A glimpse of the ASPECS process documented with the FIPA DPDF template

Massimo Cossentino[1], Stéphane Galland[2], Nicolas Gaud[2], Vincent Hilaire[2], and Abderrafiaa Koukam[2]

[1]Istituto di Calcolo e Reti ad Alte Prestazioni
Consiglio Nazionale delle Ricerche
Palermo, Italy
cossentino@pa.icar.cnr.it
[2]Université de Technologie de Belfort Montbéliard.
90010 Belfort Cedex, France
vincent.hilaire@utbm.fr
(33) 384 583 009

Abstract. The FIPA DPDF working group aims to propose a definition of method fragment to be used during a situational method engineering process, the fundamental elements it is composed of and the metamodel it is based on. Using the FIPA DPDF template, this paper presents the fragments issued from the methodology ASPECS. The process of this methodology, the underlying metamodel and the workproducts related to the first phase, dedicated to system requirements analysis, are presented.

1 Introduction

It is currently admitted in mainstream software engineering and agent oriented software engineering that there is no one-size-fit-all methodology or process. Indeed, as stated in [5] "traditional rigid IS engineering methods are inadequate to provide the necessary support in new IS developments. New methods, more flexible and better adaptable to the situation of every IS development project, must be constructed".

One solution is proposed by the situational method engineering paradigm. Situational method engineering paradigm provides means for constructing ad-hoc software engineering processes following an approach based on the reuse of portions of existing design processes, the so-called method fragments, stored in a repository, called method base.

The Foundation for Intelligent Physical Agents (FIPA) is part of the IEEE Computer Society and promotes agent-based technology and the interoperability of its standards with other technology. Among the current existing FIPA sub-groups the Design Process Documentation and Fragmentation working group aims to propose a definition of method fragment to be used during a situational method engineering process, the fundamental elements it is composed of and the metamodel it is based on.

The result of the work of the working group members is the definition of a template in order to document method fragments [**?**]. This paper illustrates the use of this template for a specific methodology, namely ASPECS [1].

The structure that follows respect the FIPA DPDF template. The section 2 introduces ASPECS with it global process and the metamodel which defines the underlying concepts of the methodology. After this initial section the FIPA DPDF template contains a section per phase of the process. Due to the lack of space only a part of the first phase of ASPECS is described in section 3. Eventually section 4 concludes.

2 Documented introduction to ASPECS

2.1 Global process overview

The ASPECS life cycle consists of three phases that are explained below and illustrated by the figure 1. The **System Requirements** phase aims at identifying a hierarchy of organisations, whose global behaviour may fulfil 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 organisations. Each organisation will therefore be responsible for exhibiting a behaviour that fulfils the requirements it is responsible for. This activity is called Organisation Identification and it produces an initial hierarchy of organisations that will later be extended and updated, with further iterations, in order to obtain the global organisation hierarchy representing the system structure and behaviour. The behaviour of each organisation is realised by a set of interacting roles whose goals consist in contributing to the fulfilment of (a part of) the requirements of the organisation within which they are defined. In order to design modular and reusable organisation models, roles are specified without making any assumptions on the structure of the agent that 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 behaviour and these skills are modelled by means of a capacity. Besides, an entity that wants to play a role has to be able to provide a concrete realisation 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 behaviour is able to 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 among 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 organisation structure is mapped into a holarchy (hierarchy of holons) in charge of realising the expected behaviours. Each of the previously identified organisations 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 (that may be decomposed in two phases), 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* [4]. Secondly, it aims at detailing how to deploy the application over various computational nodes. 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 system 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 modelling approaches (i.e. object-oriented ones) with parts designed with ASPECS.

