Communication Ontological Description

Composed Process Fragment

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

Last saved on: 23/11/10 15:41

Index

Fragment Description

Fragment Goal

Describing semantic agent communications in terms of exchanged knowledge (referred to an ontology), content language and interaction protocol.

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.

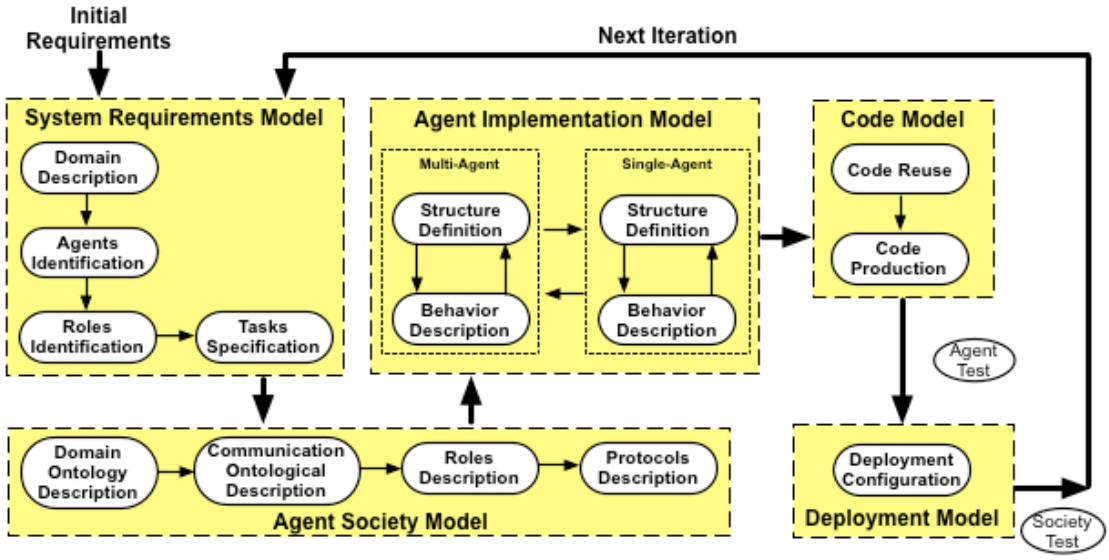


Figure 1. The PASSI design process

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

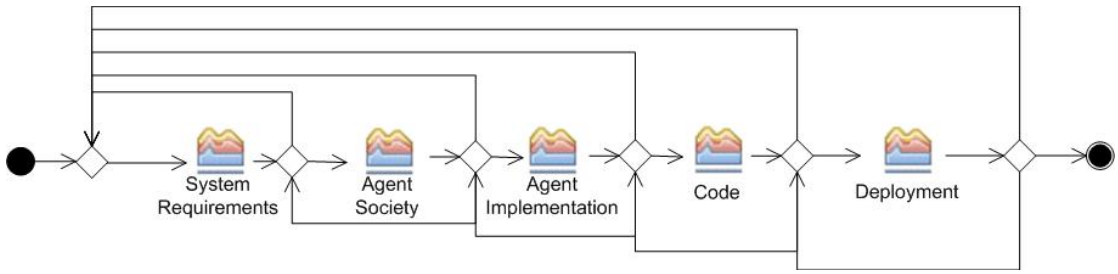


Figure 2. The PASSI process phases

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

Consider the PASSI process (see Figure 2) and the "Agent Society" phase with its outcome "Agent Society Model". Now, let us consider the "Communication Ontological 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.

