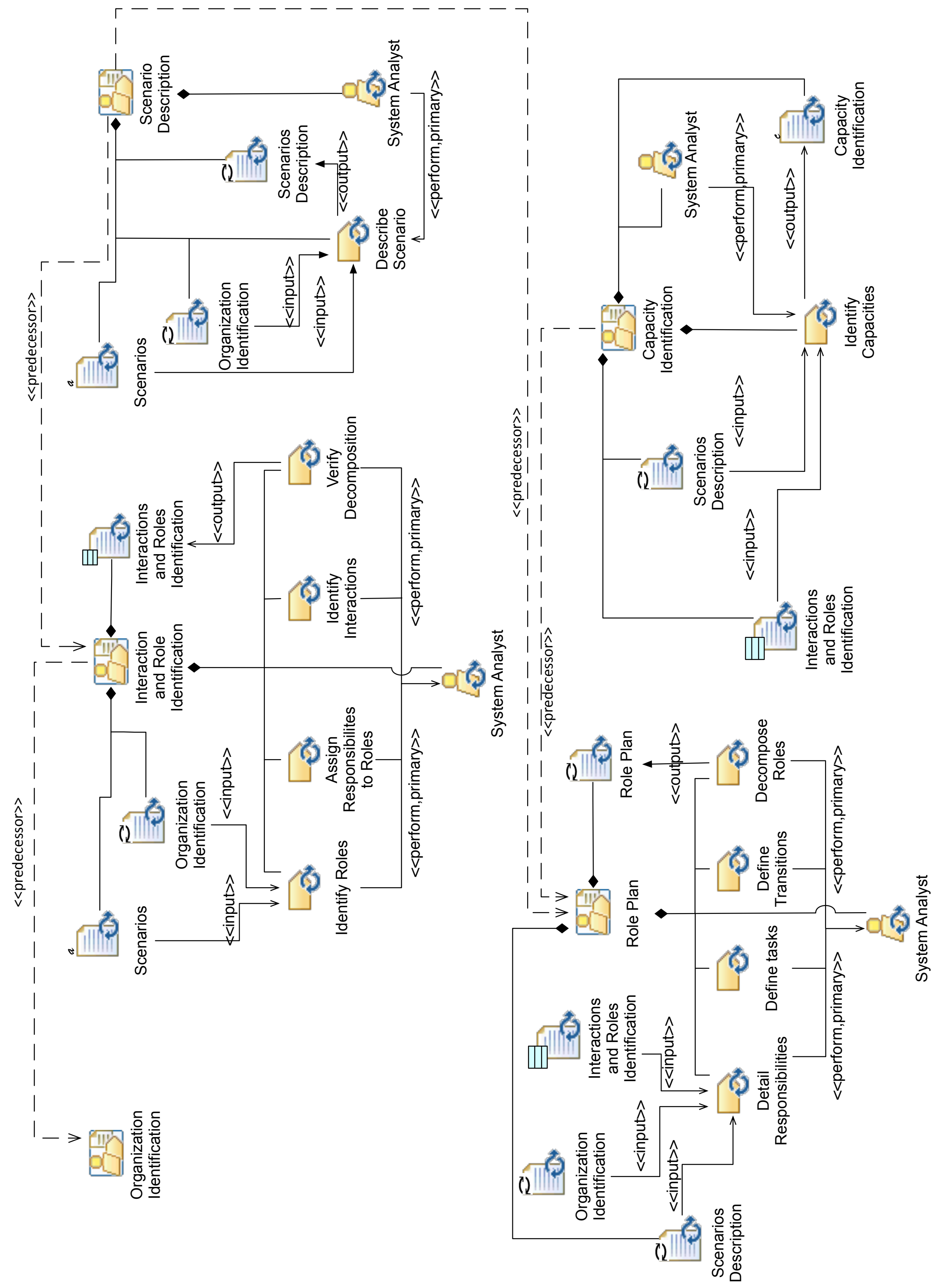


Fig. 6 The System Requirement Analysis Phase described in terms of Activities and Workproducts II

2.1.1 Process roles

Two roles are involved in the System Requirements Phase: the System analyst and the Domain expert. They are described in the following subsections.

System analyst

S/he is responsible of:

1. Use cases identification during the Domain Requirements Description (DRD) activity. Use cases are used to represent system requirements.
2. Use cases refinement during the DRD activity. Use cases are refined with the help of a Domain Expert.
3. Definition of an ontology for the conceptualization of the problem during the Problem Ontology Description (POD) activity.
4. Use cases clustering during the Organization Identification (OID) activity. The System Analyst analyzes the use case diagrams resulting from the first activity and the domain concepts resulting from the second activity and attempts to assign use case to organizations in charge of their realization.
5. Identification of interacting roles for the previously identified organizations and use cases constitutes the Interaction and Role Identification (IRI) activity.
6. Refinement of the interactions between roles during the Scenario Description (SD) activity by means of scenarios designed in form of sequence diagrams thus depicting the details of role interaction.
7. Refinement of role behaviors during Role Plan (RP) activity by means of state-transition diagrams specifying each role behavior.
8. Identification of capacities that are required by roles or provided by the organizations during the Capacity Identification (CI) activity. The capacities are added to the class diagram depicting the organizations composed of interacting roles.

Domain expert

The domain expert has knowledge about the domain of the problem to be solved. S(he) is responsible of:

1. Assisting the System Analyst with the proper knowledge on the domain.
2. Contributing in the decision whether requirements are correctly identified (end of the Domain Requirements Phase).

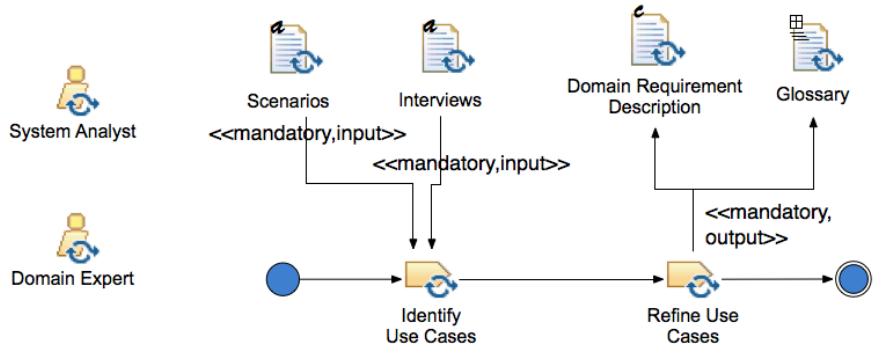


Fig. 7 The flow of tasks of the Domain Requirement Description activity

2.1.2 Activity details

Domain Requirement Description

The Domain Requirements Description (DRD) activity aims at an initial requirements elicitation. The expected result is a description of the application behavior. Several different approaches can be used. For example, use cases diagrams and documented version to introduce user annotations can specify functional and non-functional requirements. Tasks of this activity are detailed in Table 4 and their flow is represented in Fig. 8.

Table 4 Task descriptions of the Domain Requirement Description activity

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

Problem Ontology Description

The Domain Ontology Description activity defines a conceptual overview of the domain concerned. This ontology aims at the conceptualization of experts knowledge that will provide the applications context. Moreover, this ontology helps to understand the problem to be solved and allows requirements refinements. Among the subsequent refinements, the identification of organizations, roles and capacities is one of the most important. Tasks of this activity are detailed in Table 5 and their flow is represented in Fig. 9.

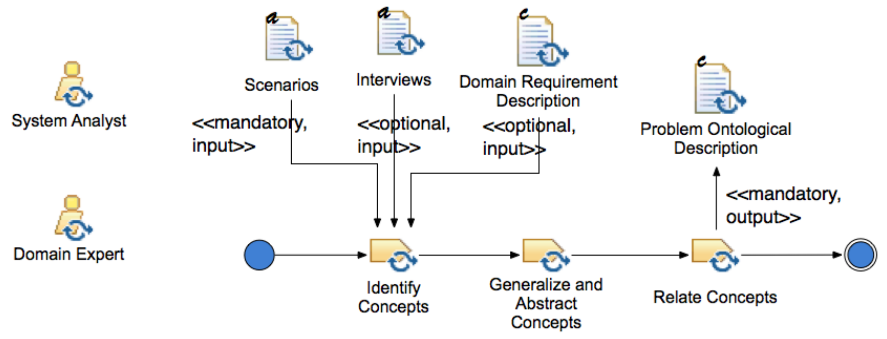


Fig. 8 The flow of tasks of the Problem Ontology Description activity

Table 5 Task descriptions of the Problem Ontology Description activity

Activity	Task	Task description	Roles involved
Problem Ontology Description	Identify Concepts	Recurrent and most significant concepts are identified in use case descriptions	System Analyst (perform), Domain Expert (assist)
Problem Ontology Description	Generalize and Abstract Concepts	Concepts are arranged by means of inheritance and abstraction.	System Analyst (perform), Domain Expert (assist)
Problem Ontology Description	Relate Concepts	Determine concept relationships and especially composition ones.	System Analyst (perform), Domain Expert (assist)

Organisation Identification

The Organization Identification activity goal consists in assigning to each requirement a behavior, which is not detailed at this level and which is represented by an organization. The meaning of this assignment is that a requirement should be satisfied by an organization. The behavior represented by the organization is the result of the interacting roles within a common context. The latter is conceptualized in the POD defined in the previous activity. This activity starts from the requirements defined in the DRD activity. From this starting point different approaches are possible to define organizations and assign requirements. A possible way is to jointly use a structural (ontological-based) approach based on the structural features already identified and a functional approach based on requirement clustering. Tasks of this activity are detailed in Table 6 and their flow is represented in Fig. 10.

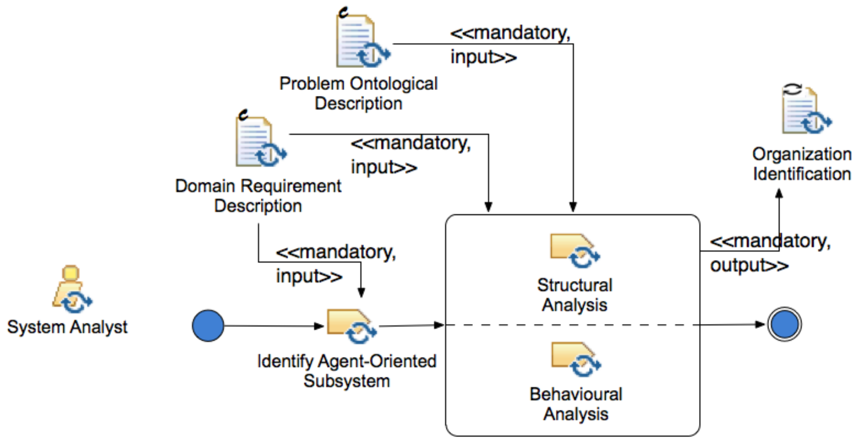


Fig. 9 The flow of tasks of the Organization Identification activity

Table 6 Task descriptions of the Organization Identification activity

Activity	**Task**	**Task description**	**Roles involved**
Organization Identification	Identify Agent-Oriented Sub-system	Identify the parts of the systems that will be realized by using an agent-oriented approach.	System Analyst (perform)
Organization Identification	Structural Analysis	Cluster use cases considering their associations and related ontological concepts.	System Analyst (perform)
Organization Identification	Behavioral Analysis	Cluster use cases dealing with related pieces of system behavior, consider organizational design patterns too.	System Analyst (perform)

Interaction and Role Identification

The Interaction and Role Identification activity should decompose the behavior represented by an organization into finer grain behaviors represented by roles. Tasks this activity are detailed in Table 7 and their flow is represented in Fig. 11.

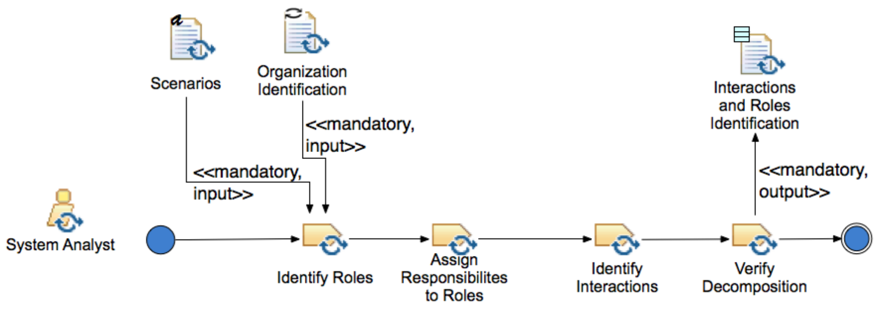


Fig. 10 The flow of tasks of the Interaction and Role Identification activity

Table 7 Task descriptions of the Interaction and Role Identification activity

Activity	**Task**	**Task description**	**Roles involved**
Interaction and Role Identification	Identify Roles	Identify roles from problem ontology and the previously identified organizations.	System Analyst (perform)
Interaction and Role Identification	Assign Responsibilities to Roles	Responsibilities correspond to a part of the requirements associated to the role's organization.	System Analyst (perform)
Interaction and Role Identification	Identify Interactions	Role interactions can be deduced from the study of the relationship between use cases and the associated roles.	System Analyst (perform)
Interactions and Roles Identification	Verify Decomposition	Verify the goodness of the decomposition by studying the contributions from organizations at a given level to roles belonging to the upper level.	System Analyst (perform)

Scenario Description

The Scenario Description activity should describe a set of possible interactions within an organization. The objective is to refine and explore the possible sequences of interactions between roles of a same organization. Obviously the required elements are organizations, with both roles and interactions, and requirements assigned. The challenge of this activity is to specify the fulfillment of requirements by the interactions of organization roles. Tasks of this activity are detailed in Table 8 and their flow is represented in Fig. 12.