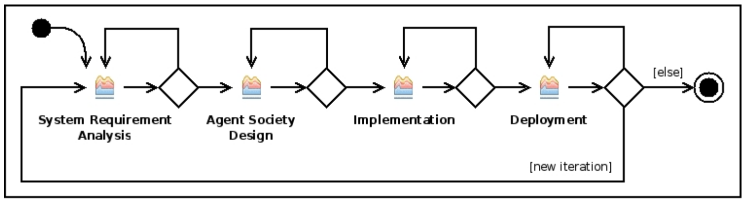

Fig. 1. ASPECS phases

2.2 Metamodel

The Problem Domain metamodel (see Figure 2), describing the concepts of the first phase, 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 organisation that fulfils them. An organisation is composed of Roles, which are interacting within scenarios while executing their Role plans. An organisation has a context that is described in terms of an ontology. Roles participate to the achievement of their organisation goals by means of their Capacities. In this subsection we will discuss the three most important elements of this domain: organisation, role, capacity. Definitions of all ASPECS metamodels can be found in [1] and on the ASPECS website[1].

An organisation is defined by a collection of roles that take part in systematic institutionalised patterns of interactions with other roles in a common context. This context consists in a shared knowledge, social rules/norms, social feelings, and it is defined according to an ontology. The aim of an organisation is to fulfil some requirements. An organisation can be seen as a tool to decompose a system and it is structured as an aggregate of several disjoint partitions. Each organisation aggregates several roles and it may itself be decomposed into sub-organisations.

In our approach, a Role defines an expected behaviour as a set of role tasks ordered by a plan, and a set of rights and obligations in the organisation context. The goal of each Role is to contribute to the fulfilment of (a part of) the requirements of the organisation within which it is defined.

In order to cope with the need of modelling system boundaries and system interactions with the external environment, we introduced two different types of roles: Common Role and Boundary Role. A Common Role is located inside the designed system and interacts with either Common or Boundary Roles. A Boundary Role is located at the boundary between the system and its environment and it is responsible for interactions happening at this border (i.e. GUI, Database wrappers, etc).

Roles use their capacities for participating to organisational goals fulfilment; a Capacity is a specification of a transformation of a part of the designed system or its environment. This transformation guarantees resulting properties if the system satisfies a set of constraints before the transformation. It may be considered as a specification of the pre- and post-conditions of a goal achievement. This concept is a high level abstraction that proved to be very useful for modelling a portion of the system capabilities without making any assumption about their implementations as it should be at the initial analysis stage.

A Capacity describes what a behaviour is able to do or what a behaviour may require to be defined. As a consequence, there are two main ways of using this concept:

– it can specify the result of some role interactions, and consequently the results that an organisation as a whole may achieve with its behaviour. In this sense, it is possible to say that an organisation may exhibit a capacity.
– capacities may also be used to decompose complex role behaviours by abstracting and externalising a part of their tasks into capacities (for instance by delegating these tasks to other roles). In this case the capacity may be considered as a behavioural building block that increases modularity and reusability.

[1] http://janus-project.org

In order to complete the description of the possibilities offered by the application of our definitions of Organisation, Roles and Capacity, let us consider the need of modelling a complex system behaviour. We assume it is possible to decompose it from a functional point of view, and in this way we obtain a set of more finer grained (less complex) behaviours. Depending on the considered level of abstraction, an organisation can be seen either as a unitary behaviour or as a set of interacting behaviours. The concept of organisation is inherently a recursive one [2]. The same duality is also present in the concept of holon as it will be shown later in this article. Both are often illustrated by the same analogy: the composition of the human body. The human body, from a certain point of view, can be seen as a single entity with an identity, its own behaviour and personal emotions. Besides, it may also be regarded as a cluster/aggregate of organs, which are themselves made up of cells, and so on. At each level of this composition hierarchy, specific behaviours emerge. The body has an identity and a behaviour that is unique for each individual. Each organ has a specific mission: filtration for kidneys, extraction of oxygen for lungs or blood circulation for the heart. An organisation is either an aggregation of interacting behaviours, and a single behaviour composing an organisation at an upper level of abstraction; the resulting whole constitutes a hierarchy of behaviours that has specific goals to be met at each level. This recursive definition of organisation will form the basis of the analysis activities performed within ASPECS. In most systems, it is somewhat arbitrary as to where we leave off the partitioning and what subsystems we take as elementary (cf. [6, chap. 8]). This remains a pure design choice.

