Domain Requirements Description

Process Fragment

Author: M. Cossentino, V. Seidita
Last saved on: 18/10/10 13:26

Index

Fragment Description

Fragment Goal

Representing system functional and non-functional requirements

Fragment Origin

The presented fragment has been extracted from *PASSI* (Process for Agent Societies Specification and Implementation) design process.

PASSI (Process for Agent Societies Specification and Implementation) is a step-by-step requirement-to-code methodology for designing and developing multi-agent societies. The methodology integrates design models and concepts from both Object-Oriented software engineering and artificial intelligence approaches.

PASSI has been conceived in order to design FIPA-compliant agent-based systems, initially for robotics and information systems applications.

Systems designed by using the PASSI process are usually composed of peer-agents (although social structures can be defined). According to FIPA specifications agents are supposed to be mobile, and they can interact by using semantic communications referring to an ontology and an interaction protocol.

PASSI is suitable for the production of medium-large MAS (up to a hundred agent-kinds each one instantiated in an unlimited number of agents in the running platform).

The adoption of patterns and the support of specific CASE tools (PTK) allows a quick and affordable production of code for the JADE platform. This encourages the use of this process even in time/cost-constrained projects or where high quality standards have to be met.

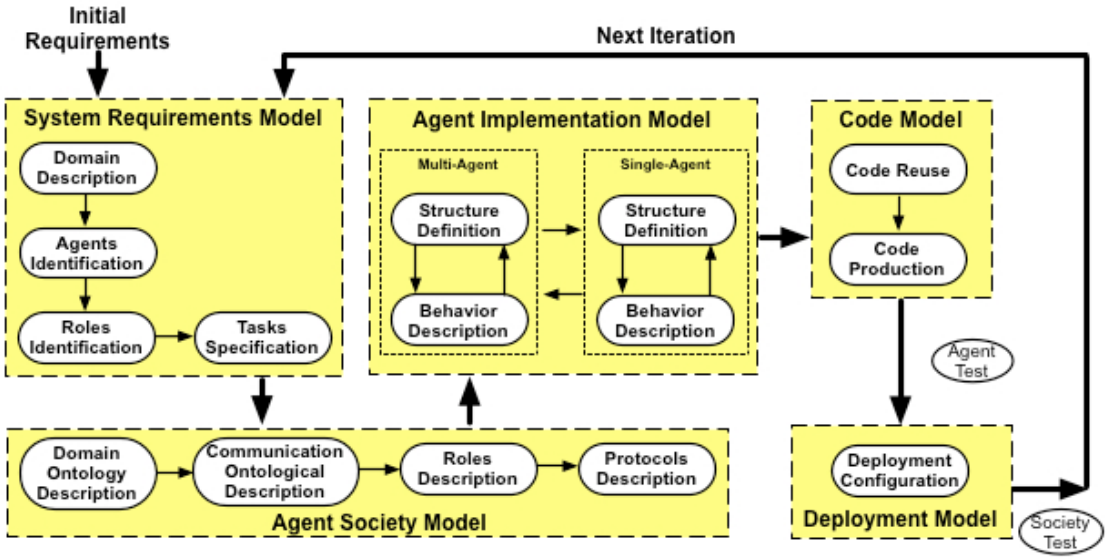

Figure 1. The PASSI design process

The design process is composed of five models (see Figure 1): the System Requirements Model is a model of the system requirements; the Agent Society Model is a model of the agents involved in the solution in terms of their roles, social interactions, dependencies, and ontology; the Agent Implementation Model is a model of the solution architecture in terms of classes and methods (at two different levels of abstraction: multi and single-agent); the

Code Model is a model of the solution at the code level and the Deployment Model is a model of the distribution of the parts of the system (i.e. agents) across hardware processing units, and their movements across the different available platforms.