Role Plan

The Role Plan activity details the behavior of each role by a general plan. This plan is a partial fulfillment of the organization objectives. The plan is composed of RoleTasks that are elementary units of action. The different plans should conform

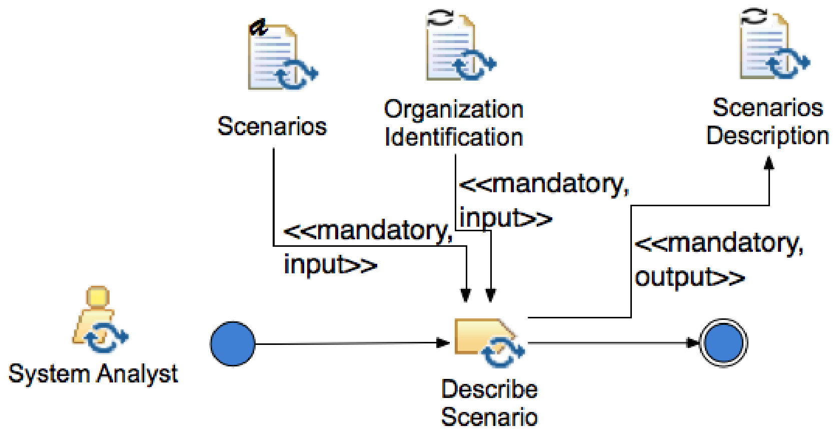


Fig. 11 The flow of tasks of the Scenario Description activity

Table 8 Task descriptions of the Scenario Description activity

Activity	Task	Task description	Roles involved
Scenario Description	Describe Scenario	Describe scenarios as a set of interacting roles working to achieve a required behavior of the system.	System Analyst (perform)

to the scenarios described in the previous activity. One different diagram is drawn for detailing the behavior of each organization. Very complex plans may require a specific diagram is depicted for each role. Tasks of this activity are detailed in Table 9 and their flow is represented in Fig. 13.

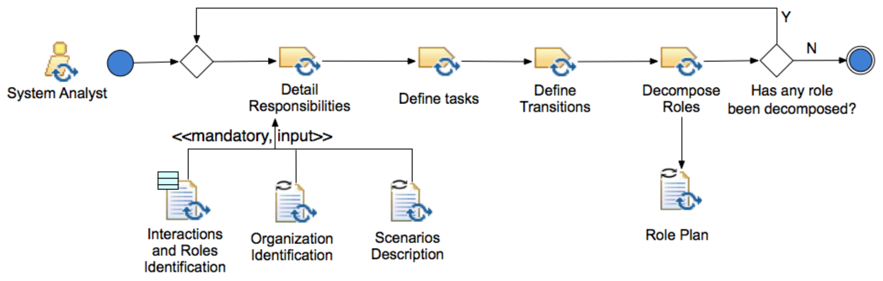


Fig. 12 The flow of tasks Role Plan activity

Table 9 Task descriptions of the Role Plan activity

Activity	Task	Task description	Roles involved
Role Plan	Detail Responsibilities	Detail responsibilities assigned to the role studied in the current diagram.	System Analyst (perform)
Role Plan	Define Tasks	Identify a set of RoleTasks for accomplishing the assigned responsibilities.	System Analyst (perform)
Role Plan	Define Transitions	Define transitions among the various tasks and the set of associated conditions.	System Analyst (perform)
Role Plan	Decompose Roles	Decompose roles if they require external contributions and/or are not correctly encapsulated/coherent.	System Analyst (perform)

Capacity Identification

The Capacity Identification activity contributes to the definition of generic behaviors. The underlying principle is to abstract the know-how that are necessary for playing the roles of an organization. This additional abstraction will allow the modularization and parametrization of the system to be (especially these choices can be made during the next phase). Indeed, the abstraction represented by a capacity can be fulfilled by any mean some unknown at this stage. The identified capacities are added to the already designed IRI diagram by adding relationships between capacities and roles. Tasks of this activity are detailed in Table 10 and their flow is represented in Fig. 14.

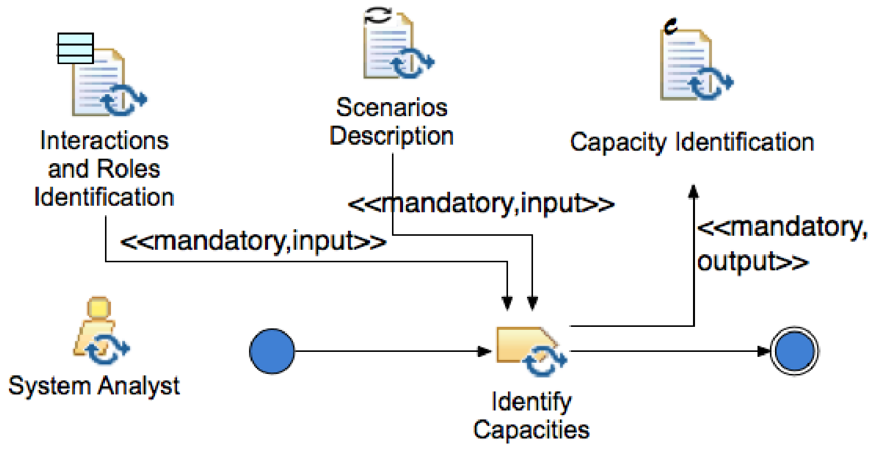


Fig. 13 The flow of tasks of the Capacity Identification activity

Table 10 Task descriptions of the Capacity Identification activity

Activity	Task	Task description	Roles involved
Capacity Identification	Identify Capacities	Identify the generic part of the role behaviour and distinguishing it from all behaviours which could depend on internal properties and data of the entity which will play the role.	System Analyst (perform)

2.1.3 Work Products

The System Requirements phase generates eight work products. Their relationships with the MAS meta- model elements are described in the following Figure 15.

This diagram represents the System Requirements model in terms of Work Products. Each of these reports one or more elements from the ASPECS MAS metamodel; each MAS metamodel element is represented using an UML class icon (yellow filled) and, in the documents, such elements can be Defined, reFined, Quoted, Related or their Relationships Quoted.

Work Product Kinds

Work product kinds are briefly described in Table 11. In the following of this paragraph a description of the notation adopted in each work product will be provided. Generally speaking, ASPECS uses UML as a modeling language but because of the specific needs of agent and holonic organizational design, the UML semantics and notation are used as reference points, and they have been extended using specific profiles (stereotypes); in fact, UML diagrams are often used to represent concepts that are not completely considered in UML and the notation has been modified to better meet the need of modeling agents. Almost all UML diagrams are associated with a textual document documenting and completing the content of the diagram.

Domain Requirement Description

The global objective of the Domain Requirement Description (DRD) activity is gathering needs and expectations of application stake-holders and providing a complete description of the behavior of the application to be developed. In the proposed approach, these requirements should be described by using the specific language of the application domain and a user perspective. This is usually done by adopting use case diagrams for the description of functional requirements (see Fig. 16) ; besides, conventional text annotations are applied to use cases documentation for describing non-functional requirements. The use cases are elicited from text scenarios and stake-holders interviews.

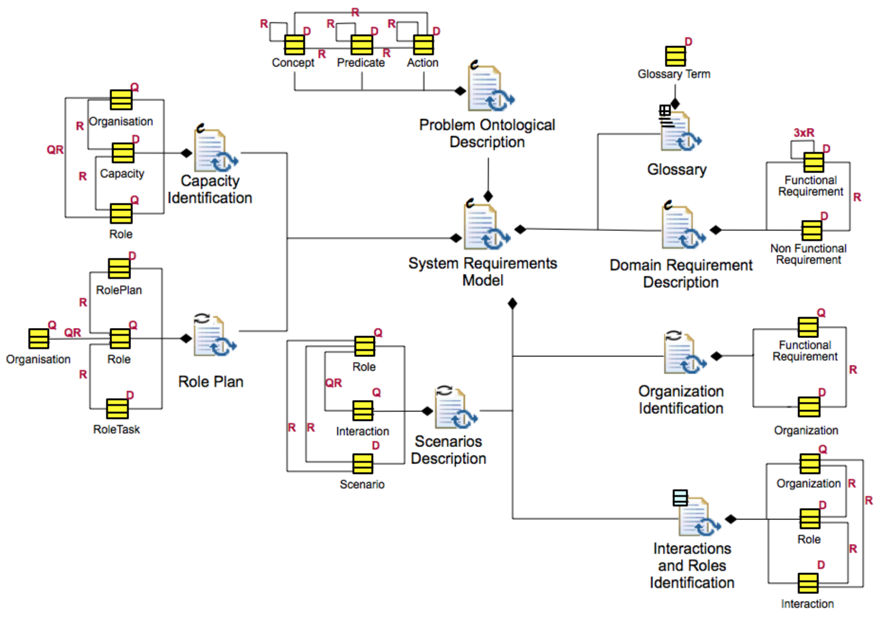


Fig. 14 ASPECS System Requirements Phase Workproducts

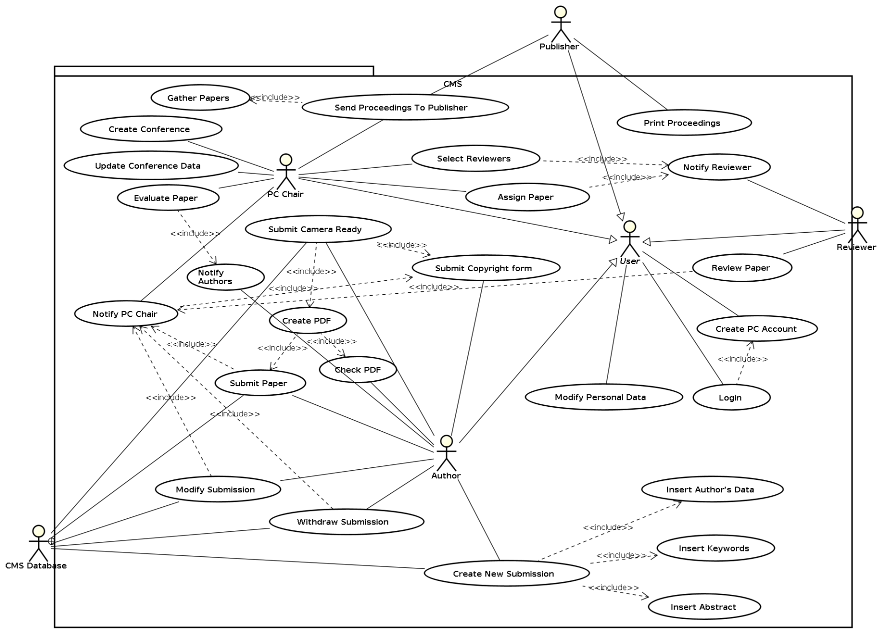


Fig. 15 CMS Case Study: Domain Requirement Description

Table 11 ASPECS System Requirements Phase Workproduct kinds

Name	**Description**	**Workproduct kinds**
Domain Requirement Description (DRD)	A document composed by standard UML use case diagrams, textual documentation of use cases and textual description of non-functional requirements	Composite (Structured + Behavioral)
Glossary	A structured text document defining terms used in the development phases.	Structured
Problem Ontology Description (POD)	The ontology is described in terms of concepts (categories of the domain), predicates (assertions on concept properties), actions (performed in the domain, affecting the status of concepts) and their relationships. A profile for UML class diagrams is used to describe the ontology.	Composite (Structured + Structural)
Organization Identification (OID)	A class diagram reporting use cases and organizations as packages containing them. First level organizations can be decomposed in smaller ones in order to describe an organizational hierarchy.	Behavioral
Interactions and Roles Identification (IRI)	A class diagram where each role is represented by a stereotyped class and interactions are represented by associations between roles. Roles are positioned inside packages representing organizations they belong to.	Structural
Scenarios Description (SD)	Scenarios are described using stereotyped sequence diagrams	Behavioral
Role Plan (RP)	An activity diagram where swimlanes are used to partition activities of different roles and one swimlane is left for hosting external events.	Behavioral
Capacity Identification (CI)	A stereotyped class diagram where capacities are related to the roles that require them through associations. Capacity description is completed by a table detailing, for each capacity, the following fields: Name, Input, Output, Requires, Ensures, and Textual Description.	Composite (Structured + Structural)

Problem Ontology Description

The global objective of the Problem Ontology Description is to provide an overview of the problem domain. Problem ontology is modeled by using a class diagram where concepts, predicates and actions are identified by specific stereotypes. Problem Ontology of the CMS Case study is depicted in Fig. 17. The POD is the result of the analysis of DRD, scenarios and stake-holders interviews.

Organization Identification

The workproduct of the Organization Identification activity (OID) refines the use case diagram produced by the DRD activity and adds organizations as packages encapsulating the fulfilled use cases. The use case diagram presented in Fig. 18

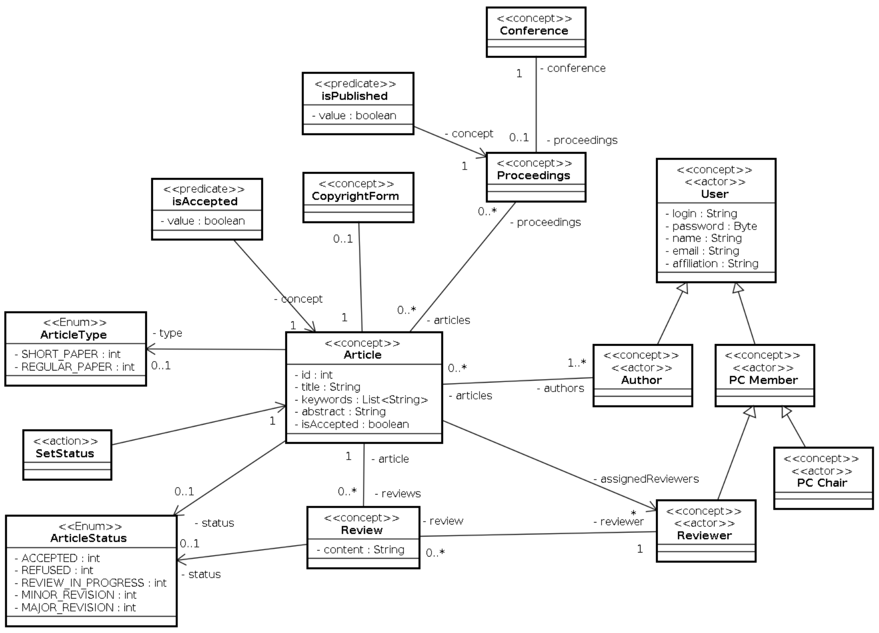


Fig. 16 CMS Case Study: Problem Ontology Description

presents the organization identification diagram for the CMS Case study. The OID uses DRD and POD.

