The SODA Methodology: Meta-model and Process Documentation

Ambra Molesini and Andrea Omicini

Abstract

The SODA methodology deals with MAS analysis and design, and focuses on critical issues such as agent coordination and MAS-environment interaction. After its first formulation, in order to further meet the needs of complex MAS engineering, SODA was extended to embody both the *layering principle* and the Agents & Artifacts (A&A) meta-model. As a result, both the SODA meta-model and the SODA process were re-defined, also to include two new phases—Requirement Analysis and Architectural Design. This chapter is then devoted to the documentation of the complete SODA process according to the FIPA standard.

1 Introduction

SODA (Societies in Open and Distributed Agent spaces) [13] is an agent-oriented methodology for the analysis and design of agent-based systems, which adopts the Agents & Artifacts (A&A) meta-model [10], and introduces a *layering principle* as an effective tool for scaling with system complexity, applied throughout the analysis and design process [2, 3, 9]. Since its first version [9], SODA is not concerned with *intra-agent* issues: designing a multi-agent system (MAS) with SODA amounts at defining agents in terms of their required observable behaviour as well as their role in the MAS. Then, whichever methodology one may choose to define the structure

A. Molesini (✉)
DISI, Alma Mater Studiorum – Università di Bologna, Viale Risorgimento 2, 40136 Bologna, Italy
e-mail: ambra.molesini@unibo.it

A. Omicini
DISI, Alma Mater Studiorum – Università di Bologna, Via Sacchi 3, 47521 Cesena, Italy
e-mail: andrea.omicini@unibo.it

M. Cossentino et al. (eds.), *Handbook on Agent-Oriented Design Processes*,
DOI 10.1007/978-3-642-39975-6_13,

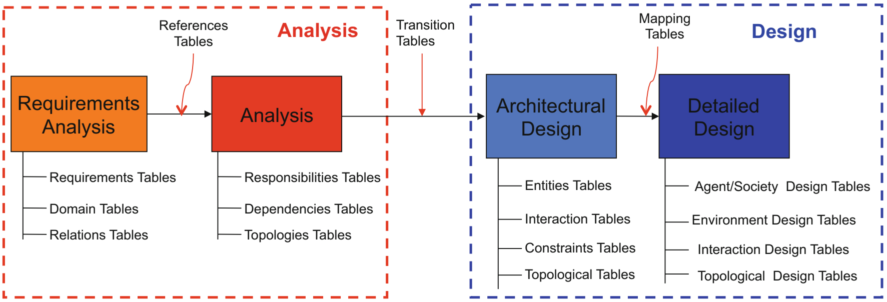


Fig. 1 An overview of the SODA process

and inner functionality of individual agents, it could be used in conjunction with SODA. Instead, SODA focus on *inter-agent* issues, like the engineering of agent societies and MAS environment.

When designing a new system in SODA, three things are to be understood: which activities have to be performed, which functions are available and required, and how activities and functions relate to each other. Accordingly, SODA abstractions are logically divided into three categories: (1) the abstractions for modelling/designing the system's active part (task, role, agent, etc.); (2) those for the reactive part (function, resource, artifact, etc.); and (3) those for interaction and organisational rules (relation, dependency, interaction, rule, etc.).

The SODA *process* is organised in two phases, each structured in two sub-phases: the *Analysis phase*, including the Requirements Analysis and the Analysis steps, and the *Design phase*, including the Architectural Design and the Detailed Design steps. Each sub-phase models the system through a subset of the SODA abstractions: in particular, each subset always includes at least one abstraction for each of the above categories—that is, at least one abstraction for the system's active part, one for the reactive part, one for interaction and organisational rules.

Figure 1 overviews the methodology by describing each step in terms of a set of relational tables. In the remainder of this chapter, the SODA process is described first as a whole process then through its four steps, following the FIPA standard [1].

Useful references about the SODA methodology and process are the following:

- A. Omicini. *SODA: Societies and Infrastructures in the Analysis and Design of Agent-based Systems* [9].
- A. Molesini, A. Omicini, A. Ricci, E. Denti. *Zooming Multi-Agent Systems* [3].
- A. Molesini, A. Omicini, E. Denti, A. Ricci. *SODA: A Roadmap to Artefacts* [2].
- A. Molesini, E. Denti, A. Omicini. *Agent-based Conference Management: A Case Study in SODA* [6].
- A. Molesini, E. Nardini, E. Denti, A. Omicini. *Advancing Object-Oriented Standards Toward Agent-Oriented Methodologies: SPEM 2.0 on SODA* [4].
- A. Molesini, E. Nardini, E. Denti, A. Omicini. *Situated Process Engineering for Integrating Processes from Methodologies to Infrastructures* [5].

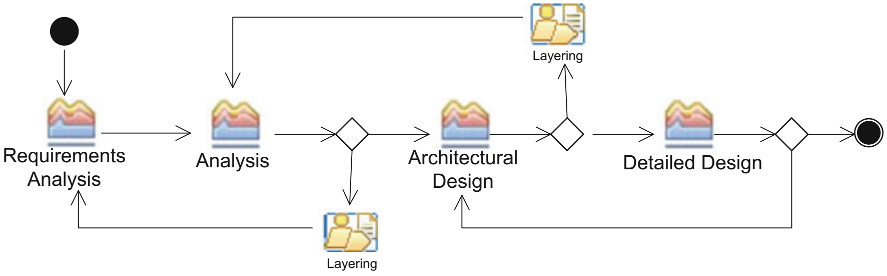


Fig. 2 Phases of the SODA process

- A. Molesini, A. Omicini. *Documenting SODA: An Evaluation of the Process Documentation Template* [7].

1.1 Process Life Cycle

SODA includes two phases, each structured in two sub-phases: the *Analysis* phase, which includes the *Requirements Analysis* and the *Analysis* steps, and the *Design* phase, which includes the *Architectural Design* and the *Detailed Design* steps. SODA phases and steps are arranged according to an iterative process model (see Fig. 2):

Requirements Analysis covers all the phases related to actor identification, requirements elicitation and analysis, and analysis of the existing environment.
Analysis investigates all the aspects related to the problem domain trying to understand the tasks satisfying the requirements, their connected functions, the environment topology and all the dependencies among these entities.
Architectural Design defines a set of admissible architectures for the final system.
Detailed Design determines the best system architecture and designs the environment and the system interactions.

Each step in SODA produces several sets of relational tables, each describing a specific MAS Meta-model Element (MMMElement) and its relationships with other MMMElements. The details of each step will be discussed in the following section.

1.2 Meta-model

The meta-model adopted by SODA is represented in Fig. 3, where SODA abstract entities are depicted along with their mutual relations, and distributed according to the four SODA steps: Requirements Analysis, Analysis, Architectural Design and Detailed Design.

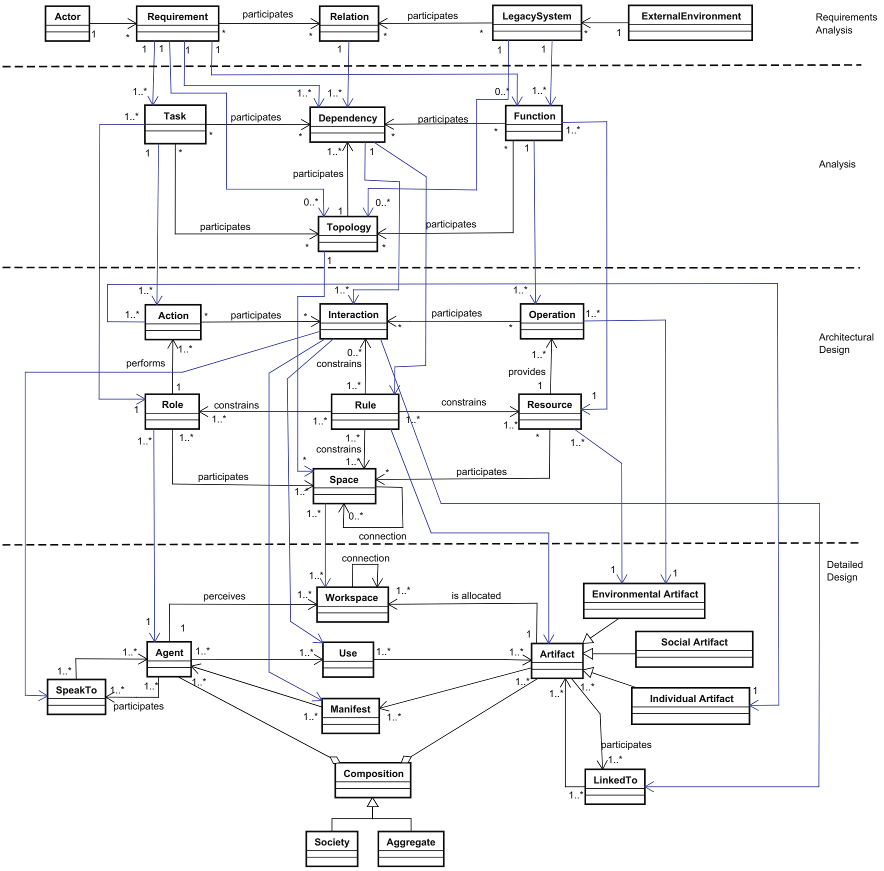


Fig. 3 The SODA meta-model

1.2.1 Requirements Analysis

Several abstract entities are introduced for requirement modelling: in particular, *Requirement* and *Actor* are used for modelling the customers' requirements and the requirement sources, respectively, while the notion of *External Environment* is used as a container of the *Legacy Systems* that represent the legacy resources of the environment. The relationships between requirements and legacy systems are then modelled in terms of *Relation* entities.

1.2.2 Analysis

The Analysis step expresses the abstract requirement representation in terms of more concrete entities such as *Tasks* and *Functions*. Tasks are activities requiring one or more competences, while functions are reactive activities aimed at supporting tasks. The relations highlighted in the previous step are now the starting point for the definition of *Dependencies*—interactions, constraints, etc.—among the abstract entities. The structure of the environment is also modelled in terms of *Topologies*, that is, topological constraints over the environment. Topologies are often derived from functions, but can also constrain/affect task achievement.

1.2.3 Architectural Design

The main goal of this stage is to assign responsibilities to achieve tasks to *Roles*, and responsibilities to provide functions to *Resources*. To this end, roles should be able to perform *Actions*, and resources should be able to execute *Operations* providing one or more functions. The dependencies identified in the previous phase become here *Interactions* and *Rules*. *Interactions* represent the acts of the interaction among roles, among resources and between roles and resources; rules, instead, enable and bound the entities' behaviour. Finally, the topology constraints lead to the definition of *Spaces*, that is, conceptual places structuring the environment.

1.2.4 Detailed Design

The active and passive parts are expressed in the Detailed Design in terms of individual entities (*Agents* and *Artifacts*) as well as of composite entities, such *Societies* and *Aggregates*. Agents are intended here as autonomous entities able to play several roles, whereas the resources identified in the previous step are now mapped onto suitable artifacts.

Artifacts have "types" according to the following taxonomy [11]:

- An *individual artifact* handles the interaction of a single agent within a MAS and essentially works as a mediator between the agent and the MAS itself. Since they can be used to shape admissible interactions of individual agents in MAS, individual artifacts play an essential role in engineering both organisational and security concerns in MAS.
- An *environmental artifact* brings an external resource within a MAS by mediating agent actions and perceptions over resources. As such, environmental artifacts play an essential role in enabling, disciplining and governing the interaction between agents and MAS environment.
- A *social artifact* rules social interactions within a MAS by mediating interactions between individual, environmental and possibly other social artifacts. Social artifacts in SODA play the role of the coordination artifacts that embody the rules around which societies of agents can be built.

Interactions between agents and artifacts in SODA take the form of *Use* (agent to artifact), *Manifest* (artifact to agent), *SpeakTo* (agent to agent) and *LinkedTo* (artifact to artifact).

In SODA, a group of individual entities can be abstracted away as a single composite entity. In particular, a group of interacting agents and artifacts can be seen as a SODA *Society* when its overall behaviour is essentially an autonomous, proactive one; it can be seen as a SODA *Aggregate*, instead, when its overall behaviour is essentially a functional, reactive one. Finally, SODA *Workspaces* take the form of an open set of artifacts and agents: artifacts can be dynamically added to or removed from workspaces, and agents can dynamically enter (join) or exit workspaces (Table 1).

Table 1 The SODA entities definitions

Concepts	Definition	Step
Actor	*System's stakeholder*	Requirements Analysis
Requirement	*Service that the stakeholder requires from a system and the constraints under which it operates and is developed*	Requirements Analysis
Legacy System	*Legacy resources*	Requirements Analysis
External Environment	*Legacy environment in which the new system will execute*	Requirements Analysis
Relation	*A tie among the entities of the Requirements Analysis*	Requirements Analysis
Task	*An activity aimed at the satisfaction of a specific requirement*	Analysis
Function	*Function or service aimed at supporting task accomplishment*	Analysis
Topology	*Topological constraints over the environment. Often derived from legacy systems and requirements, however also functions and tasks could induct some topological constraints*	Analysis
Dependency	*Any kind of dependency relationships among abstract entities, as a conceptual premise to any sort of interaction*	Analysis
Role	*An entity responsible to accomplish some tasks*	Architectural Design
Action	*An activity that changes the environment in order to meet roles design objectives*	Architectural Design
Resource	*Entity that provides functions*	Architectural Design
Operation	*A resource access point in order to achieve a function*	Architectural Design
Space	*Conceptual places structuring the environment*	Architectural Design
Rule	*Any prescription over roles, resources, interactions, and spaces*	Architectural Design
Interaction	*Any interaction among roles and resources*	Architectural Design
Agent	*Pro-active components of the systems, encapsulating the autonomous execution of some kind of activities inside an environment*	Detailed Design
Artifact	*Passive components of the systems such as resources and media that are intentionally constructed, shared, manipulated and used by agents to support their activities, either cooperatively or competitively*	Detailed Design
Individual Artifact	*Mediator between an individual agent and the MAS*	Detailed Design

(continued)

Table 1 (continued)

Concepts	Definition	Step
Social Artifact	*Mediator of social interactions within a MAS*	Detailed Design
Environmental Artifact	*Mediator of the interaction between MAS and the external environment*	Detailed Design
Composition	*A collection of agents and artifacts working together as an ensemble*	Detailed Design
Society	*A composition whose overall behaviour is essentially an autonomous, proactive one*	Detailed Design
Aggregate	*A composition whose overall behaviour is essentially a functional, reactive one*	Detailed Design
Workspace	*Conceptual containers of agents and artifacts, providing a notion of locality for MAS*	Detailed Design
Use	*The act of interaction between agent and artifact: agent uses artifact*	Detailed Design
Manifest	*The act of interaction between artifact and agent: artifact manifests itself to agent*	Detailed Design
SpeakTo	*The act of interaction among agents: agent speaks to another agent*	Detailed Design
LinkedTo	*The act of interaction among artifact: artifact is linked to another artifact*	Detailed Design

1.3 Guidelines and Techniques

SODA exploits a technique called *Layering* that can be applied to the overall process before the Detailed Design step. In SODA, during the Analysis phase and the Architectural Design step, the system is described in principle by all the layers defined, and could then be modelled by a number of different—although related—*design views*. This of course does not hold for the Detailed Design step since the developer should be provided with a single system representation among all the potentially admissible ones based on the Architectural Design layers.

Accordingly, the next section presents the SODA Layering technique.

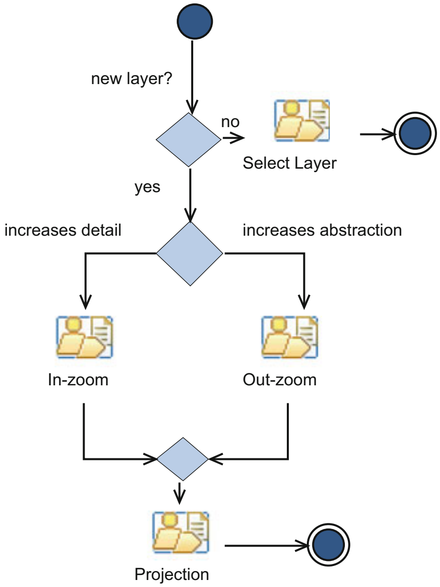
1.3.1 Layering

Complexity is inherent in real-life systems. While *modelling* complex systems and understanding their behaviour and dynamics is the most relevant concern in many areas, such as economics, biology or social sciences, also the complexity of *construction* becomes an interesting challenge in artificial systems like software ones. An integral part of a system development methodology must therefore be devoted to controlling and managing complexity.

To this end, SODA introduces *Layering*, a conceptual tool to deal with the complexity of system representation. Using Layering, a system in SODA can be represented as composed by different *layers of abstraction*, with a *layering operation* to conceptually move between them.

