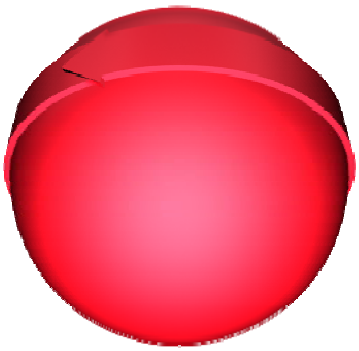
ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

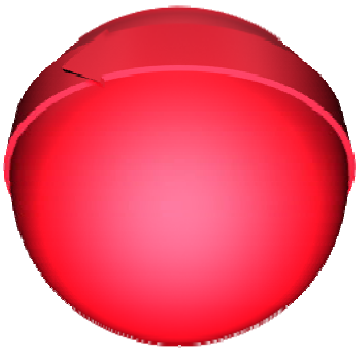
SODA



Societies in Open Distributed Agent spaces

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

SODA



Societies in Open Distributed Agent spaces

Date:	22nd October 2012
Version:	Second Draft
Pages:	
Responsible:	Ambra Molesini
Authors:	Ambra Molesini
	Andrea Omicini
Contact:	ambra.molesini@unibo.it

Contents

1 Introduction

SODA (Societies in Open and Distributed Agent spaces) [1, 6, 7, 9] 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 the system complexity, applied throughout the analysis and design process. 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 agent structure and inner functionality, it could be used in conjunction with SODA. So, SODA focus on *inter-agent* issues, like the engineering of societies and environment for MAS.

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: *i)* the abstractions for modelling/designing the system's active part (task, role, agent, etc.); *ii)* those for the reactive part (function, resource, artifact, etc.); and *iii)* 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*, which includes the Requirements Analysis and the Analysis steps, and the *Design phase*, including the Architectural Design and the Detailed Design steps. Each sub-phase models (designs) the system by exploiting 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, and another for interaction and organisational rules.

An overview of the methodology structure is shown in Figure 1.1: each step is practically described in terms of a set of relational tables, listed in the figure. In the remainder of this document, the SODA process will be described first as a whole process then through its four steps.

Useful reference about the SODA process are the following:

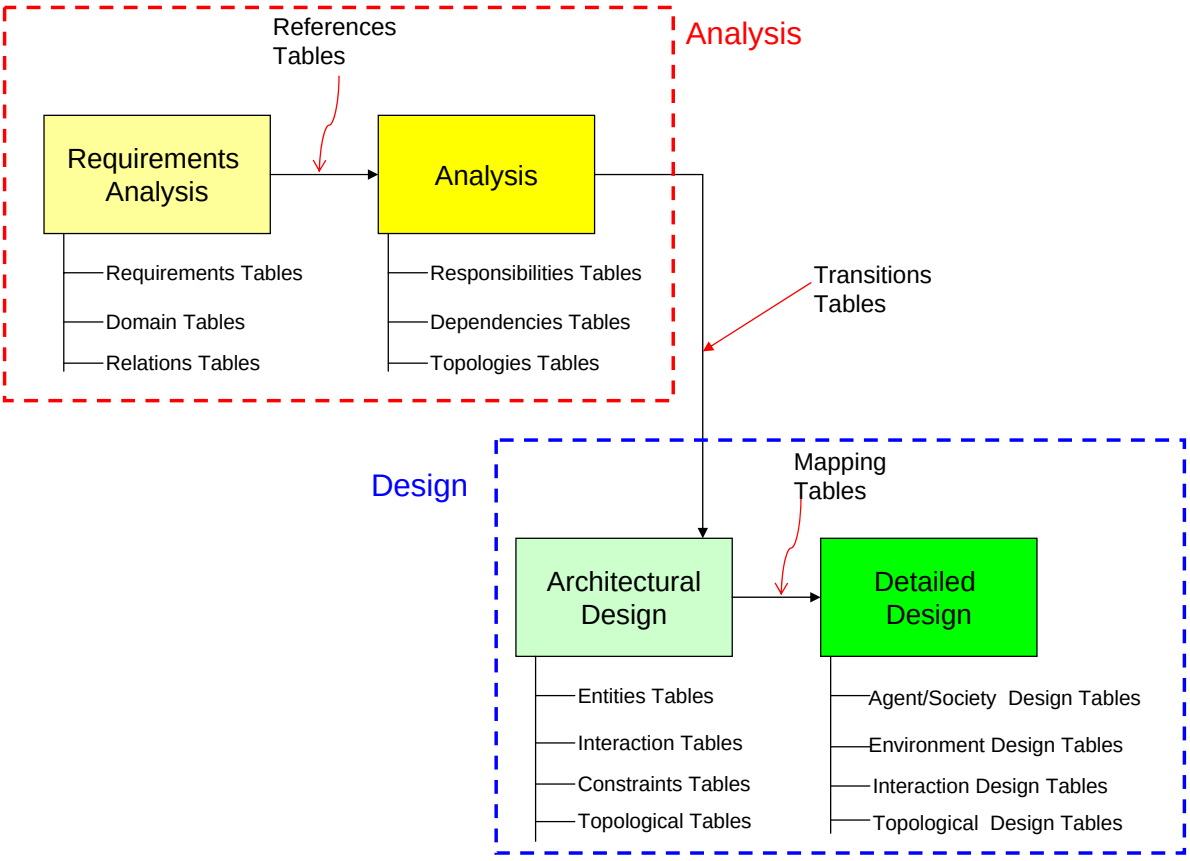


Figure 1.1: An overview of the SODA process

- 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 [7].
- A.Molesini, A.Omicini, E.Denti, A.Ricci. SODA: A Roadmap to Artefacts [6].
- A.Molesini, E.Denti, A.Omicini. Agent-based Conference Management: A Case Study in SODA [2].
- A.Molesini, E.Nardini, E.Denti, A.Omicini Advancing Object-Oriented Standards Toward Agent-Oriented Methodologies: SPEM 2.0 on SODA [3].
- A.Molesini, E.Nardini, E.Denti, A.Omicini. Situated Process Engineering for Integrating Processes from Methodologies to Infrastructures

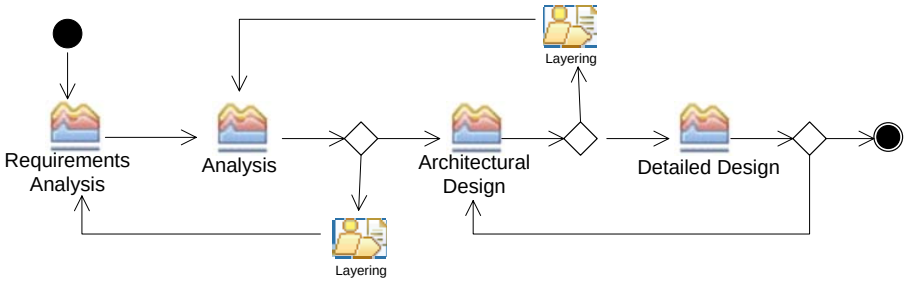


Figure 1.2: The SODA process phases

[4].

- A.Molesini, A.Omicini. Documenting SODA: An Evaluation of the Process Documentation Template [5].

1.1 Process Lifecycle

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, including the *Architectural Design* and the *Detailed Design* steps.

SODA phases and steps are arranged according to an iterative process model (see Figure 1.2):

- *Requirements Analysis* covers all the phases related to the actor identification, the requirements elicitation and analysis, and the 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 family 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 of which describes a specific MAS Meta-model Element (MMMElement) and its relationships with the other MMMElements. The details of each step will be discussed in the following section.

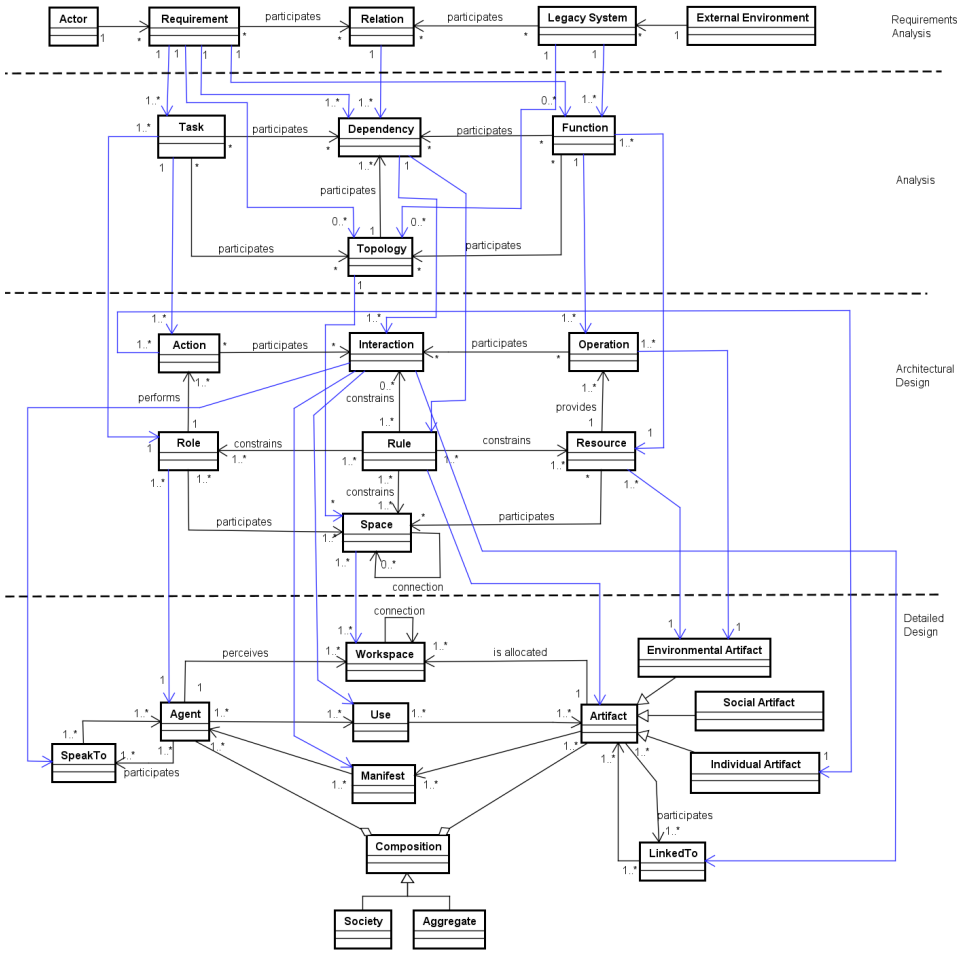


Figure 1.3: The SODA Meta-model

1.2 Meta-model

The meta-model adopted by SODA is represented in Figure 1.3, 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.

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 suitable *Relation* entities.

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*, i.e. topological constraints over the environment.

Topologies are often derived from functions, but can also constrain / affect task achievement.