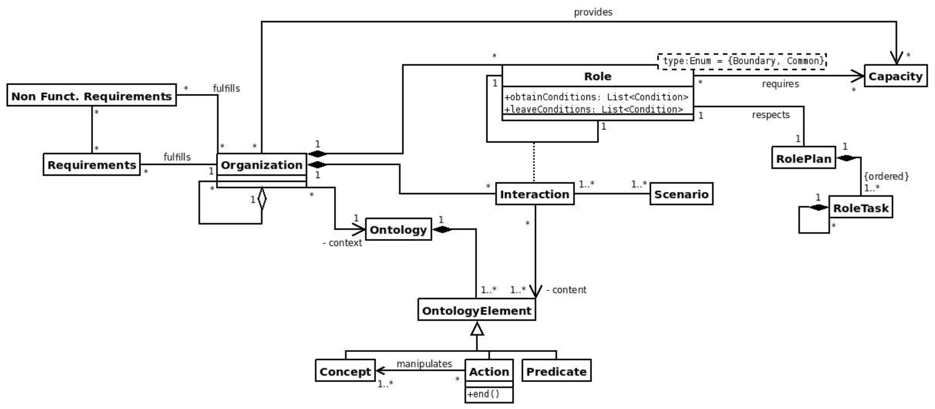

Fig. 2. Metamodel of the ASPECS problem domain

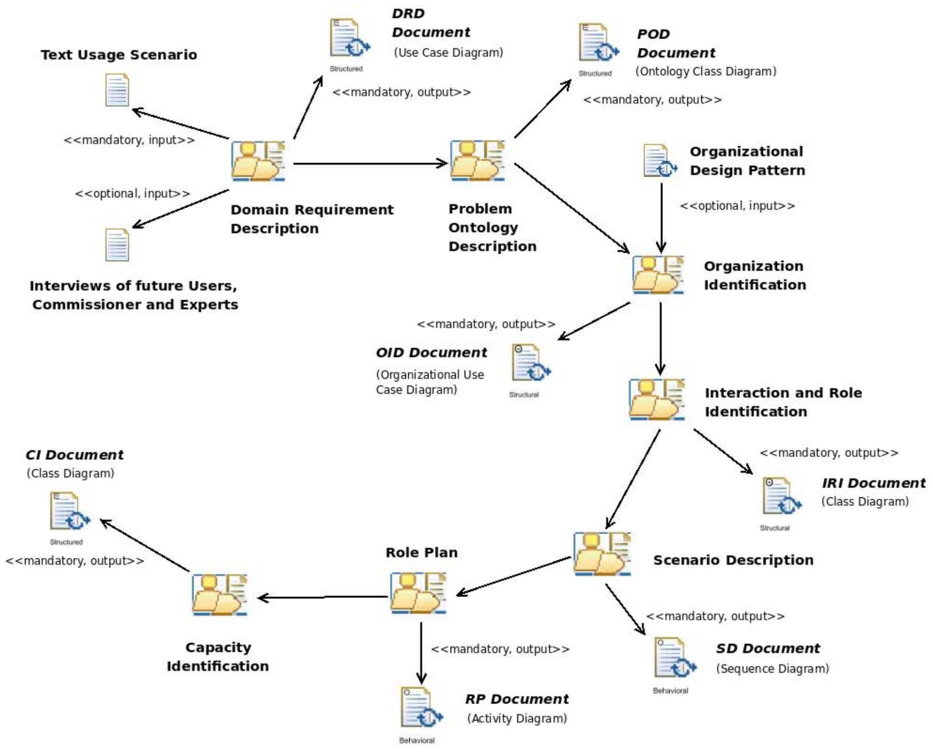

Fig. 3. System Requirements Phase: activities and workproducts

3 Phase: Domain

3.1 Process roles

Two roles are involved in the System Requirements discipline: 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 conceptualisation of the problem during the Problem Ontology Description (POD) activity.
4. Use cases clustering during the Organisation 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 organisations in charge of their realisation.
5. Identification of interacting roles for the previously identified organisations 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 behaviours during Role Plan (RP) activity by means of state-transition diagrams specifying each role behaviour.
8. Identification of capacities that are required by roles or provided by the organisations during the Capacity Identification (CI) activity. The capacities are added to the class diagram depicting the organisations composed of interacting roles.