Layering can be represented as a *capability pattern* [8]—that is, a reusable portion of the process, as shown in Fig. 4, where the layering process is detailed. In particular, the layering process has two activities: (1) the selection of a specific layer

Fig. 4 The layering process



for refining/completing the abstractions models in the methodology process (Select Layer activity), and (2) the creation of a new layer in the system by *in-zooming*—that is, increasing the system detail—or *out-zooming*—that is, increasing the system abstraction—activities. In the last case, the layering process ends with the projection activity where the abstractions are projected "as they are" from one layer to another so as to maintain the consistency in each layer.

In general, when working with SODA, the reference layer, called *core layer*, is labelled with C, and is by definition *complete*—that is, it contains all the entities required to fully describe a given abstract layer. Any other layer—labelled with either $C + i$, for more detailed layers, where i is the number of in-zoom steps from the C layer, or $C - i$, for more abstract layers, where i is the number of out-zoom steps from the C layer—contains just the entities (in/out-) zoomed from another layer, along with the entities projected "as they are" from other layers. So, in general, the other (non-core) layers are not required to be complete—though of course they might be so, as in the case of layer $C + 1$ in Fig. 5. The projected entities are identified by means the prefix "+" if they are projected from a more abstract layer to a more detailed layer (see entity E2 in Fig. 5), with "–" otherwise—see entity E1 in Fig. 5.

Figure 6 depicts a more detailed view of the Layering capability pattern showing the flow of activities, the process roles involved and the input and work products of each activity.

1.3.2 Process Roles

One role is involved in the Layering pattern: the Layering Expert. Layering Expert is responsible for

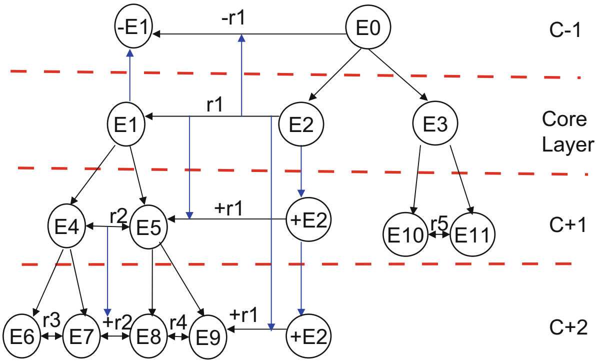


Fig. 5 An example of layering

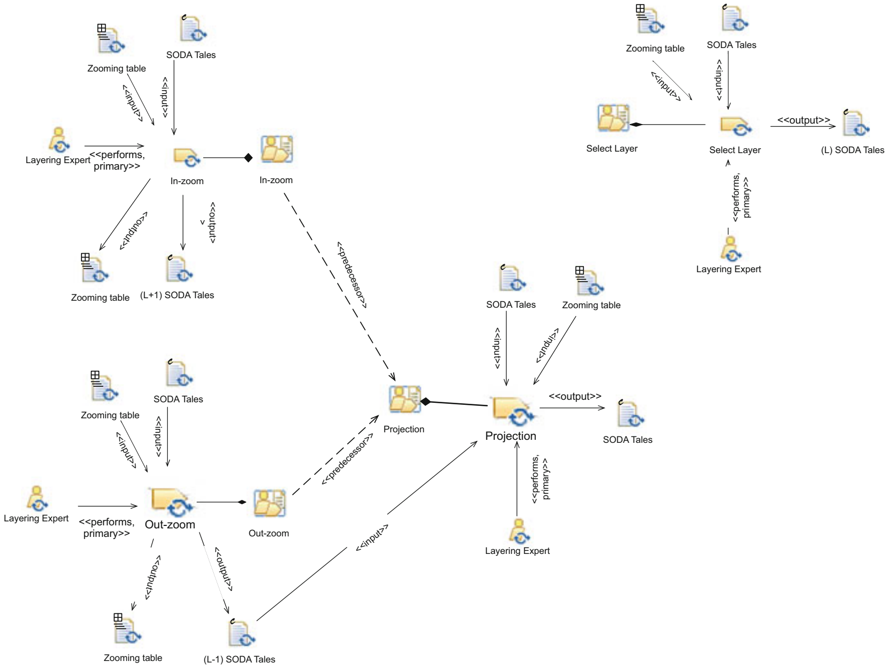


Fig. 6 The layering flow of activities, roles and work products

- Selecting the specific abstraction layer
- Either in-zooming or out-zooming the system by creating the specific Zooming table or modifying an existing Zooming table
- Projecting the necessary entities in the new created layer
- Partially filling all the newly created SODA tables

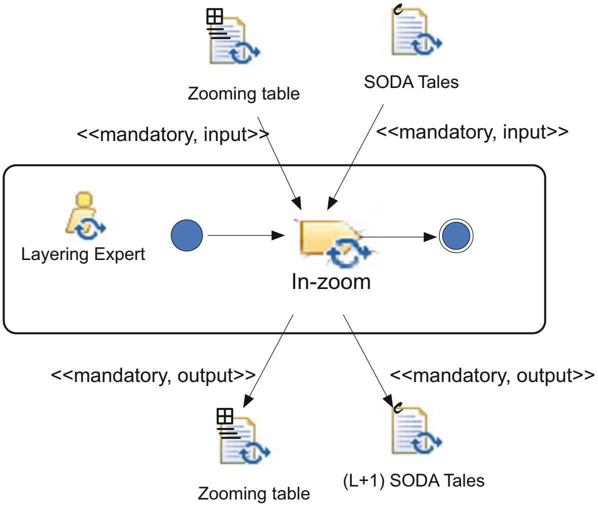


Fig. 7 The task in the in-zoom activity

1.3.3 Activity Details

In-Zoom Activity

The flow of tasks inside *in-zoom* activity is reported in Fig. 7; the tasks are detailed in the following table.

Activity	Task	Task description	Role involved
In-zoom	In-zoom	*Allowing the creation of a new, more detailed layer modifying the zooming table, and introducing the work products for the new layer*	Layering Expert *(perform)*

Out-Zoom Activity

The flow of tasks inside *out-zoom* activity is reported in Fig. 8; the tasks are detailed in the following table.

Activity	Task	Task description	Role involved
Out-zoom	Out-zoom	*Allowing the creation of a new, more abstract layer modifying the Zooming table, and introducing the work products for the new layer*	Layering Expert *(perform)*

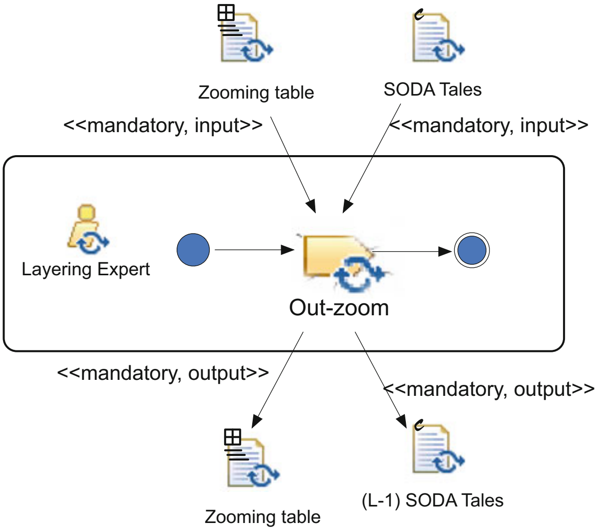


Fig. 8 The task in the out-zoom activity

Select Layer Activity

The flow of tasks inside *Select Layer* activity is reported in Fig. 9; the tasks are detailed in the following table.

Activity	Task	Task description	Role involved
Select Layer	Select Layer	*Allowing the selection of a specific layer in order to either redefine or complete it*	Layering Expert *(perform)*

Projection Activity

The flow of tasks *Projection activity* this activity is reported in Fig. 10; the tasks are detailed in the following table.

Activity	Task	Task description	Role involved
Projection	Projection	*Allowing the projection of a non-zoomed entity from a layer to another in order to preserve the layer consistency*	Layering Expert *(perform)*

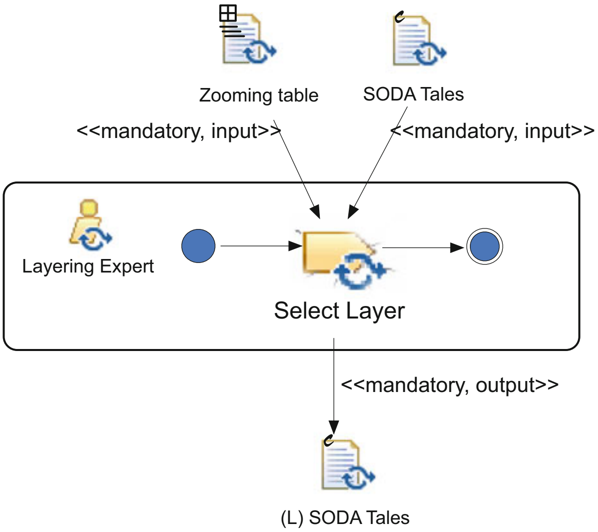


Fig. 9 The task in the select layer activity

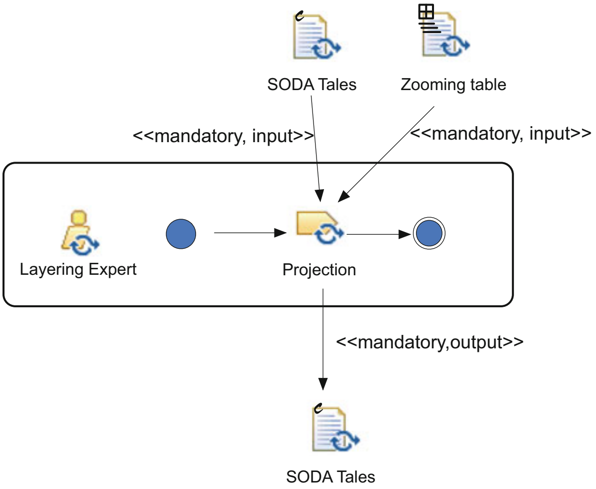


Fig. 10 The task in the projection activity

1.3.4 Work Products

Layering generates one work product: the Zooming table. Its relationships with the MMMElements are described in Fig. 11.

This diagram represents the Layering in terms of the Work Product and its relationships with the SODA meta-model (Sect. 1.2) elements. Each MMMElement is represented using an UML class icon (yellow) and, in the documents, such

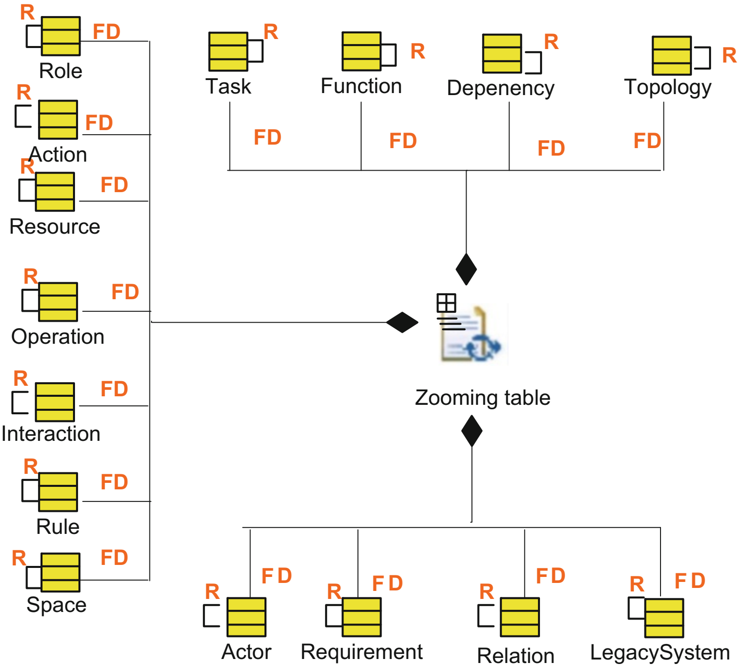


Fig. 11 The layering work products

elements can be Defined, reFined, Quoted, Related or Relationship Quoted as defined in [12] and briefly reported in the following:

- *Defined* (D label)—this means that the element is introduced for the first time in the design in this artifact (the MMMElement is instantiated in this artifact)
- *reFined* (F label)—this means that the MMMElement is refined in the work product (for instance by means of attribute definition)
- *Related* (R label)—this means that an already defined element is related to another, or, from a different point of view, that one of the MAS meta-model relationships is instantiated in the document
- *Quoted* (Q label)—this means that the element was already defined, and it is reported in this artifact only to complete its structure, but no work has to be done on it
- *Relationship Quoted* (RQ label)—this means that the relationship is reported in the work product, but it was defined in another part of the process

Kinds of Work Products

Layering is represented by means of a Zooming table ($(C)Z_t$)—see Fig. 12. The Zooming table formalises the in-zoom of a layer into the more detailed layer; of course, the same table can be used to represent the dual out-zoom process. One column of the table contains the name of the abstraction at layer C, while the

Fig. 12 $(L)Z_t$

Layer L	Layer $L+1$
out-zoomed entity	*in-zoomed entities*

Fig. 13 $(C)Z_t$

Layer C	Layer $C+1$
E1	E4, E5, r2, +E2, +r1
E3	E10, E11, r5

Fig. 14 $(C-1)Z_t$

Layer $C-1$	Layer C
-E1, -r1, E0	E2, E3

Fig. 15 $(C+1)Z_t$

Layer $C+1$	Layer $C+2$
E4	E6, E7, r3, +r2
E5	E8, E9, +E2, r4, +r1

other column reports the name of the corresponding zoomed abstractions at the subsequent layer $C+1$ (in-zooming) or $C-1$ (out-zooming).

In general when in-zooming an entity from layer C to layer $C+1$, we obtain a new group of entities, but also a set of relationships among these new entities that allow the entities' coordination as shown in Fig. 5.

Examples of Work Products

Figures 13, 14, and 15 report the Zooming tables modelling the example proposed in Fig. 5. In particular, Fig. 13 shows the relationships between layer C—the core layer—and layer $C+1$, where entity E1 is in-zoomed into E4 and E5, and E3 is in-zoomed into E10 and E11. The E2 entity and r1 relationship are projected from C to $C+1$: this is reported in the in-zoom table of E1, since the relation between E1 and E2 in layer C has to be maintained also in layer $C+1$ in order to maintain consistency. In addition, two new relationships are necessary after the in-zoom operation: r2 comes from the in-zooming of E1 in order to coordinate the E4 and E5; in a similar way r5 comes from the in-zooming of E3. Figure 14 reports the relation between layer $C-1$ and layer C. Here there is an out-zoom operation, E2 and E3 are collapsed in E0, while E1 and r1 are projected for consistency reason. Finally, Fig. 15 depicts the relation between layer $C+1$ and layer $C+2$ where E4 is in-zoomed in E6, E7 and r3; E5 is in-zoomed in E8, E9 and r4; and E2, r1 and r2 are projected.

2 Phases of the SODA Process

2.1 The Requirements Analysis

The goals of Requirements Analysis are (1) characterising both the customers' requirements and the legacy systems with which the system should interact, as well as (2) highlighting the relationships among requirements and legacy systems. Requirements can be categorised in [14]:

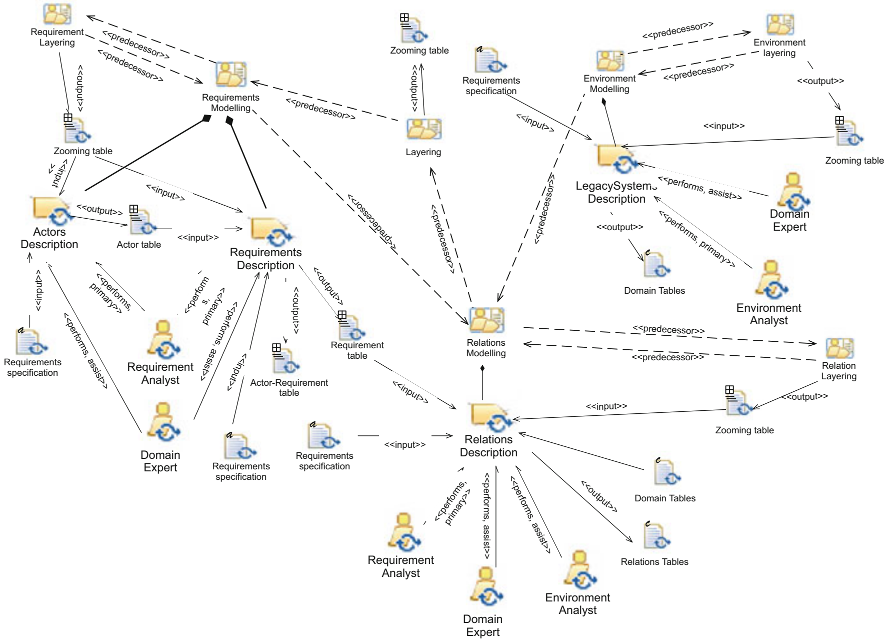


Fig. 16 The requirements analysis flow of activities, roles and work products

- *Functional Requirements*—statements about which functionalities the system should provide, how the system should react to particular inputs and how the system should behave in particular situations.
- *Non-Functional Requirements*—constraints on the services and functions offered by the system such as timing constraints, constraints on the development process, standards, security, privacy, etc. Non-functional requirements could be more critical than functional requirements. If these are not met, the system is useless.
- *Domain Requirement*—requirements that come from the application domain of the system, and that reflect features of that domain. Domain requirements could be new functional requirements, constraints on existing requirements or define specific computations. If domain requirements are not satisfied, the system may be unworkable.

