Task Specification

Process Fragment

Author(s): M. Cossentino, V. Seidita

Last saved on: 28/09/10 14:18

Index

Fragment Description

Fragment Goal

Describing the behaviour of each agent (agent's plan) by considering its activities and communications.

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.

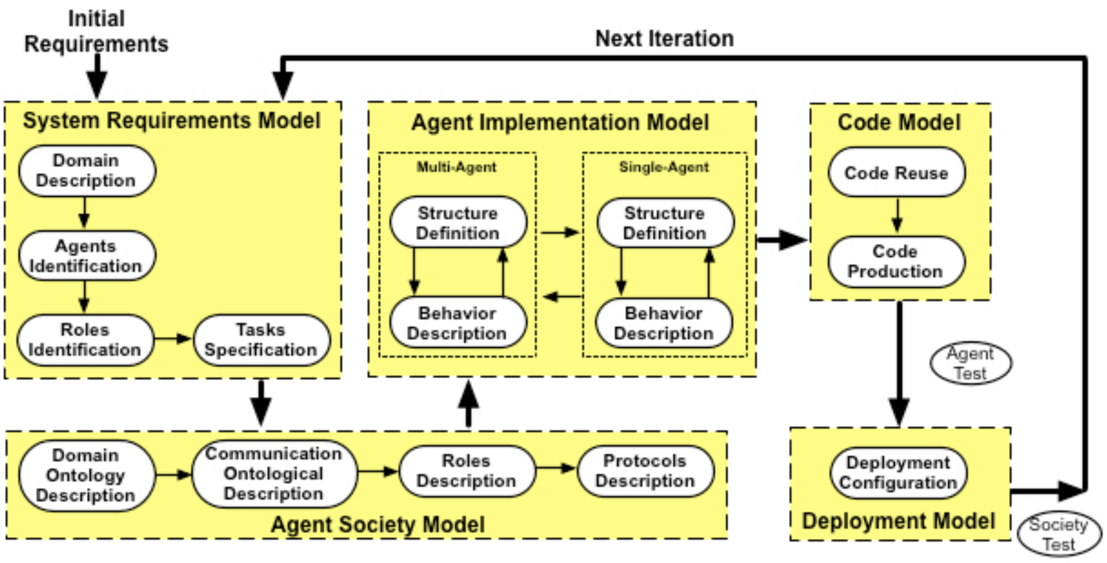

Figure 1. The PASSI design process

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 Process lifecycle

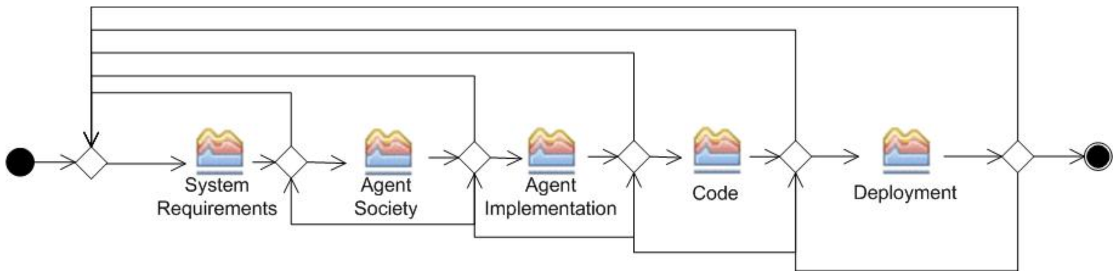

Figure 2. The PASSI process phases

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

This fragment aims to describe the behaviour of each agent. One different diagram is designed for each agent whose behaviour is modelled in terms of flow of activities and communication as a plan. The UML Model of this portion of process, Task Specification Diagram, can be designed by adopting a standard UML notation (Activity diagram).