Useful references about the PASSI process are the following:

- M. Cossentino. From Requirements to Code with the PASSI Methodology. In Agent-Oriented Methodologies, B. Henderson-Sellers and P. Giorgini (Editors). Idea Group Inc., Hershey, PA, USA. 2005.
- M. Cossentino, S. Gaglio, L. Sabatucci, and V. Seidita. The PASSI and Agile PASSI MAS Meta-models Compared with a Unifying Proposal. Lecture Notes in Computer Science, vol. 3690. Springer-Verlag GmbH. 2005. pp. 183-192.
- M. Cossentino and L. Sabatucci. Agent System Implementation in Agent-Based Manufacturing and Control Systems: New Agile Manufacturing Solutions for Achieving Peak Performance. CRC Press, April 2004.
- M. Cossentino, L. Sabatucci, and A. Chella. Patterns reuse in the PASSI methodology. In Engineering Societies in the Agents World IV, 4th International Workshop, ESAW 2003, Revised Selected and Invited Papers, volume 3071 of Lecture Notes in Artificial Intelligence. Springer-Verlag, 2004. pp. 294-310
- M. Cossentino, L. Sabatucci, A. Chella - A Possible Approach to the Development of Robotic Multi-Agent Systems - IEEE/WIC Conf. on Intelligent Agent Technology (IAT'03). October, 13-17, 2003. Halifax (Canada)
- Chella, M. Cossentino, and L. Sabatucci. Designing JADE systems with the support of case tools and patterns. Exp Journal, 3(3):86-95, Sept 2003.

The PASSI Process lifecycle

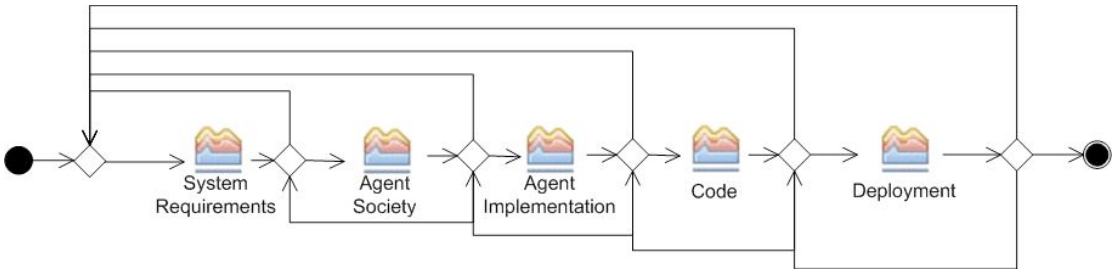

Figure 2. The PASSI process phases

PASSI includes five phases (see Figure 2) arranged in an iterative/incremental process model:

- System Requirements: It covers all the phases related to Req. Elicitation, analysis and agents/roles identification
- Agent Society: All the aspects of the agent society are faced: ontology, communications, roles description, Interaction protocols
- Agent Implementation: A view on the system's architecture in terms of classes and methods to describe the structure and the behavior of single agent.
- Code: A library of class and activity diagrams with associated reusable code and source code for the target system.
- Deployment: How the agents are deployed and which constraints are defined/identified for their migration and mobility.

Each phase produces a document that is usually composed aggregating UML models and work products produced during the related activities. Each phase is composed of one or more sub-phases each one responsible for designing or refining one or more artefacts that are part of the corresponding model. For instance, the System Requirements model includes an agent identification diagram that is a kind of UML use case diagrams but also some text documents like a glossary and the system use scenarios.

Fragment Overview

Consider the PASSI process (Figure 1) and the "System Requirements" phase with its outcome "System Requirements Model". Now, consider the "Domain Requirements Description" (red colored in Figure 3) activity and the consequent outcome (the "Communication Ontological Description" composite document).

This activity aims to model the social interactions and dependencies among the agents involved in the solution and to face the following agent society aspects are faced: communication and role description. The activity and its main outcome has been considered for being extracted from PASSI and for becoming a process fragment.