In this step, we take into account several abstract entities to model the system's requirements: actors, requirements, external environment, legacy systems and relations. The Requirements Analysis involves three different process roles, and eight work products, as described in Fig. 16. Figure 17 presents the Requirements Analysis process composed by three main activities—Requirements Modelling, Environment Modelling, and Relations Modelling—and several different layering activities—see Sect. 1.3.1.

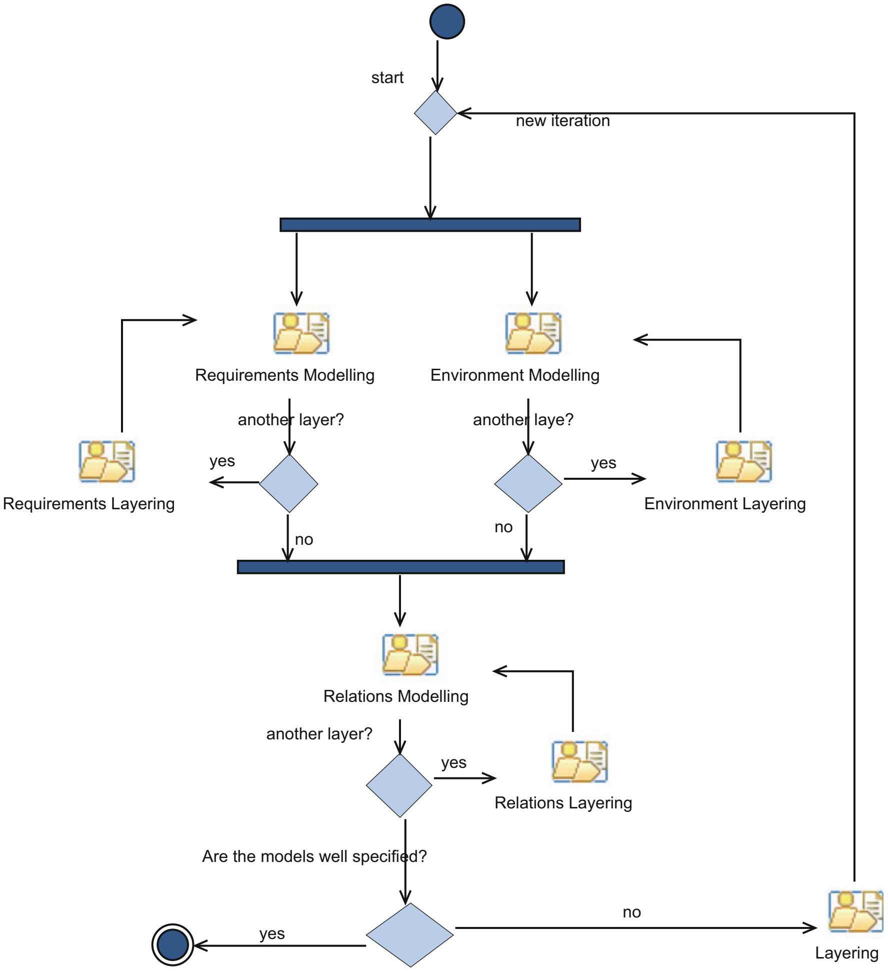


Fig. 17 The requirements analysis process

2.1.1 Process Roles

Three roles are involved in the Requirements Analysis: the Requirement Analyst, the Environment Analyst and the Domain Analyst.

Requirement Analyst

The Requirement Analyst is responsible for

- The identification of the main actors and system stakeholders
- The identification of the system functional and non-functional requirements
- The analysis of the system's requirements

- The identification of the any kinds of relationship among requirements, and between requirements and legacy systems

Environment Analyst

The Environment Analyst is responsible for

- The identification of the legacy systems already present in the environment
- The analysis of the legacy systems

In addition, the Environment Analyst should help the Requirement Analyst in identification of the any kinds of relationship between requirements and legacy systems.

Domain Expert

The Domain Expert supports the Requirement Analyst and the Environment Analyst during the description of the application domain.

2.1.2 Activity Details

For the details about the different Layering activities please refer to Sect. 1.3.1.

Requirements Modelling Activity

The Requirements Modelling activity is composed of the following tasks:

Activity	Task	Task description	Role involved
Requirements Modelling	Actors Description	*Identification of the actors and their description*	Requirement Analyst *(perform)* Domain Expert *(assist)*
Requirements Modelling	Requirements Description	*Identification of the requirements and their description*	Requirement Analyst *(perform)* Domain Expert *(assist)*

The flow of tasks inside the Requirements Modelling activity is reported in Fig. 18.

Environment Modelling Activity

The Environment Modelling activity is composed of the following tasks:

Activity	Task	Task description	Role involved
Environment Modelling	Legacy Systems Description	*Identification of the legacy systems and their description*	Environment Analyst *(perform)* Domain Expert *(assist)*

The flow of tasks inside the Environment Modelling activity is reported in Fig. 19.

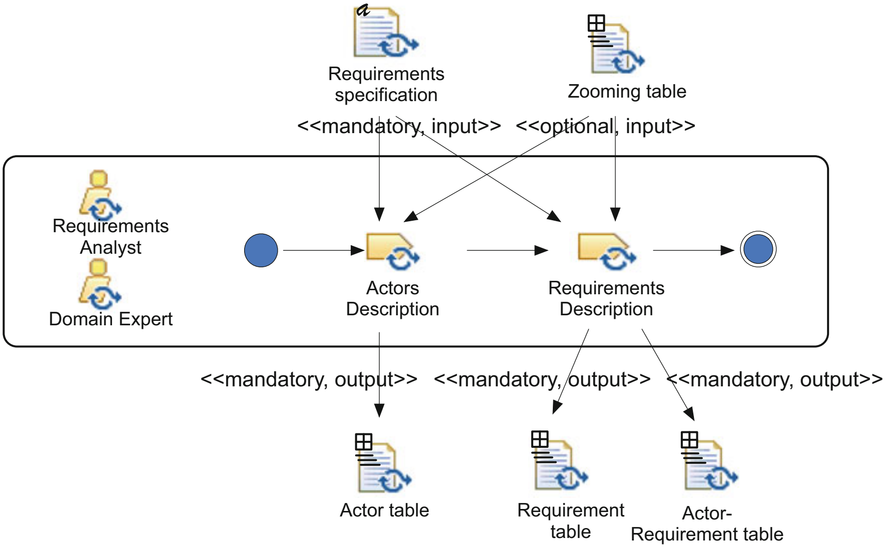


Fig. 18 The flow of tasks in the requirements modelling activity

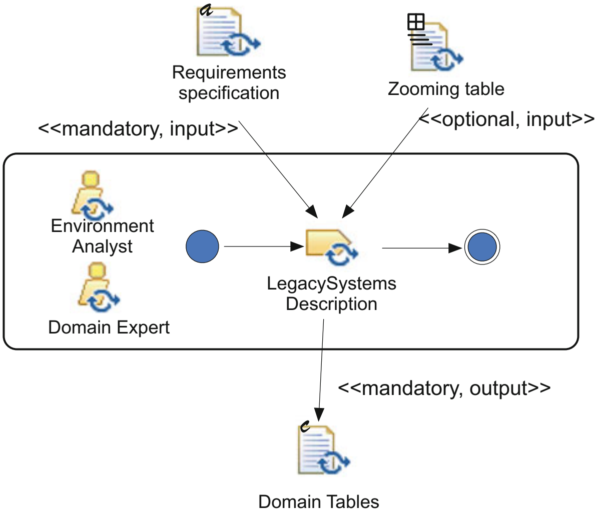


Fig. 19 The flow of tasks in the environment modelling activity

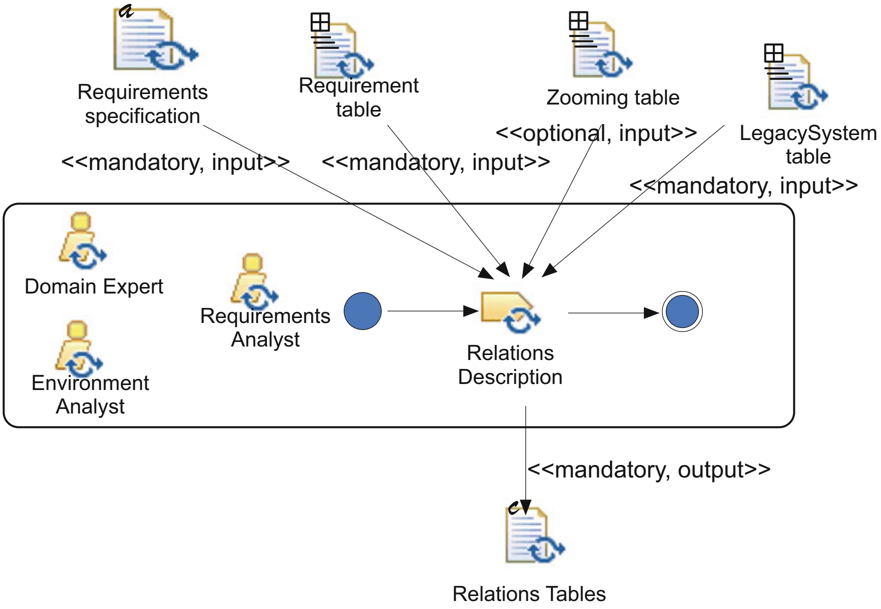


Fig. 20 The flow of tasks in the relations modelling activity

Relations Modelling Activity

The Relations Modelling activity is composed by the following tasks:

Activity	Task	Task Description	Role Involved
Relations Modelling	Relations Description	*Identification of the relations and their description*	Environment Analyst *(perform)* Domain Expert *(assist)* Environment Analyst *(assist)*

The flow of tasks inside the Relations Modelling activity is reported in Fig. 20.

2.1.3 Work Products

The Requirements Analysis step consists of three sets of tables: Requirements Tables, Domain Tables and Relations Tables. Figure 21 reports the relationships among the work products of this step and the MMMElements of the Requirements Analysis. In Fig. 21, the relationships among the Zooming table and the MMMElements of the Requirements Analysis are also reported—see Sect. 1.3.1 for details.

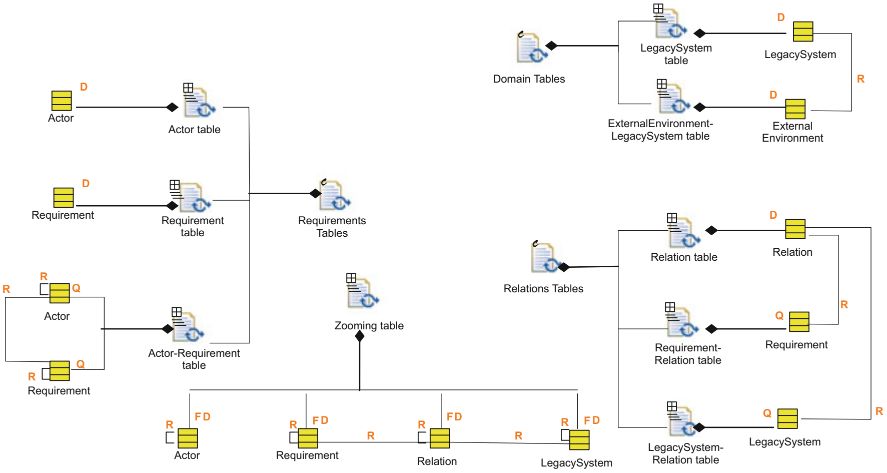


Fig. 21 The requirement analysis work products

Kinds of Work Products

Table 2 describes all the work products of the Requirements Analysis. In particular, the first entry (Requirements Specification) represents the input of the all SODA process, the second set is the outcome of the Requirement Modelling activity, the third set is the outcome of the Environment Modelling activity and the last set is the outcome of the Relation Modelling activity.

Requirements Tables

Figure 22 provides an example of the Requirements Tables for the conference management case study [6].

Domain Tables

In the conference management system case study, there is only one Legacy System, called “WebServer”, which represents the container for the web application of the conference: the reason to include it in the description is that the conference management system will obviously interact with it, and such an interaction should be captured and constrained. Figure 23 presents Legacy System table where the WebServer system is described.

In our system, there is just one relation, called “Web”, which involves all the abstract entities since all requirements need to access the web server to retrieve or store information.

Relations Tables

An example of the Relations Tables in illustrated in Fig. 24.

Table 2 Requirements Analysis work products kinds

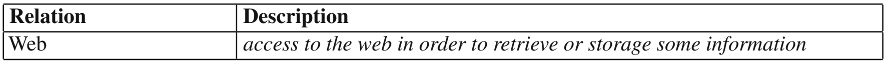
Name	Description	Work product kind
Requirements Specification	*A description of the problem to be solved*	Free Text
Requirements Tables	*A composition of others tables that defines the abstract entities tied to the concept of "requirement"*	Composite (structured)
Actor table ($(L)Ac_t$)	*Description of each single actor*	Structured
Requirement table ($(L)Re_t$)	*Description of each single requirement*	Structured
Actor-Requirement table ($(L)AR_t$)	*Specification of the collection of the requirements associated to each actor*	Structured
Domain Tables	*A composition of others tables that defines the abstract entities tied to the concept of "external environment"*	Composite (structured)
Legacy System table ($(L)LS_t$)	*Description of each single legacy system*	Structured
External Environment – Legacy System table ($(L)EELS_t$)	*Specification of the legacy systems associated to the external environment*	Structured
Relations Tables	*A composition of others tables that links the abstract entities with each other*	Composite (structured)
Relation table ($(L)Rel_t$)	*Description of all the relationships among abstract entities*	Structured
Requirement-Relation table ($(L)RR_t$)	*Specification of the relations where each requirement is involved*	Structured
Legacy System – Relation table ($(L)LSR_t$)	*Specification of the relations where each legacy system is involved*	Structured

Requirement	**Description**
ManageStartUp	*creating call for papers and defining the rules of the organisation*
ManageSubmission	*managing users registration, papers submission and keywords insertion*
ManagePartitioning	*partitioning papers basing on the conference structure*
ManageReviewers	*managing reviewers registrations and insertion of the keywords representing their expertise area*
ManageAssignment	*managing the assignment process according to the organisation rules*
ManageReview	*managing the review process and sending reviews to authors*

Fig. 22 Requirement table $(C)Re_t$

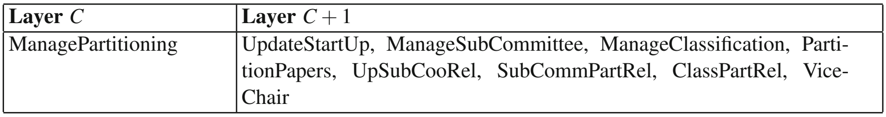
Legacy System	**Description**
WebServer	*the container for the web application of the conference*

Fig. 23 Legacy System table $(C)LS_t$



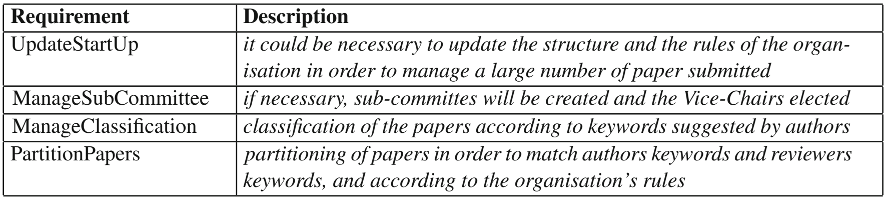
Relation	Description
Web	*access to the web in order to retrieve or storage some information*

Fig. 24 Relation table $(C)Rel_t$



Layer C	Layer $C+1$
ManagePartitioning	UpdateStartUp, ManageSubCommittee, ManageClassification, PartitionPapers, UpSubCooRel, SubCommPartRel, ClassPartRel, Vice-Chair

Fig. 25 Zooming table $((C)Z_t)$: paper partitioning in-zoom



Requirement	Description
UpdateStartUp	*it could be necessary to update the structure and the rules of the organisation in order to manage a large number of paper submitted*
ManageSubCommittee	*if necessary, sub-committes will be created and the Vice-Chairs elected*
ManageClassification	*classification of the papers according to keywords suggested by authors*
PartitionPapers	*partitioning of papers in order to match authors keywords and reviewers keywords, and according to the organisation's rules*

Fig. 26 Requirement table $(C + 1)Re_t$

Requirements Tables at Layer $C + 1$

In Figs. 25 and 26, we report some examples of the SODA tables modelling the conference management systems at layer $C + 1$.

2.2 The Analysis

In the Analysis step, SODA takes into account four abstract entities to analyse the system: tasks, functions, dependencies and topologies. Figure 27 presents the Analysis process, while Fig. 28 presents the flow of activities, the roles involved and the work products.

2.2.1 Process Roles

One role is involved in the Analysis step: the System Analyst.