Interaction and Role Identification

The result of the Interaction and Role Identification is a class diagram where classes represent roles (stereotypes are used to differentiate common and boundary roles), packages represent organizations and relationships describe interactions among roles or contributions (to the achievement of a goal) from one organization to another. Fig. 19 shows the interaction and role identification diagram for the CMS Case study. The organizations come from OID. Scenarios are used to identify roles and interactions.

Scenario Description

Scenarios of the Scenario Description (SD) activity are drawn in form of UML sequence diagrams and participating roles are depicted as object-roles. The role name is specified together with the organization it belongs to. Fig. 20 shows the Scenario Description diagram for the example of the *Paper Submission* use case for

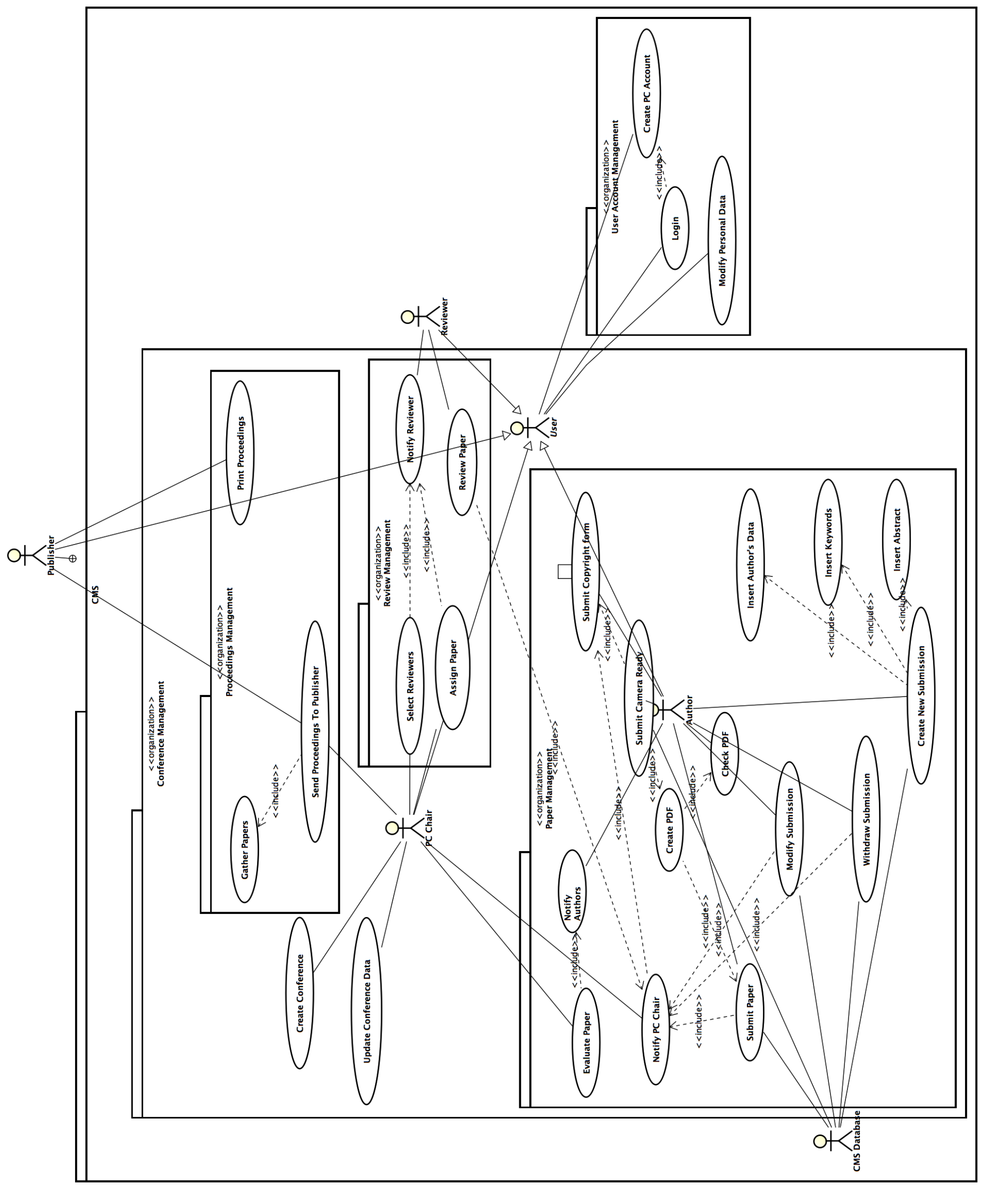


Fig. 17 CMS Case Study: Organization Identification Description

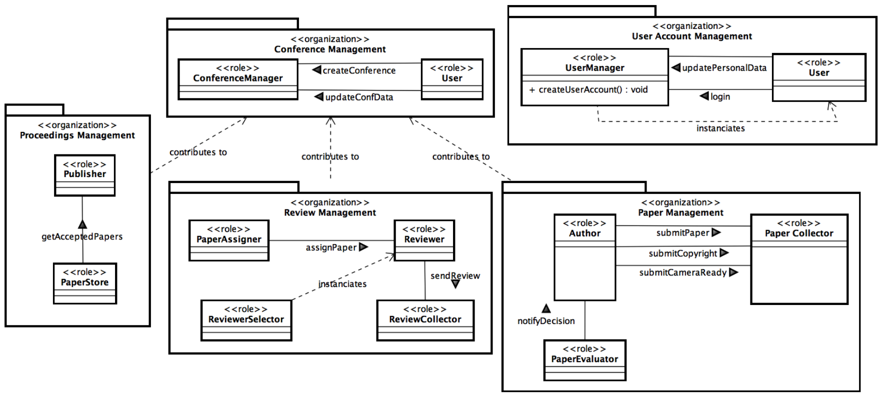
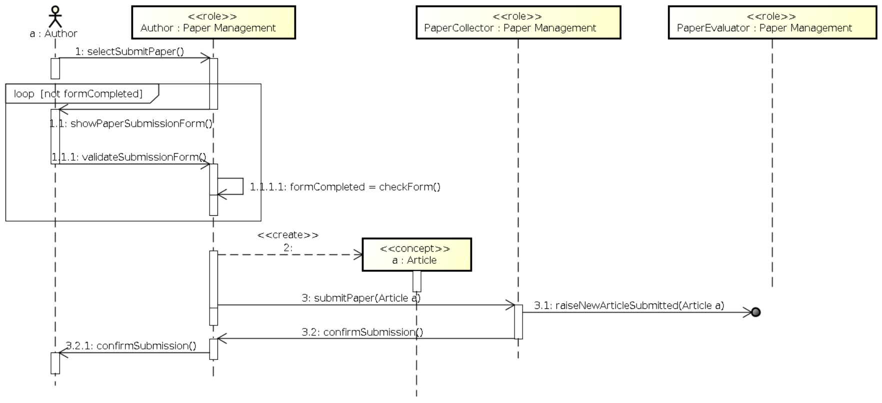


Fig. 18 CMS Case Study: Interaction and Role Identification



a : Author
<<role>> Author : Paper Management
<<role>> PaperCollector : Paper Management
<<role>> PaperEvaluator : Paper Management
1: selectSubmitPaper()
loop [not formCompleted]
1.1: showPaperSubmissionForm()
1.1.1: validateSubmissionForm()
1.1.1.1: formCompleted = checkForm()
<<create>> 2:
<<concept>> a : Article
3: submitPaper(Article a)
3.1: raiseNewArticleSubmitted(Article a)
3.2: confirmSubmission()
3.2.1: confirmSubmission()

Fig. 19 CMS Case Study: Example of the Scenario Description for the *Paper Submission* use case for the *Paper Management* organization

the *Paper Management* organization of the CMS Case study. In this use case, only two of the three roles of the organization are interacting. Organizations, roles and interactions come from OID and IRI. Obviously, scenarios are used.

Role Plan

The resulting work product of the Role Plan (RP) activity is an UML activity diagram reporting one swimlane for each role. Activities of each role are positioned in the corresponding swimlane and interactions with other roles are depicted in form of signal events or object flows corresponding to exchanged messages. Fig. 21 shows

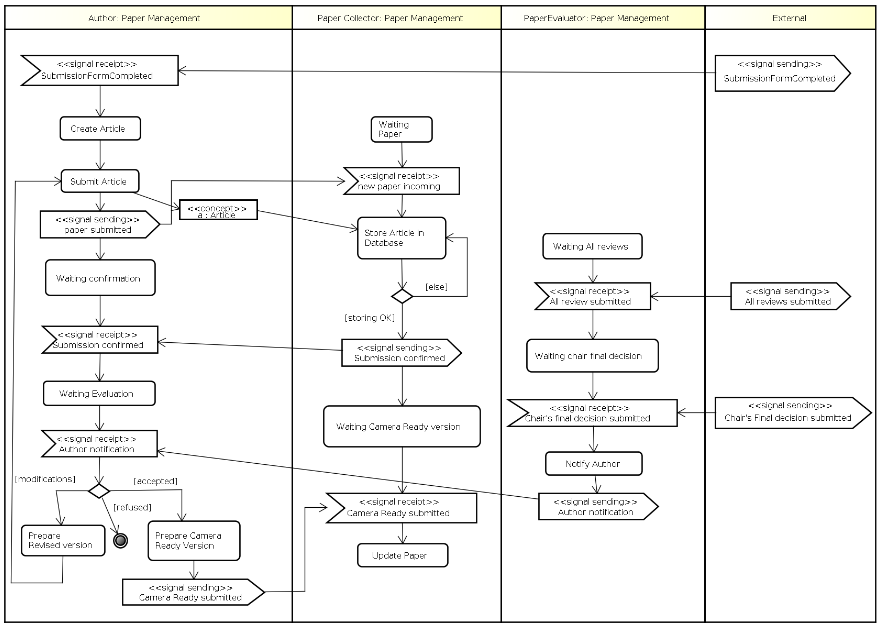

Author: Paper Management
Paper Collector: Paper Management
PaperEvaluator: Paper Management
External
<<signal receipt>> SubmissionFormCompleted
<<signal sending>> SubmissionFormCompleted
Create Article
Submit Article
<<concept>> a : Article
<<signal sending>> paper submitted
Waiting confirmation
<<signal receipt>> Submission confirmed
Waiting Evaluation
<<signal receipt>> Author notification
[modifications]
[accepted]
[refused]
Prepare Revised version
Prepare Camera Ready Version
<<signal sending>> Camera Ready submitted
Waiting Paper
<<signal receipt>> new paper incoming
Store Article in Database
[else]
[storing OK]
<<signal sending>> Submission confirmed
Waiting Camera Ready version
<<signal receipt>> Camera Ready submitted
Update Paper
Waiting All reviews
<<signal receipt>> All review submitted
<<signal sending>> All reviews submitted
Waiting chair final decision
<<signal receipt>> Chair's final decision submitted
<<signal sending>> Chair's Final decision submitted
Notify Author
<<signal sending>> Author notification

Fig. 20 CMS Case Study: Part of the Role Plan Description for the *Paper Management* organization

part of the Role Plan Description Diagram for the *Paper Management* organization of the CMS Case study. Roles and interactions come from IRI. SD are also used.

Capacity Identification

The workproduct produced by the Capacity Identification activity is a refinement of the IRI diagram including capacities (represented by classes) and relating them to the roles that require them. Fig. 22 shows the Capacity Identification Diagram for the *Paper Management* organization of the CMS Case study. Organizations and roles come from OID and IRI.

2.2 Agent Society Design

The complete sequence of activities is described in figure 23, **??** and **??**. As it was the case for the first phase the description in terms of activities and workproudtcs has been divided in two figures for clarity reasons. Each figure is complete in terms

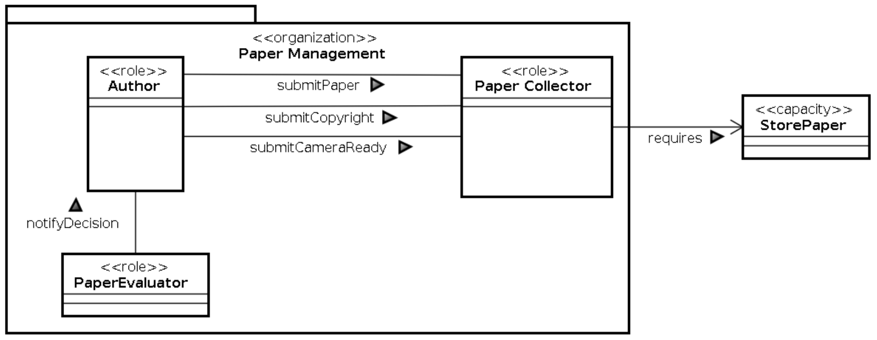


Fig. 21 CMS Case Study: Capacity Identification for the *Paper Management* organization

of activities and corresponding roles and workproducts. The second figure add the required predecessor reference(s) to activitie(s) presented in the first figure.