Domain expert The domain expert has knowledge about the domain of the problem to be solved and is able to decide if the requirements are identified (end of the Domain Requirements Phase).

3.2 Activity details

Domain Requirement Description (DRD) The global objective of the Domain Requirements Description (DRD) activity is gathering needs and expectations of application stake-holders and providing a complete description of the behaviour 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; besides, conventional text annotations are applied to use cases documentation for describing

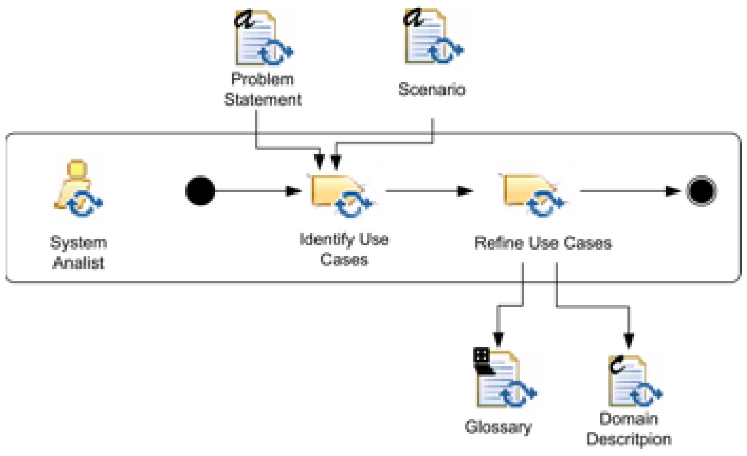

Fig. 4. Domain Requirement Description activity

non-functional requirements. In ASPECS, we advocate the use of a combination between use-case driven and goal-oriented requirements analysis where the description of functional requirements is completed by the one of associated goals and goal failures

Table 1. ASPECS Domain Requirement Description tasks

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

Organisation Identification (OID) The goal of the Organisation Identification activity is to bind each requirement to a global behaviour, embodied by an organisation. Each requirement is then associated to a unique organisation in charge of fulfilling it. As already said, an organisation is defined by a set of roles, their interactions and a common context. The associated context is defined according to a part of the Problem Ontology, described in the previous activity. Starting from use cases defined in the DRD activity, different approaches could

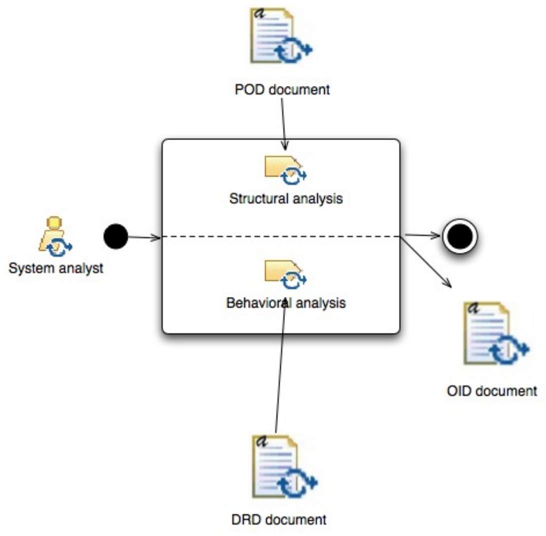

Fig. 5. Organisation Identification activity

be used to cluster them and identify organisations. We advocate the use of a combination between a structural (or ontological) approach mainly based on the analysis of the problem structure described in the POD and a functional approach based on requirement clustering.

Structural analysis focuses on the identification of the system structure. It is mainly based on the association between use cases and related ontological concept. In structural organisation identification, use cases that deal with the same ontological concepts are often put together in the same organisation. This approach assumes the same knowledge is probably shared or managed by the different members of the organisation. The structure of the ontology itself can often constitute a good guideline to identify organisations, their composition relationships, and later their roles.

Behavioural analysis aims at identifying a global behaviour for the organisation intended to fulfil the requirements described in the corresponding use case diagram. The set of organisation roles and their interactions have to generate this higher-level behaviour. For this task, the use of *Organisational Design Patterns* may be useful to the designer. In behavioural organisation identification, use cases dealing with related pieces of the system behaviour are grouped (for instance an use case and another related to it by an include relationship). This means that members of the same organisation share similar goals.