System Analyst

The System Analyst is responsible for

- Mapping the MMMElements of the Requirements Analysis to the MMMElements of the Analysis
- Identifying new tasks coming from system analysis and description of the all tasks (new tasks and tasks coming from the mapping)
- Identifying new functions coming from system analysis and description of the all functions (new tasks and tasks coming from the mapping)

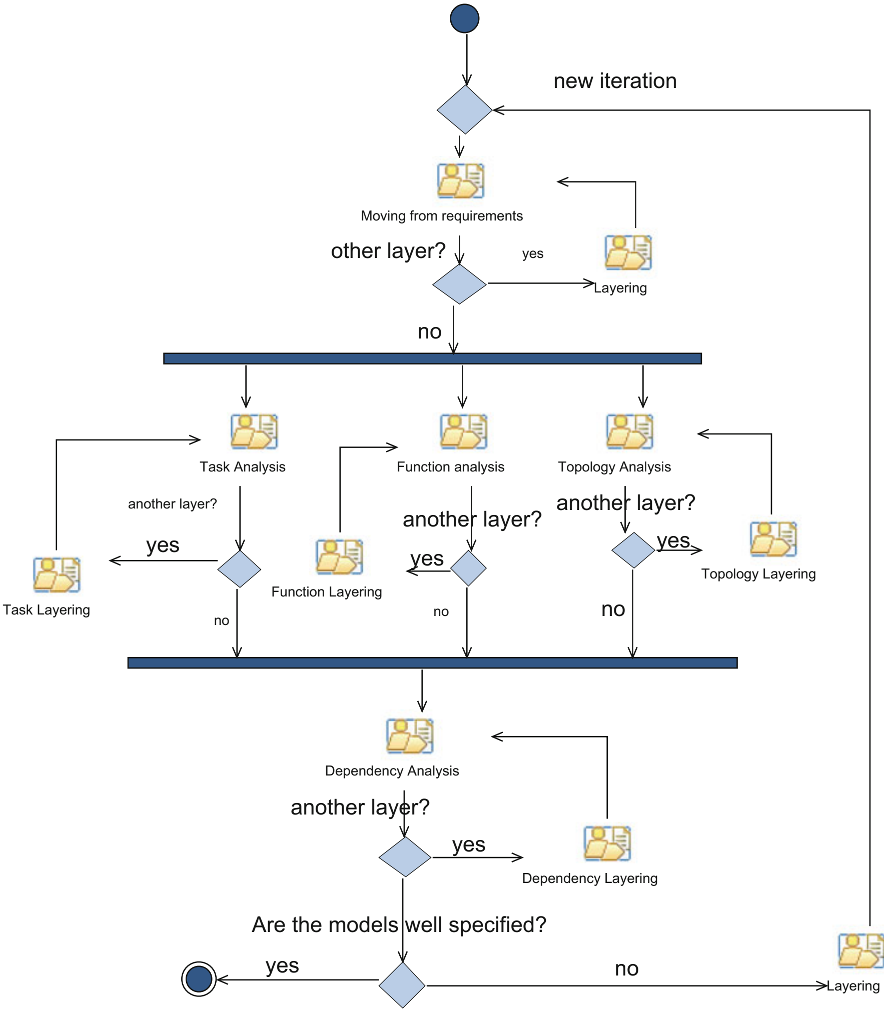


Fig. 27 The analysis process

- Identifying new dependencies coming from system analysis and description of the all dependencies (new dependencies and dependencies coming from the mapping)
- Identifying new topologies coming from system analysis and description of the all topologies (new topologies and topologies coming from the mapping)

2.2.2 Activity Details

For the details about the different Layering activities, please refer to Sect. 1.3.1.

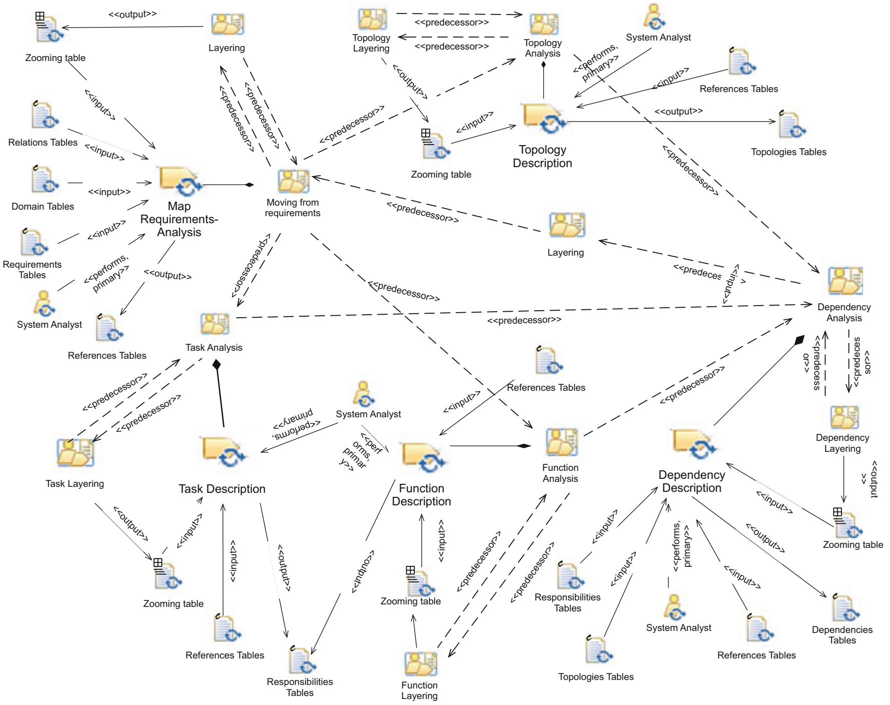


Fig. 28 The analysis flow of activities, roles and work products

Moving from Requirements Activity

The Moving from Requirements activity is composed of the following tasks:

Activity	Task	Task description	Role involved
Moving from Requirements	Map Requirements-Analysis	*Mapping of the MMMElements defined in Requirements Analysis to the Analysis MMMElements*	System Analyst *(perform)*

The flow of tasks inside the Moving from Requirements activity is reported in Fig. 29.

Task Analysis Activity

The Task Analysis activity is composed of the following tasks:

Activity	Task	Task description	Role involved
Task Analysis	Task Description	*Identification of the tasks and their description*	System Analyst *(perform)*

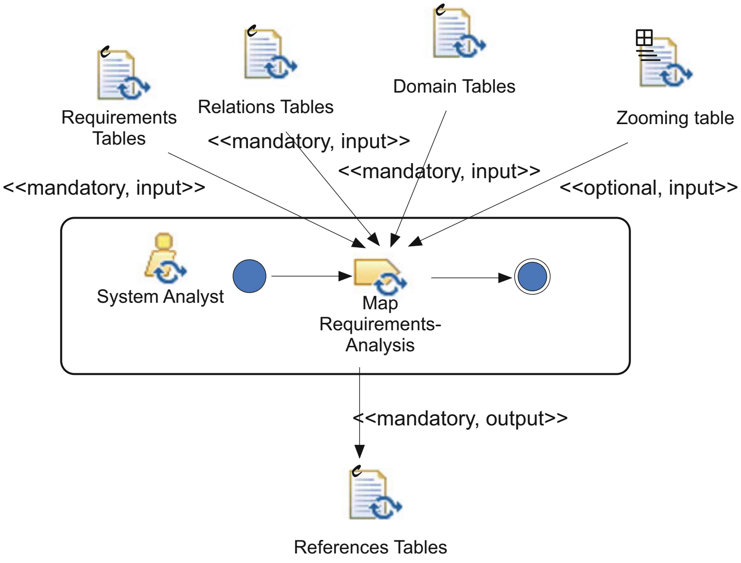


Fig. 29 The flow of tasks in the moving from requirements activity

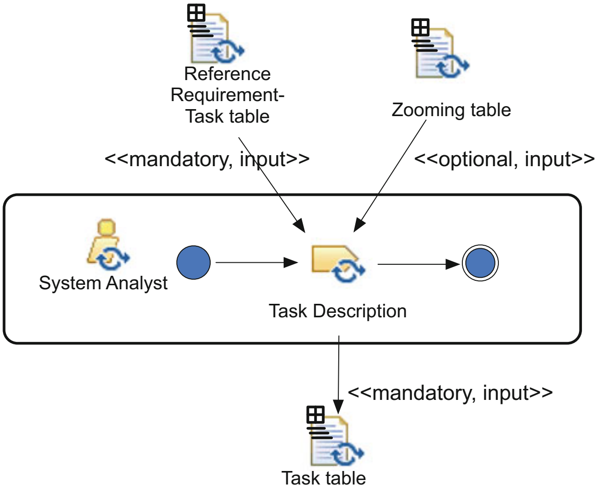


Fig. 30 The flow of tasks in the task analysis activity

The flow of tasks inside the Task Analysis activity is reported in Fig. 30.

Function Analysis Activity

The flow of tasks inside this activity is reported in Fig. 31; the tasks are detailed in the following table.

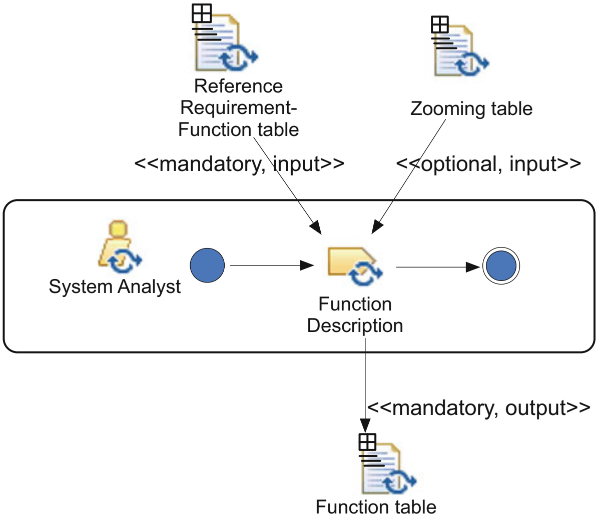


Fig. 31 The flow of tasks in the function analysis activity

Activity	Task	Task description	Role involved
Function Analysis	Function Description	*Identification of the functions and their description*	System Analyst *(perform)*

Dependency Analysis Activity

The flow of tasks inside this activity is reported in Fig. 32, and the tasks are detailed in the following table.

Activity	Task	Task description	Role involved
Dependency Analysis	Dependency Description	*Identification of the system dependencies and their description. Identification of relations with tasks, functions and topology*	System Analyst *(perform)*

Topology Analysis Activity

The flow of tasks inside this activity is reported in Fig. 33; the tasks are detailed in the following table.

Activity	Task	Task description	Role involved
Topology Analysis	Topology Description	*Identification of the topological constraints and their description. Identification of relations with tasks and functions*	System Analyst *(perform)*

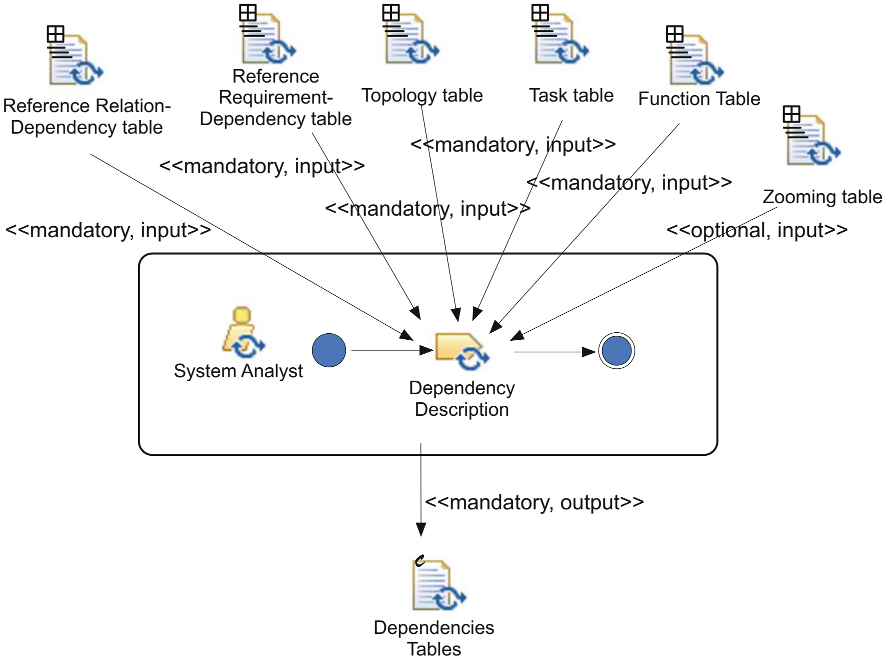


Fig. 32 The flow of tasks in the dependency analysis activity

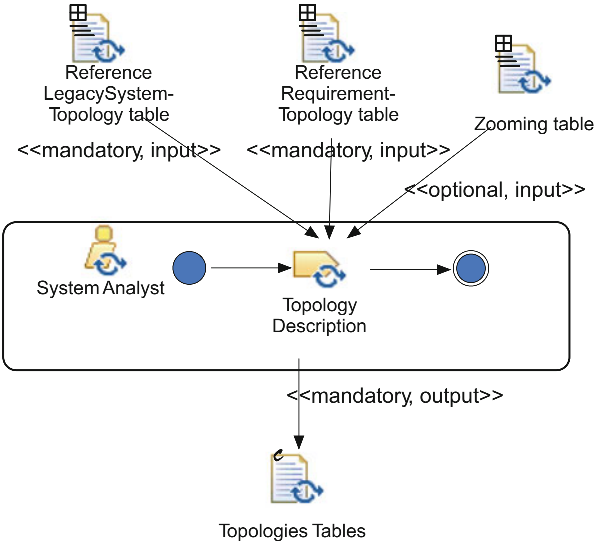


Fig. 33 The flow of tasks in the topology analysis activity

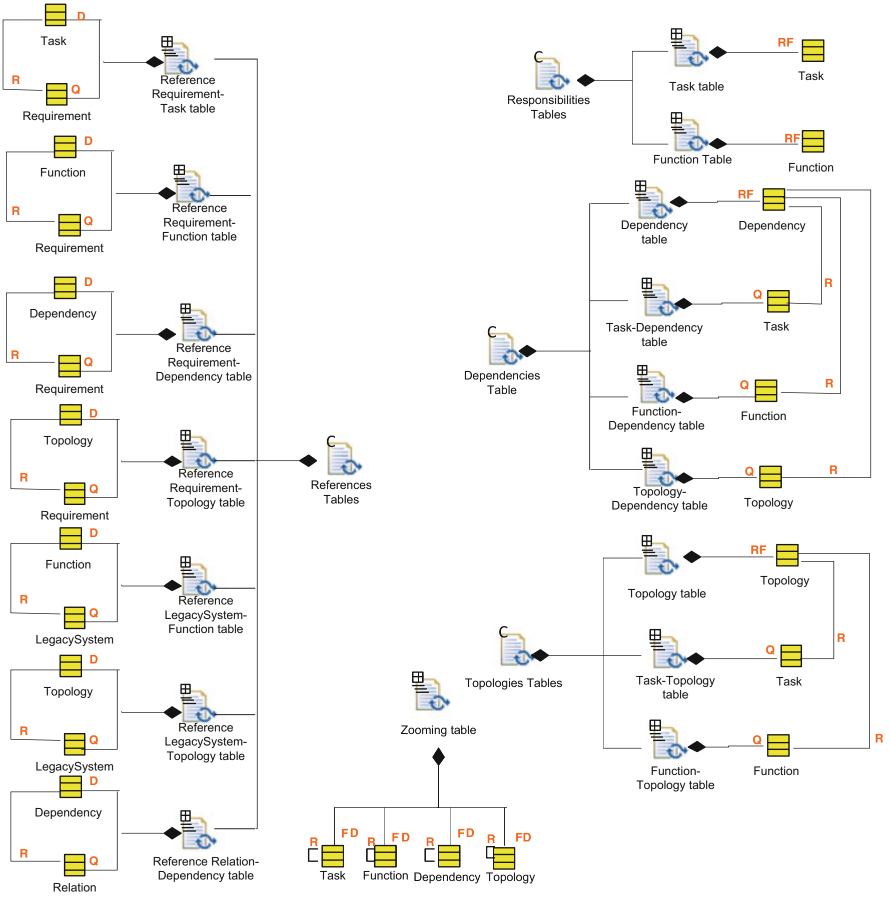


Fig. 34 The analysis work products

2.2.3 Work Products

The Analysis step exploits four sets of tables: Reference Tables, Responsibilities Tables, Dependencies Tables and Topologies Tables. Figure 34 reports the relationships among the work products of this step and the MMMElements of the Analysis. In Fig. 34 are also reported the relationships among the Zooming table and the MMMElements of the Analysis—see Sect. 1.3.1 for details.