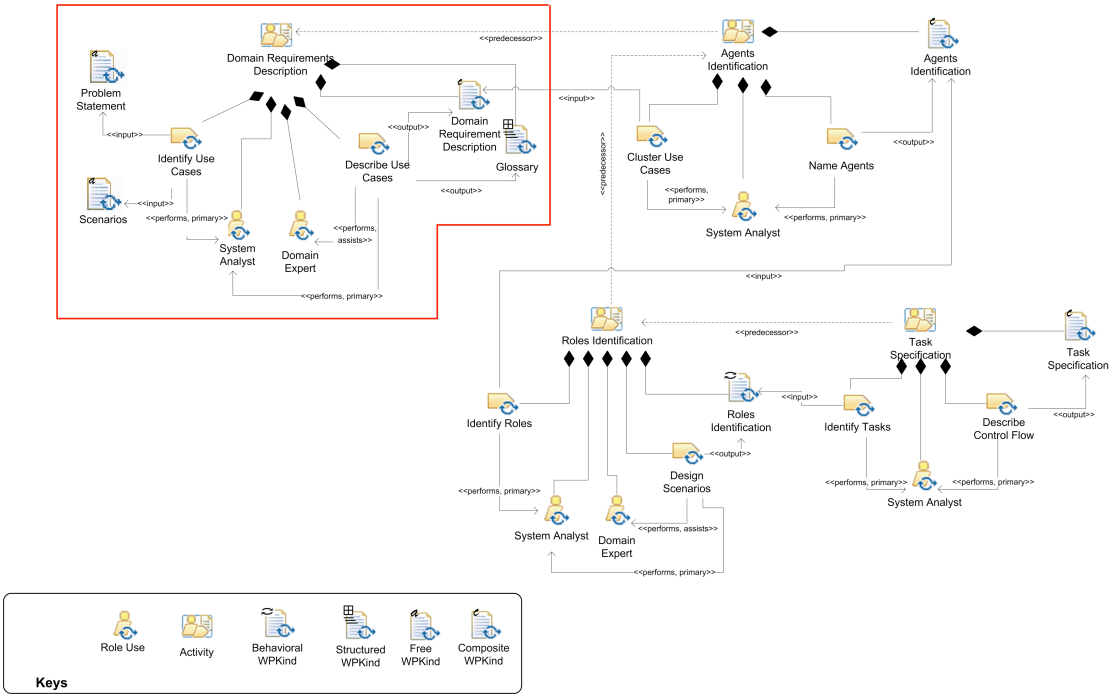

Figure 3. The System Requirements Phase (structural view)

Fragment System metamodel

The portion of metamodel of this fragment is:

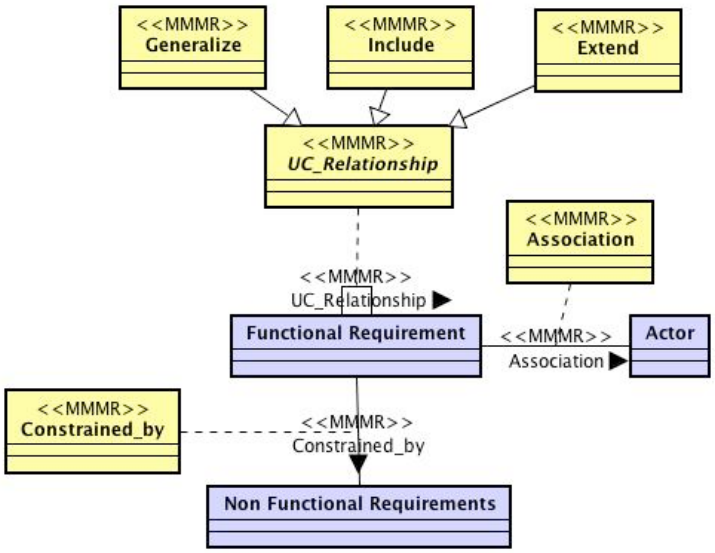

Figure 4. The fragment system metamodel

This fragment refers to the MAS metamodel adopted in PASSI and contributes to define and describe the elements reported in Figure 4.

Definition of System metamodel elements

This fragment underpins the following model elements:

Functional requirement - Functional requirements describe the functions that the software is to execute. (from IEEE SEBOK 2004)
Non-Functional requirement - Non functional requirements constrain the solution and are sometimes known as constraints or quality requirements. (from IEEE SEBOK 2004)
Actor - An external entity (human or system) interacting with the multi-agent system.

Definition of System metamodel relationships

Generalize – (see UML definition)
Include – (see UML definition)
Extend – (see UML definition)
Association – (see UML definition)
Constrained_by – It relates a Functional Requirement to a Non Functional Requirement. It means that a functionality of the system has to be realised under some non functional constraints (see FURPS+ to have examples of non functional requirements).

System metamodel Input/Output

Input, output system metamodel elements to be designed in the fragment are detailed in the following tables.