Architectural Design The main goal of this stage is to assign responsibilities of achieving tasks to *Roles*, and responsibilities of providing 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*, i.e. conceptual places structuring the environment.

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 of 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 1.1: 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
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 then described in principle by all the layers defined, and could then be modelled by a number of different – albeit 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, next section presents the SODA's 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 a conceptual tool aimed at dealing with the complexity of the system representation: the system is represented as composed by different layers of abstraction, between which one could conceptually move by means of a *layering operation*. This tool in SODA is called "Layering" and can be represented as a *capability pattern* [8]—i.e., a reusable portion of the process, as shown in Figure 1.4 where the layering process is detailed. In particular, the layering process has two activities: *(i)* the selection of a specific layer for refining / completing the abstractions models in the methodology process (Select Layer activity), and *(ii)* the creation of a new layer in the system by *in-zooming* – i.e., increasing the system detail – or *out-zooming* – i.e., increasing the system abstraction – activities. In latter case, the layering process terminates with the projection activity needed to project the abstractions from one layer to another "as they are", 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 Figure 1.5. The projected entities are

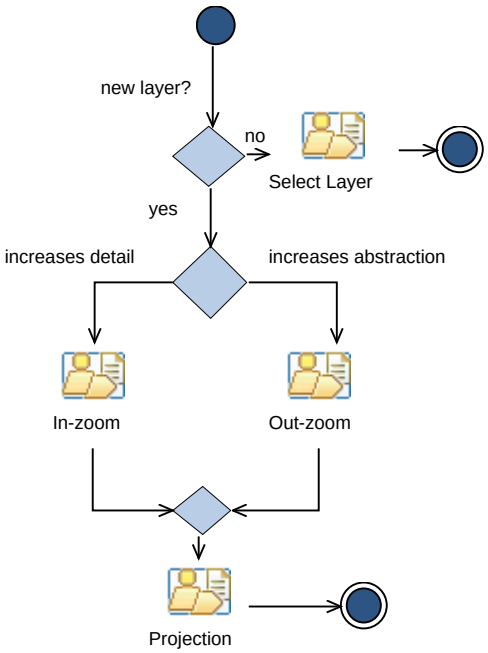


Figure 1.4: The Layering process

identified by means the prefix + if they are projected from a more abstract layer to a more detailed layer (see entity E2 in Figure 1.5), with − otherwise (see entity E1 in Figure 1.5).

Figure 1.6 depicts a more detailed view of the Layering capability pattern showing the flow of activities, the process roles involved, finally the input and work products of each activity.

Process roles

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

- 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 new created SODA's tables.

Activity Details

Select Layer activity The flow of tasks inside this activity is reported in Figure 1.7; the tasks are detailed in the following table.

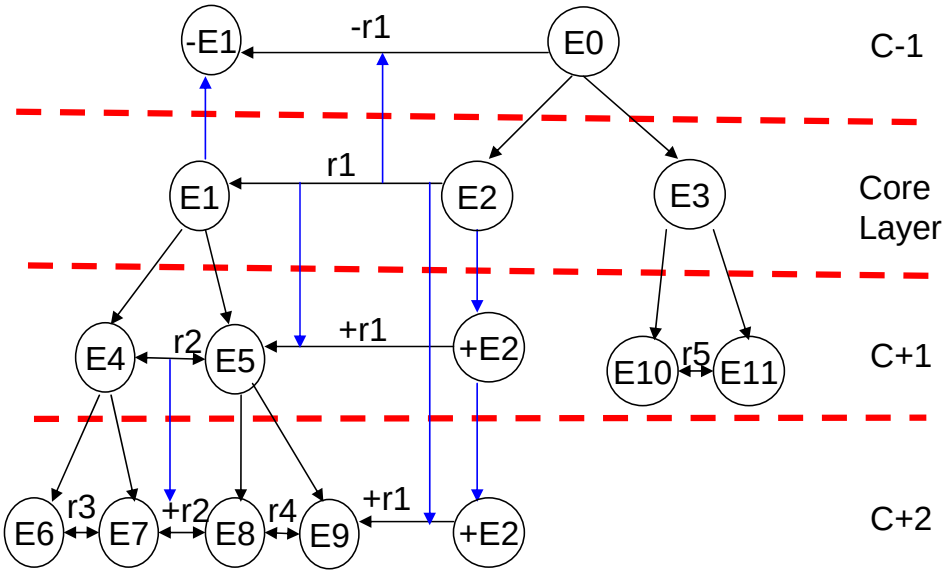


Figure 1.5: An Example of Layering

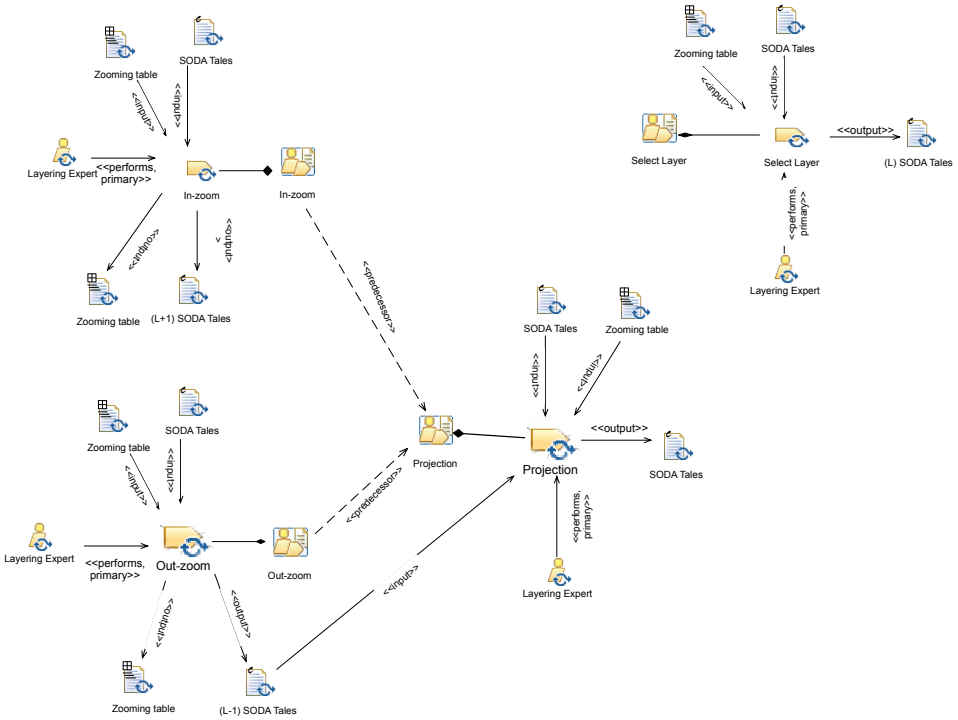


Figure 1.6: The Layering flow of activities, roles, and work products

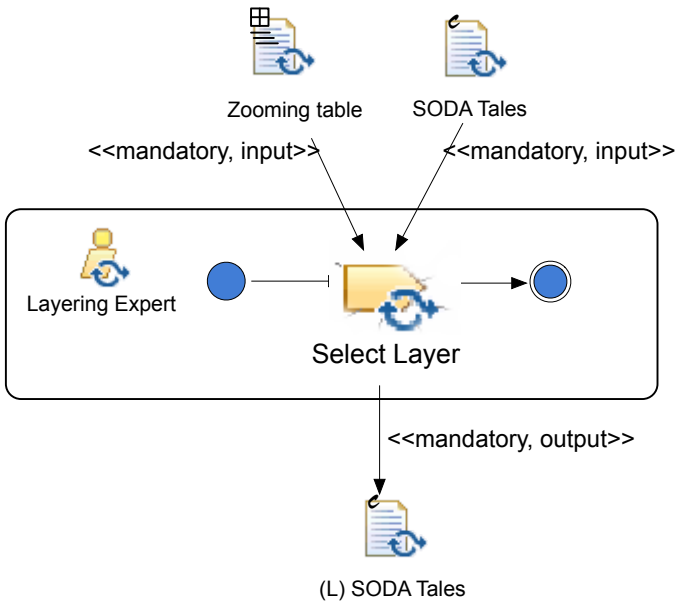


Figure 1.7: The flow of task in the Select Layer activity

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)

In-zoom activity The flow of tasks inside this activity is reported in Figure 1.8; 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 this activity is reported in Figure 1.9; 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)

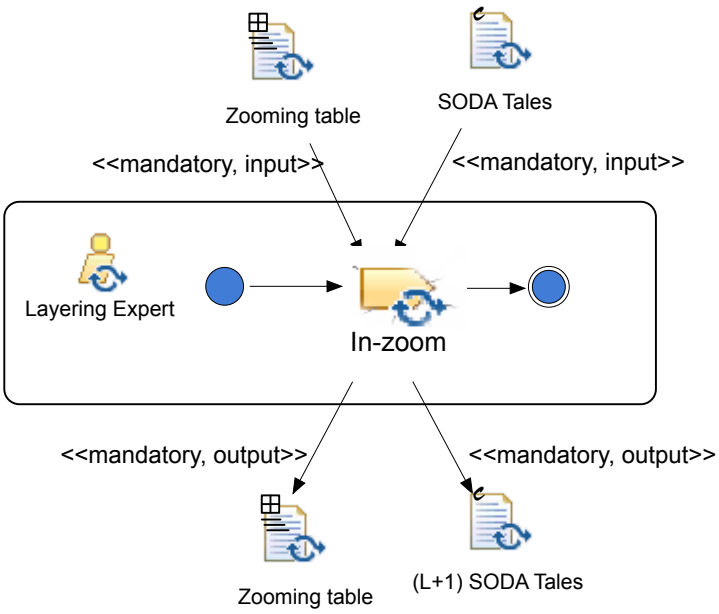


Figure 1.8: The flow of task in the In-zoom activity

Projection activity The flow of tasks inside this activity is reported in Figure 1.10; 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)

Work products

Layering generates one work product: the Zooming table. Its relationships with the MMMElements are described in Figure 1.11.