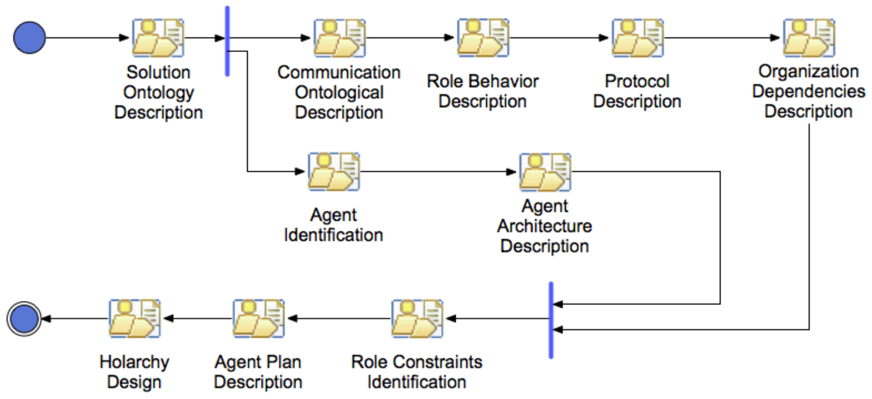

Solution Ontology Description
Communication Ontological Description
Role Behavior Description
Protocol Description
Organization Dependencies Description
Agent Identification
Agent Architecture Description
Holarchy Design
Agent Plan Description
Role Constraints Identification

Fig. 22 The Agent Society Design phase flow of activities

2.2.1 Process Roles

In this second phase of the ASPECS process three roles are employed: Agent Designer, Domain Expert, and System Analyst. The first is a generic role which generally corresponds to a senior developer or a software architect with a good expertise in multiagent systems. The second is an expert in the problem domain able to help the designer in analyzing the problem and understanding its peculiarities. The third, collaborates to some design activities by bringing her/his understanding of requirements to be satisfied.

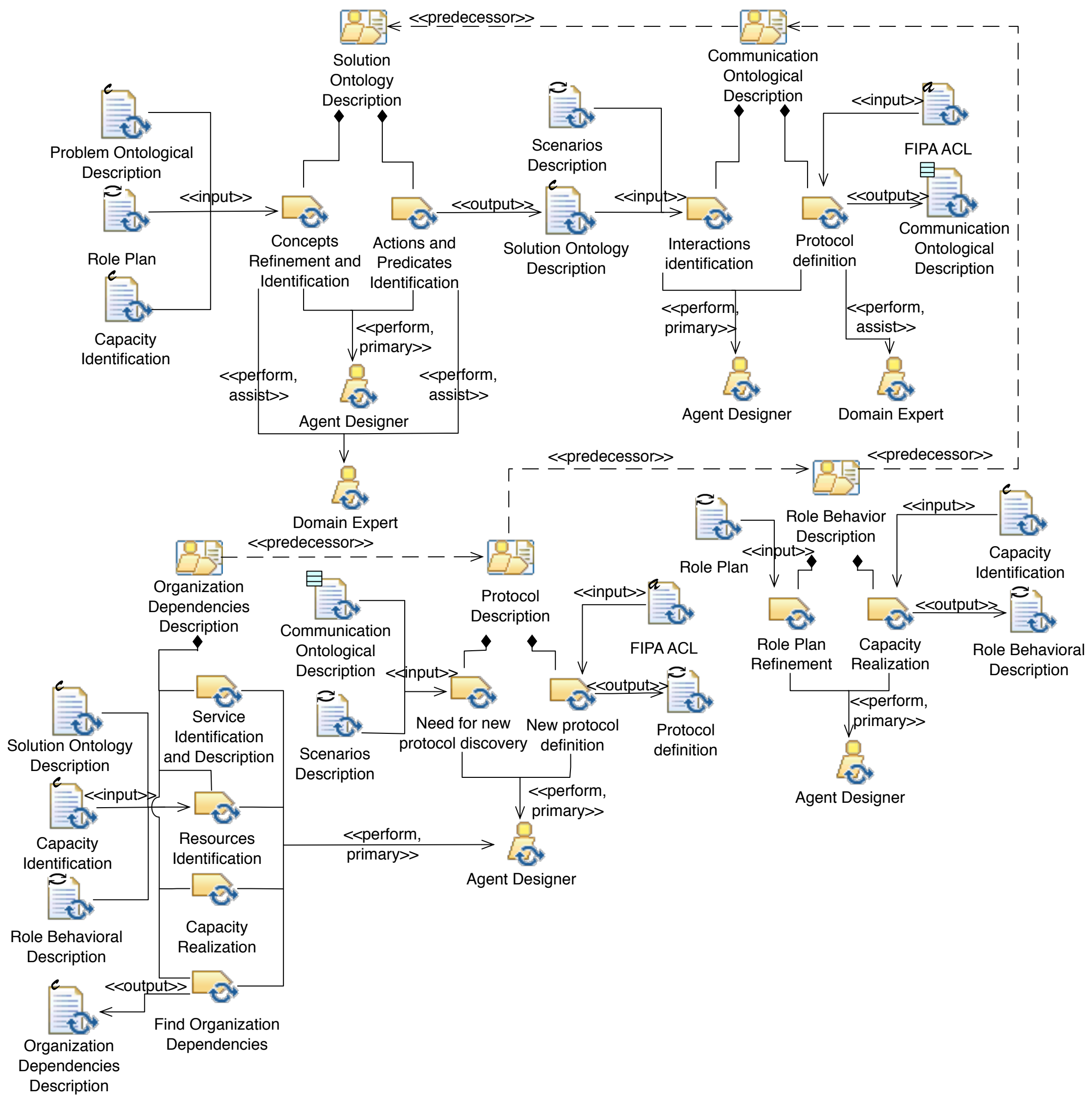


Fig. 23 The Agent Society Phase described in terms of Activities and Workproducts I

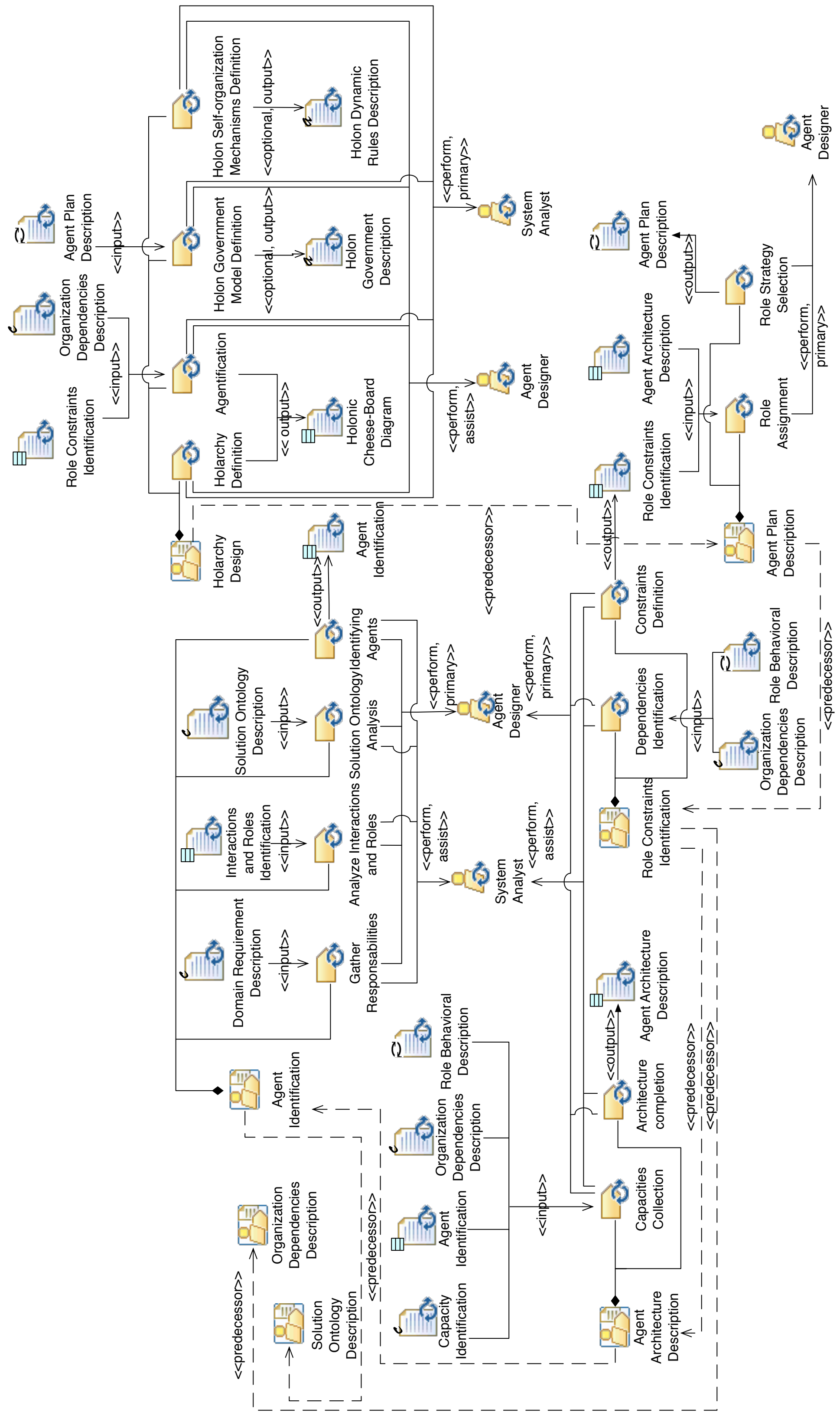

Role Constraints Identification
Organization Dependencies Description
Agent Plan Description
<<input>>
Holarchy Definition
Agentification
Holonic Cheese-Board Diagram
<< output>>
Holon Government Model Definition
<<optional, output>>
Holon Government Description
Holon Self-organization Mechanisms Definition
Holon Dynamic Rules Description
<<perform, primary>>
<<perform, assist>>
System Analyst
Agent Designer
Holarchy Design
Agent Identification
<<output>>
<<predecessor>>
Role Assignment
Role Strategy Selection
Agent Architecture Description
Agent Plan Description
Domain Requirement Description
Interactions and Roles Identification
Solution Ontology Description
Gather Responsabilities
Analyze Interactions and Roles
Solution Ontology Analysis
Identifying Agents
Constraints Definition
Dependencies Identification
Role Behavioral Description
Role Constraints Identification
Architecture completion
Capacities Collection
Capacity Identification
Solution Ontology Description
Agent Architecture Description

Fig. 24 The Agent Society Phase described in terms of Activities and Workproducts II

Domain Expert

S/he assists in:

1. The refinement and identification of concepts.
2. The identification of actions and predicates.

Agent Designer

S/he is responsible for:

1. Refining concepts of Problem Ontology and identification of new solution-dependent concepts during Solution Ontology Description (SOD) activity.
2. Action and predicate identification and their association with previously identified concepts during SOD activity.
3. Identifying individual agents and their personal goals during Agent Identification activity. The Agent designer analyzes the solution ontology to identify individuals who are usually concepts linked to actions. Action's analysis is considered as a useful guideline to identify agent responsibilities since an individual acts according to personal motivations and goals.
4. Defining agent's architecture in Agent Architecture Description (ADD) activity.
5. Clustering interactions in communications or conversations during Communication Ontological Description (COD) activity.
6. Describing role plans using statecharts during the Role Behavior Description activity.
7. Describing tasks and actions needed by roles to realize a capacity.
8. Defining purpose-specific interaction protocols during Protocol Description (PD) activity when the description of communications done during COD and Scenario Description activities does not match any of the existing FIPA protocols.
9. Identifying resources and services related to the different roles of solution's organizations during Organization Dependencies Description (ODD) activity.
10. Identifying constraints between roles during Role Constraints Identification (RCI) activity.
11. Identifying agents's responsibilities, defining lowest level roles and identifying agents during the Agent Identification (AI) activity.
12. Determining agent's personal plan during Agent Plan Description (APD) activity.
13. Defining the complete solution structure during Holarchy Design (HD) activity: mapping between agent and role, agent composition.
14. Specifying the rules that govern the dynamics of holarchies during HD activity.

System Analyst

S/he assists in:

1. Identifying agents's responsibilities, defining lowest level roles and identifying agents during the Agent Identification (AI) activity.
2. Defining agent's architecture in Agent Architecture Description (ADD) activity.
3. Identifying constraints between roles during Role Constraints Identification (RCI) activity.

2.2.2 Activity Details

Solution Ontology Description

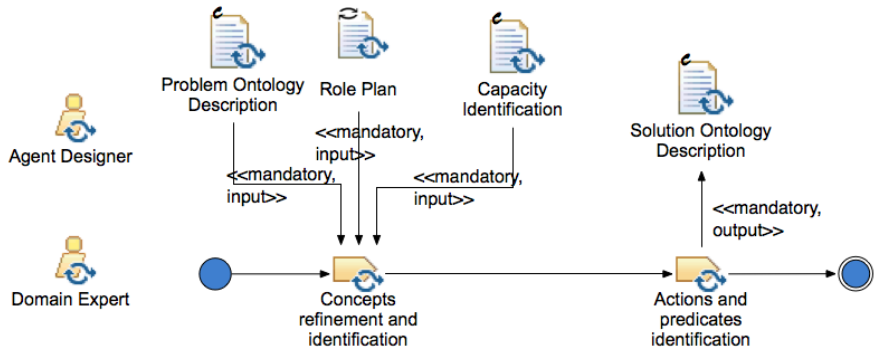


Fig. 25 The flow of tasks of the Solution Ontology Description (SOD)

The objective of this activity consists in refining the problem ontology described during POD by adding new concepts related to the agent-based solution and refining the existing ones.

For concept identification, a guideline consists in looking into previously identified organizations. The hierarchical composition of two organizations generally hides a composition between concepts. This can also be done by looking into role plans and scenarios. Identification of actions and predicates can be facilitated by results of the capacity identification activity. As we have already said, if a capacity deals with some ontological knowledge, manipulated concepts should be connected to an action in the corresponding ontology. Moreover, the solution ontology will also be exchanged in agent communications; an indication about which knowledge will be necessary to roles' behaviors and in which activities it will be used can be found in role plans and scenarios. From these, actions and predicates can generally easily be identified. Of course, this description of the ontology is also an iterative process, and it is generally refined during the communication ontology description (the second activity of the Agent Society Phase). Concepts, predicates and actions of this ontology will also be used to describe information exchanged during communications among roles. Tasks of this activity are detailed in Table 12 and their flow is represented in Fig. 26.