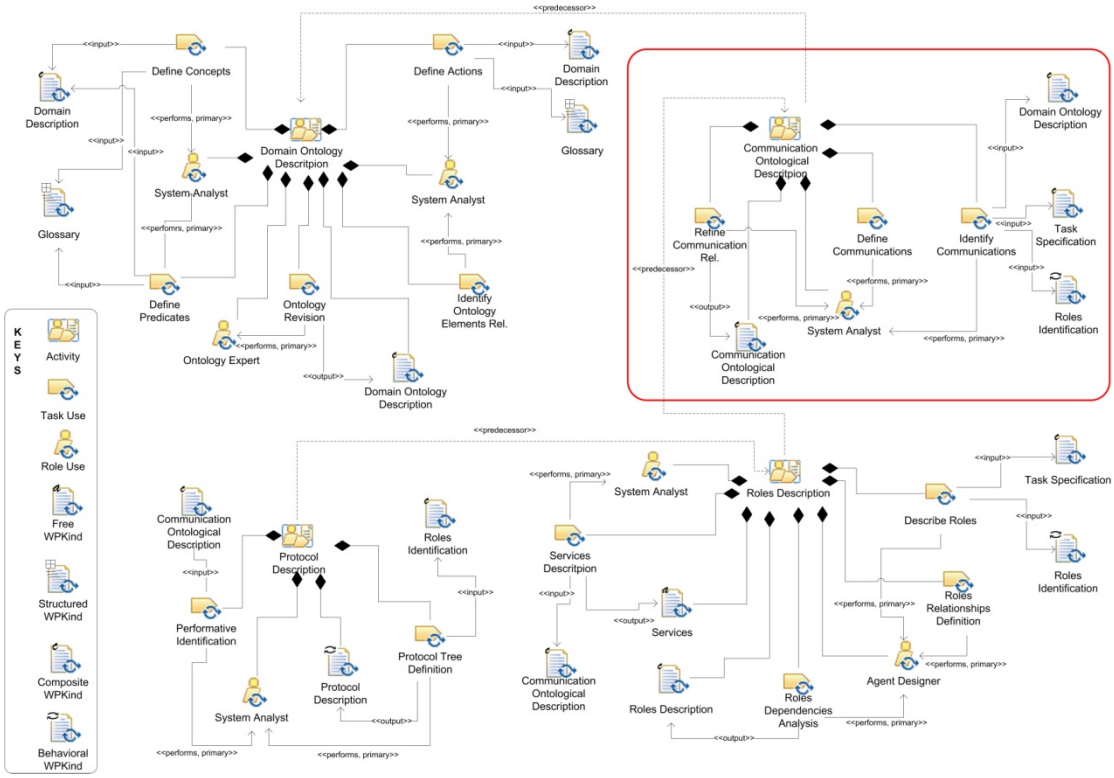


Figure 3. The communication ontological description fragment within the PASSI Agent Society model structural view

Fragment System metamodel

The portion of metamodel of this fragment is:

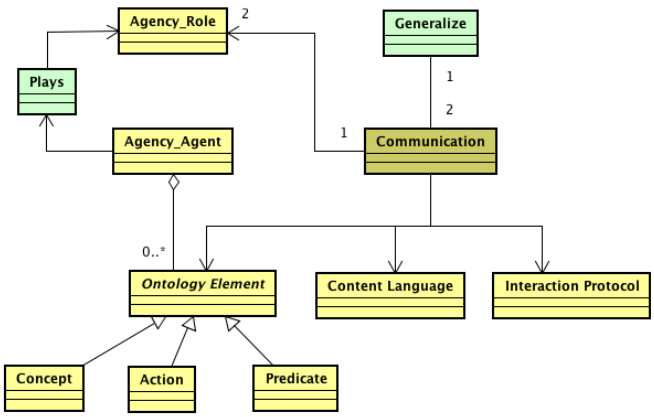


Figure 4. The fragment System metamodel

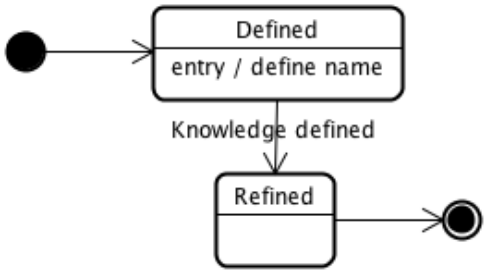
This fragment refers to the MAS meta-model adopted in PASSI and contributes to define and describe the elements reported in Figure 4.

Definition of System metamodel elements

This fragment underpins the following model elements:

Agency_Agent – an autonomous entity capable of pursuing an objective through its autonomous decisions, actions and social relationships. It is capable of performing actions in the environment it lives; it can communicate directly with other agents, typically using an Agent Communication Language; it possesses resources of its own; it is capable of perceiving its environment; it has a (partial) representation of this environment in form of an instantiation of the domain ontology (knowledge); it can offer services; it can play several, different (and sometimes concurrent or mutually exclusive) agency_roles.

Each agent may be refined by adding knowledge items necessary to store/manage communication contents. The Agency_agent statechart is:



Description of the Agency_Agent states:

Defined: An Agency_Agent is in this state once it is instantiated in the system model. The agent's unique name has to be defined.

Refined: An Agency_Agent moves in this state once its knowledge chunks are defined.