This diagram represents the Layering in terms of the Work Product and its relationships with the SODA meta-model (Section 1.2) elements. Each MMMElement is represented using an UML class icon (yellow) and, in the documents, such 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 artefact (the MMMElement is instantiated in this artefact);
- *reFined* (F label): this means that the MMMElement is refined in the work product (for instance by means of attribute definition);

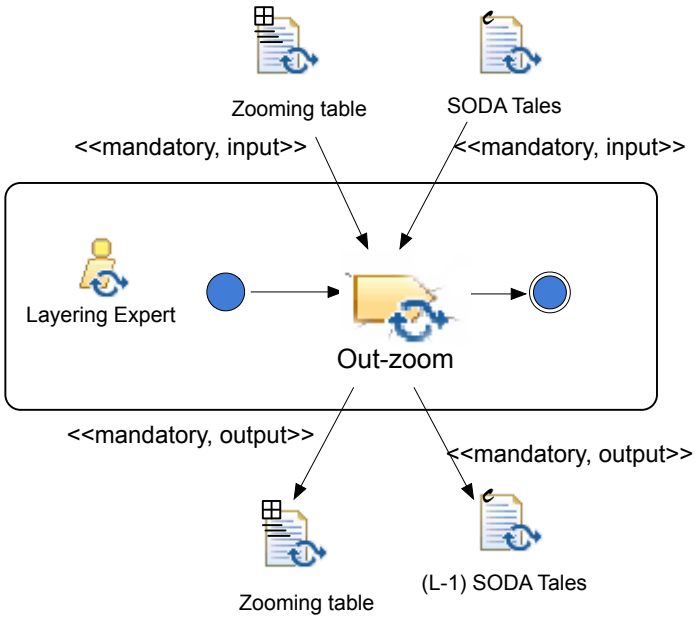


Figure 1.9: The flow of task in the Out-zoom activity

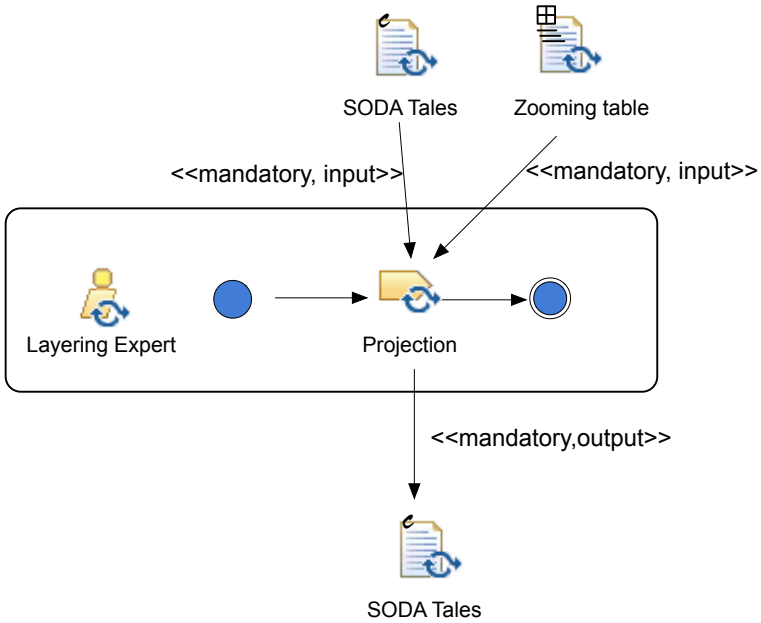


Figure 1.10: The flow of task in the Projection activity

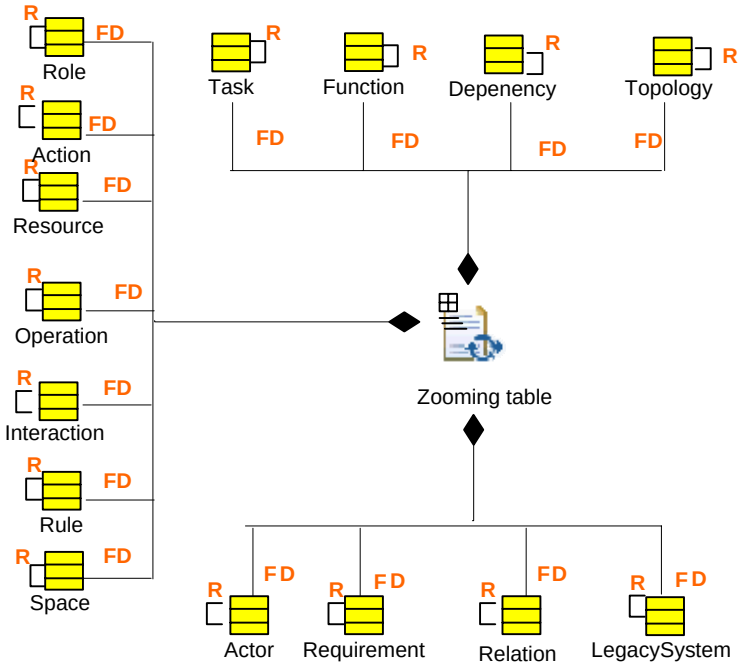


Figure 1.11: The Layering work products

- *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 has been already defined and it is reported in this artefact 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 has been defined in another part of the process.

Kinds of work products Layering is represented by means of a Zooming table ($(C)Z_t$)—see Figure 1.12. 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 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 at layer C at 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 showed in Figure 1.5.

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

Figure 1.12: $(L)Z_t$

Examples of work products In Figure 1.13, Figure 1.14, Figure 1.15 are reported the Zooming tables modelling the example proposed in Figure 1.5. In particular, Figure 1.13 shows the relationships between layer C – the core layer – to layer $C + 1$, where the 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 $C + 1$ and it is reported inside the in-zoom of E1 since in layer C E1 has a relation with E2, and such a relation 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 in-zoom of E1 in order to coordinate the E4 and E5; in a similar way r5 comes from the in-zoom of E3.

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

Figure 1.13: $(C)Z_t$

Figure 1.14 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.

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

Figure 1.14: $(C - 1)Z_t$

Finally, Figure 1.15 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.

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

Figure 1.15: $(C+1)Z_t$

2
Phases of the SODA Process

2.1 The Requirements Analysis

The goals of Requirements Analysis are *(i)* characterising both the customers' requirements and the legacy systems with which the system should interact, as well as *(ii)* highlighting the relationships among requirements and legacy systems. Requirements can be categorised in [13]:

- *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. In particular:

Actor — a user of the systems (or the systems itself) that needs several functionalities from the systems. We use the system as an actor in order to express several non-functional requirements as security, standards and so on. Actors are used to make it possible tracing the sources of requirements.

Requirement — a system description statement that have to clarify a system functionality, or a system constraints, or a domain description.

External Environment — is the external world, made by those legacy systems that will interact with the system.

Legacy System — is an individual legacy system.

Relation — is a relationship among Requirements, or among Legacy Systems, or between Requirements and Legacy Systems.

The Requirements Analysis involves three different process roles, and eight work products, as described in Figure 2.2. Figure 2.1 presents the Requirements Analysis process composed by three main activities (Requirements modelling, Environment modelling, and Relations modelling) and several different layering activities (Subsection 1.3.1).

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.

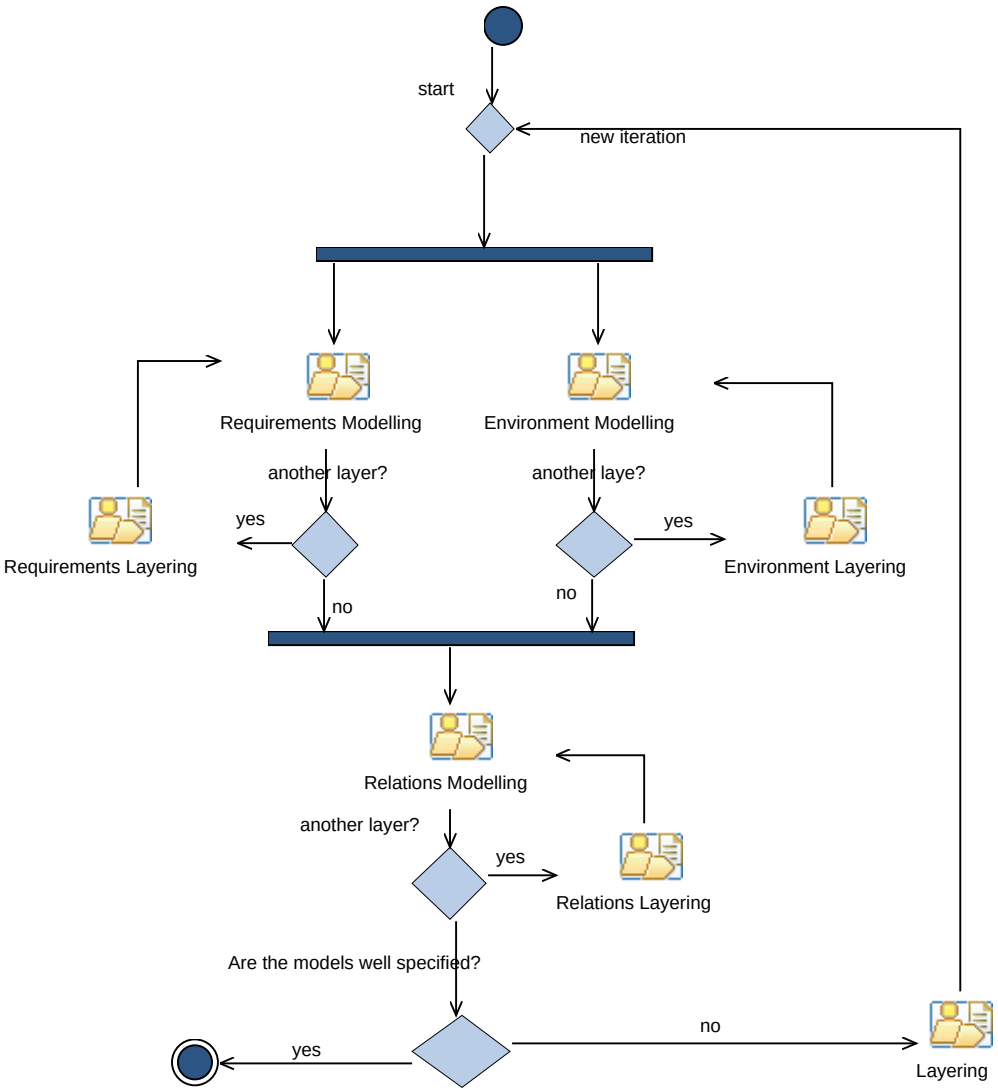


Figure 2.1: The Requirements Analysis process

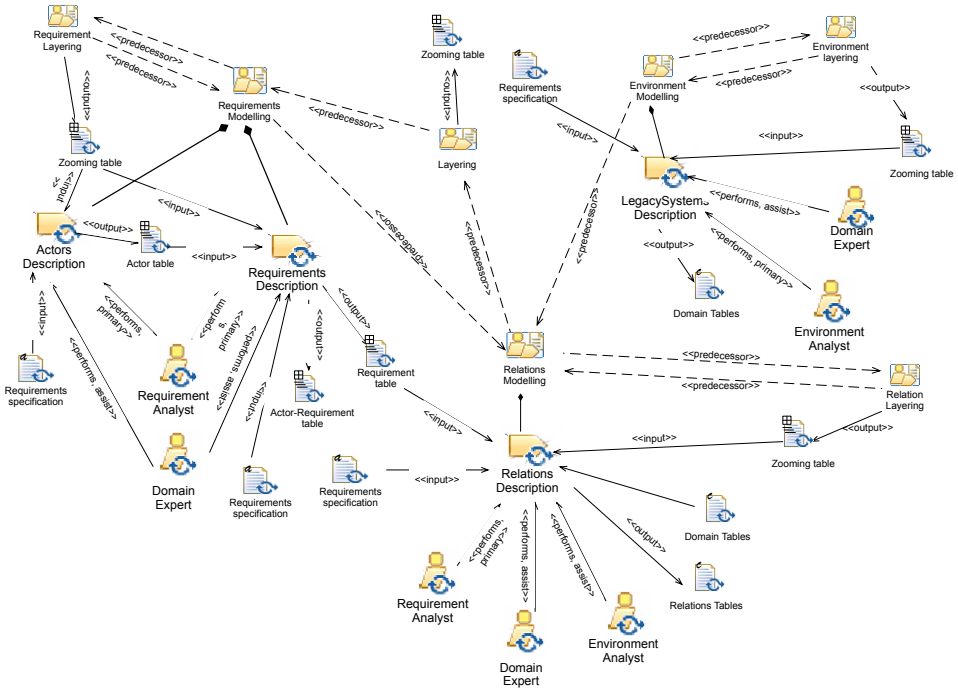


Figure 2.2: The Requirements Analysis flow of activities, roles, and work products