Table 12 Task descriptions of the Solution Ontology Description activity

Activity	**Task**	**Task Description**	**Roles involved**
Solution Ontology Description	Concepts refinement and identification	Existing concepts are refined and new ones maybe identified	Agent Designer (perform), Domain expert (assist)
Solution Ontology Description	Actions and predicates identification	Actions and predicates are identified with the help of the domain expert	Agent Designer (perform), Domain expert (assist)

Communication Ontological Description

This activity aims at describing communications among roles in terms of ontology, interaction protocol and content language.

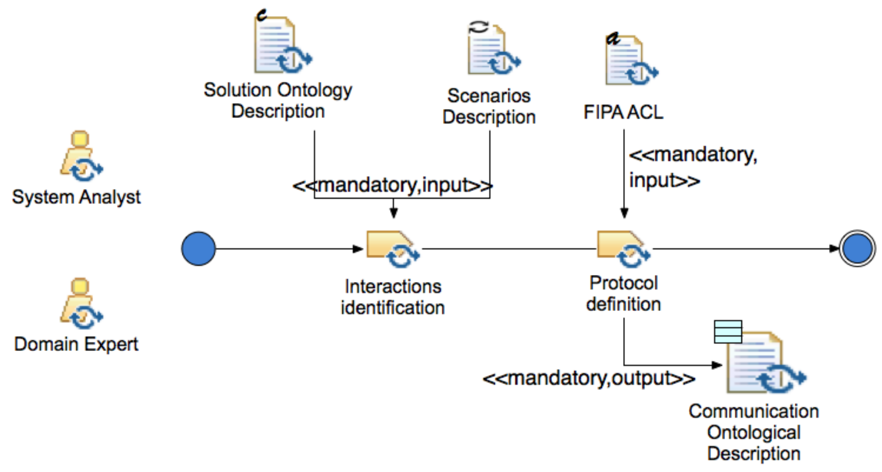


Fig. 26 The flow of tasks of the Communication Ontological Description (COD)

A communication is an interaction between two or more roles where the content (language, ontology,and encoding) and the control of the communication (protocol) are explicitly detailed. A communication mainly consists of FIPA speech acts and protocols. The model of role communication we adopt is based on the assumption that two roles during their interaction, have to share a common ontology. This common knowledge is represented in the communication by a set of Ontology elements. A communication is an interaction composed of several messages ordered by a Protocol. Each set of interactions between two roles has to be clustered in one or more communications. At this stage, we could regard the previous studied interactions as messages of the communication.Interactions linking a boundary role to another boundary one are generally refined in communications (usually these interactions are mediated by the environment and therefore).They can be based on events or other stimuli sent from one role to the other. Interactions between classical (i.e. non-boundary) roles are refined in communications with a defined protocol, a referred ontology and an associated content language. As a consequence, in communications, the sequence of messages will be ruled by a proper interaction protocol, and the content will be related to the exchanged information or required exhibition of a capacity through an explicit reference to the adopted ontology. This is in accordance with the FIPA specifications[6] (that we largely adopt), where communications consist of speech acts [14]. FIPA also specifies a relevant number of interaction protocols but a new one, if necessary, can be designed.

This activity should also describe data structures required in each role to store exchanged data. These are based upon the elements of the solution ontology. Tasks of this activity are detailed in Table 13 and their flow is represented in Fig. 27.

Table 13 Task descriptions of the Communication Ontology Description activity

Activity	**Task**	**Task Description**	**Roles involved**
Communication Ontology Description	Interactions identification	Among the existing interactions some are refined into communications	Agent Designer (perform)
Communication Ontology Description	Protocol definitions	For each communication, a protocol and speech acts are defined	Agent Designer (perform), Domain expert (assist)

Role Behavior Description

This activity aims at defining the complete life-cycle of a role by integrating previously identified RoleTasks, capacities, communications/conversations in which it is involved and the set of (or part of) requirements it has to fulfill. The behavior of the role is described by a set of AgentTasks and AgentActions.

Role Behavior Description is a refinement at the Agent Society level of outputs produced by the Role Plan activity during the System Requirements phase. Each RoleTask is refined and decomposed into a set of atomic behavioral units. The behavior of each role is now described using a statechart or an activity diagram.

If a role requires capacities or provides services, this activity has to describe tasks and actions in which they are really used or provided. The designer describes the dynamical behavior of the role starting from the Role Plan drawn in the previous phase, and the capacities used by the role.

Tasks of this activity are detailed in Table 14 and their flow is represented in Fig. 28.

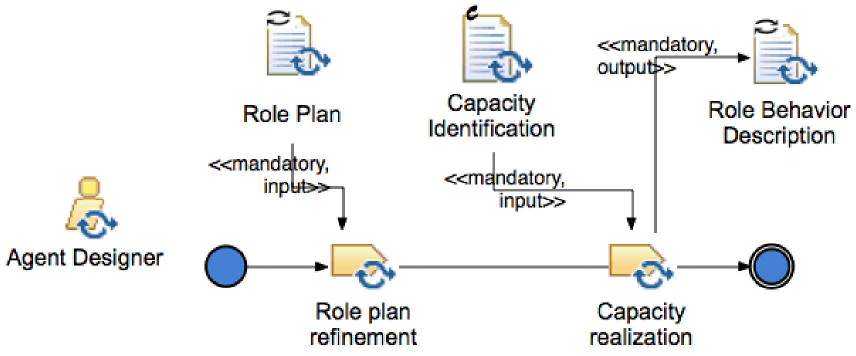


Fig. 27 The flow of tasks of the Role Behavior Description (RBD) activity

Table 14 Task descriptions of the Role Behavior Description activity

Activity	Task	Task Description	Roles involved
Role Behavior Description	Role Plan Refinement	Role plans are described in form of a statechart	Agent Designer (perform)
Role Behavior Description	Capacity Realization	If a role requires capacities or provides services, tasks and actions in which they are really used or provided have to be described.	Agent Designer (perform)

Protocol Description

The aim of this activity is to define purpose-specific interaction protocols whose need may arise when the description of communications done during the COD (Communication Ontology Description), and SD (Scenario description) activities does not match any of the existing FIPA protocols. The designer starts from scenarios and the ontological description of communications in order to find the need for a new specific interaction protocol. If this is the case, then he can proceed to the definition of a new protocol that is compliant with the interactions described in scenarios and the semantics of the communication. It is advisable to refer to the FIPA Interaction protocols library[3] in order to see if a satisfying protocol already exists and if not, probably an existing one can be the basis for changes that can successfully solve the specific problem. Tasks of this activity are detailed in Table 15 and their flow is represented in Fig. 29.

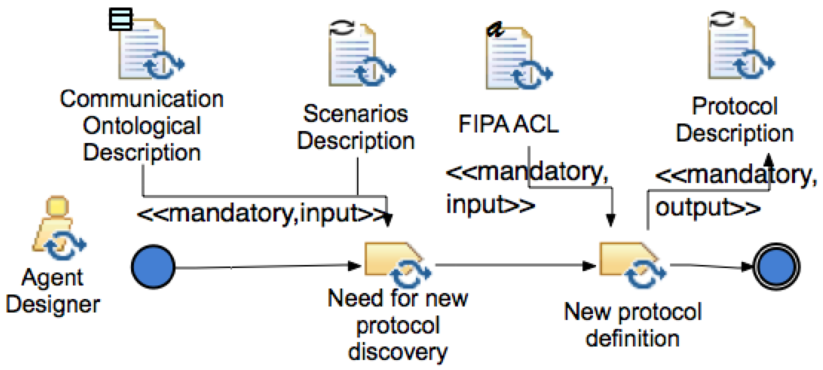


Fig. 28 The flow of tasks of the Protocol Description (PD)

Table 15 Task descriptions of the Protocol Description activity

Activity	**Task**	**Task Description**	**Roles involved**
Protocol Description	Need for new protocol discovery	The designer starts from scenarios and the ontological description of communications in order to find the need for a new specific interaction protocol.	Agent Designer (perform)
Protocol Description	Role plan refinement	Refine the defined role plan by decomposing activities into atomic behavior units	Agent Designer (perform)

[3] FIPA Interaction Protocols specifications: `http://www.fipa.org/repository/ips.php3`

Organization Dependencies Description

The first aim of this activity is to identify resources manipulated by roles; this often implies identifying new capacities devoted to manipulate these resources. Moreover, since organizations depend on each other through service exchange, services provided by roles (while exploiting their capacities and accessing resources) can be identified in this activity.

Identification of resources can be done by identifying resources that can be accessed and manipulated by different roles: databases, files, hardware devices, etc. Resources are external to the role; a capacity is generally used to interface the role with the new resource. Each capacity needs a realization in form of a service that is purposefully defined for that.

When capacities and services are completely described, a verification is necessary to ensure that each capacity is associated with its set of possible service-level realizations. This matching between service and capacity allows the initialization of a repository that may be used to inform agents on how to dynamically obtain a given capacity. Moreover, it is also a verification of the goodness of the system hierarchical decomposition. This matching should validate the contribution that organizations acting at a given level give to upper-level organizations. It will be assumed that an

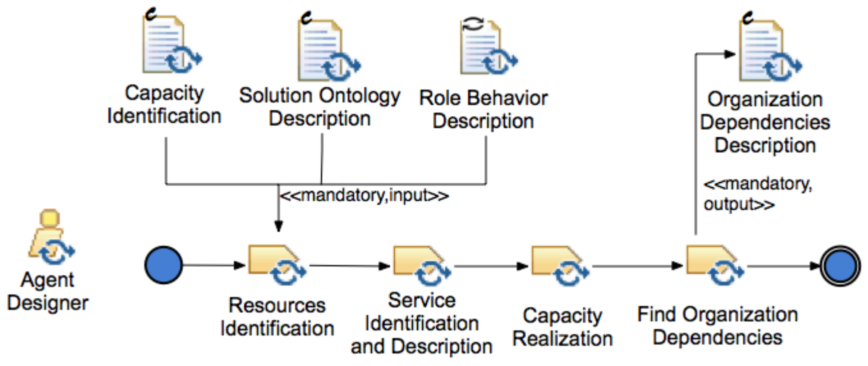

Capacity Identification
Solution Ontology Description
Role Behavior Description
Organization Dependencies Description
<<mandatory,input>>
<<mandatory, output>>
Agent Designer
Resources Identification
Service Identification and Description
Capacity Realization
Find Organization Dependencies

Fig. 29 The flow of tasks of the Organizations Dependencies Description (ODD)

organization can provide a service if the service is provided by one of its roles. A role may provide a service if it owns a capacity with a compatible description, and it manages/supervises a workflow where:

- the service is implemented in one of its own behaviors; or
- the service is obtained from a subset of other roles of the same organization by interacting with them using a known protocol; this is a service composition scenario where the work plan is a priori known, and performance may be someway ensured; or

- the service is obtained from the interaction of roles of the same organization, but the work plan is not a priori known and the service results in an emergent way.

In the last cases, the management of a service provided by the global organization behavior is generally associated to one of the organization's roles. It can also be managed by at least one of the representatives of the various holons playing roles defined in the corresponding organization. Service exchange is often a need arising from the beginning of the requirements analysis phase. In the proposed approach, some services can be identified by exploiting relationships among use cases, especially those crossing different organizational packages. If two use cases belonging to two different organizations are related by an *include* or *extend* relationship, it is very likely that the corresponding organizations will cooperate in a service exchange. Tasks of this activity are detailed in Table 16 and their flow is represented

Table 16 Task descriptions of the Organization Dependencies Description activity

Activity	**Task**	**Task Description**	**Roles involved**
Organizations Dependencies Description	Resource Identification	Identify resources manipulated by roles.	Agent Designer (perform)
Organizations Dependencies Description	Service Identification and Description	Identify and describe services provided by roles and organizations	Agent Designer (perform)
Organizations Dependencies Description	Capacity Realization	Match capacities with services that can realize them and with resources accessed by capacities	Agent Designer (perform)
Organizations Dependencies Description	Find Organization Dependencies	Check dependencies between organizations for possible capacities to be published/realized	Agent Designer (perform)

in Fig. 30.

Agent Identification

The Agent Identification (AI) activity consists in identifying the agents that will compose the lowest level of the system hierarchy and their responsibilities. Responsibilities are modeled using the notion of individual goals and they will be the basis to determine the agent architecture in the next activity. The Domain Requirements Description, Interactions and Role Identification and Solution Ontology Description Document are the main inputs for this activity. Results of this activity may be reported by adopting TROPOS[2] goal and actor diagram. Other options may be preferred like, for instance, the system and agent overview diagrams proposed in PROMETHEUS[13] .