Kinds of Work Products

Table 3 describes all the work products of the Analysis. In particular, the first set of work products is the outcome of the Moving from Requirements activity, the second set is the outcome of the Task Analysis and Function Analysis activities, the third is

Table 3 Requirements Analysis work products kinds

Name	Description	Work product kind
References Tables	*A composition of others tables that allow to move from Requirements Analysis to Analysis*	Composite (structured)
Reference Requirement-Task table ($(L)RRT_t$)	*Specification of the mapping between each requirement and the generated tasks*	Structured
Reference Requirement-Function table ($(L)RRF_t$)	*Specification of the mapping between each requirement and the generated functions*	Structured
Reference Requirement-Topology table ($(L)RRTo_t$)	*Specification of the mapping between each requirement and the generated topologies*	Structured
Reference Requirement-Dependency table ($(L)RReqD_t$)	*Specification of the mapping between each requirement and the generated dependencies*	Structured
Reference Legacy System-Function table ($(L)RLSF_t$)	*Specification of the mapping between each Legacy System and the corresponding functions*	Structured
Reference Legacy System-Topology table ($(L)RLST_t$)	*Specification of the mapping between Legacy Systems and topologies*	Structured
Reference Relation-Dependency table ($(L)RRelD_t$)	*Specification of the mapping between relations and dependencies*	Structured
Responsibilities Tables	*A composition of others tables that defines the abstract entities tied to the concept of "responsibilities centre"*	Composite (structured)
Task table ($(L)T_t$)	*Description of the all tasks*	Structured
Function table ($(L)F_t$)	*Description of the all functions*	Structured
Topologies Tables	*A composition of others tables that express the topological constraints over the environment*	Composite (structured)
Topology table ($(L)Top_t$)	*Description of the topological constraints*	Structured
Task-Topology table ($(L)TTop_t$)	*Specification of the list of the topological constraints where each task is involved*	Structured
Function-Topology table ($(L)FTop_t$)	*Specification of the list of the topological constraints where each function is involved*	Structured
Dependencies Tables	*A composition of others tables that relates functions and tasks with each other*	Composite (structured)
Dependency table ($(L)D_t$)	*Description of the all dependencies among abstract entities*	Structured
Task-Dependency table ($(L)TD_t$)	*Specification of the set of dependencies where each task is involved*	Structured
Function-Dependency table ($(L)FD_t$)	*Specification of the list of dependencies where each function is involved*	Structured
Topology-Dependency table ($(L)TopD_t$)	*Specification of the list of dependencies where each topology is involved*	Structured

Requirement	**Task**
ManageStartUp	*start up*
ManageSubmission	*paper submission* *user registration*
ManageReviewers	*reviewer registration*
ManagePartitioning	*paper partitioning*
ManageAssignment	*assignment papers*
ManageReview	*review process*

Fig. 35 Reference Requirement-Task table $(C)RRT_t$

Task	**Description**
start up	*insertion of the setup information*
submission	*the paper has to be submitted and the keywords have to be indicated*
user registration	*user inserts his data*
reviewer registration	*reviewer inserts his data and the keywords representing his expertise areas*
paper partitioning	*partitioning of the set of papers according to the conference rules*
assignment papers	*assignment papers to reviewers*
review process	*creation and submission of the reviews*

Fig. 36 Task table $(C)T_t$

Topology	**Description**
place	*it is the locus where functions are allocated*

Fig. 37 Topology table $(C)Top_t$

the outcome of the Topology Analysis activity and the last set is the outcome of the Dependency Analysis activity.

References Tables

Figure 35 represents an example of the References Tables for the conference management case study.

Responsibilities Tables

Figure 36 represents an example of the Responsibilities Tables for the conference management case study. Figure 37 represents an example of the Topologies Tables for the conference management case study.

Dependencies Tables

Figure 38 represents an example of the Dependencies Tables for the conference management case study.

Responsibilities Tables at Layer $C + 1$

Figures 39 and 40 report some examples of the SODA tables modelling the conference management systems at layer $C + 1$.

Dependency	**Description**
RegSubDep	*paper submission to be done after author registration*
RegAssDep	*the paper assignment has to be done after reviewers registration*
PartAssDep	*the paper assignment has to be done after the conclusion of the paper partitioning process*
AssRevDep	*the paper revision has to be started only after the conclusion of the paper assignment process*
WebAccessDep	*access to website for retrieving or storing information*
StartUpInfDep	*access of all the information bout start up process*
UserInfDep	*access to all the users' information*
ReviewerInfDep	*access to all the reviewers' information*
PaperInfDep	*access to all the paper information*
PartInfDep	*access to all the information about partitioning process*
SubInfDep	*access to all the information about submission process*
AssInfDep	*access to all the information about assignment process; a reviewer cannot be the author of the papers assigned to him*
ReviewInfDep	*access to all the information about review process*

Fig. 38 Dependency table $(C)D_t$

Layer C	**Layer** $C+1$
paper partitioning	modifying startup, create sub-committees, Vice-Chair elections, paper classification, partition papers, NewOrganisationDep, ElectionDep

Fig. 39 Zooming table $(C)Z_t$

Task	**Description**
modifying startup	*update the structure and the rules of the organisation*
create sub-committees	*creation of sub-committees*
Vice-Chair elections	*for each sub-committee elect the Vice-Chair*
papers classification	*classification of papers according to the keywords*
partition papers	*partitioning papers according to their classification*

Fig. 40 Task table $(C + 1)T_t$

2.3 The Architectural Design

In this step, we take into account several abstract entities in order to design the system's general architecture: role, resource, action, operation, interaction, environment and place. Figure 41 presents the Architectural Design process, while Fig. 42 presents the flow of activities, the involved roles and the work products.

2.3.1 Process Roles

One role is involved in the Architectural Design: the Architectural Designer.

Architectural Designer

The Architectural Designer is responsible for

- Mapping the MMMElements of the Analysis to the MMMElements of the Architectural Design

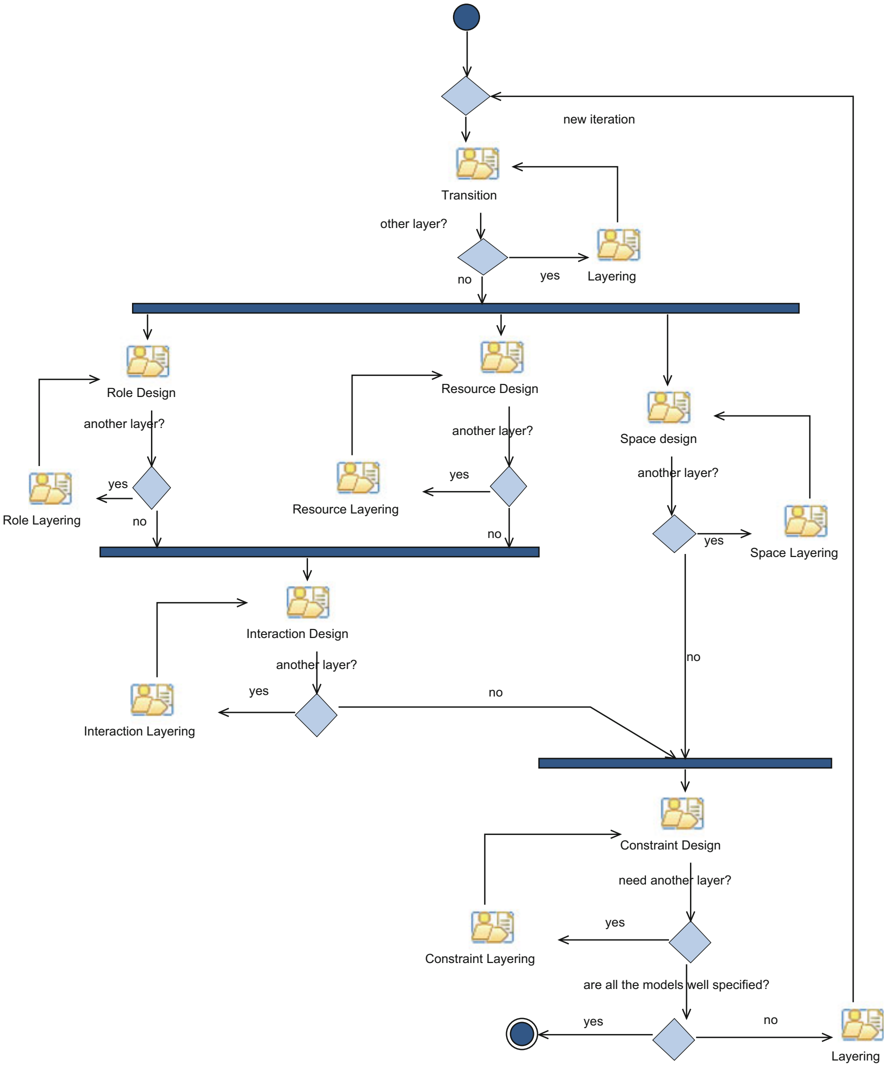


Fig. 41 The architectural design process

- Assigning tasks to roles
- Assigning functions to resources
- Identifying new actions coming from system design and describing all the actions (new actions and actions coming from the mapping)
- Identifying operations coming from system design and describing all the operations (new operations and operations coming from the mapping)
- Identifying new interactions coming from system design and describing all the interactions (new interactions and interactions coming from the mapping)

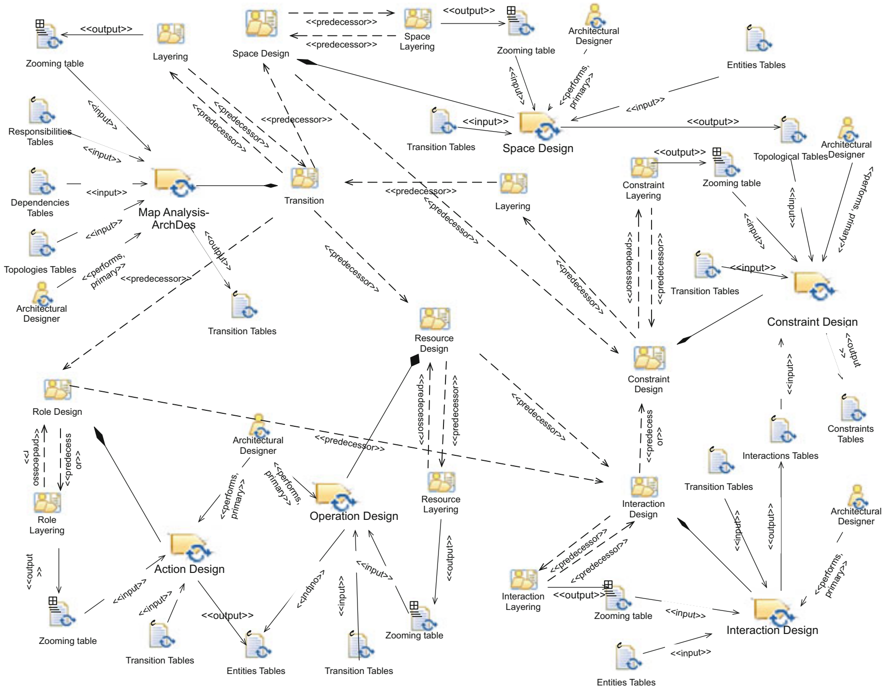


Fig. 42 The architectural design flow of activities, roles, and work products

- Identifying new rules coming from system design and describing all the rules (new rules and rules coming from the mapping)
- Identifying new spaces coming from system design and describing all the spaces (new spaces and spaces coming from the mapping)

2.3.2 Activity Details

For the details about the different Layering activities, please refer to Sect. 1.3.1.

Transition Activity

The Transition activity is composed of the following tasks:

Activity	Task	Task description	Role involved
Transition	Map Analysis-ArchDes	*Mapping of the MMMElements defined in Analysis to the Architectural Design MMMElements so as to generate the initial version of the Architectural Design models*	Architectural Designer *(perform)*

The flow of tasks inside the Transition activity is reported in Fig. 43.

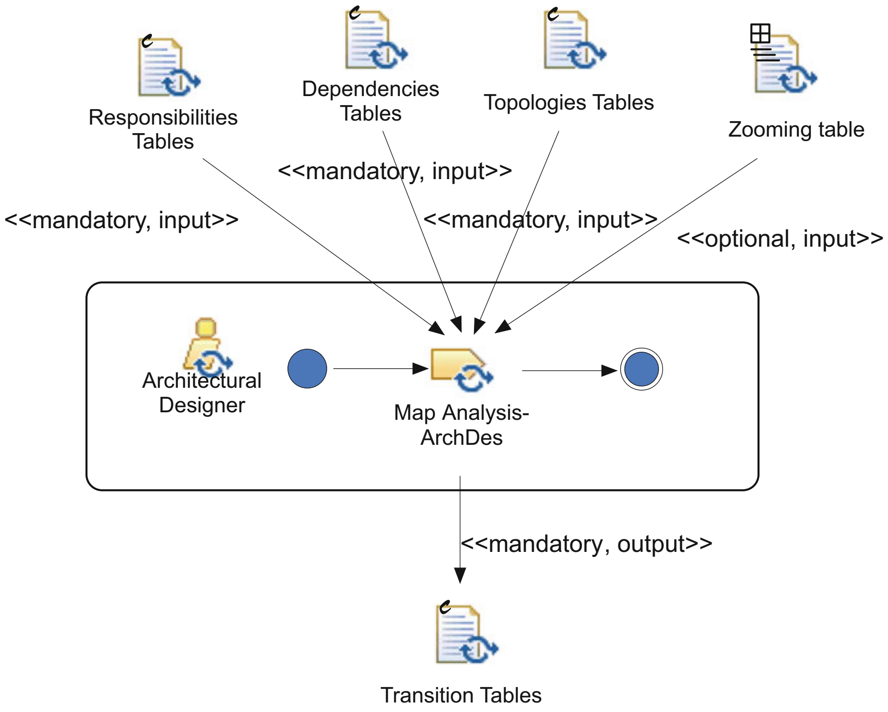


Fig. 43 The flow of tasks in the transition activity

Role Design Activity

The Role Design activity is composed by the following tasks:

Activity	Task	Task Description	Role Involved
Role Design	Action Design	*Assignment of tasks to roles and identification of the actions necessary in order to achieve each specific task*	Architectural Designer *(perform)*

The flow of tasks inside the Role Design activity is reported in Fig. 44.

Resource Design activity

The Resource Design activity is composed by the following tasks:

Activity	Task	Task Description	Role Involved
Operation Design	Resource Design	*Assignment of functions to resources and identification of the operations necessary for providing each specific function*	Architectural Designer *(perform)*

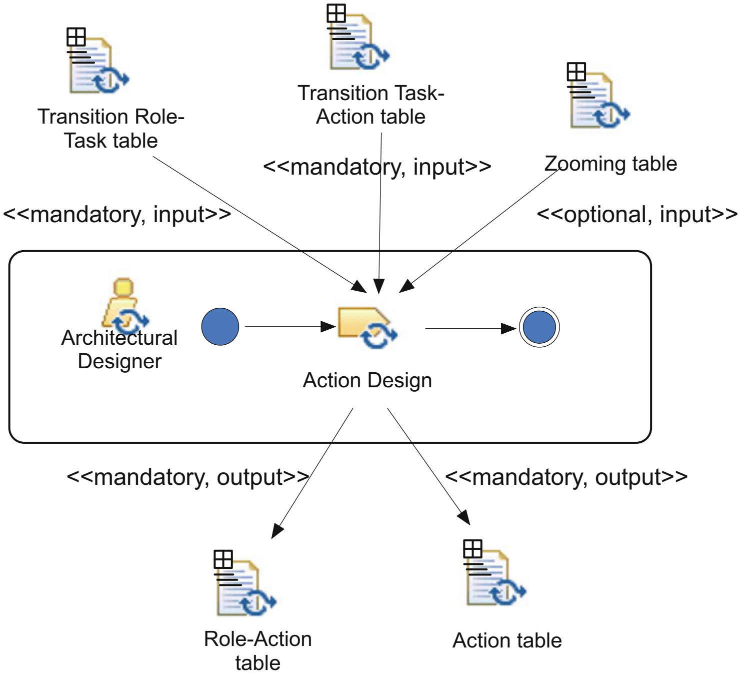


Fig. 44 The flow of tasks in the role design activity

The flow of tasks inside the Resource Design activity is reported in Fig. 45.

Constraint Design Activity

The Constraint Design activity is composed by the following tasks:

Activity	Task	Task Description	Role Involved
Constraint Design	Constraint Design	*Identification of the rules that enable and bound the entities' behaviour starting from the dependencies analysed in the previous step*	Architectural Designer *(perform)*

The flow of tasks inside the Constraint Design activity is reported in Fig. 46.