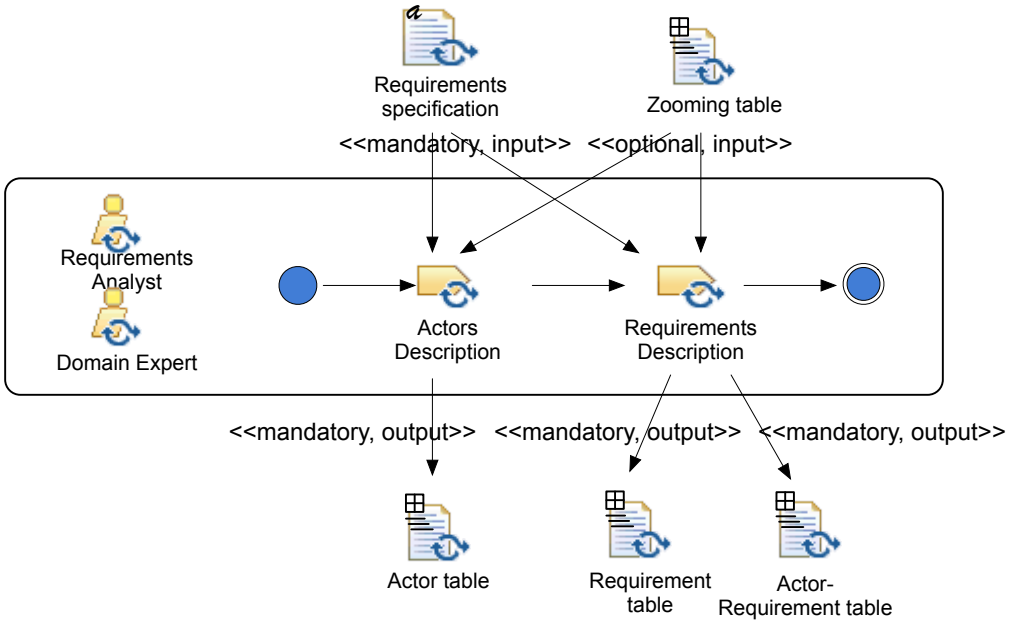


Figure 2.3: The flow of tasks in the Requirements Modelling activity

2.1.2 Activity Details

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

Requirements Modelling activity

The Requirements Modelling activity is composed by 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 Figure 2.3.

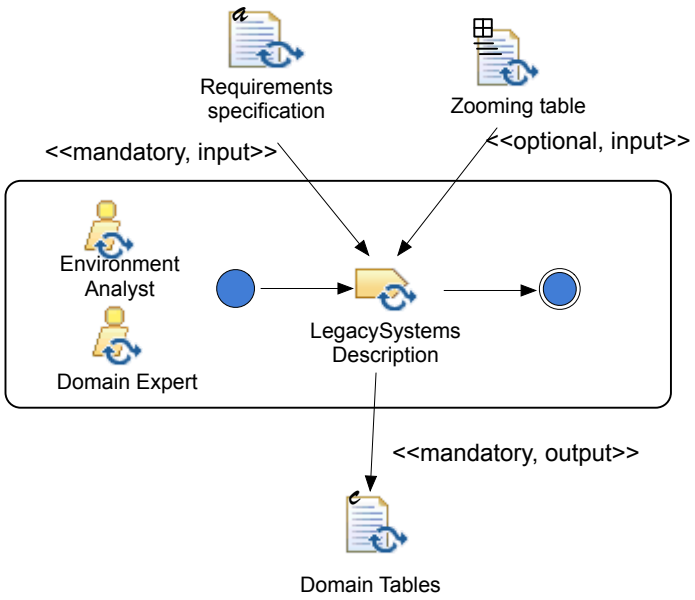


Figure 2.4: The flow of tasks in the Environment Modelling activity

Environment Modelling activity

The Environment Modelling activity is composed by 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 Figure 2.4.

Relations Modelling activity

The Relations Modelling activity is composed by the following tasks:

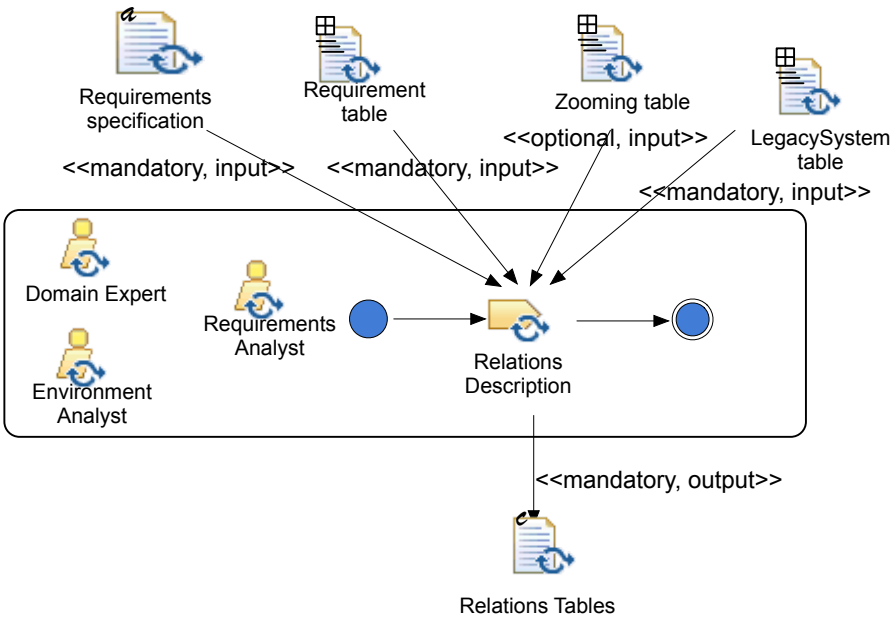


Figure 2.5: The flow of tasks in the Relations Modelling activity

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 Figure 2.5.

2.1.3 Work products

The Requirements Analysis step consists of three sets of tables: Requirements Tables, Domain Tables, and Relations Tables. Figure 2.6 reports the relationships among the work products of this step and the MMMElements of the Requirements Analysis. In Figure 2.6 the relationships among the Zooming table and the MMMElements of the Requirements Analysis are also reported—see Subsection 1.3.1 for details.

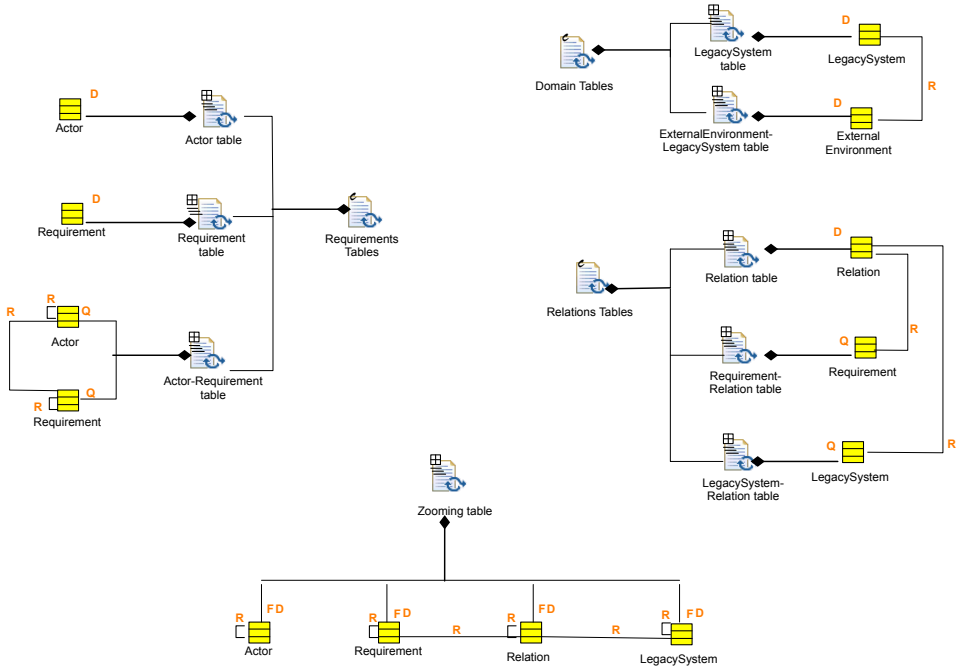


Figure 2.6: The Requirement Analysis work products

Kinds of work products

Table 2.1 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.

Table 2.1: 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

Example of work products

Figure 2.7 depicts an example of the structure of the Requirements Tables:

- *Actor table* $(L)Ac_t$ lists all the actors and describes them
- *Requirement table* ($(L)Re_t$) lists all the requirements and describes them
- *Actor-Requirement table* ($(L)AR_t$) specifies the list of the requirements for each actor

Actor	Description
actor name	*actor description*

Requirement	Description
requirement name	*requirement description*

Actor	Requirement
actor name	*requirement names*

Figure 2.7: Requirements Tables, in top-down order: $(L)Ac_t$, $(L)Re_t$, $(L)AR_t$

Figure 2.8, Figure 2.9, Figure 2.10 provide an example of the Requirements Tables for the conference management case study [2].

Actor	**Description**
Organisers	*a team of people that organise a conference*
Chair	*person responsible for the managing of the scientific papers*
Author	*a person that writes one or more paper*
PCMember	*a person member of program committee, he/she can do reviews or he/she can assign his/her reviews to some other reviewers*
Reviewer	*a person making a review*

Figure 2.8: Actor table $(C)Ac_t$

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*

Figure 2.9: Requirement table $(C)Re_t$

Figure 2.11 depicts an example of the structure of the Domain Tables:

- *External Environment-Legacy System table ($(L)EELS_t$)* specifies the list of the legacy systems for external environment.
- *Legacy System table ($(L)LS_t$)* lists all the legacy systems and describe them.