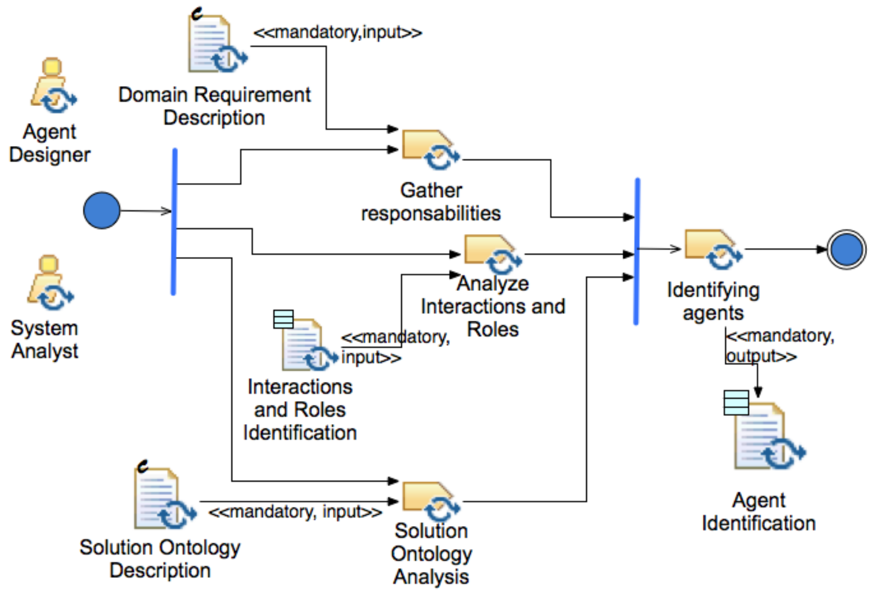


Fig. 30 The flow of tasks of the Agent Identification (AI) activity

The work flow of this activity begins with the identification of Agents goals; this mainly consists on gathering organization responsibilities located at the lowest level of the system organizational hierarchy. These responsibilities are expressed in terms of requirements described by using a combination between a use-case driven and a goal-oriented approach. A second task exploits the Interactions and Role Identification activity results. Indeed, Agents are conceived to play lowest-level roles; their personal goals should at least correspond to the set of the goals pursued by these roles. In order to be able to play these roles, agents have also to provide an implementation for the capacities required by these roles. This aspect will be studied in the next activity. A third task, consists in the analysis of the Solution Ontology in order to identify concepts that represent system individuals (concepts linked to ontology actions). The results of these three tasks are effectively considered as useful guidelines to identify agent responsibilities since an individual acts according to personal motivations and goals. Tasks of this activity are detailed in Table 17 and their flow is represented in Fig. 31.

Agent Architecture Description

The Agent Architecture Description (AAD) activity defines the architecture to be adopted by agents. Such agent architecture should at least define the set of roles that the agent should play and the minimal set of services implementing the capacities required by these roles. The association between Agents and Agent Roles allows the

Table 17 Task descriptions of the Agent Identification activity

Activity	Task	Task Description	Roles involved
Agent Identification	Gather responsibilities	Identify the set of lowest level requirements	Agent Designer (perform), System Analyst (assist)
Agent Identification	Analyze Interactions and Roles	Find lowest level roles	Agent Designer (perform), System Analyst (assist)
Agent Identification	Solution Ontology analysis	Analyze the Solution Ontology to identify concepts linked to actions	Agent Designer (perform), System Analyst (assist)
Agent Identification	Identifying agents	Combine the results of previous tasks to identify agents	Agent Designer (perform), System Analyst (assist)

identification of the set of capacities that are required by Agent Roles in order to be played by Agents. In this activity, a UML class diagram is used to describe agents and their capacities realizations in terms of attributes and methods.

Based on the previous activity, Agent Identification, this activity is composed of two tasks: Capacities collection, and Architecture completion. The first task consists in collecting the capacities needed by the roles played by the agent, the second in adding all the necessary agent architectural elements (for instance knowledge slots necessary to manage the concepts related to role playing and capacities enacting). Tasks of this activity are detailed in Table 18 and their flow is represented in Fig. 32.

Table 18 Task descriptions of the Agent Architecture Description activity

Activity	Task	Task Description	Roles involved
Agent Architecture Description	Capacities Collection	Capacities needed by the agent to play roles are related to the agent.	Agent Designer (perform), System Analyst (assist)
Agent Architecture Description	Agent Architecture Description	Other agents' architectural details (for instance knowledge) are added to the architecture definition.	Agent Designer (perform), System Analyst (assist)

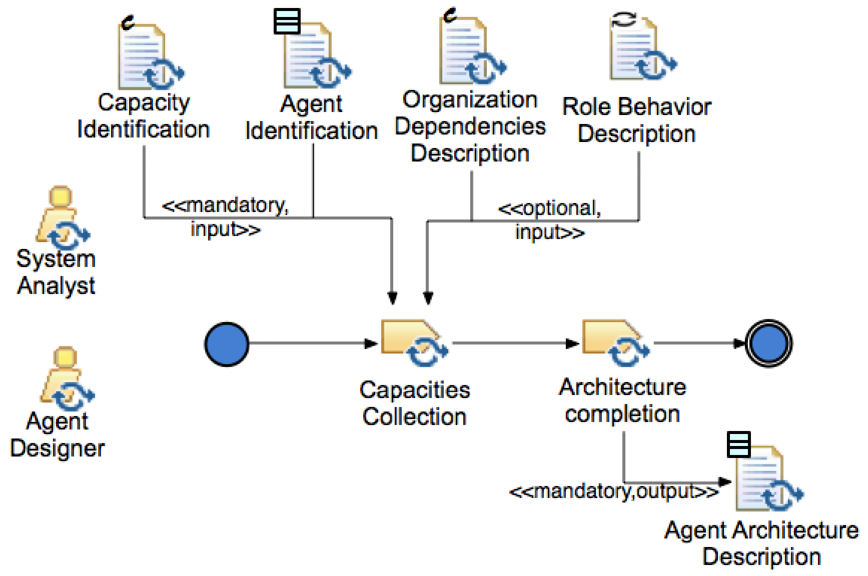


Fig. 31 The flow of tasks of the Agent Architecture Description (AAD) activity

Role Constraints Identification

This activity aims at identifying constraints among roles. The objective consists in characterizing roles that have to be played simultaneously, priorities or mutual exclusion between role, preconditions for one role generated by another one, etc. Concurrency constraints are also important because they will drive the definition of role scheduling policies. Constraints between roles that prevent their concurrent execution and force a precise execution sequence have to be defined as well. Roles can be played simultaneously if and only if they allow an exchange of information between two different organizations. A means for realizing this exchange may consists in using the agent internal context when both roles belong to the same agent. This constitutes an alternative to the use of services and a simplification of information transfer.

The first task of the work consists in identifying constraints between roles thanks to roles' dependencies and associated knowledge as described in the Organization Dependency Description and Role Behavior Description activities. This allows the definition of precise constraints between roles in the second activity's task. Results of this activity are added to the Organization Dependencies Description diagram thus refining that work product. Tasks of this activity are detailed in Table 19 and their flow is represented in Fig. 33.

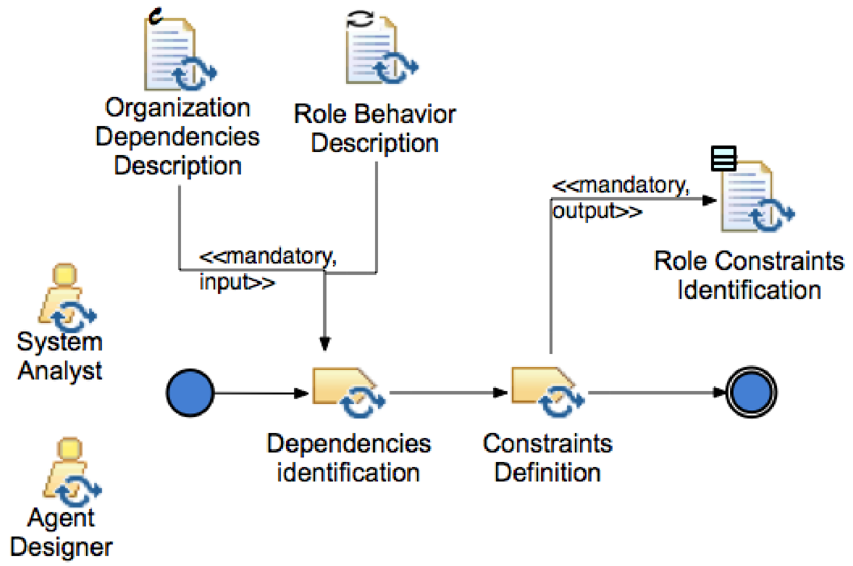

Organization Dependencies Description
Role Behavior Description
<<mandatory, input>>
<<mandatory, output>>
Role Constraints Identification
System Analyst
Agent Designer
Dependencies identification
Constraints Definition

Fig. 32 The flow of tasks of the Role Constraints Identification (RCI) activity

Table 19 Task descriptions of the Role Constraints Identification activity

Activity	Task	Task Description	Roles involved
Role Constraints Identification	Dependencies Identification	Identification of possible conflicts, dependencies and the constraints that relate roles	Agent Designer (perform), System Analyst (assist)
Role Constraints Identification	Constraints Definition	Formalization of constraints in the output diagram	Agent Designer (perform), System Analyst (assist)

Agent Plan Description

This activity refines the design of the agent internal architecture. Plans of each agent are defined according to its goals starting from the Agent Architecture Description and Role Constraints Identification work products. In the plan, the designer makes explicit the strategy the agent will use for choosing the roles it will play. In this activity, a statechart or activity diagram is used to describe the plan of each agent.

In this activity, each agent within the system is associated with the set of roles it has to play according to the set of capacities it owns. In a second step, the strategy for selecting the current role-playing relationship is defined. It may consist in a scheduling algorithm or a rule-based or norm-based mechanism. Tasks of this activity are detailed in Table 20 and their flow is represented in Fig. 34.

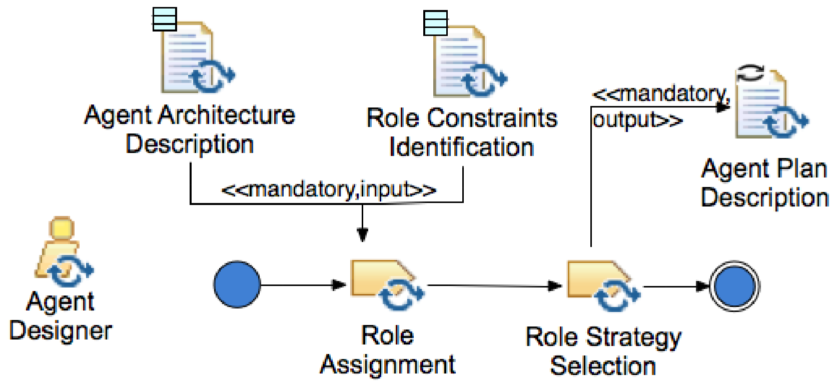


Fig. 33 The flow of tasks of the Agent Plan Description (APD) activity

Table 20 Task descriptions of the Agent Plan Description activity

Activity	**Task**	**Task Description**	**Roles involved**
Agent Plan Description	Role Assignment	Select, for each agent the roles it plays	Agent Designer (perform)
Agent Plan Description	Role Strategy Selection	Define a role strategy selection for each agent	Agent Designer (perform)

Holarchy Design

A super-holon is a community of holons that cooperate to achieve a commonly agreed objective. In order to allow a fruitful evolution of the super-holon structure at runtime, certain rules have to be defined.

Trying to enumerate all possible issues would probably result in an incomplete or domain dependent list. Conversely, we prefer to only discuss the most common and important rules that must be defined in the presented approach. The first rule is about managing **Inclusion/Exclusion of holons members**. Once a super-holon has been created, new members may request to join it or the super-holon itself may require new members to achieve its goal/task (the new member admission process is called **Merging**). Two aspects of that should be analyzed: (i) who is in charge for taking the decisions and how this happens (head, vote, etc.); (ii) how the requesting process could be started (who is to be contacted by the external holon who wants to enter the super-holon to play a role within it). The decision process for the admission of a new member can be done accordingly to several different internal policies representing different levels of involvement of the holon members' community: federation is positioned at one side of the spectrum; in this configuration, all members are equal when a decision has to be taken. Opposite to that there is the dictatorship, where heads are omnipotent; a decision taken by one of them does not have to be validated by any other member. In this government form, members loose most of their autonomy having to request permission of the head to provide a service or request a collective action.

Even if these configurations may be useful in specific domains, generally an intermediate configuration will be desirable. In order to identify the right choice, it is, firstly, necessary to analyze the functionalities that will be provided by the holon, and then define the level of authority that the Head will have.

A versatile solution to this problem consists in establishing a voting mechanism for each functionality. In order to parameterize a voting mechanism, three elements have to be defined: **Requester**, **Participants** and **Adoption Mechanism**.

The vote Requester defines which members are allowed to request for a vote; Participants are members who are authorized to participate in the voting process, and finally, the adoption mechanism defines how a proposal is accepted or rejected. This scheme can be easily modeled as an organization as shown in figure 35.

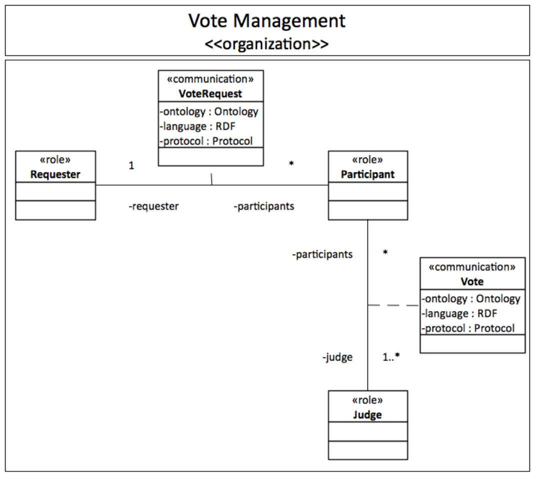


Fig. 34 Voting Organization

Solution Ontology gives important information for the definition of the holonic structure: rules and constraints are generally described in the ontology. Normally, at this step, these constraints have been already associated to each organization.

Table 21 also provides a set of question that may guide the design of holon government and the determination of rules that govern holons dynamic. Answer to these questions facilitates the definition of the future super-holons functioning: not authorized process, decisions are made by a vote, the heads take the decision, and so on.