Interaction Design Activity

The Interaction Design activity is composed by the following tasks:

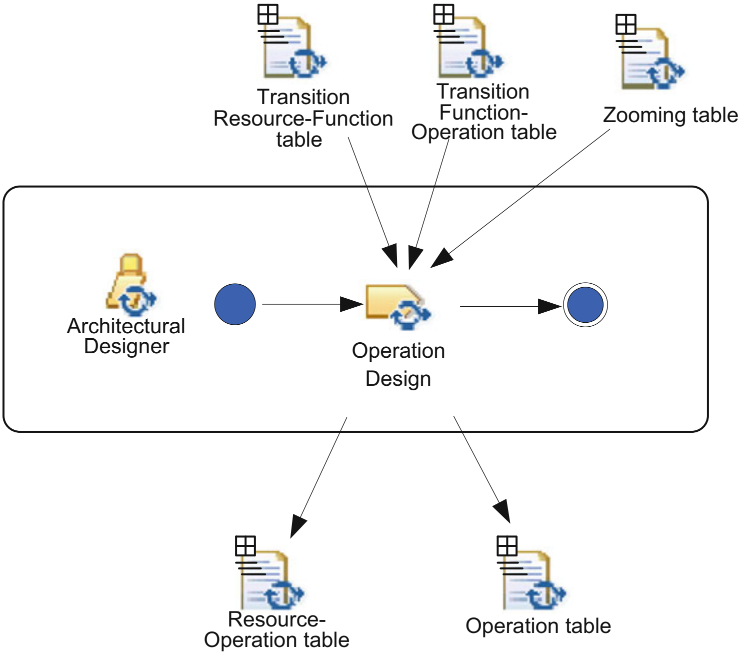


Fig. 45 The flow of tasks in the resource design activity

Activity	Task	Task Description	Role Involved
Interaction Design	Interaction Design	*Identification of the interactions hat represent the acts of the interaction among roles, among resources and between roles and resources starting from the dependencies analysed in the previous step*	Architectural Designer *(perform)*

The flow of tasks inside the Interaction Design activity is reported in Fig. 47.

Space Design Activity

The Space Design activity is composed by the following tasks:

Activity	Task	Task Description	Role Involved
Space Design	Space Design	*Identification of the spaces starting from the topology constraints analysed in the previous step*	Architectural Designer *(perform)*

The flow of tasks inside the Space Design activity is reported in Fig. 48.

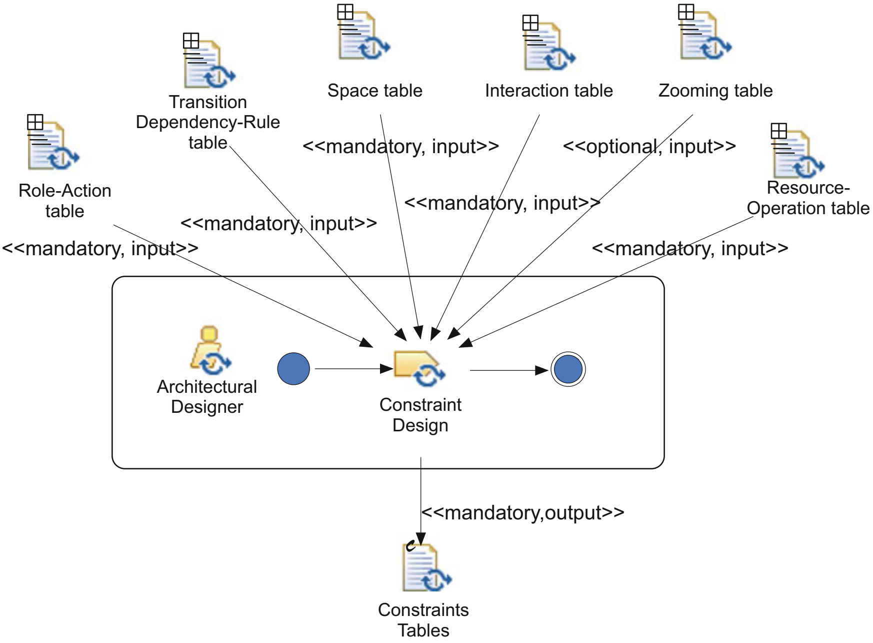


Fig. 46 The flow of tasks in the constraint design activity

2.3.3 Work Products

The Architectural Design step consists of five sets of tables: Transition Tables, Entities Tables, Interactions Tables, Constraints Tables and Topological Tables. Figure 49 reports the relationships among the work products of this step and the MMMElements of the Architectural Design step. In Fig. 49 are also reported the relationships among the Zooming table and the MMMElements of the Architectural Design—see Sect. 1.3.1 for details.

Kinds of Work Products

Table 4 describes all the work products of the Architectural Design. In particular, the first set of work products is the outcome of the Transition activity, the second is the outcome of the Role Design and Resource Design activities, the third is the outcome of the Interaction Design activity, the fourth is the outcome of the Constraint Design activity and the last is the outcome of the Space Design activity.

Transition Tables

Figure 50 presents an example of the Transition Tables for the conference management case study.

Entities Tables

Figure 51 presents an example of the Entities Tables for the conference management case study.

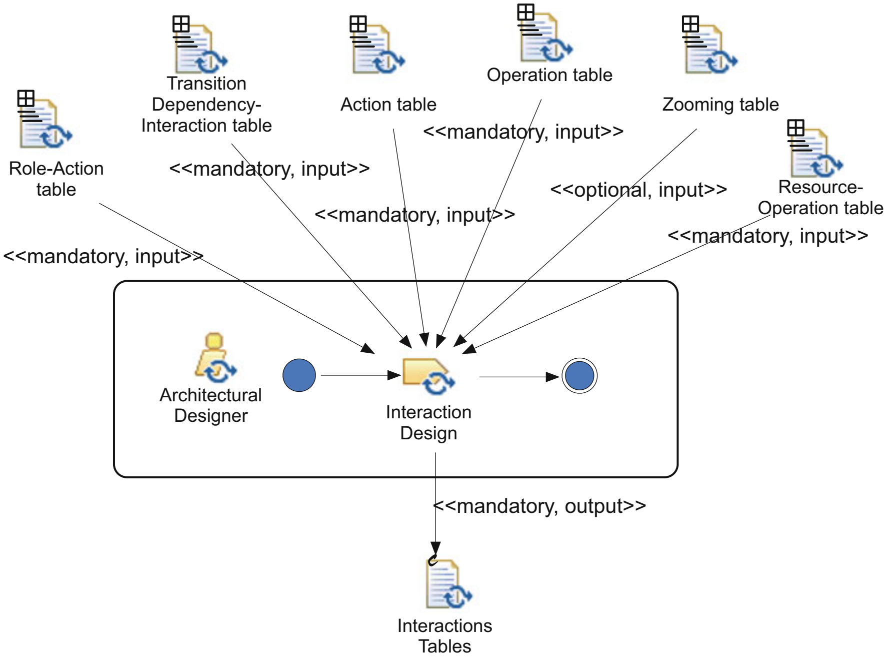


Fig. 47 The flow of tasks in the interaction design activity

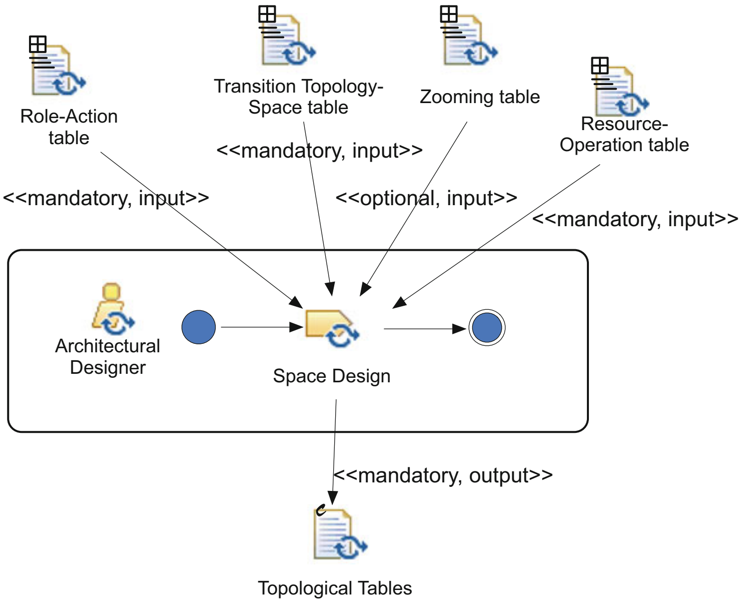


Fig. 48 The flow of tasks in the space design activity

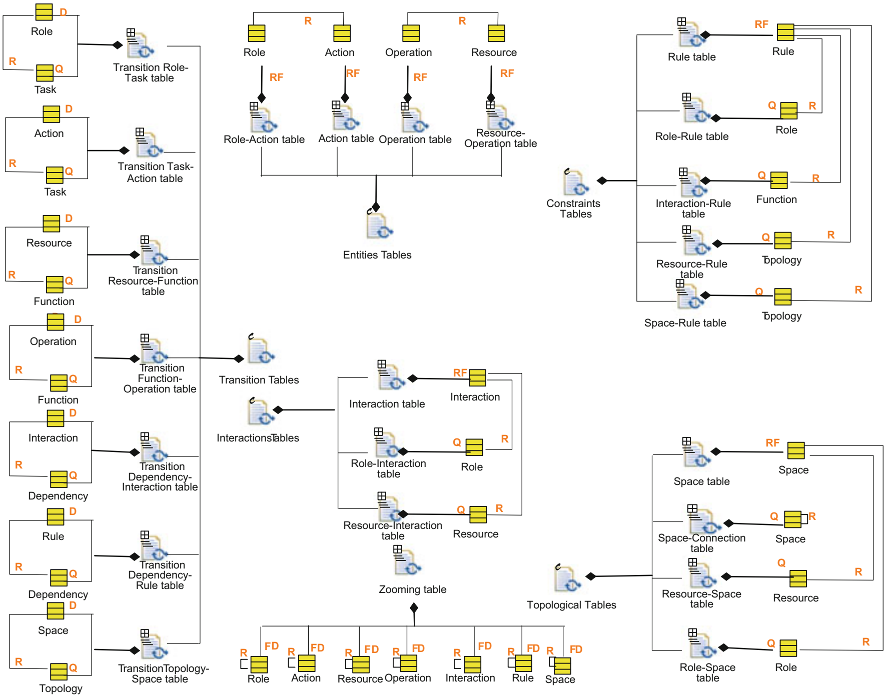


Fig. 49 The architectural design work products

Interactions Tables

Figure 52 presents an example of the Interactions Tables for the conference management case study.

Constraints Tables

Figure 53 presents an example of the Constraints Tables for the conference management case study.

Topological Tables

Figure 54 presents an example of the Topological Tables for the conference management case study.

2.4 The Detailed Design

The goal of Detailed Design is to choose the most adequate representation level for each architectural entity, thus leading to depict one (detailed) design from the many potential alternatives outlined in the Architectural Design step. Figure 55

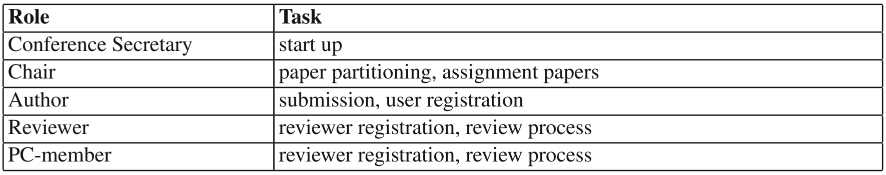
Table 4 Architectural Design work products kinds

Name	Description	Work product kind
Transition Tables	*A composition of others tables that links the Analysis step with the Architectural Design step*	Composite (structured)
Transition Role-Task table ($(L)TRT_t$)	*Specification of the mapping between each role and the tasks assigned to it*	Structured
Transition Task-Action table ($(L)TTA_t$)	*Specification of the mapping between each task and the generated actions*	Structured
Transition Resource-Function table ($(L)TRF_t$)	*Specification of the mapping between each functions and the functions assigned to it*	Structured
Transition Function-Operation table ($(L)TFO_t$)	*Specification of the mapping between each functions and the generated operations*	Structured
Transition Dependency-Interaction table ($(L)TDI_t$)	*Specification of the mapping between each dependency and the generated interactions*	Structured
Transition Dependency-Rule table ($(L)TDRu_t$)	*Specification of the mapping between each dependency and the generated rules*	Structured
Transition Topology-Space table ($(L)TTopS_t$)	*Specification of the mapping between each topology and the generated spaces*	Structured
Entities Tables	*A composition of others tables that describes both the active entities (able to perform some actions in the system) and the passive entities which provide services*	Composite (structured)
Action table ($(L)A_t$)	*Description of the actions executable by some roles*	Structured
Operation table ($(L)O_t$)	*Description of the operations provided by resources*	Structured
Role-Action table ($(L)RA_t$)	*Specification of the actions that each role can do*	Structured
Resource-Operation table ($(L)RO_t$)	*Specification of the operations that each resource can provide*	Structured
Interactions Tables	*A composition of others tables that describe the interaction between roles and resources*	Composite (structured)
Interaction table ($(L)I_t$)	*Description of the single interactions*	Structured
Action-Interaction table ($(L)AcI_t$)	*Specification of the interactions where each action is involved*	Structured
Operation-Interaction table ($(L)OpI_t$)	*Specification of the interactions where each operation is involved*	Structured
Constraints Tables	*A composition of others tables that describes the constraints over the entities behaviours*	Composite (structured)
Rule table ($(L)Ru_t$)	*Description of the rules*	Structured
Rule-Interaction table ($(L)IRu_t$)	*Specification of the constraints over the interactions*	Structured

(continued)

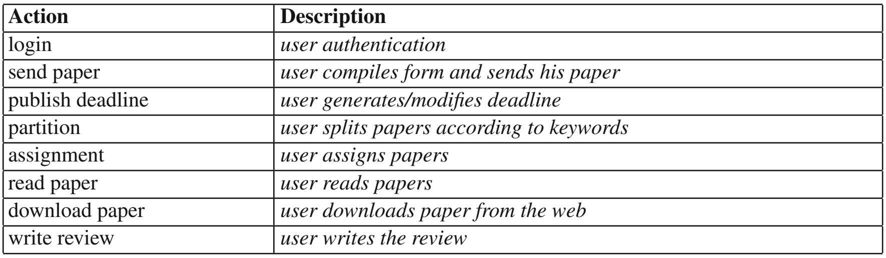
Table 4 (continued)

Name	Description	Work product kind
Resource-Rule table $((L)ReI_t)$	*Specification of the rules where each resource is involved*	Structured
Role-Rule table $((L)RoRu_t)$	*Specification of the rules where each role is involved*	Structured
Space-Rule table $((L)SRu_t)$	*Specification of the rules where each space is involved*	Structured
Topological Tables	*A composition of others tables that describes the logical structure of the environment*	Composite (structured)
Space table $((L)S_t)$	*Description of the spaces*	Structured
Space-Connection table $((L)SC_t)$	*Specification of the connections among the spaces of a given layer (the hierarchical relations between spaces are expressed via the Zooming Table)*	Structured
Resource-Space table $((L)ReS_t)$	*Specification of the all spaces where resources is involved*	Structured
Role-Space table $((L)RoS_t)$	*Specification of the all spaces where role is involved*	Structured



Role	Task
Conference Secretary	start up
Chair	paper partitioning, assignment papers
Author	submission, user registration
Reviewer	reviewer registration, review process
PC-member	reviewer registration, review process

Fig. 50 Transition role-task table $(C)TRT_t$



Action	Description
login	*user authentication*
send paper	*user compiles form and sends his paper*
publish deadline	*user generates/modifies deadline*
partition	*user splits papers according to keywords*
assignment	*user assigns papers*
read paper	*user reads papers*
download paper	*user downloads paper from the web*
write review	*user writes the review*

Fig. 51 Action table $(C)A_t$

presents the Detailed Design process, while Fig. 56 presents the flow of activities, the involved roles and the work products.

2.4.1 Process Roles

One role is involved in the Detailed Design: the Detailed Designer.