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

Content Language – A language with a precisely defined syntax semantics and pragmatics, which is the basis of communication between independently designed and developed agents. (from [1])

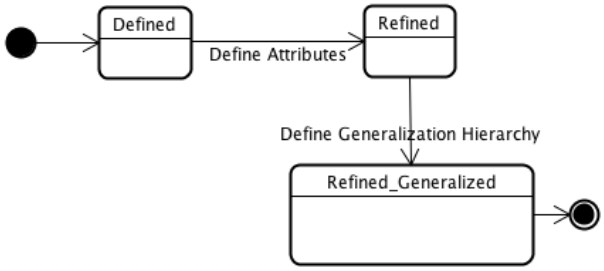
Ontology Element *(abstract class)* – An ontology is composed of concepts, actions and predicates. An Ontology element is an abstract class used as a placeholder for the ontology constituting elements (either concepts, predicates or actions).

(Interaction) Protocol – It is a pattern specifying the sequence of message types within a communication. Usually message types are identified by the performative (or speech act) associated to the message.
Concept - Description of a certain identifiable entity of the domain
Action – It expresses an activity, carried out by an agent.
Predicate – Description of a property of an entity of the domain

Definition of System metamodel relationships

This fragment underpins the following relationships among the model elements:

- **Communication** *(SMMR Association Class)* – An interaction among two agents, referring an Agent Interaction Protocol and a piece of the domain ontology (knowledge exchanged during the interaction). Usually it is composed of several messages, each one associated with one communicative act (or performative).
  The communication construct is an association class and its life-cycle within this fragment is depicted by the following statechart:



  Description of the Communication class states:
  Defined: the construct is defined once instantiated in the new system model
  Refined: the construct is refined once the values of its attributes (content language, ontology, interaction protocol) have been defined
  Refined_Generalized: the construct enters the state Refined_Generalized once a specific generalization hierarchy is defined for reusing common elements in the system communications.

- **Generalize** - See standard UML semantics
- **Plays** (*SMMR Association*) – It specifies that an agent may play one (or more) role(s).

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**	
SMME	**SMMR**	**SMME**	**SMMR**	**SMME**	**SMMR**	**SMME**	**SMMR**
Scenario	Message_RR	Agency_Agent*	Generalize		Communi-cation (definition of (Concept	Concept	

					or Predicate or Action), Content Language, Interaction Protocol)		
Role	AA -OntoRel	Agency_ Role*	Plays			Predicate	
Agent	CA-OntoRel		Communi-cation			Action	
Concept	CC-OntoRel					Content Language	
Predicate	CP-OntoRel					Interaction Protocol	
Action	PP-OntoRel						
Content Language	PA-OntoRel						
Interaction Protocol							

* Some obtained by 1:1 transformation from Problem Domain Agent and Role

Definition of input system metamodel elements and relationships

Role - A portion of the behaviour of an agent that is characterized by a goal (accomplishing some specific functionality) and/or provides a service.
Scenario - An instance of a use case describing a concrete set of actions. A scenario is composed of the following fields:

- Name: used to identify the scenario
- Participating actors: the list of participating actors (frequently actor instances are used)
- Flow of events: describing the flow of events step by step.

Ontology Element *(abstract class)* – An ontology is composed of concepts, actions and predicates. An Ontology element is an abstract class used as a placeholder for the ontology constituting elements (either concepts, predicates or actions).
Concept - Description of a certain identifiable entity of the domain
Action – It expresses an activity, carried out by an agent.
Predicate – Description of a property of an entity of the domain

Stakeholders

Roles involved in this fragment are:

- System Analyst.

Their responsibilities are described in the following subsections.

System Analyst

(S)He is responsible for:

1. Communications identification. It consists in introducing an association for each communication between two agents, looking at exchanged messages in the scenario.

2. Communications definition. The description of agents' communication in terms of ontology, content language and interaction protocol.
3. Communication relationships refinement. The identification of association classes in order to link each communication to the three fundamental element of communication itself (ontology, language and protocol).

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.