This activity aims at refining the solution architecture, and it is composed of four tasks: firstly, the agentification task consists in identifying all the required holons and associating them with the set of roles they can play. This allows the identification of the set of capacities required by each holon thus giving precise indications on the architecture that should be adopted for it. Indeed, the holon architecture is

Related aspects	Questions to answer
The addition of a new functioning rules	• Is it possible to add new rules ? • By what process can we add of new functioning rule ?
Integration/Exclusion of a member	• How are integrated the new members ? • How to exclude a member ?
New tasks acceptance	• How are the new tasks accepted or refused ?
Choice and Acceptance of new objectives for the super-holon	• Is it possible to add new goals to the super-holon • If yes, how ?
Super-holon actions	• Who can take decisions about action selection ?
Multi-Part authorized	• Are the members authorized to join other holons? • If yes, under which conditions ?
Holon Destruction: Leaving of one of the members	• Is it possible that a part leaves the super-holon without implying its destruction ? • If yes, under which conditions ?

Table 21 A part of the Holon Government Description Template

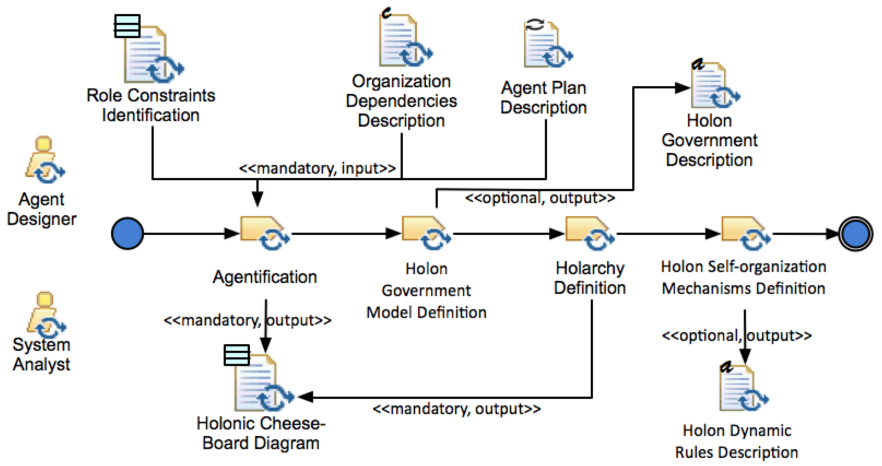

Role Constraints Identification
Organization Dependencies Description
Agent Plan Description
Holon Government Description
<<mandatory, input>>
<<optional, output>>
Agent Designer
System Analyst
Agentification
Holon Government Model Definition
Holarchy Definition
Holon Self-organization Mechanisms Definition
<<mandatory, output>>
<<optional, output>>
Holonic Cheese-Board Diagram
<<mandatory, output>>
Holon Dynamic Rules Description

Fig. 35 The flow of tasks of the Holarchy Design (HD)

at least defined by the set of roles that the holon should play, the minimal set of services that implement capacities required by its role. The second task focusses on composed holons and aims at identifying a government type for each of them. The objective consists in describing the various rules used to take decisions inside each super-holon. At this point, all the previously described elements are merged in order to obtain the complete set of holons (composed or not) involved in the solution. In this way, the complete holarchy of the system at the instance level is described.

The description obtained with the previous tasks is just the initial structure of the system, the last objective is now to specify rules that govern holons' dynamics in the system (creation, new member integration, specific self-organization mechanisms, scheduling policies for roles) in order to support a dynamic evolution of the system holarchy. Tasks of this activity are detailed in Table 22 and their flow is represented in Fig. 36.

Table 22 Task descriptions of the Holarchy Design activity

Activity	**Task**	**Task Description**	**Roles involved**
Holarchy Design	Agentification	Identification of the holons for all levels and their composition relationships	Agent Designer (perform), System Analyst (assist)
Holarchy Design	Holon Government Model Definition	For each holon, definition of a decision making mechanism	Agent Designer (perform), System Analyst (assist)
Holarchy Design	Holarchy Definition	Definition of the initial holarchies properties	Agent Designer (perform), System Analyst (assist)
Holarchy Design	Holon Self-Organization Mechanisms Definition	Definition of the rules that will describe the dynamics of holarchies	Agent Designer (perform), System Analyst (assist)

2.2.3 Work Products

The System Requirements phase generates eight work products. Their relationships with the MAS metamodel elements are described in the following Figure 37. This diagram represents the Agent Society Model in terms of Work products. Each of these latter reports one or more element from the ASPECS metamodel Agency Domain.

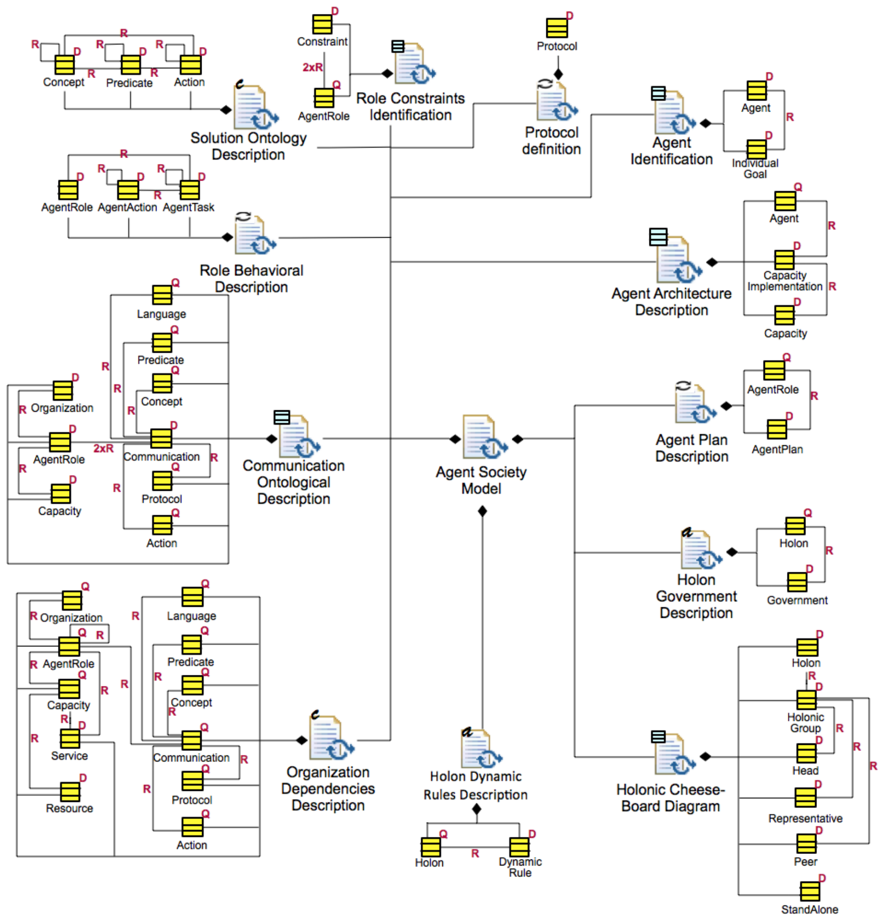


Fig. 36 The Agent Society Phase Model documents structure

Work Product Kinds

Table 23 briefly describes the nature and content of each work-product kind. Almost every work-product could be considered as a composite one as each UML diagram is associated to a structured text document detailing the content of the diagram. We consider that this documentation is part of the diagram, the nature of the work-product is thus defined with respect to the description given in the diagram.

Due to space concerns we present the three most representative examples of workproducts.

Name	Description	Work Product Kind
Solution Ontology Description (SOD)	A UML Class Diagram with a specific profile and its associated documentation describing concepts, actions, predicates and their relationships of the agent-based solution ontology	Structural & Structured Text
Agent Identification (AI)	Tropos goal and actor diagram or PROMETEHEUS agent overview diagram and their associated textual documentation to identify agents and their individual goals	Behavioral & Structured Text
Agent Architecture (AA)	A UML Class Diagram with a specific profile and its associated documentation describing attributes and behavioral methods of each agents and their personal capacities	Structural & Structured Text
Communication Ontological Description (COD)	A UML Class Diagram with a specific profile and its associated documentation describing agent's roles and their communications. Interactions, previously identified are now clustered in conversation or communications and represented by an association. Attributes of each communication (Ontology) and each conversation (Ontology, Content Language, Interaction Protocol) are specified in an association class. Each conversation is oriented from the initiator of the conversation to the other participant roles.	Structural & Structured Text
Role Behavior Description (RBD)	UML State machine Diagram describing role's internal behavior in terms of AgentActions, AgentTasks and AgentRoleStates.	Behavioral & Structured Text
Protocol Description (PD)	A UML Class Diagram with a specific profile (or AUML) and its associated documentation defining purpose-specific interaction protocols when communications' description does not match any of the existing FIPA protocols	Behavioral & Structured Text
Organization Dependencies Description (ODD)	A UML Class Diagram with a specific profile and its associated documentation describing service, resources, capacities, roles of each organization. Refinement of Capacity Identification diagram	Structural & Structured Text
Role Constraints Identification (RCI)	Refinement of ODD diagram with the additional constraints between roles to be played simultaneously.	Structural & Structured Text
Agent Plan Description (APD)	UML State Machine diagram with specific profile and its associated documentation defining agent's personal plan according to its individual motivations and pursued goals. The plan represents the strategy used by the agent to choose the roles to play. Each agent is associated to the set of roles it has to play according to the set of capacities that it owns.	Behavioral & Structured Text
Holarchy Design (HD), Holonic Cheese-board	Cheese-board diagram and its associated documentation describing the complete structure of the system mapping the hierarchy of organizations to a holarchy. This mapping is based on the association between holons composing the holarchy with the set of roles they play in the various groups (instance of organizations) composing the system	Structural & Structured Text
Holarchy Design (HD), Holon Government Description	Text document describing the various rules used to take decisions inside each super-holon: the holon decision-making process.	Free text
Holarchy Design (HD), Holon Dynamic Rules Description	Text document specifying holons self-organization mechanisms (creation, new member integration, scheduling policies for roles) in order to support a dynamic evolution of the system holarchy	Free text

Table 23 Work Product Kinds of the Agent Society Phase

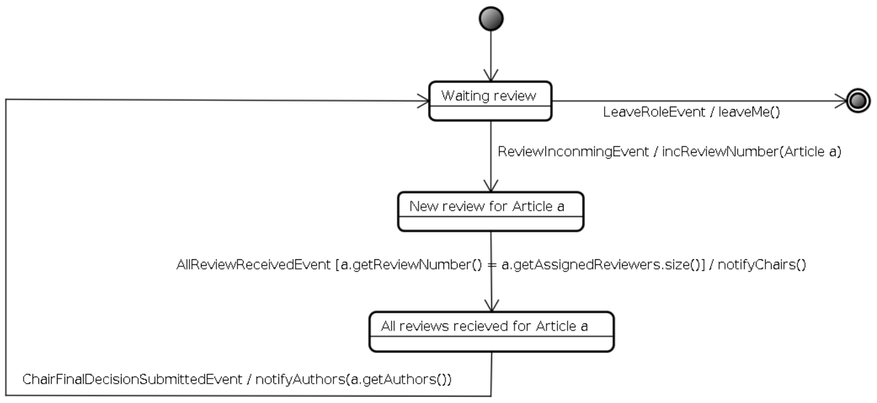

Waiting review
LeaveRoleEvent / leaveMe()
ReviewInconmingEvent / incReviewNumber(Article a)
New review for Article a
AllReviewReceivedEvent [a.getReviewNumber() = a.getAssignedReviewers.size()] / notifyChairs()
All reviews recieved for Article a
ChairFinalDecisionSubmittedEvent / notifyAuthors(a.getAuthors())

Fig. 37 Role Behavior Description diagram - a portion of the CMS case study

Role Behavior Description

This activity produces a behavioral workproduct where role behaviors are refined and detailed under the form of states and transitions. Transitions classically specify the dynamic conditions under which roles change their states. The notation is that of the UML state machine diagram for describing a single role behavior. Figure 38 shows an example of notation of the Role Behavior Diagram for the CMS case study. Roles come from the IRI diagram and their general behavior from the PD diagram.

Role Constraints Identification

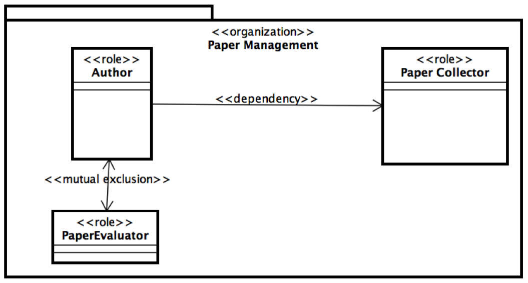


Fig. 38 Role Constraints Identification diagram - a portion of the CMS case study

This activity produces a structural workproduct that describes roles and the constraints that are to be verified at runtime. The notation used is an UML class diagram where roles are represented as classes and constraints are expressed as stereotyped relationships between roles. Figure 39 shows an example of notation of the Role

Constraints Identification diagram for the CMS case study. It is a refinement of the ODD diagram.

Holarchy Design

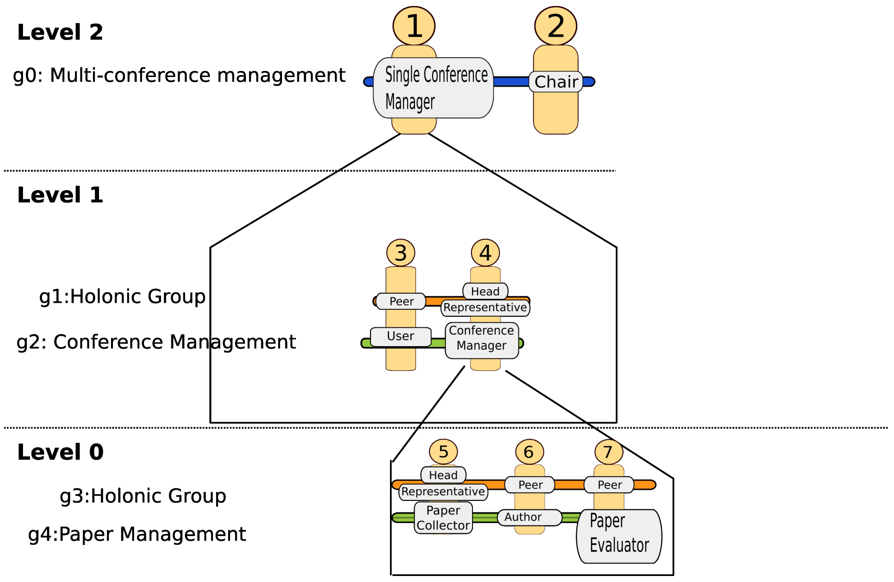


Fig. 39 Holarchy Design diagram - a portion of the CMS case study

The Holarchy Design activity produces a diagram called cheeseboard diagram that represents the different levels of the Holarchy, and, for each level, the different holons and their corresponding groups and roles. A free text description can complement this diagram to explain the adopted design choices. Figure 40 shows an example of notation of the Holarcy Design diagram for the CMS case study. Holons come from AI diagrams, groups and levels come from ODD diagrams.