Interaction	**Description**
UserInfInteraction	*accessing user information*
ReviewerInfInteraction	*accessing reviewer information*
PaperInfInteraction	*accessing paper information*
PartInfInteraction	*accessing partitioning information*
SubInfInteraction	*accessing submission information*
AssInfInteraction	*accessing assignment information*
ReviewInfInteraction	*accessing review information*
WebAccessInteraction	*accessing website*

Fig. 52 Interaction table $(C)I_t$

Rule	**Description**
RegSubRule	*the submission has to be done after the author registration*
RegAssRule	*the assignment has to be done after reviewer registration*
PartAssRule	*the assignment has to be done after partitioning*
AssRevRule	*write review after the assignment*
UserInfRule	*user can access & modify only his own information*
ReviewerInfRule	*reviewer can access & modify only his own information*
AuthorInfRule	*author can access & modify only public information of owned paper(s)*
MatchRule	*papers can be partitioned according to their keywords*
SubInfRule	*send paper only before deadline submission*
AutRevRule	*PC-Member/Reviewer cannot review his own papers*
ReviewRule	*PC-Member/Reviewer cannot access private information about owned papers*
WebAccessRule	*access to the system must be authorised*

Fig. 53 Rule table $(C)Ru_t$

Space	**Description**
S-place	*the space where resources have to be allocated*

Fig. 54 Space table $(C)S_t$

Detailed Designer

The Detailed Designer is responsible for

- Mapping the MMMElements of the Architectural Design to the MMMElements of the Detailed Design
- Identifying the most suitable system architecture among all the possibilities provided in the Architectural Design step
- Assigning roles to agents
- Assigning actions to individual artifacts
- Assigning roles to societies
- Assigning resources to environmental artifacts
- Assigning resources to aggregate
- Assigning operations to environmental artifacts
- Assigning rules to artifacts
- Designing artifacts usage interfaces

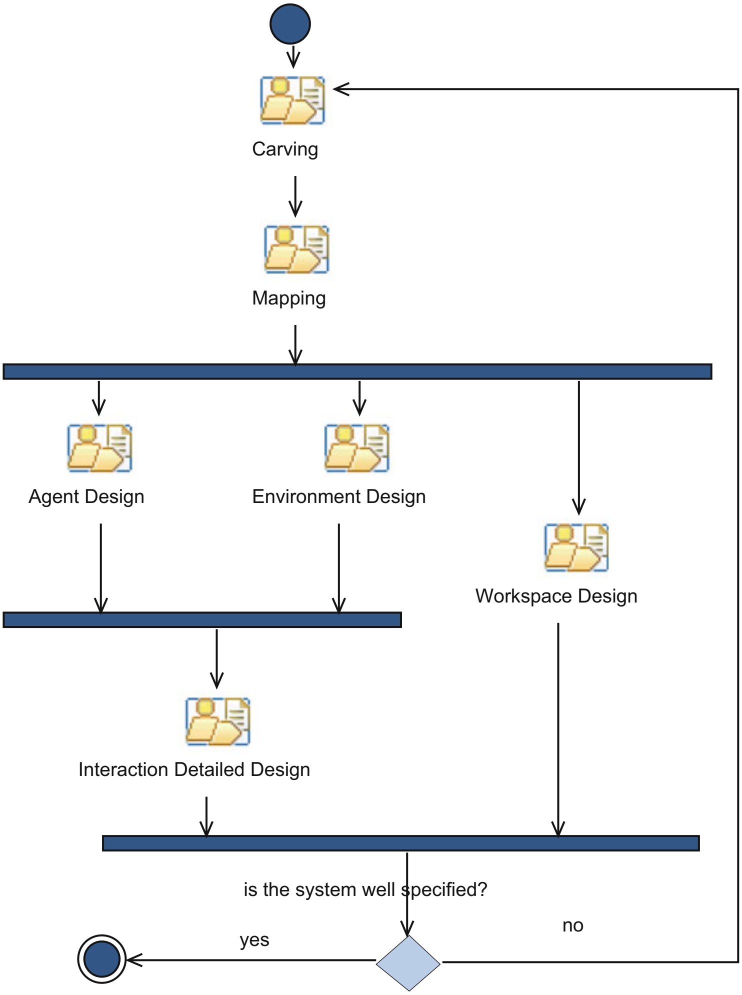


Fig. 55 The detailed design process

- Assigning interactions to uses and designing the specific protocols
- Assigning interactions to manifests and designing the specific protocols
- Assigning interactions to speakTo and designing the specific protocols
- Assigning interactions to linkedTo and designing the specific protocols
- Assigning spaces to workspaces and designing them

2.4.2 Activity Details

Carving Activity

The Carving activity is composed of the following tasks:

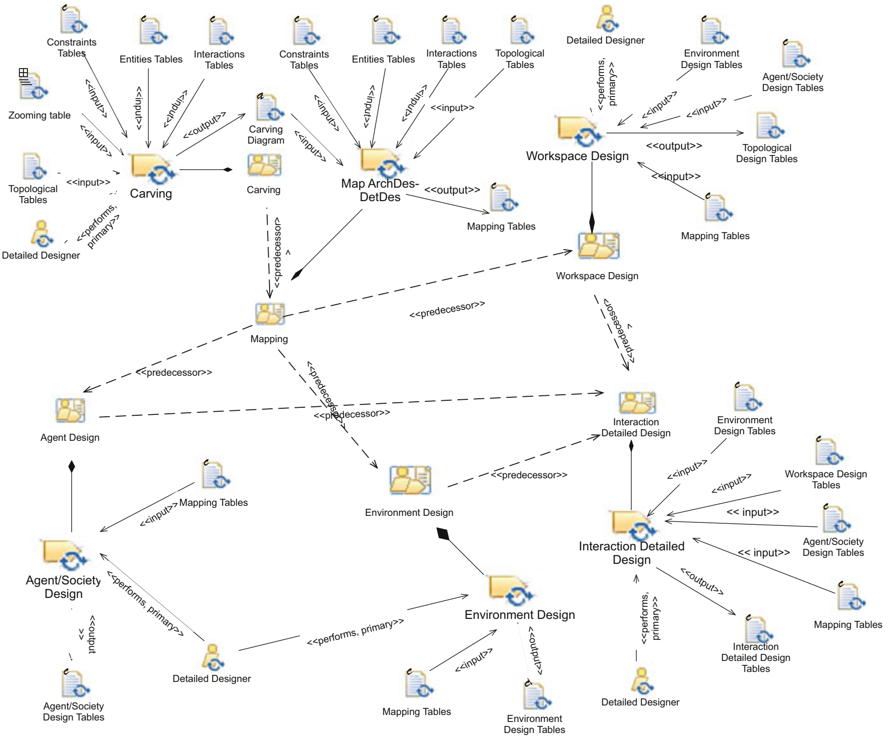


Fig. 56 The detailed design flow of activities, roles, and work products

Activity	Task	Task description	Role involved
Carving	Carving	*For each entity the appropriate layer of representation is chosen*	Detailed Designer *(perform)*

The flow of tasks inside the Carving activity is reported in Fig. 57.

Mapping Activity

The Mapping activity is composed of the following tasks:

Activity	Task	Task description	Role involved
Mapping	Map ArchDes-DetDes	*Mapping of the MMMElements defined in Architectural Design to Detailed Design MMMElements so as to generate the initial version of the Detailed Design models*	Detailed Designer *(perform)*

The flow of tasks inside the Mapping activity is reported in Fig. 58.

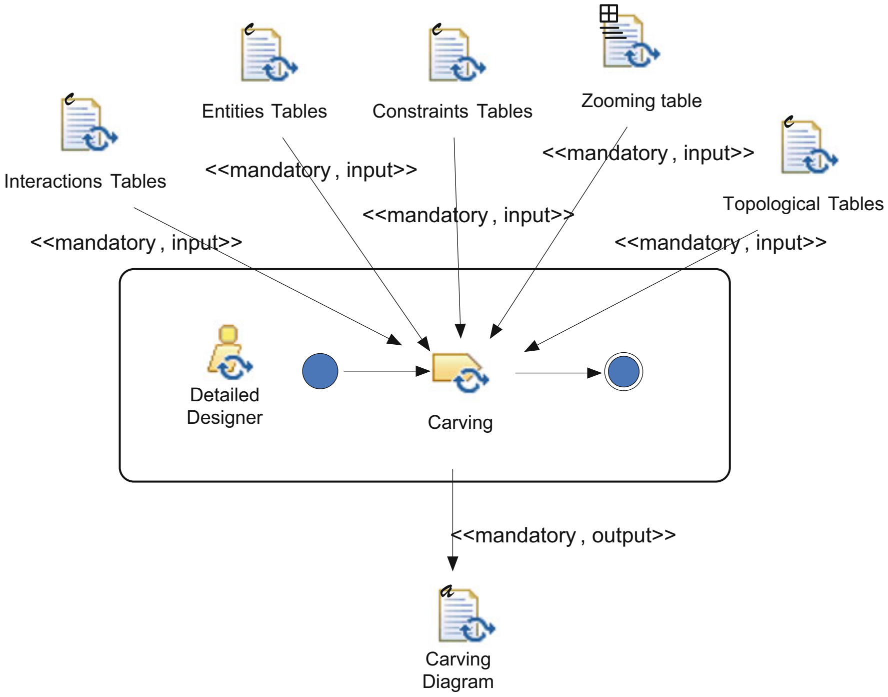


Fig. 57 The flow of tasks in the carving activity

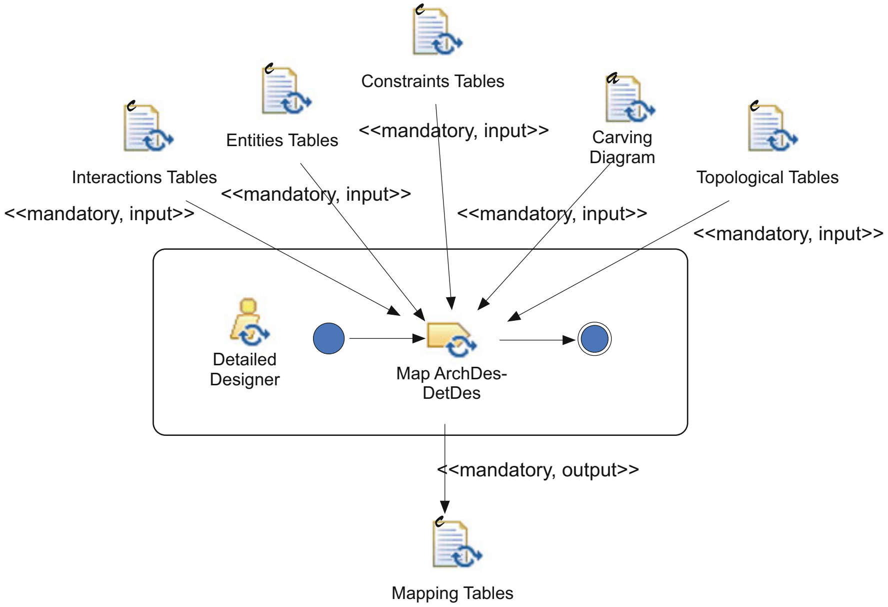


Fig. 58 The flow of tasks in the mapping activity

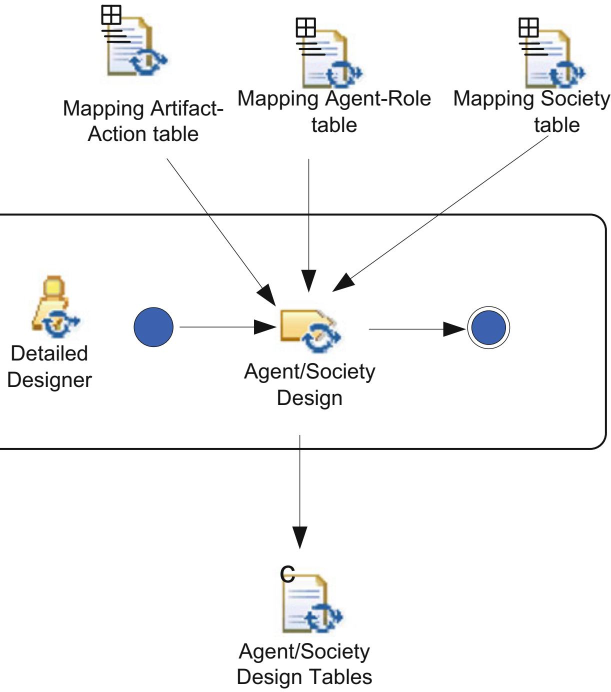

Mapping Artifact-Action table
Mapping Agent-Role table
Mapping Society-Role table
Detailed Designer
Agent/Society Design
C
Agent/Society Design Tables

Fig. 59 The flow of tasks in the agent design activity

Agent Design activity

The Agent Design activity is composed of the following tasks:

Activity	Task	Task description	Role involved
Agent Design	Agent/ Society Design	*Design of Agents and Societies*	Detailed Designer *(perform)*

The flow of tasks inside the Agent Design activity is reported in Fig. 59.

Environment Design Activity

The Environment Design activity is composed of the following tasks:

Activity	Task	Task description	Role involved
Environment Design	Environment Design	*Design of Artifacts and Aggregates*	Detailed Designer *(perform)*

The flow of tasks inside the Environment Design activity is reported in Fig. 60.

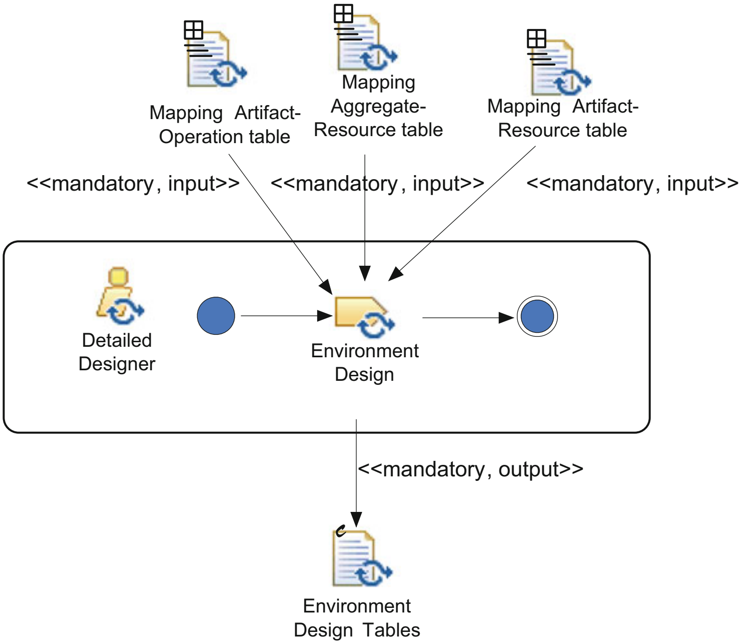

Mapping Artifact-Operation table
Mapping Aggregate-Resource table
Mapping Artifact-Resource table
<<mandatory, input>>
<<mandatory, input>>
<<mandatory, input>>
Detailed Designer
Environment Design
<<mandatory, output>>
Environment Design Tables

Fig. 60 The flow of tasks in the environment design activity

Interaction Detailed Design Activity

The Interaction Detailed Design activity is composed of the following tasks:

Activity	Task	Task description	Role involved
Interaction Detailed Design	Interaction Detailed Design	*Design of the interaction protocols for Uses, Manifests, SpeakTo and LinkedTo identified in the carving*	Detailed Designer *(perform)*

The flow of tasks inside the Interaction Detailed Design activity is reported in Fig. 61.

Workspace Design Activity

The Workspace Design activity is composed by the following tasks:

Activity	Task	Task description	Role involved
Workspace Design	Workspace Design	*Design of workspaces starting from spaces identified in the carving*	Detailed Designer *(perform)*

The flow of tasks inside the Workspace Design activity is reported in Fig. 62.

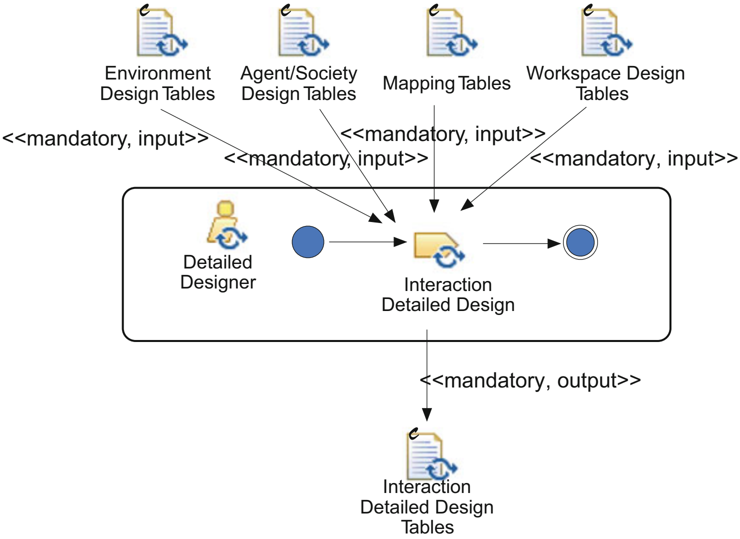


Fig. 61 The flow of task in the interaction detailed design activity

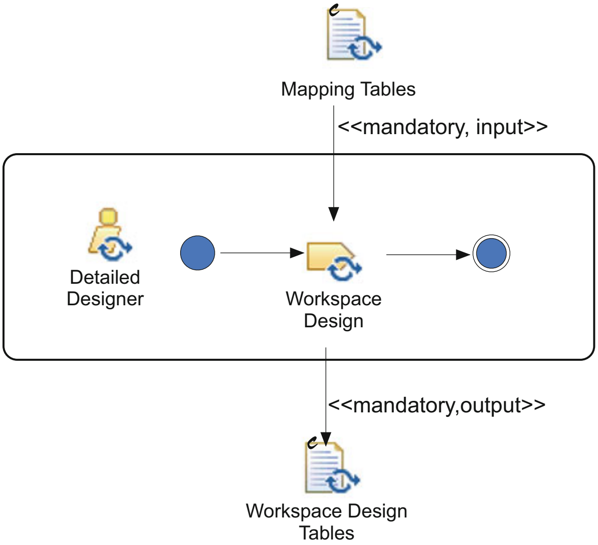


Fig. 62 The flow of task in the workspace design activity

2.4.3 Work Products

The Detailed Design step exploits several sets of tables: namely, Mapping Tables, Agent/Society Design Tables, Environment Design Tables, Interaction Detailed Design Tables and Workspace Design Tables. Figure 63 reports the relationships among the work products and the MMMElements of the Detailed Design step.