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 2.12 presents the External Environment-Legacy System table and Figure 2.13 presents Legacy System Table where the WebServer system is described. Figure 2.14 depicts an example of the structure of the Relations Tables:

Actor	**Requirement**
Organisers	ManageStartUp
Chair	ManageSubmission, ManagePartitioning, ManageAssignment, ManageReview
Author	ManageSubmission
PCMember	ManageReviewers, ManageReview
Reviewer	ManageReviewers,

Figure 2.10: Actor-Requirement table $(C)AR_t$

External Environment	Legacy System
external environment name	*legacy system names*

Legacy System	Description
legacy system name	*legacy system description*

Figure 2.11: Domain Tables, in top-down order: $(L)EELS_t$, $(L)LS_t$

- *Relation table ($(L)Rel_t$)* lists all the relationship among abstract entities and provides a description to them.
- *Requirement-Relation table ($(L)RR_t$)* specifies the list of relations where requirement is involved.
- *Legacy System-Relation table ($(L)LSR_t$)* specifies the list of relations where the legacy system is involved.

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. The Relations Tables are illustrated in Figure 2.15, Figure 2.16, and Figure 2.17. In the following figures we report the SODA tables modelling the conference management systems at layer $C + 1$.

External Environment	**Legacy System**
External	WebServer

Figure 2.12: External Environment-Legacy System table $(C)EELS_t$

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

Figure 2.13: Legacy System table $(C)LS_t$

Relation	Description
relation name	*relation description*

Requirement	Relation
requirement name	*relation names*

Legacy System	Relation
legacy system name	*relation names*

Figure 2.14: Relations Tables, in top-down order: $(L)Rel_t$, $(L)RR_t$, $(L)LSR_t$.

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

Figure 2.15: Relation table $(C)Rel_t$

Requirement	**Relation**
ManageStartUp	Web
ManageSubmission	Web
ManagePartitioning	Web
ManageAssignment	Web
ManageReview	Web

Figure 2.16: Relation-Requirement table $(C)RR_t$

Legacy-System	**Relation**
WebServer	Web

Figure 2.17: Relation LegacySystem table $(C)RLS_t$

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

Figure 2.18: Zooming table ($(C)Z_t$): Paper Partitioning in-zoom

Actor	**Description**
Organisers	*a team of people that organise a conference*
Vice-Chair	*person responsible for the managing a set of scientific papers relating to a specific topic*

Figure 2.19: Actor table $(C+1)Ac_t$

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*

Figure 2.20: Requirement table $(C+1)Re_t$

Actor	**Requirement**
+Organisers	UpdateStartUp
+Chair	ManageSubCommittee, ManageClassification, PartitionPapers
Vice-Chiar	ManageClassification, PartitionPapers, +ManageSubmission, +ManageAssignment, +ManageReview

Figure 2.21: Actor-Requirement table $(C+1)AR_t$

Relation	**Description**
UpSubCooRel	*if necessary, the organisation structure and the conference rules have to be changed before the creation of the sub-committes*
SubCommPartRel	*if necessary, the sub-committes have to be created before starting the paper assignment*
ClassPartRel	*paper classification should be done before starting the paper assignment*

Figure 2.22: Relation table $(C + 1)Rel_t$

Requirement	**Relation**
UpdateStartUp	+Web, UpSubCooRel
ManageSub- Committee	+Web, UpSubCooRel, SubCommPartRel
ManageClassi- fication	+Web, ClassPartRel
PartitionPapers	+Web, SubCommPartRel, ClassPartRel

Figure 2.23: Relation-Requirement table $(C + 1)RR_t$

2.2 The Analysis

In the Analysis step, SODA takes into account four abstract entities to analyse the system: tasks, functions, dependencies, and topologies:

Task — is an activity that requires one or more competences and the use of functions.

Function — is an reactive activity that aimed at supporting tasks.

Dependency — is any relationship (interactions, constraints...) among other (tasks and/or functions) abstract entities.

Topology — is the expression of any topological need of the environment's structure, often derived from functions. It is important to note that topology could influence tasks, given that topology may affect the achievement of tasks.

Figure 2.24 presents the Analysis process, while Figure 2.25 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
- identification of new tasks coming from system analysis and description of the all tasks (new tasks and tasks coming from the mapping)
- identification of new functions coming from system analysis and description of the all functions (new tasks and tasks coming from the mapping)
- identification of new dependencies coming from system analysis and description of the all dependencies (new dependencies and dependencies coming from the mapping)
- identification of new topologies coming from system analysis and description of the all topologies (new topologies and topologies coming from the mapping)

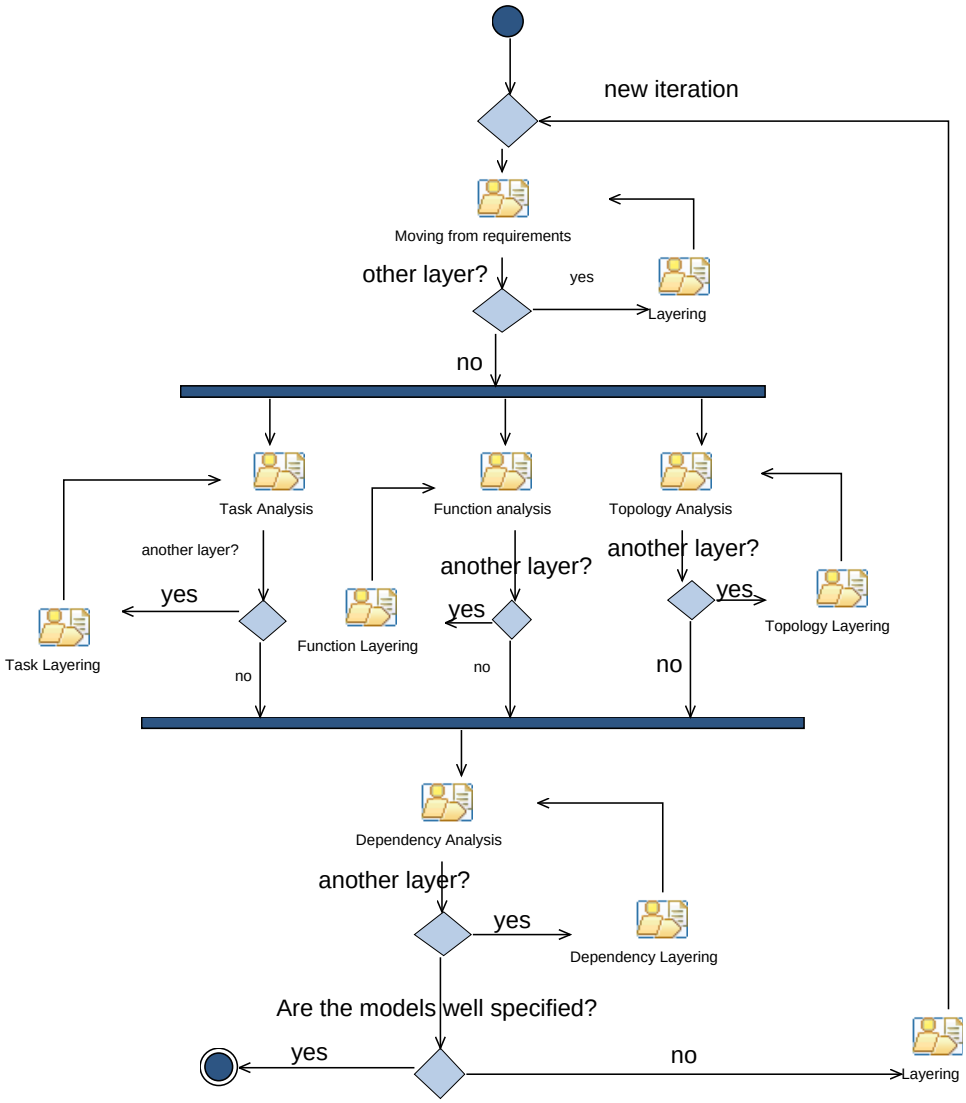


Figure 2.24: The Requirements Analysis process

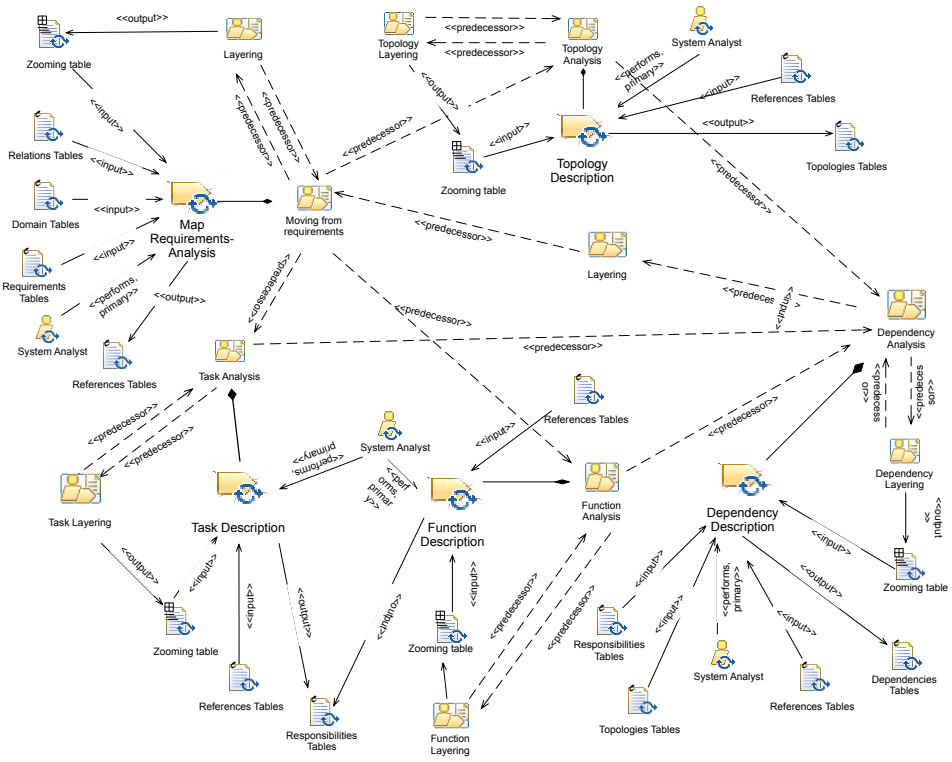


Figure 2.25: The Analysis flow of activities, roles and work products

2.2.2 Activity Details

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

Moving from Requirements activity

The Moving from Requirements activity is composed by 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 Figure 2.26.