Located inside the PASSI System Requirement phase reported in the following Figure 3, this fragment includes the activity "Task Specification" (red box)

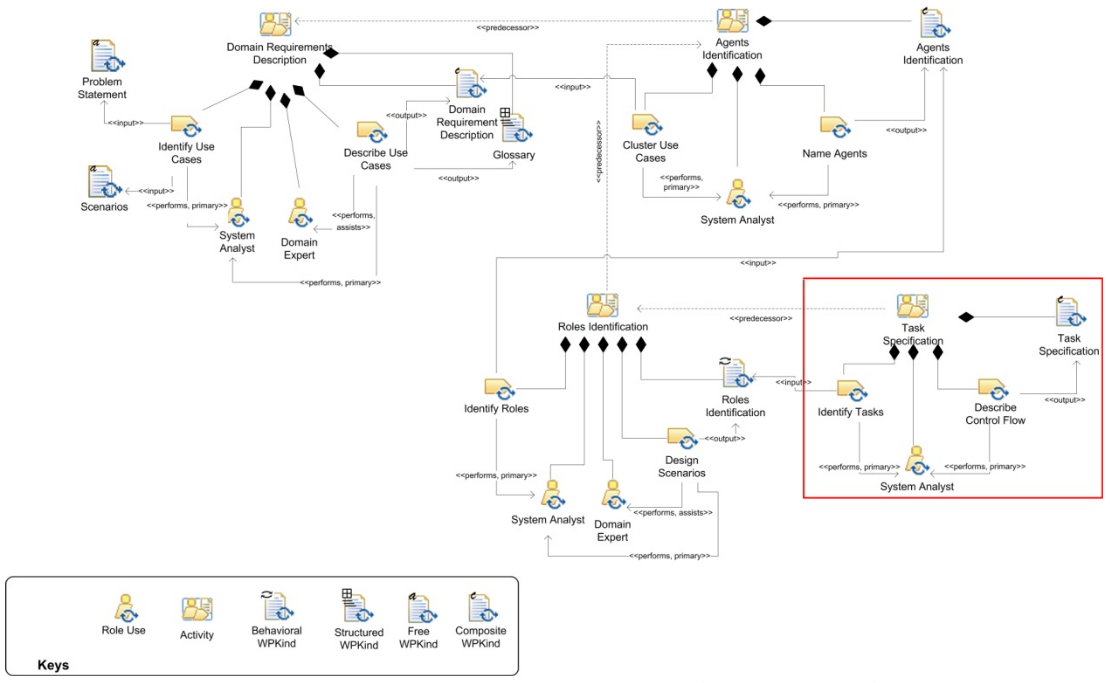

Figure 3. ***The System Requirements Phase (structural view)***

Fragment System metamodel

The portion of metamodel of this fragment is:

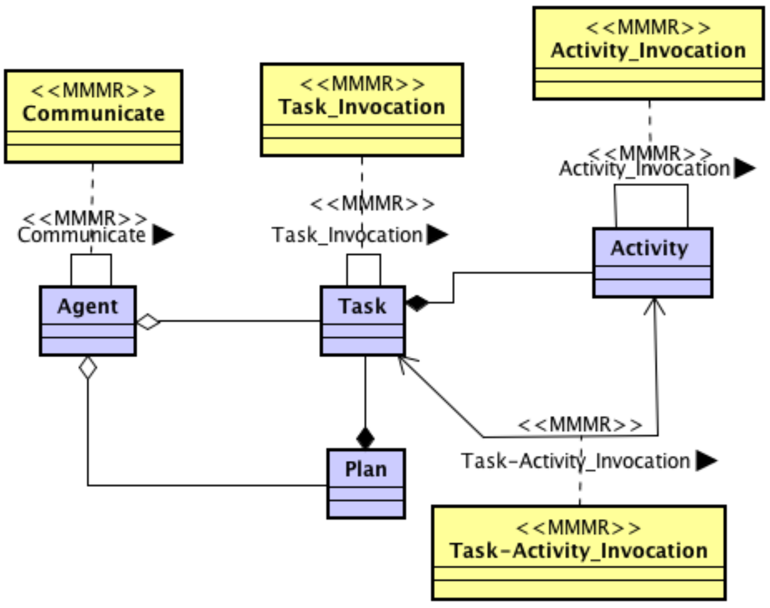

Figure 4. The fragment MAS metamodel

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 and Relationships

Element	Type	Definition
Task	MMME	A task specifies the computation that generates the effects of the behavioural feature. Its granularity addresses the significance of a non decomposable group of atomic actions that cannot be directly addressed without referring to their belonging task.
Activity	MMME	The composing unit of a task. An activity takes a set of inputs and converts them into a set of outputs, though either or both sets may be empty. An Activity can either be decomposable in other activities or atomic.
Plan	MMME	The behaviour of an Agent is specified within its plan. It is the description of how to combine and order Tasks and interactions to fulfil a (part of a) requirement.
Communicate	MMMR	The communication between two agents is at this level of abstraction still not completely defined. It can be composed by one or more atomic messages (for instance Message_RR messages defined in RID PASSI fragment)
Task_Invocation	MMMR	Flow of control within the Plan uses Task_Invocation messages to enable Tasks invocation from other Tasks
Activity_Invocation	MMMR	Flow of control within the Plan uses Activity_Invocation messages to enable Activities invocation from other Activities
Task-Activity_Invocation	MMMR	Flow of control within the Plan uses Task-Activity_Invocation messages to enable Tasks or Activities invocation from other Tasks or Activities

Keys:
MMME= MAS Metamodel Element
MMMR= MAS Metamodel Relationship

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
Role	Role-Role (Message_RR)	Task	Agent-Agent (Communicate)			Agent	
Agent	Role-Actor (Message_RA)	Activity	Task-Task (Task_Invocation)				
Actor	Role-Agent (Plays)	Plan	Agent-Task (aggregate)				
			Task-Activity (Composite)				
			Task-Plan (Composite)				
			Agent-Plan (Aggregate)				
			Activity-Activity (Activity_Invocation)				
			Task-Activity (Task-Activity_Invocation)				

Definition of input system metamodel elements and relationships

Actor - An external entity (human or system) interacting with the multi-agent system.
Agent - A system requirements domain agent.
PASSI considers three different levels of abstraction in the agent definition:
1) the system requirements domain agent is a responsibility center; this means that each agent will rationally act to achieve its goals (usually defined in terms of functionalities it should ensure).
2) during the Agency Domain design phases, the agent is an autonomous entity capable of pursuing an objective through its autonomous decisions, actions and social relationships.
3) In the solution domain phases, an agent is a significant software unit, each agent is an instance of an agent class.

Generally speaking, an Agent is an entity which:
- is capable of actions in an environment;

- can communicate directly with other agents typically using an Agent Communication Language;
- is driven by a set of functionalities it has to accomplish;
- possesses resources of its own;
- is capable of perceiving its environment;
- has only a partial representation of this environment in form of an instantiation of the domain ontology (knowledge);
- can offer services;
- can play several, different (and sometimes concurrent or mutually exclusive) roles.

Role - A portion of the social behaviour of an agent that is characterized by a goal (accomplishing some specific functionality) and/or it provides a service.

Message_RA - An interaction between an agent and a role. This can be a message as well as another kind of interaction (for instance GUI-based)

Message_RR - A message exchanged between two roles.

Plays - It specifies which agent plays a role

Stakeholder

Only one role is involved in this fragment, that is:

- System Analyst

Her/His responsibilities are described in the following subsection.

System Analyst

He is responsible for:

1. Tasks identification during the TSp. sub-phase.
2. Description of the control flow during the TSp. sub-phase.

Fragment workflow

Workflow description

The process that is to be performed in order to obtain the result is represented in the following as a SPEM 2.0 diagram.