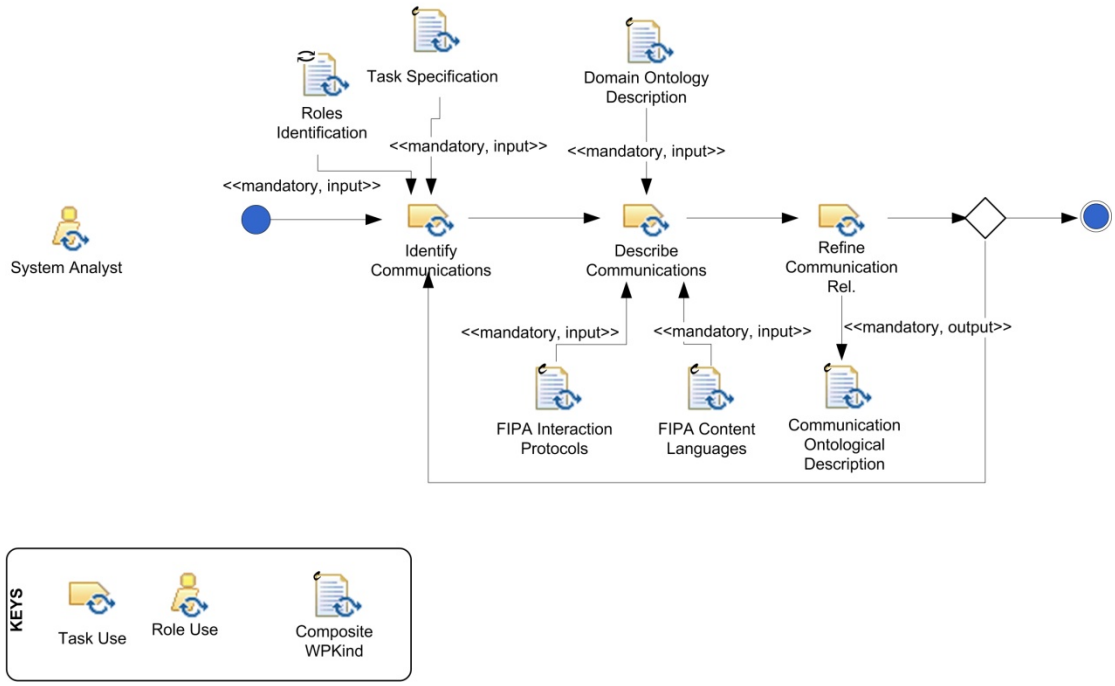


Figure 5. The flow of tasks of this fragment

Activity description

The fragment encompasses the following *work breakdown elements*:

Name	Kind	Description	Roles involved
Identify Communications	Task	It consists in defining communications among agents looking at exchanged messages in the scenario.	System Analyst (performs)

Describe Communications	Task	It consists in the description of agents' communications in terms of ontology, content language and interaction protocol. Agents' knowledge structures necessary to deal with communication contents have to be introduced in the agents.	System Analyst (performs)
Refine Communication Relationships	Task	The identification of general communication association classes in order to enhance reuse and improve the architecture. Use of generalize association among general and specialized communications.	System Analyst (performs)

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	**SMME**	**SMMR**	**SMME**	**SMMR**
Identify Communications	Scenario, Role, Agent, Content Language, Interaction Protocol	Message_RR	Agency_Agent* Agency_Role*	Plays Communication[defined
Describe Communications	Concept, Predicate, Action, Interaction Protocol, Content Language	AA-OntoRel, CA-OntoRel, CC-OntoRel, CP-OntoRel, PP-OntoRel, PA-OntoRel	Agency_Agent* [refined]	Communication [refined]
Refine Communication Relationships				Generalize, Communication [refined_generalized]

WP Input/Output

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

Input	Output
Agent Identification Document	Communication Ontological Description Document
Task Specification Document	
Domain Ontology Description Document	
FIPA Interaction Protocols	
FIPA Content Languages	

Deliverable

Communication Ontological Description (COD) Document

This fragment produces a composite document composed by a class diagram (whose classes represent agents and communications) and a text document describing the elements reported in the diagram.
The Communication Ontological Description (COD) diagram is a representation of the agents' (social) interactions; this is a structural diagram (for instance a class diagram) that shows all agents and all their interactions (lines connecting agents).
According to FIPA standards, communications consist of speech acts [1] and are grouped by FIPA in several interaction protocols [2] that define the sequence of expected messages. As a consequence, each communication is characterized by three attributes, which we group into an association class. The attributes are: ontology (a piece of the ontology defined in the PASSI DOD fragment), content language (see [3]), interaction protocol. This is the characterization of the communication itself (a communication with different ontology, content language or interaction protocol is certainly different from this one) and its name is used to uniquely refer this communication (which can have, obviously, several instances at runtime since it may be enacted more than once).

The following table describes the knowledge items assigned to agents (as attributes) in order to conveniently store/manage the received/outgoing communication content.