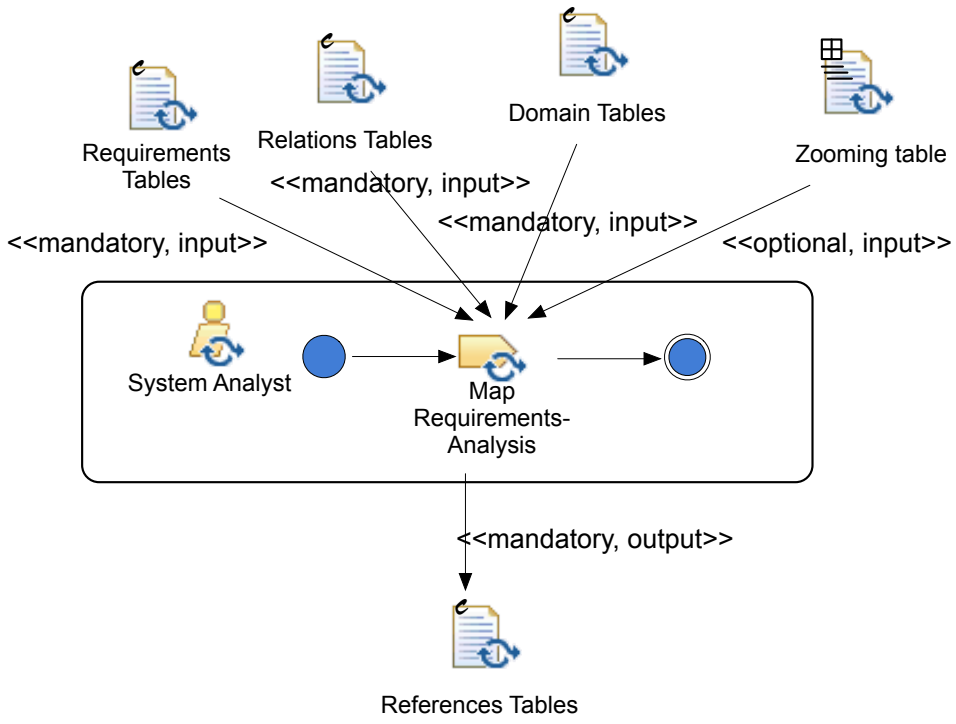


Figure 2.26: The flow of tasks in the Moving from requirements activity

Task Analysis activity

The Task Analysis activity is composed by the following tasks:

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

The flow of tasks inside the Task Analysis activity is reported in Figure 2.27.

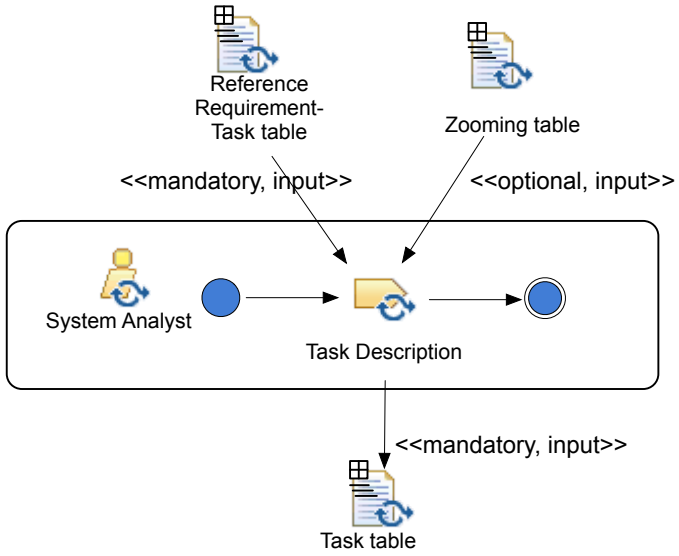


Figure 2.27: The flow of tasks in the Task Analysis activity

Function Analysis activity

The flow of tasks inside this activity is reported in Figure 2.28 and the tasks are detailed in the following table.

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 Figure 2.29 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)

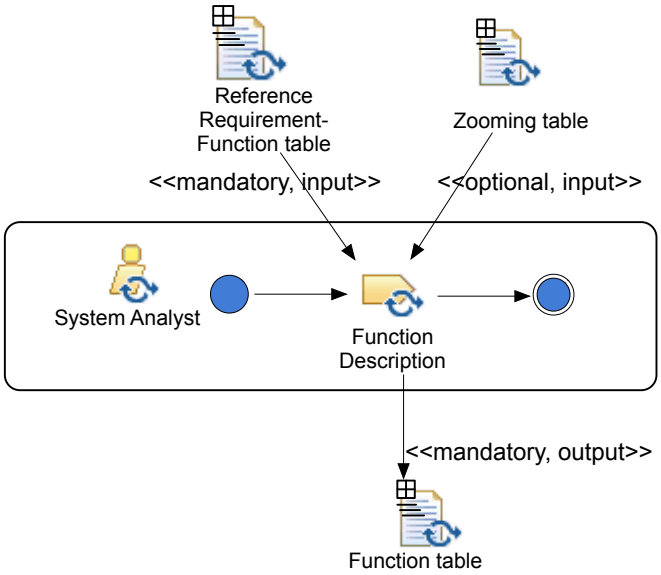


Figure 2.28: The flow of tasks in the Function Analysis activity

Topology Analysis activity

The flow of tasks inside this activity is reported in Figure 2.30 and 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)

2.2.3 Work products

The Analysis step exploits four sets of tables: Reference Tables, Responsibilities Tables, Dependencies Tables and Topologies Tables. Figure 2.31 reports the relationships among the work products of this step and the MMMElements of the Analysis. In Figure 2.31 are also reported the relationships among the Zooming table and the MMMElements of the Analysis—see Subsection 1.3.1 for details.