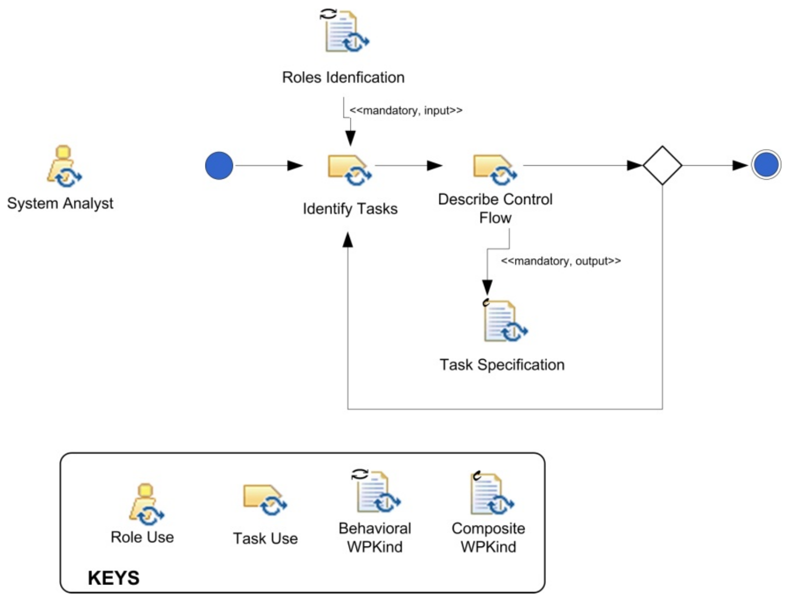

Figure 5. The flow of activity of this fragment

Activity description

The fragment encompasses the following *work breakdown elements*:

Name	Kind	Description	Roles involved
Identify Tasks	Task	It consists in the identification of the behavioural features each agent performs, and the (agent) roles involved in fulfilling the requirements that are under its responsibility. It consists in the identification of the activities that each agent performs playing one role.	System Analyst (perform)
Describe the control flow	Task	It consists in introducing the communication relationships among tasks of different agents and the control flow among tasks of the same agent	System Analyst (perform)

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 Tasks	Role	Role-Role (Message_RR)	Task	
	Agent	Role-Actor (Message_RA)	Activity	
	Actor	Role-Agent (Plays)	Plan	
Describe the control flow	Role	Role-Role (Message_RR)		Agent-Agent (Communicate)
	Agent	Role-Actor (Message_RA)		Task-Task (Task_Invocation)
	Actor	Role-Agent (Plays)		Agent-Task (aggregate)
				Task-Activity (Composite)
				Task-Plan (Composite)
				Agent-Plan (Aggregate)
				Activity-Activity (Activity_Invocation)
				Task-Activity (Task-Activity_Invocation)

WP Input/Output

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

Input	**Output**
Roles Identification document	Task Specification Document

Deliverable

Task Specification Document

This fragment delivers a Task Specification composite document. This is composed by:

- one or more Task Specification Diagram(s), each diagram is used to specify the plan of a different agent. Usually these diagrams are UML activity diagrams reporting two swim-lanes: the leftmost contains tasks belonging to external agents, the rightmost contains tasks belonging to the agent whose plan is defined in the diagram.
- a table describing each task in terms of list of activities, data and internal plan.

A Task Specification diagram represents the plan of the agent behaviour. It shows the relationships among the external stimuli received by the agent and its behaviour (expressed in terms of fired tasks). Relationships between activities represent communications between tasks of different agents (these communications cross the border separating the two swim-lanes) or invocation messages triggering tasks execution within the agent (this messages are all located within the rightmost swim-lane).

Task Specification Diagram: example of notation

Common UML Activity diagrams are used to specify agent's plan.