2.3 Implementation and Deployment

The objectives of the Implementation and Deployment phase are to implement and deploy an agent-oriented solution resulting from the Agent Society Design phase. Due to space restrictions this phase will not be fully described according to the

FIPA Standard but only synthesized in this section. The interested reader could find informations concerning this phase in [5] and on the ASPECS website[4].

The complete sequence of activities is described in figure 41.

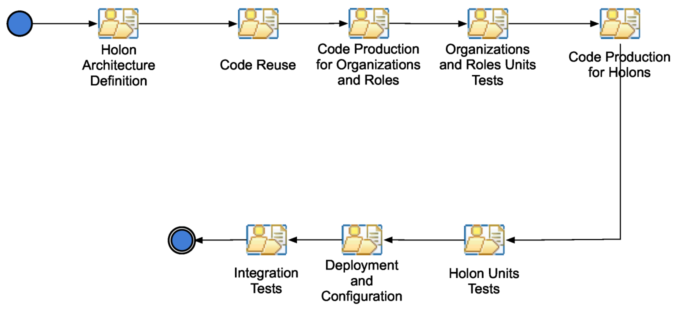

Holon Architecture Definition
Code Reuse
Code Production for Organizations and Roles
Organizations and Roles Units Tests
Code Production for Holons
Holon Units Tests
Deployment and Configuration
Integration Tests

Fig. 40 The Implementation and Deployment phase flow of activities

2.3.1 Process Roles

The roles involved in the activities and tasks of this phase are two: Developer and Tester.

2.3.2 Activity Details

Some details about the activities of this phase are synthesized in Table 24. In the follow the different objectives for each activity are introduced.

The Holon Architecture Definition activity should define the architecture of each agent/holon identified within the Agent Society Design phase. For each organization, a description detailed enough to allow implementation should be provided. The role playing relationship of each agent/holon and the sets of capacities/services are completely defined.

The Code Reuse activity aims at integrating the set of organizational patterns identified during the OID (Organization Identification) and IRI (Interactions and Roles Identification) activities and for the Holon Government identification task. The second aim is to allow the reuse of code from other existing applications.

The Code Production for Organizations and Roles activity should produce code for each organization and role. The ideal case is when the chosen implementation platform natively supports these concepts as it happens for Janus [8].

[4] `http://www.aspecs.org`

The Organizations and Roles Units Tests activity is the first level of test in ASPECS. The principle is to test at organization and role levels for each specific context.

The Code Production for Holons activity focusses on code production for each Holon. The principle is to implement, depending on the platform primitives, the results of the Holarchy Design activity from the previous phase.

The Holon Unit Test activity is the second level of test in the ASPECS process. The focus is on holon's behavior validation. Each holon is thus individually tested.

The Deployment and Configuration activity details the concrete deployment of the application. The elements to be detailed are, for example, a network configuration, holon(s) physical location, external devices, external applications, ...

Integration Tests is the third and final test activity of the ASPECS process. The testing scope is on system functionalities and on the interfaces between system parts.

2.3.3 Work Products

The Implementation and Deployment phase generates eight work products which are listed in the table 25.

Work Product Kinds

Work product kinds are briefly described in Table 25.

3 Work product dependencies

Fig. 39 and Fig. 40 describe the dependencies among the work products of the first two ASPECS phases.

References

1. Bernon, C., Cossentino, M., Pavón, J.: An overview of current trends in european aose research. Informatica **29**(4), 379–390 (2005)
2. Bresciani, P., Giorgini, P., Giunchiglia, F., Mylopoulos, J., Perini., A.: TROPOS: An Agent-Oriented Software Development Methodology. Journal of Autonomous Agents and Multi-Agent Systems **8**(3), 203–236 (2004)
3. Cossentino, M., Galland, S., Gaud, N., Hilaire, V., Koukam, A.: How to control emergence of behaviours in a holarchy. In: Self-Adaptation for Robustness and Cooperation in Holonic Multi-Agent Systems (SARC), Workshop of the second IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO). Isola di San Servolo, Venice, Italy (2008)

Table 24 Activity details for the Implementation and Deployment phase

Activity name	Task	Task description	Roles involved
Holon Architecture Definition	Static architecture definition	Each architecture designed in the Holarchy Design is detailed	Developer (perform)
	Dynamic architecture definition	A dynamic architecture is defined to take into account the Holarchy Design	Developer (perform)
Code Reuse	Organizations Pattern Integration	OID model is used to identify organizational patterns	Developer (perform)
	Role-Interactions Pattern Integration	IRI model is used to identify role and interactions patterns	Developer (perform)
	Holonic Pattern Integration	Holarchy Design is used to identify holonic patterns	Developer (perform)
	Pattern Integration	Identified patterns are merged and adapted	Developer (perform)
Code Production for Organizations and Roles	Organization, Role code production	IRI model and Role plan models are detailed with platform specific primitives	Developer (perform)
Organizations and Roles Units Tests	Organization test	SD model are used to define organizations tests	Tester (perform), Developer (assist)
	Role test	RP models are used to define roles tests	Tester (perform), Developer (assist)
Code Production for Holons	Holons Code-Production	The HD model is used to define tests dealing with rules about holon government, task management, and new members entrance	Developer (perform)
Holon Unit Test	Unit Test Definition	The HD model is used to define the most adapted implementation and platform specific primitives	Tester (perform), Developer (assist)
Deployment and Configuration	Holon Partitioning	The HD model is used to establish a partition between the various holons used to develop the application	Developer (perform)
	Dynamic Configuration Rules	The configuration specification is used to define rules for kernel distribution/integration	Developer (perform)
Integration Tests	Interfaces Tests	The HD model is used to define interface tests between sub-systems	Tester (perform), Developer (assist)
	Functionality Tests	The DRD model is used to define functionality tests	Tester (perform), Developer (assist)

Name	Description	Work Product Kind
Holarchy Architecture Definition (HAD)	A UML Class Diagram with a specific profile	Structural & Structured Text
Code Reuse (AI)	A document describing the reused code	Structured
Code Production for Organizations and Roles (CPOR)	A document describing code for organizations and roles	Structured
Organizations and Roles Unit Tests (ORUT)	A document describing tests for organizations and roles.	Structured
Code Production for Holons (CPH)	A document describing code for holons.	Structured
Holons Unit Tests (HUT)	A document describing tests for holons.	Structured
Deployment and Configuration	A UML Deployment Diagram with a specific profile representing where the agents are located, the resources and communication channels.	Structural
Integration Test (IT)	A document describing integration tests.	Structured

Table 25 Work Product Kinds of the Implementation and Deployment Phase

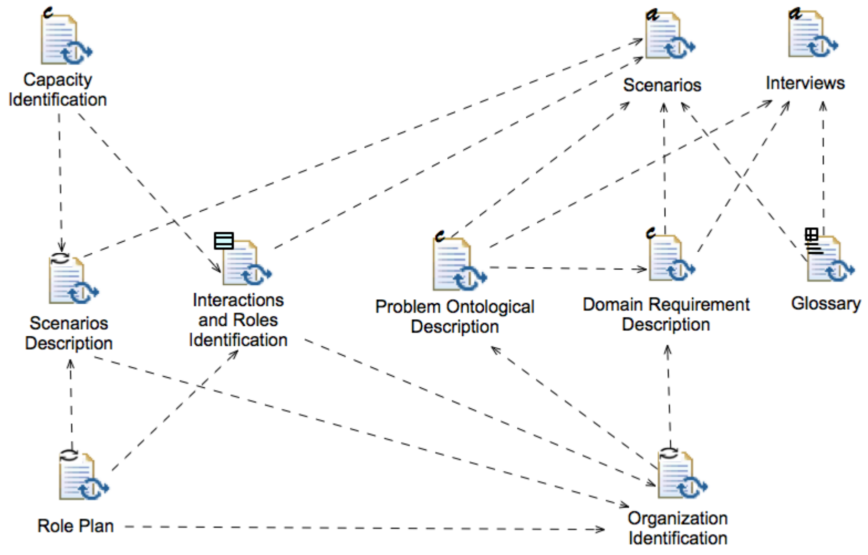

Capacity Identification
Scenarios
Interviews
Scenarios Description
Interactions and Roles Identification
Problem Ontological Description
Domain Requirement Description
Glossary
Role Plan
Organization Identification

Fig. 41 The Work Product Dependency Diagram of the System Requirements Phase

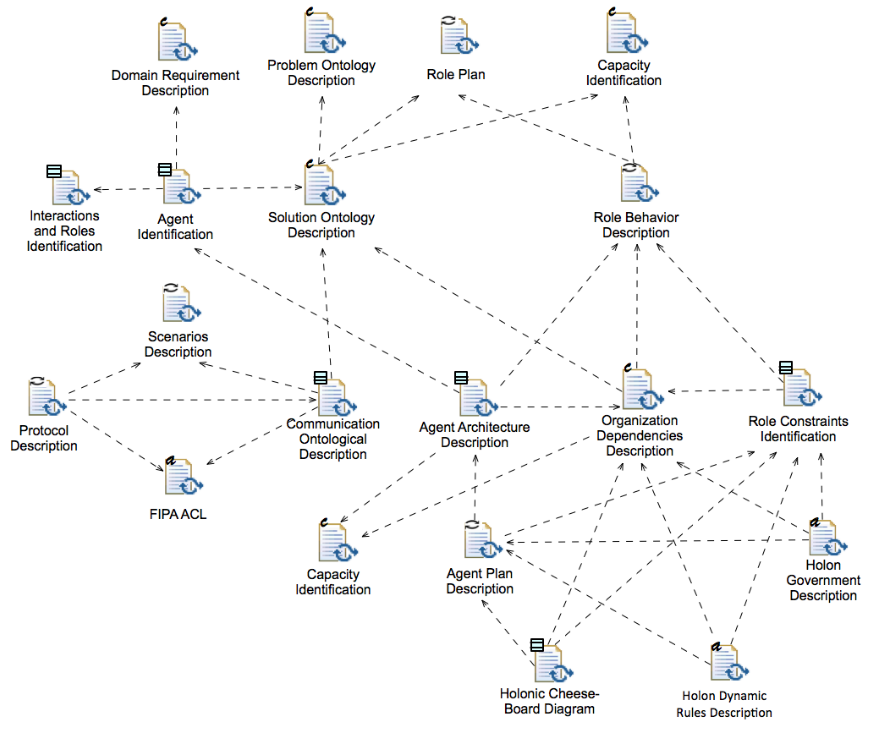

Domain Requirement Description
Problem Ontology Description
Role Plan
Capacity Identification
Interactions and Roles Identification
Agent Identification
Solution Ontology Description
Role Behavior Description
Scenarios Description
Protocol Description
Communication Ontological Description
Agent Architecture Description
Organization Dependencies Description
Role Constraints Identification
FIPA ACL
Capacity Identification
Agent Plan Description
Holon Government Description
Holonic Cheese-Board Diagram
Holon Dynamic Rules Description

Fig. 42 The Work Product Dependency Diagram of the Agent Society Phase

4. Cossentino, M., Gaud, N., Galland, S., Hilaire, V., Koukam, A.: Holomas'07. pp. 237–246. Regensburg, Germany (2007)
5. Cossentino, M., Gaud, N., Hilaire, V., Galland, S., Koukam, A.: Aspecs: an agent-oriented software process for engineering complex systems. Autonomous Agents and Multi-Agent Systems **20**(2), 260–304 (2010). DOI 10.1007/s10458-009-9099-4
6. Foundation For Intelligent Physical Agents: FIPA ACL Message Structure Specification (2002). Standard, SC00061G
7. Foundation For Intelligent Physical Agents: FIPA Communicative Act Library Specification (2002). Standard, SC00037J
8. Gaud, N., Galland, S., Hilaire, V., Koukam, A.: An Organisational Platform for Holonic and Multiagent Systems. In: PROMAS-6@AAMAS'08. Estoril, Portugal (2008)
9. Gerber, C., Siekmann, J., Vierke, G.: Holonic multi-agent systems. Tech. Rep. DFKI-RR-99-03, DFKI - GmbH (1999)
10. Gruber, T.: Toward principles for the design of ontologies used for knowledge sharing. International Journal Human-Computer Studies **43**(Issues 5-6), 907–928 (1995)
11. Object Management Group: MDA Guide, v1.0.1, OMG/2003-06-01 (2003)
12. Odell, J., Nodine, M., Levy, R.: A metamodel for agents, roles, and groups. In: J. Odell, P. Giorgini, J. Müller (eds.) AOSE, LNCS. Springer (2005)
13. Padgham, L., Winikoff, M.: Prometheus: A methodology for developing intelligent agents. In: AOSE (2002)
14. Searle, J.: Speech Acts. Cambridge University Press, Cambridge, UK (1969)
15. Simon, H.A.: The Science of Artificial, 3rd edn. MIT Press, Cambridge, Massachusetts (1996)

16. Wilber, K.: Sex, Ecology, Spirituality. Shambhala (1995). URL `http://207.44.196.94/~wilber/20tenets.html`