Kinds of work products

Table 2.2 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

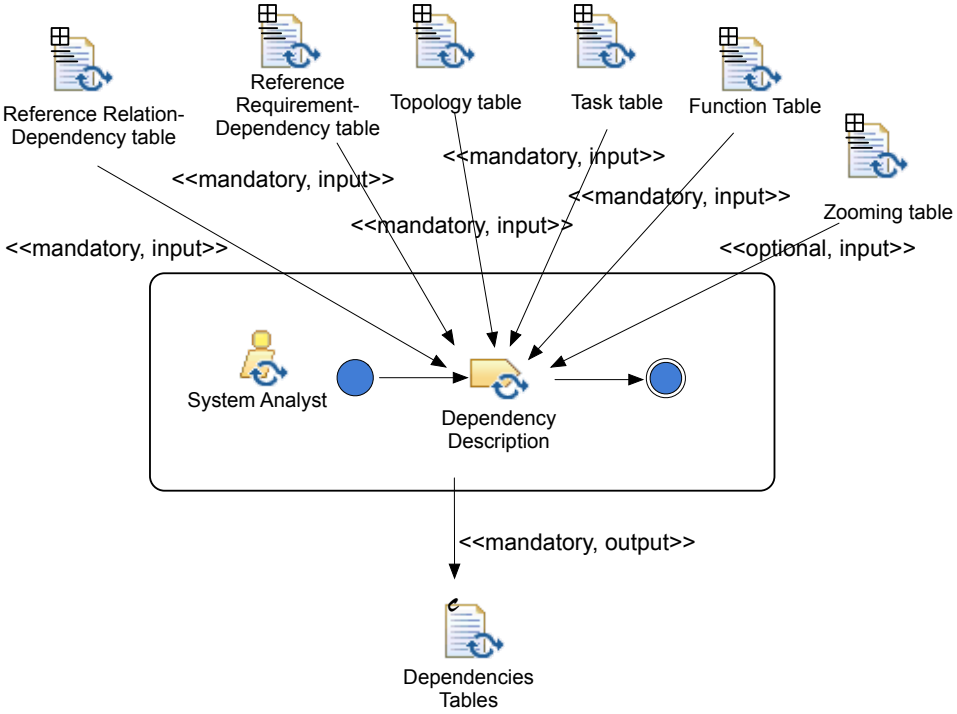


Figure 2.29: The flow of tasks in the Dependency Analysis activity

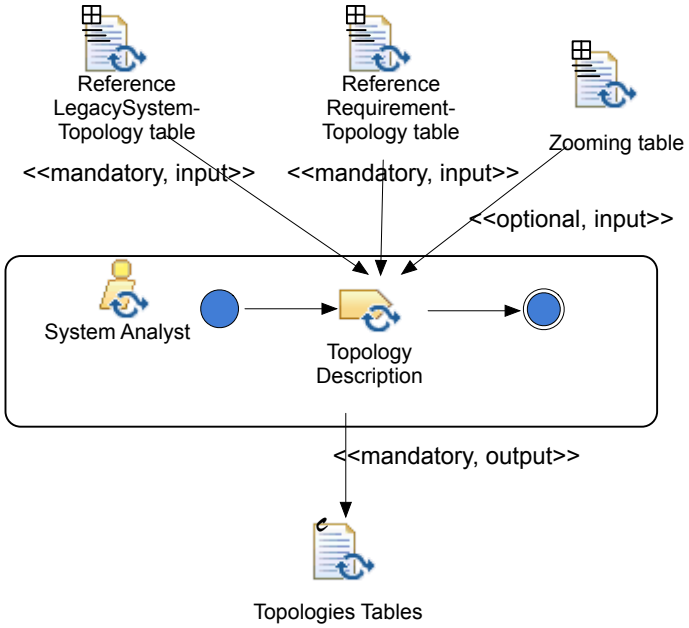


Figure 2.30: The flow of tasks in the Topology Analysis activity

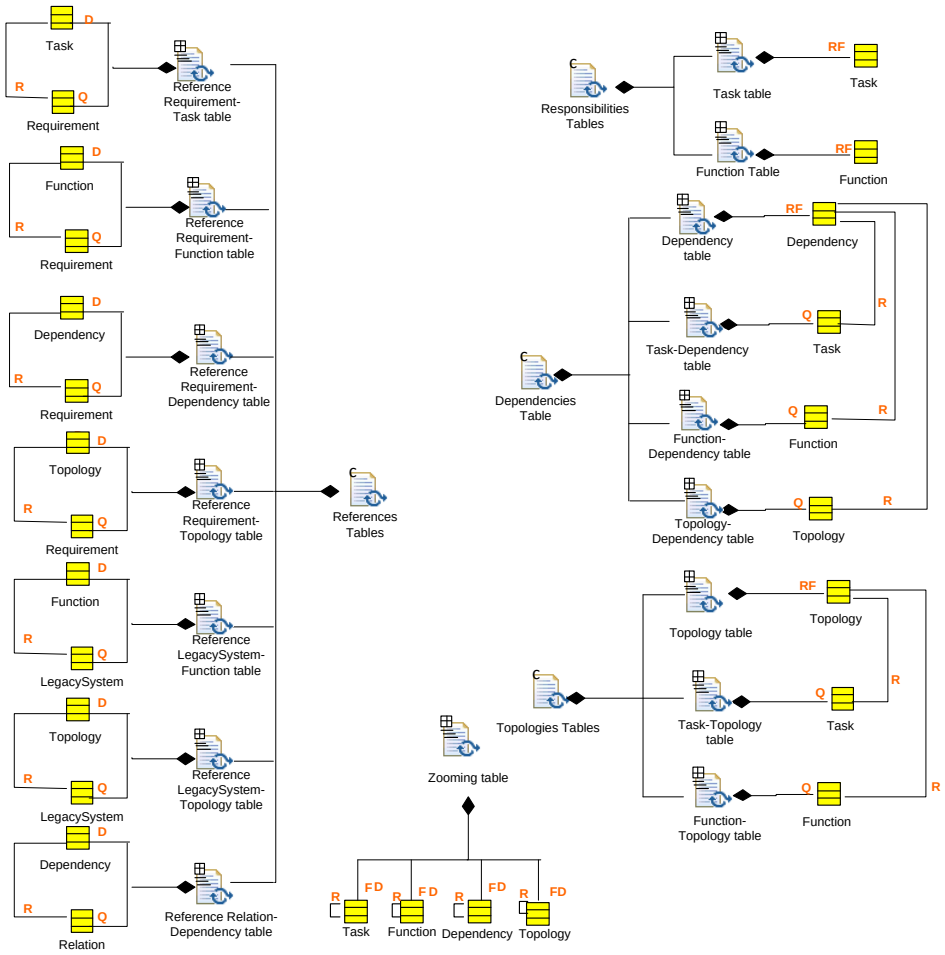


Figure 2.31: The Analysis work products

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

Table 2.2: 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

Example of work products

Figure 2.32 depicts an example of the structure of the References Tables:

- *Reference Requirement-Task table(*$(L)RRT_t$*)* specifies the mapping between requirements and tasks
- *Reference Requirement-Function table (*$(L)RRF_t$*)* specifies the mapping between requirements and resources
- *Reference Legacy System-Function table (*$(L)RLSF_t$*)* specifies the mapping between legacy systems and functions
- *Reference Legacy System-Topology table (*$(L)RLST_t$*)* specifies the mapping between legacy systems and topologies
- *Reference Relation-Dependency table (*$(L)RRD_t$*)* specifies the mapping between relations and dependencies

The following Figures represent the References Tables for the conference management case study.

Since our system is composed of two layers, a similar set of tables will be defined also for the $C+1$ layer. Figure 2.40 shows the Reference Requirement-Task table at layer $C+1$. Figure 2.42 depicts an example of the structure of the Responsibilities Tables:

- *Task table (*$(L)\,T_t$*)* lists all the tasks and describes them
- *Function table(*$(L)F_t$*)* lists all the functions and describes them

Requirement	Task
requirement name	*task names*

Requirement	Function
requirement name	*function names*

Requirement	Topology
requirement name	*topology names*

Requirement	Dependency
requirement name	*dependency names*

Legacy System	Function
legacy system name	*function names*

Legacy System	Topology
legacy system name	*topology names*

Relation	Dependency
relation name	*dependency names*

Figure 2.32: References Tables, in top-down order: $(L)RRT_t$, $(L)RRF_t$, $(L)RRTo_t$, $(L)RReqD_t$, $(L)RLSF_t$, $(L)RLST_t$, $(L)RRelD_t$

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

Figure 2.33: Reference Requirement-Task table $(C)RRT_t$

Requirement	**Function**
ManageStartUp	*management process* *management user*
ManageSubmission	*management user* *management paper*
ManageReviewers	*management reviewer* *management paper*
ManagePartitioning	*management partitioning* *management paper*
ManageAssignment	*management assignment* *management paper*
ManageReview	*management review* *management paper*

Figure 2.34: Reference Requirement-Function table $(C)RRF_t$

Requirement	**Topology**

Figure 2.35: Reference Requirement-Topology table $(C)RRTo_t$

Requirement	**Dependency**
ManageSubmission	RegSubDep
ManageReviewers	RegAssDep
ManageAssignment	PartAssDep
ManageReview	AssRevDep

Figure 2.36: Reference Requirement-Dependency table $(C)RReqD_t$

Legacy System	**Function**
WebServer	webSite

Figure 2.37: Reference Legacy System-Function table $(C)RLSF_t$

Legacy System	**Topology**

Figure 2.38: Reference Legacy System-Topology table $(C)RLST_t$

Relation	**Dependency**
Web	WebAccessDep

Figure 2.39: Reference Relation-Dependency table $(C)RRD_t$

Requirement	**Task**
UpdateStartUp	*modifying startup*
ManageSubcommittee	*create sub-committees, Vice-Chair elections*
ManageClassification	*paper classification*
PartitionPapers	*partition papers*

Figure 2.40: Reference Requirement-Task table $(C + 1)RRT_t$

Relation	**Dependency**
UpSubCooRel	UpSubCooDep
SubCommPartRel	SubCommPartDep
ClassPartRel	ClassPartDep

Figure 2.41: Reference Relation-Dependency table $(C + 1)RRD_t$

The following Figures represent the Responsibilities Tables for the conference management case study. Figure 2.45 depicts an example of the structure of the Topologies Tables:

- *Topology table*$((L)\,Top_t)$ lists all the topological requirements and provides them with a description
- *Task-Topology table* $((L)\,TTop_t)$ specifies the list of the topological requirements influencing the task
- *Function-Topology table* $((L)FTop_t)$ specifies the list of the topological requirements affected by the function

The following figures represent the Topologies Tables for the conference management case study. Figure 2.49 depicts an example of the structure of the Dependencies Tables:

- *Dependency table*$((L)D_t)$ lists all the dependency among abstract entities and provides them with a description
- *Task-Dependency table*$((L)\,TD_t)$ specifies the list of the dependencies where the task is involved

Task	Description
task name	*task description*

Function	Description
function name	*function description*

Figure 2.42: Responsibilities Tables, in top-down order: $(L)\,T_t$, $(L)F_t$

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*

Figure 2.43: Task table $(C)\,T_t$

Function	**Description**
management user	*managing and storing user information*
management reviewer	*managing and storing reviewer information*
management review	*managing and storing review information*
management paper	*managing and storing paper information*
management assignment	*managing and storing assignment information*
management partitioning	*managing and storing partitioning information*
management process	*managing and storing start-up information*
webSite	*web interface of the conference*

Figure 2.44: Function table $(C)F_t$

Topology	Description
Topology name	*topology description*

Task	Topology
task name	*topology names*

Function	Topology
function name	*topology names*

Figure 2.45: Topologies Tables in top-down order: $(L)\,Top_t$, $(L)\,TTop_t$, $(L)FTop_t$

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

Figure 2.46: Topology table $(C)\,Top_t$

Task	**Topology**
start up	place
submission	place
user registration	place
reviewer registration	place
paper partitioning	place
assignment papers	place
review process	place

Figure 2.47: Task-Topology table $(C)\,TTop_t$

Function	**Topology**
management user	place
management reviewer	place
management review	place
management paper	place
management assignment	place
management partitioning	place
management process	place
webSite	place

Figure 2.48: Function-Topology table $(C)FTop_t$

Dependency	Description
dependency name	*dependency description*

Task	Dependency
task name	*dependency names*

Function	Dependency
function name	*dependency names*

Topology	Dependency
topology name	*dependency names*

Figure 2.49: Dependencies Tables in top-down order $(L)D_t$, $(L)\,TD_t$, $(L)FD_t$ and $(L)\,TopD_t$

- *Function-Dependency table ($(L)FD_t$)* specifies the list of the dependencies where the function is involved

The following figures represent the Dependencies Tables for the conference management case study.

In the following figures we report 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*

Figure 2.50: Dependency table $(C)D_t$

Task	**Dependency**
start up	WebAccessDep, StartUpInfDep
submission	WebAccessDep, RegSubDep, UserInfDep, PaperInfDep, SubInfDep
user registration	WebAccessDep, RegSubDep, UserInfDep,
reviewer registration	WebAccessDep, RegAssDep, ReviewerInfDep
paper partitioning	WebAccessDep, PartAssDep, RegAssDep, RegSubDep, PaperInfDep, PartInfDep
assignment papers	WebAccessDep, PartAssDep, AssRevDep, UserInfDep, ReviewerInfDep, PaperInfDep, AssInfDep
review process	WebAccessDep, AssRevDep, PaperInfDep, ReviewInfDep

Figure 2.51: Task-Dependency table $(C)TD_t$

Function	**Dependency**
management user	WebAccessDep, UserInfDep
management reviewer	WebAccessDep, ReviewerInfDep
management review	WebAccessDep, ReviewInfDep
management paper	WebAccessDep, PaperInfDep, SubInfDep
management assignment	WebAccessDep, AssInfDep
management partitioning	WebAccessDep, PartInfDep
management process	WebAccessDep, StartUpInfDep
webSite	WebAccessDep, StartUpInfDep

Figure 2.52: Function-Dependency table $(C)FD_t$

Topology	**Dependency**
place	WebAccessDep

Figure 2.53: Topology-Dependency table $(C)\,TopD_t$

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

Figure 2.54: 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*

Figure 2.55: Task table $(C+1)\,T_t$

Dependency	**Description**
UpSubCooDep	*start up before creating sub-committees*
SubCommPartDep	*sub-committees have to be created before paper assignment*
ClassPartDep	*paper classification have to be done before paper assignment*
NewOrganisationDep	*modifications to the conference information*
ElectionDep	*starting the election process for vice-chairs*

Figure 2.56: Dependency table $(C+1)D_t$

Task	**Dependency**
modifying startup	+WebAccessDep, +StartUpInfDep, UpSub-CooDep
create sub-committees	+WebAccessDep, +UserInfDep, SubComm-PartDep
Vice-Chair elections	+WebAccessDep, +UserInfDep,ElectionDep
paper classification	+WebAccessDep, +PaperInfDep
partition papers	+WebAccessDep, +PartAssDep, +RegAssDep, +RegSubDep, +PaperInfDep, +PartInfDep, ClassPartDep

Figure 2.57: Task-Dependency table $(C + 1)\,TD_t$

Task	**Topology**
modifying startup	+place
create sub-committees	+place
Vice-Chair elections	+place
papers classification	+place
partition papers	+place

Figure 2.58: Task-Topology table $(C + 1)\,TTop_t$

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.