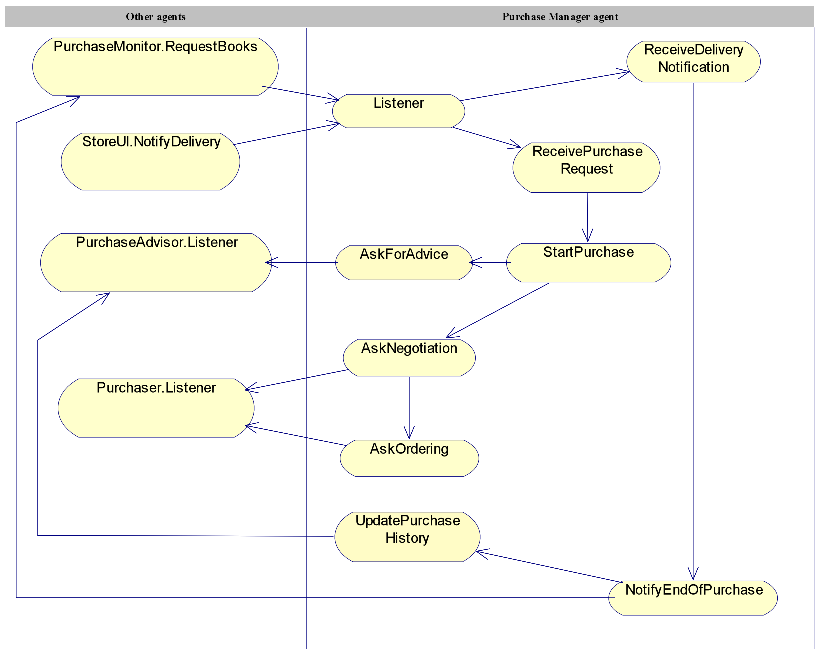

Figure 6. An example of Task Specification Diagram

The following table is used to detail each task:

Task: ***Task_name***

Description	A short description of task goal and structure
Activities	A list of the activities composing this task
Data	A list of the main data fields used by the task
Behavior	A structured description of the Task behavior (a kind of internal plan)

Deliverable relationships with the system metamodel

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

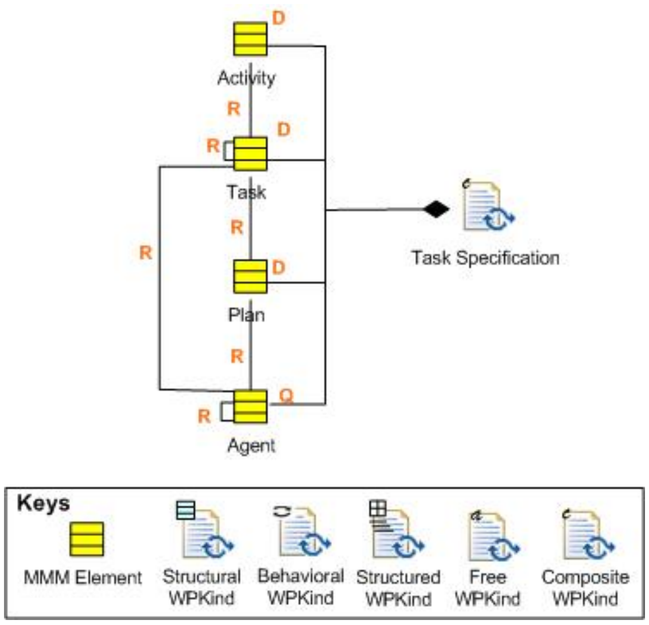

Figure 7. Structure of the fragment work-product in terms of MAS meta-model elements

Guidelines

Enactment Guidelines

In the PASSI methodology, tasks can be identified by looking at lifelines of roles in PASSI RID sequence diagrams.
This fragment is suitable for the construction of action plans for agents that support a state-based architecture instead of a goal-oriented one. Plans designed by using this fragment are static and therefore they are not suitable for agents exploiting reasoning capabilities for dynamic plan composition.

Reuse Guidelines

Composition

None

Dependency Relationship with other fragments

This fragment can be used after the definition of scenarios within a PASSI RID fragment.

Glossary

Message - an individual unit of communication between two or more agents that points out the standard FIPA message format. Usually a message is associated with a communicative act (or performative)

References

--