As regards system metamodel elements:

Input		**To Be Designed**		**To Be Refined**		**To Be Quoted**	
MMME	MMMR	MMME	MMMR	MMME	MMMR	MMME	MMMR
Scenario		Actor	Functional Requirement-Functional Requirement (Generalize, Include, Extend)				
		Functional Requirement	Functional Requirement-Actor (Association)				
		Non Functional Requirement	Functional Requirement- Non Functional Requirement (Constrained By)				

Definition of input system metamodel elements and relationships

Scenario: “A narrative description of what people do and experience as they try to make use of computer systems and applications” [M. Carrol, Scenario-based Design, Wiley, 1995]

Stakeholder

Roles involved in this fragment are:

- System Analyst
- Domain Expert

Their responsibilities are described in the following subsections.

System Analyst

He is responsible for:

1. Use cases identification
2. Use cases refinement. Use cases are refined with the help of a Domain Expert.

Domain Expert

1. He supports the system analyst during the description of the domain requirements.

Fragment workflow

Workflow description

The process that is to be performed in order to obtain the result is represented in the following as a SPEM2.0 diagram.

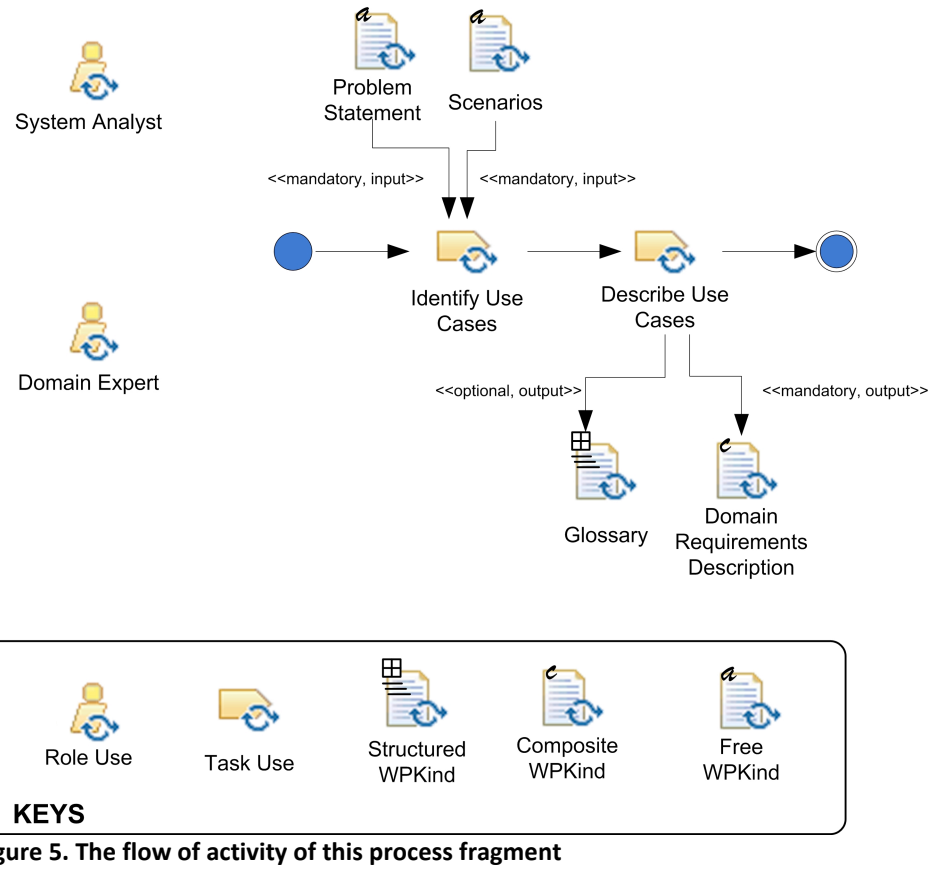

Figure 5. The flow of activity of this process fragment

Activity description

The fragment encompasses the following *work breakdown elements*:

Name	Kind	Description	Roles involved
Identify Requirements	Task	It consists in identifying use cases in order to represent the system requirements.	System Analyst (performs)
Describe Requirements	Task	Use cases are described with the help of a Domain Expert	System Analyst (performs) Domain Expert (assists)

System metamodel elements and relationships input/output

The above described *work breakdown elements* have the following input/output in terms of system metamodel components.
In the Input column, system metamodel components utilization is completed by the name of the input document reporting them in the original design process.

	Input		Output	
Activity/Task Name	**MMME**	**MMMR**	**MMME**	**MMMR**
Identify Requirements	Scenario (in Scenarios document)			
Describe Requirements			Actor, Functional Requirement, Non Functional Requirement.	Functional Requirement-Functional Requirement (Generalize, Include, Extend)
				Functional Requirement-Actor (Association)
				Functional Requirement- Non Functional Requirement (Constrained By)

WP Input/Output

Input, output work products to be designed in the fragment are detailed in the following tables.