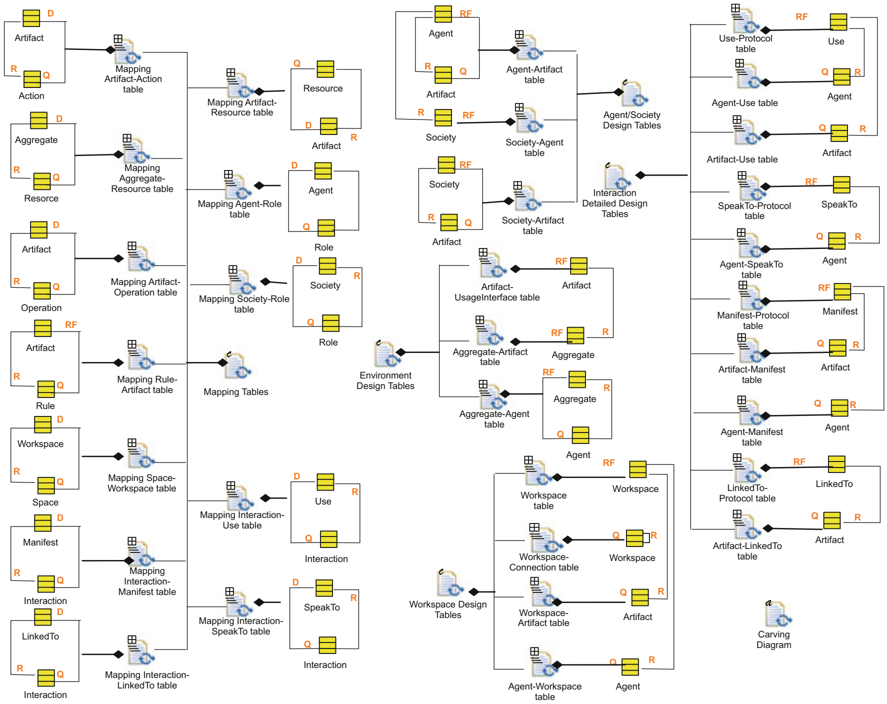


Fig. 63 The detailed design work products

Kinds of Work Products

Table 5 describes all the work products of the Detailed Design. In particular, the first entry is the outcome of the Carving Activity, the second set of work products is the outcome of the Mapping activity, the third set is the outcome of the Agent Design activity, the fourth set is the outcome of the Environment Design activity, the fifth set is the outcome of the Interaction Detailed Design activity and the last set is the outcome of the Workspace Design activity.

Carving Diagram

An example of the Carving Diagram for the conference management system is reported in Fig. 64.

Mapping Tables

Figures 65, 66, and 67 present some examples of the Mapping Tables for the conference management case study.

Agent/Society Design Tables

Figure 68 presents an example of the Agent/Society Design Tables for the conference management case study.

Table 5 Detailed Design work products kinds

Name	Description	Work product kind
Carving Diagram	*Diagram that shows the chosen system architecture*	Structured
Mapping Tables	*A composition of others tables that links the Architectural Design step with the Detailed Design step*	Composite (structured)
Mapping Agent-Role table (MAR_t)	*Specification of the mapping between roles and agents*	Structured
Mapping Society-Role table (MSR_t)	*Specification of the mapping between role and society*	Structured
Mapping Artifact-Action table ($MAAc_t$)	*Specification of the mapping between actions and individual artifacts*	Structured
Mapping Artifact-Resource table ($MArR_t$)	*Specification of the mapping between resources and artifacts*	Structured
Mapping Aggregate-Resource table ($MAggR_t$)	*Specification of the mapping between resources and aggregate*	Structured
Mapping Artifact-Operation table ($MArOp_t$)	*Specification of the mapping between operations and environmental artifacts*	Structured
Mapping Artifact-Rule table ($MArRu_t$)	*Specification of the mapping between rules and the artifacts that implement and enforce them*	Structured
Mapping Artifact-Operation table (MSW_t)	*Specification of the mapping between spaces and workspaces*	Structured
Mapping Interaction-Use table (MIU_t)	*Specification of the mapping between interactions and uses*	Structured
Mapping Interaction-Manifest table (MIM_t)	*Specification of the mapping between interactions and manifests*	Structured
Mapping Interaction-SpeakTo table ($MISp_t$)	*Specification of the mapping between interactions and speakTos*	Structured
Mapping Interaction-LinkedTo table (MIL_t)	*Specification of the mapping between interactions and linkedTos*	Structured
Agent/Society Design Tables	*A composition of others tables that depicts agents, individual artifacts, and the societies derived from the carving operation*	Composite (structured)
Agent-Artifact table (AA_t)	*Specification of the individual artifacts related to each agent*	Structured
Society-Agent table (SA_t)	*Specification of the list of agents belonging to a specific society*	Structured

(continued)

Table 5 (continued)

Name	Description	Work product kind
Society-Artifact table (SAr_t)	*Specification of the list of artifacts belonging to a specific society*	Structured
Environment Design Tables	*A composition of others tables that depicts artifacts, aggregates, agents derived from the carving operation*	Composite (structured)
Artifact-UsageInterface table (AUI_t)	*Specification of the operations provided by each artifact*	Structured
Aggregate-Artifact table ($AggArt_t$)	*Specification of the list of artifacts belonging to a specific aggregate*	Structured
Aggregate-Agent table ($AggAge_t$)	*Specification of the list of agents belonging to a specific aggregate*	Structured
Interaction Detailed Design Tables	*A composition of others tables that concerns the design of interactions among entities*	Composite (structured)
Use-Protocol table (UP_t)	*Description of the protocols for each "use"*	Structured
Agent-Use table ($AgeU_t$)	*Specification of the "use" where each agent is involved*	Structured
Artifact-Use table ($ArtU_t$)	*Specification of the "use" where each artifact is involved*	Structured
SpeakTo-Protocol table (SP_t)	*Description of the protocols for each "speakTo"*	Structured
Agent-SpeakTo table ($AgeSp_t$)	*Specification of the "speakTo" where each agent is involved*	Structured
Manifest-Protocol table (MP_t)	*Description of the protocols for each "manifest"*	Structured
Agent-Manifest table ($AgeM_t$)	*Specification of the "manifest" where each agent is involved*	Structured
Artifact-Manifest table ($ArtM_t$)	*Specification of the "manifest" where each artifact is involved*	Structured
LinkedTo-Protocol table (LP_t)	*Description of the protocols for each "linkedTo"*	Structured
Artifact-LinkedTo table ($ArtL_t$)	*Specification of the "linkedTo" where each artifact is involved*	Structured
Workspace Design Tables	*A composition of others tables that describes the structure of the environment*	Composite (structured)
Workspace table ($(L)W_t$)	*Description of the workspaces*	Structured
Workspace-Connection table ($(L)WC_t$)	*Specification of the connections among the workspaces*	Structured
Workspace-Artifact table ($(L)WArt_t$)	*Specification of the allocation of artifacts in the workspaces*	Structured
Workspace-Agent table ($(L)WA_t$)	*Specification of the list of the workspaces that each agent can perceive*	Structured

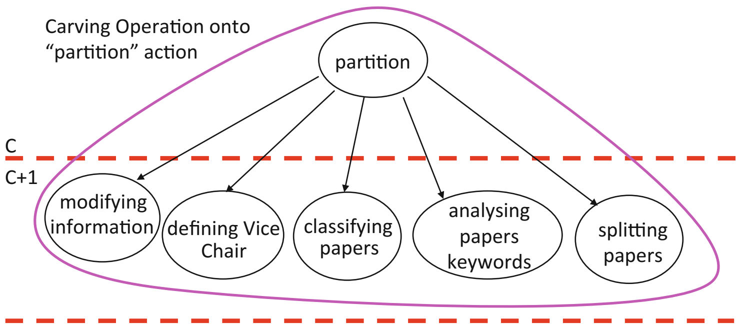


Fig. 64 Carving operation in the conference management system

Agent	**Role**
Conference Secretary Agent	Conference Secretary
Chair Agent	Chair
Author Agent	Author
Reviewer Agent	Reviewer
PC-Member Agent	PC-member

Fig. 65 Mapping agent-role table MAR_t

Artifact	**Resource**
Paper Artifact	Paper DB
People Artifact	People DB
Process Artifact	Process DB
Web Artifact	WebService

Fig. 66 Mapping artifact-resource table $MArR_t$

Artifact	**Rule**
User Artifact	UserInfRule, ReviewerInfRule
Paper Artifact	AuthorInfRule , MatchRule,
Process Artifact	RegSubRule, RegAssRule, PartAssRule, AssRevRule, SubInfRule
Review Artifact	AutRevRule, ReviewRule
Web Artifact	WebAccessRule

Fig. 67 Mapping artifact-rule table $MArRu_t$

Agent	**Artifact**
Conference Secretary Agent	Conference Secretary
Chair Agent	Chair Artifact
Author Agent	Author Artifact
Reviewer Agent	Reviewer Artifact
PC-Member Agent	PC-Member Artifact

Fig. 68 Agent-artifact table AA_t

Artifact	**Usage Interface**
Chair Artifact	read start up information, modify start up information, get info, login, partition, assignment
Author Artifact	login, registration, submit paper
Reviewer Artifact	login, registration, read paper, write review, download paper
PC-Member Artifact	login, read paper, write review, download paper
User Artifact	store user, get user, modify user
Paper Artifact	store paper, get paper, store classification, store partitioning, get partitioning, get assignment, store assignment, store review, check authors, check reviewer, check user, get review
Process Artifact	start conference process, get process, store process, next stage, deadline extension, update rule, read rule
Web Artifact	login
Review Artifact	check access to review information

Fig. 69 Artifact-usageInterface table AUI_t

Use	**Protocol**
ReadUserInfo	get user (id) information
ReadReviewInfo	check user ack get review (paperID) review
PaperInf- Interaction	check user ack get paper (paperID) paper
PartInf- Interaction	check user ack get partitioning partitioning info
SubInf- Interaction	get info info
AssInf- Interaction	check user ack get assignment assignment info
ReviewInf- Interaction	check access to review information ack get review (paperID) review
WebAccess- Interaction	login

Fig. 70 Use-protocol table UP_t

Environment Design Tables

Figure 69 presents an example of the Environment Design Tables for the conference management case study.

Interaction Detailed Design Tables

Figure 70 presents an example of the Interaction Detailed Design Tables for the conference management case study.

Workspace	Artifact
Wplace	Chair Artifact, Author Artifact, Reviewer Artifact, PC-Member Artifact, People Artifact, Process Artifact, Web Artifact, Review Artifact, Paper Artifact

Fig. 71 Workspace-artifact table WA_t

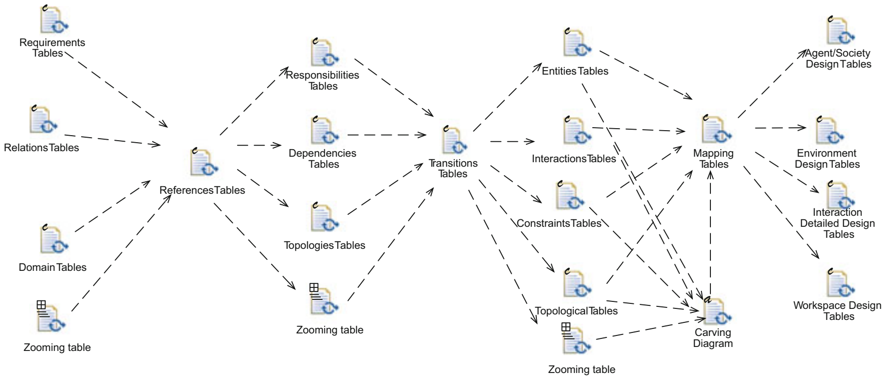


Fig. 72 The work products dependencies

Workspace Design Tables

Figure 71 presents an example of the Workspace Design Tables for the conference management case study.

3 Work Products Dependencies

Figure 72 describes the dependencies among the different SODA composite work products.

Acknowledgements This work was supported by the EU-FP7-FET Proactive project SAPERE—Self-aware Pervasive Service Ecosystems under contract no. 256873.

References

1. DPDF Working Group: FIPA design process documentation template. http://fipa.org/specs/fipa00097/ (2011)
2. Molesini, A., Omicini, A., Denti, E., Ricci, A.: SODA: a roadmap to artefacts. In: Dikenelli, O., Gleizes, M.P., Ricci, A. (eds.) Engineering Societies in the Agents World VI. Lecture Notes in Artificial Intelligence, vol. 3963, pp. 49–62. Springer, Berlin (2006). doi:10.1007/11759683_4. 6th International Workshop (ESAW 2005), Kuşadası, Aydın, 26–28 Oct 2005. Revised, Selected & Invited Papers
3. Molesini, A., Omicini, A., Ricci, A., Denti, E.: Zooming multi-agent systems. In: Müller, J.P., Zambonelli, F. (eds.) Agent-Oriented Software Engineering VI. Lecture Notes in Computer

Science, vol. 3950, pp. 81–93. Springer, Berlin (2006). doi:10.1007/11752660_7. 6th International Workshop (AOSE 2005), Utrecht, 25–26 Jul 2005. Revised and Invited Papers

4. Molesini, A., Nardini, E., Denti, E., Omicini, A.: Advancing object-oriented standards toward agent-oriented methodologies: SPEM 2.0 on SODA. In: Baldoni, M., Cossentino, M., De Paoli, F., Seidita, V. (eds.) 9th Workshop "From Objects to Agents" (WOA 2008) – Evolution of Agent Development: Methodologies, Tools, Platforms and Languages, pp. 108–114. Seneca Edizioni, Palermo (2008). http://www.pa.icar.cnr.it/woa08/materiali/Proceedings.pdf
5. Molesini, A., Nardini, E., Denti, E., Omicini, A.: Situated process engineering for integrating processes from methodologies to infrastructures. In: Shin, S.Y., Ossowski, S., Menezes, R., Viroli, M. (eds.) 24th Annual ACM Symposium on Applied Computing (SAC 2009), vol. II, pp. 699–706. ACM, Honolulu (2009). doi:10.1145/1529282.1529429
6. Molesini, A., Denti, E., Omicini, A.: Agent-based conference management: a case study in SODA. Int. J. Agent Oriented Softw. Eng. **4**(1), 1–31 (2010). doi:10.1504/IJAOSE.2010.029808
7. Molesini, A., Omicini, A.: Documenting SODA: an evaluation of the process documentation template. In: Omicini, A., Viroli, M. (eds.) WOA 2010 – Dagli oggetti agli agenti. Modelli e tecnologie per sistemi complessi: context-dependent, knowledge-intensive, nature-inspired e self-*, CEUR Workshop Proceedings, vol. 621, pp. 95–101. Sun SITE Central Europe, RWTH Aachen University, Rimini (2010). http://CEUR-WS.org/Vol-621/paper14.pdf
8. Object Management Group: Software & systems process engineering meta-model specification 2.0. http://www.omg.org/spec/SPEM/2.0/PDF (2008)
9. Omicini, A.: SODA: societies and infrastructures in the analysis and design of agent-based systems. In: Ciancarini, P., Wooldridge, M.J. (eds.) Agent-Oriented Software Engineering. Lecture Notes in Computer Science, vol. 1957, pp. 185–193. Springer, Berlin (2001). doi:10.1007/3-540-44564-1_12. 1st International Workshop (AOSE 2000), Limerick, 10 June 2000. Revised Papers
10. Omicini, A.: Formal ReSpecT in the A&A perspective. Electron. Notes Theor. Comput. Sci. **175**(2), 97–117 (2007). doi:10.1016/j.entcs.2007.03.006. 5th International Workshop on Foundations of Coordination Languages and Software Architectures (FOCLASA'06), CONCUR'06, Bonn, 31 Aug 2006. Post-proceedings
11. Omicini, A., Ricci, A., Viroli, M.: *Agens Faber*: toward a theory of artefacts for MAS. Electron. Notes Theor. Comput. Sci. **150**(3), 21–36 (2006). doi:10.1016/j.entcs.2006.03.003. 1st International Workshop "Coordination and Organization" (CoOrg 2005), COORDINATION 2005, Namur, 22 April 2005. Proceedings
12. Seidita, V., Cossentino, M., Gaglio, S.: Using and extending the SPEM specifications to represent agent oriented methodologies. In: Luck, M., Gómez-Sanz, J.J. (eds.) Agent-Oriented Software Engineering IX. Lecture Notes in Computer Science, vol. 5386, pp. 46–59. Springer, Berlin (2009). doi:10.1007/978-3-642-01338-6. 9th International Workshop (AOSE 2008), Estoril, 12–13 May 2008, Revised Selected Papers
13. SODA: Home page. http://soda.apice.unibo.it (2009)
14. Sommerville, I.: Software Engineering, 8th edn. Addison-Wesley, Reading (2007)