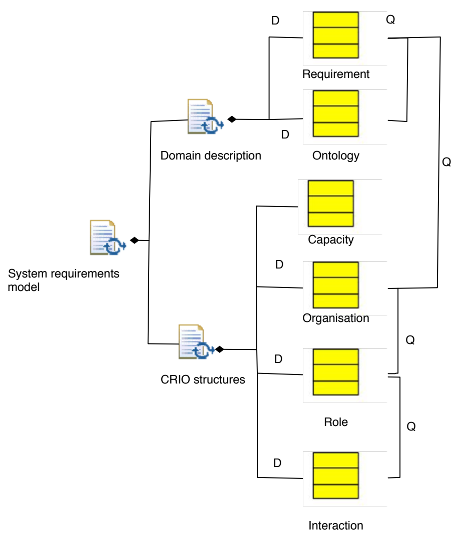

Fig. 6. ASPECS System Requirements Workproducts

Table 2. ASPECS Workproduct kinds

Name	**Description**	**Workproduct kinds**
DRD document	A text document composed by the Domain Description diagram, a documentation of use cases reported in it and the non-functional requirements of the system	Composite (Structured + Behavioural)
POD document	An ontology in the form of a class diagram stereotyped according to [3]	Structured
OID document	A class diagram reporting use cases and organisations as packages	Composite (Structured + Behavioural)
IRI document	A stereotyped class diagram	Structured
SD document	A stereotyped sequence diagram	Behavioural
RP document	An activity diagram	Behavioural
CI document	A stereotyped class diagram	Structured

3.3 Workproducts

The global objective of the Domain Requirements Description (DRD) activity is gathering needs and expectations of application stake-holders and providing a complete description of the behaviour 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; besides, conventional text annotations are applied to use cases documentation for describing non-functional requirements.

The global objective of the Problem Ontology Description is to provide an overview of the problem domain. Problem ontology is modelled by using a class diagram where concepts, predicates and actions are identified by specific stereotypes.

The workproduct of the Organisation Identification activity (OID) refines the use case diagram produced by the DRD activity and add organisations as packages encapsulating the fulfilled use cases.

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 organisations and relationships describe interactions among roles or contributions (to the achievement of a goal) from one organisation to another.

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 organisation it belongs to.

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 its swimlane and interactions with other roles are depicted in form of signal events or object flows corresponding to exchanged messages.

The workproduct produced by the Capacity Identification is a refinement of the IRI diagram by adding capacities (represented by classes) and relating them to the roles that require them.

4 Conclusion

This paper has presented the use of the FIPA DPDF working group template with a specific MAS methodology, namely ASPECS [1]. Only the first of the three phases composing ASPECS is presented and in this phase two activities are detailed. The aim were twofold, first to prove the usability of the FIPA DPDF template and second to show a glimpse of the fragmentation of the ASPECS methodology. For more details about the methodology can consult either [1]or the ASPECS website[2].

[2] http://aspecs.org

References

1. Massimo Cossentino, Nicolas Gaud, Vincent Hilaire, Stéphane Galland, and Abderrafiaa Koukam. ASPECS: an agent-oriented software process for engineering complex systems. *Autonomous Agents and Multi-Agent Systems*, 20(2):260–304, march 2010.
2. Jacques Ferber. *Multi-Agent Systems. An Introduction to Distributed Artificial Intelligence.* Addison Wesley, London, 1999.
3. FIPA. Fipa rdf content language specification. Technical Report XC00011B, 2001.
4. Nicolas Gaud, Stéphane Galland, Vincent Hilaire, and Abderrafiâa Koukam. An Organisational Platform for Holonic and Multiagent Systems. In *PROMAS-6@AAMAS'08*, Estoril, Portugal, May 12-16th 2008.
5. Jolita Ralyté and Colette Rolland. An approach for method reengineering. *Lecture Notes in Computer Science*, 2224:471–??, 2001.
6. Herbert A. Simon. *The Science of Artificial.* MIT Press, Cambridge, Massachusetts, 3rd edition, 1996.