Input	Output
Problem Statement	System Requirements document
Scenarios	Glossary

Deliverable

Domain Requirements Description Document

This fragment produces a composite document composed of use case diagrams and portions of (structured) text containing the complete documentation of the use cases in terms of: name, participating actors, entry condition, flow of events, exit condition, exceptions and special requirements.
It also reports the non functional requirements identified for the system and associated to each use case.

Domain Requirements Description Diagram: example of notation

Common UML use case diagram(s) are used to represent the system requirements.

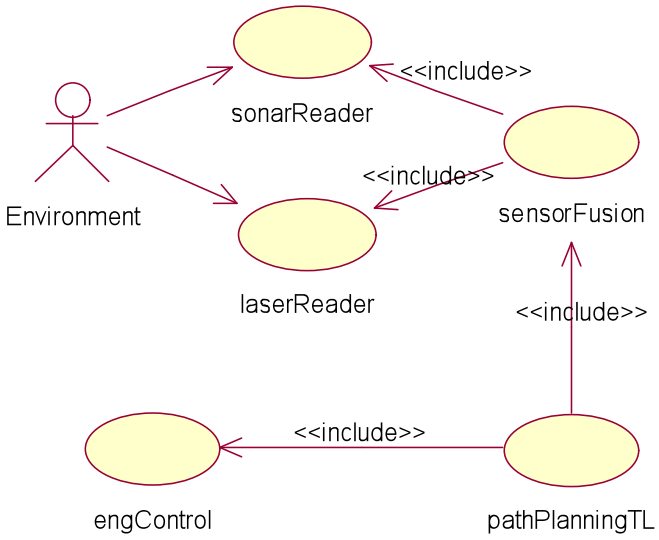

Figure 6. An example of Domain Requirements Description diagram

Deliverable relationships with the MMM

The following figure describes the structure of this fragment work products in relationship with the MAS model elements:

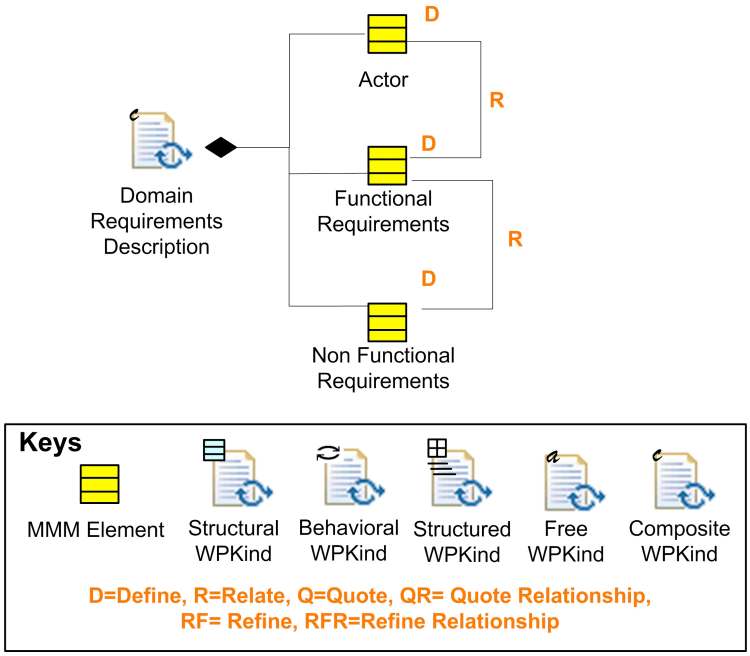

Figura 7. Structure of the fragment work-product in terms of system metamodel elements

Guidelines

Enactment Guidelines

--

Reuse Guidelines

Composition

--

Dependency Relationship with other fragments

In most approaches, this fragment is intended to be the first of the design process but a requirements elicitation fragment can be adopted before this.

References

[1] M. Cossentino. From Requirements to Code with the PASSI Methodology. In Agent-Oriented Methodologies, B. Henderson-Sellers and P. Giorgini (Editors). Idea Group Inc., Hershey, PA, USA. 2005

[2] http://pa.icar.cnr.it/passi/