Agents' Knowledge

Agent	Knowledge piece	Data Type	Description
<agent name>	<name used to instantiate a piece of the ontology in the agent>	<a piece of the system ontology>	<text description>

The following table details the communication:

Communication details

From	To	Protocol	Content Language	Content (referred to ontology)	Description

<Initiator Agent>.<Initiator Role>	<Destination Agent>.<Destination Role>	<Interaction Protocol name>	<Content Language>	<Ontology element referred to in the communication content>	<description of the communication objective>

Communication Ontological Description Diagram notation

Each agent (fill colour: yellow) is described in terms of its knowledge (pieces of the ontology described in the Domain Ontology Description fragment). There is one relationship between two agents for each communication they are involved in. In each relationship the roles played by the agents during the communication are also reported as connection roles.
Each communication (fill colour: white) is represented by the relationship among the two agents and it is detailed in the relationship attribute class. The class is identified by an unique name and it is described by the following attributes: the ontology, the content language and the interaction protocol.
The ontology field refers to an element of the DOD (Domain Ontology Description); the language addresses for the adopted FIPA content language of the communication while the protocol points out the adopted FIPA Interaction Protocol.

Example

In Figure 6, the *PurchaseManager* agent starts a communication (see *QueryForAdvice* association class) with the *PurchaseAdvisor* agent. The communication contains the *Course* ontology, the *Query* protocol and the RDF language. This means that the *PurchaseManager* wants to perform a speech act based on the FIPA query protocol in order to ask the *PurchaseAdvisor* advice on how to purchase (supplier, number of stocks, number of items per each, purchase-money) provided the Course information.

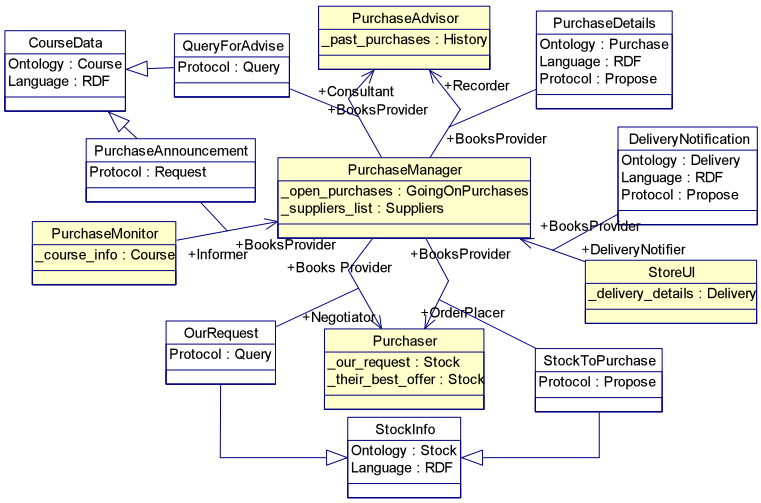
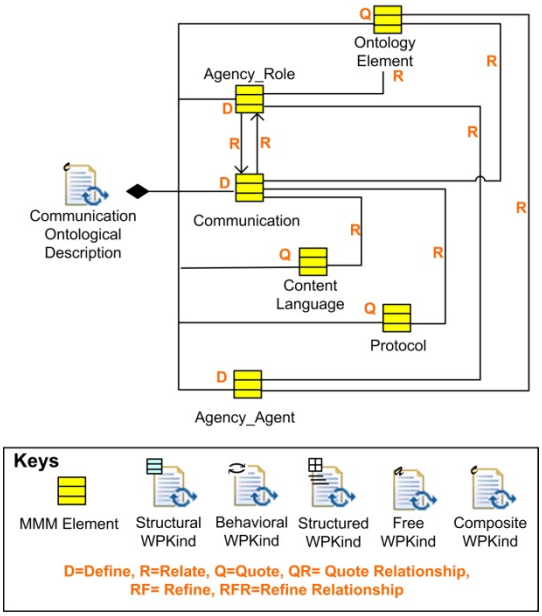


Figure 6. An example of Communication Ontological Description diagram

Deliverable relationships with the system metamodel



Guidelines

Enactment Guidelines

Agency_Roles are usually obtained by 1:1 transformation from Roles defined in the PASSI Problem Domain. This initially means Agency_Roles represent analysis roles, but new Agency_Roles may be defined if necessary/useful to improve design.
The same happens for Agency_agents.

Reuse Guidelines

Composition

--

Dependency Relationship with other fragments

This fragment usually is preceded by the PASSI Domain Ontology Description (composed) fragment.

Glossary

References

[1] J.R. Searle. Speech Acts. Cambridge Univ. Press. Cambrige. 1969.

[2] IEEE FIPA. Interaction Protocols Specifications. Available online at: http://www.fipa.org/repository/ips.php3

[3] IEEE FIPA. Content Languages Specifications. Available online at: http://www.fipa.org/repository/cls.php3