Role — is the abstraction responsible for the achievement of one or more tasks

Resource — is the abstraction that provides some functions

Action — represents an action that a role is potentially able to do

Operation — represents the operation that a resource is potentially able to provide

Interaction — represents the acts of the interaction among roles, among resources, and between roles and resources

Rule — enables and bounds the entities' behaviour

Space — is a conceptual locus in the environment

Figure 2.59 presents the Architectural Design process, while Figure 2.60 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;
- 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);

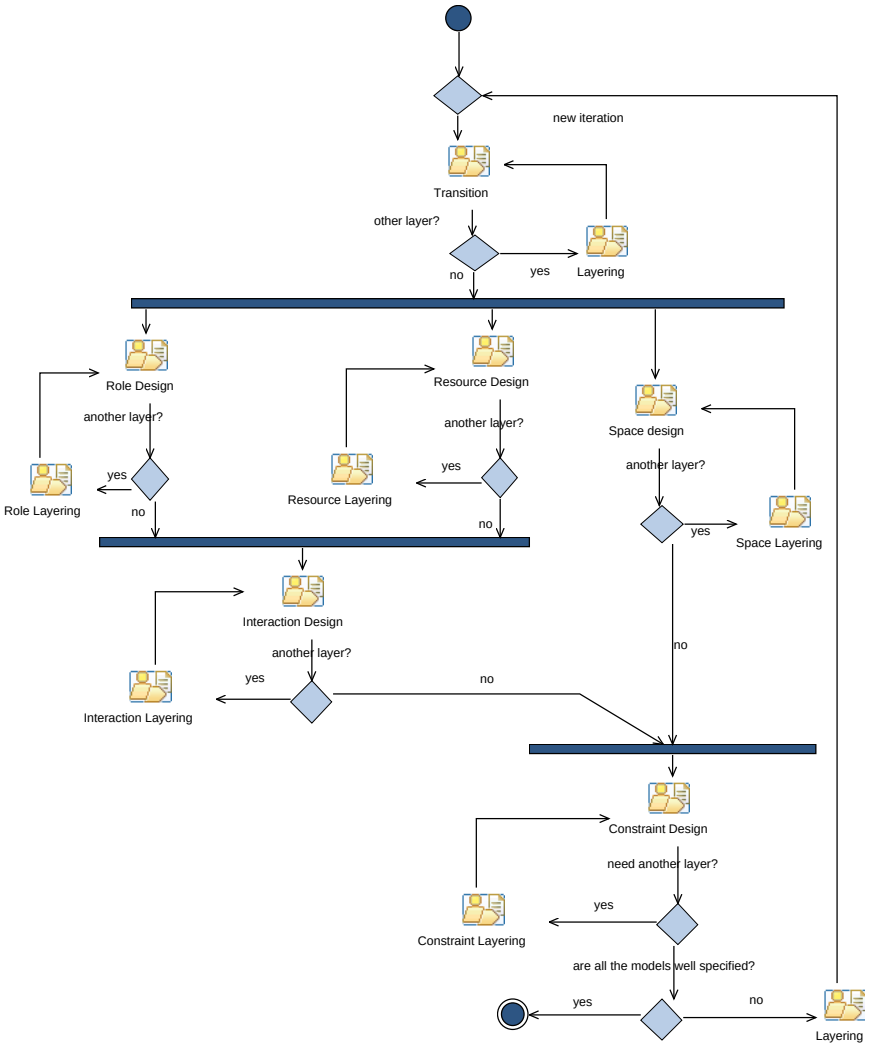


Figure 2.59: The Architectural Design process

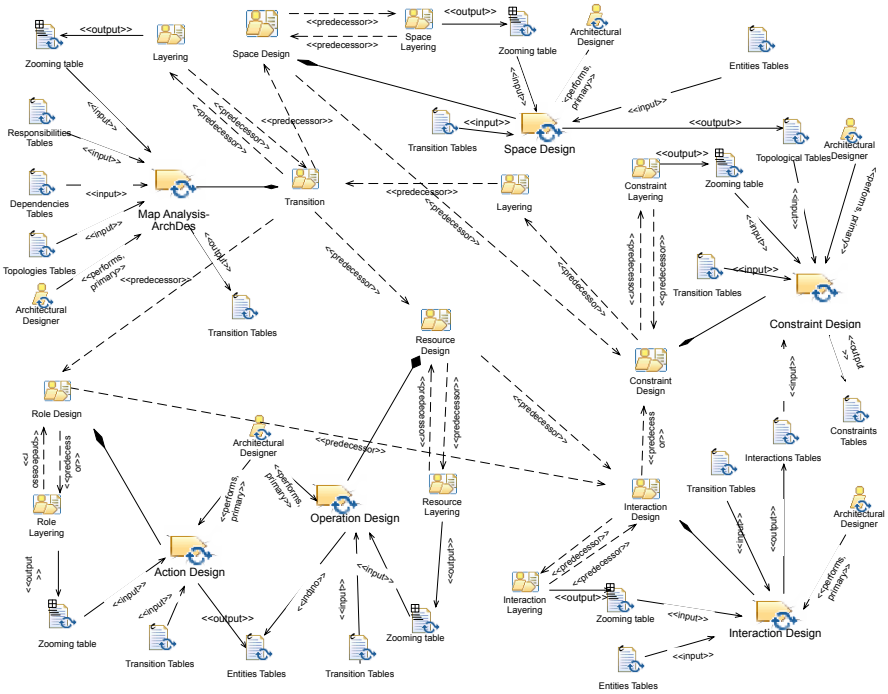


Figure 2.60: The Architectural Design flow of activities, roles, and work products

- identifying new interactions coming from system design, and describing all the interactions (new interactions and interactions coming from the mapping);
- 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 Subsection 1.3.1.

Transition activity

The Transition activity is composed by 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 Figure 2.61.

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 Figure 2.62.

Resource Design activity

The Resource Design activity is composed by the following tasks:

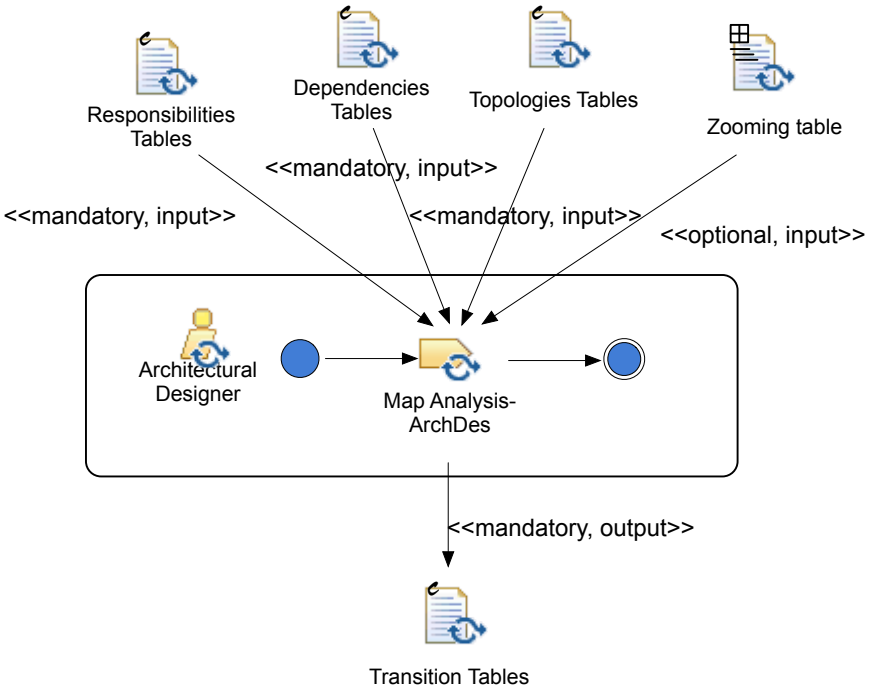


Figure 2.61: The flow of tasks in the Transition activity

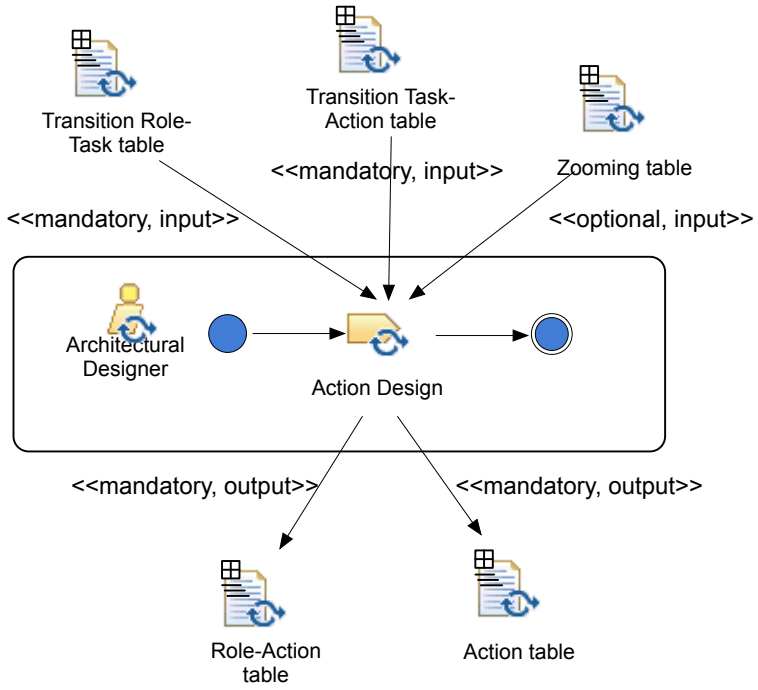


Figure 2.62: The flow of tasks in the Role Design activity

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)*

The flow of tasks inside the Resource Design activity is reported in Figure 2.63.

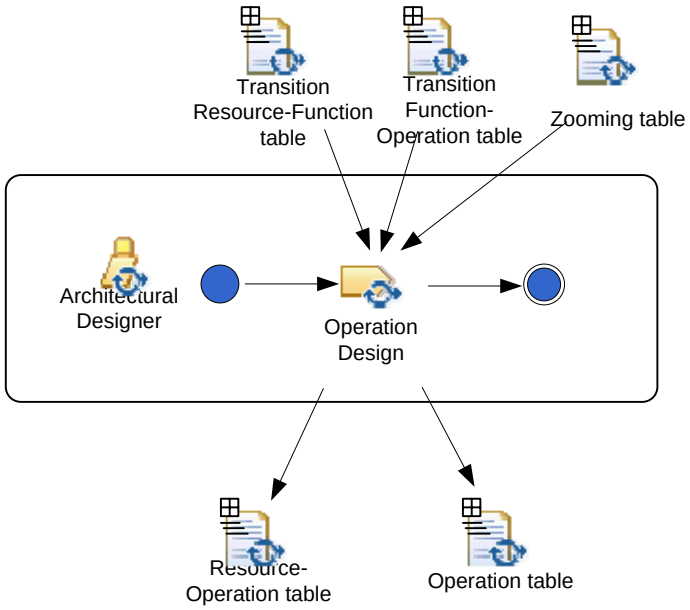


Figure 2.63: The flow of tasks in the Resource Design activity

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 Figure 2.64.

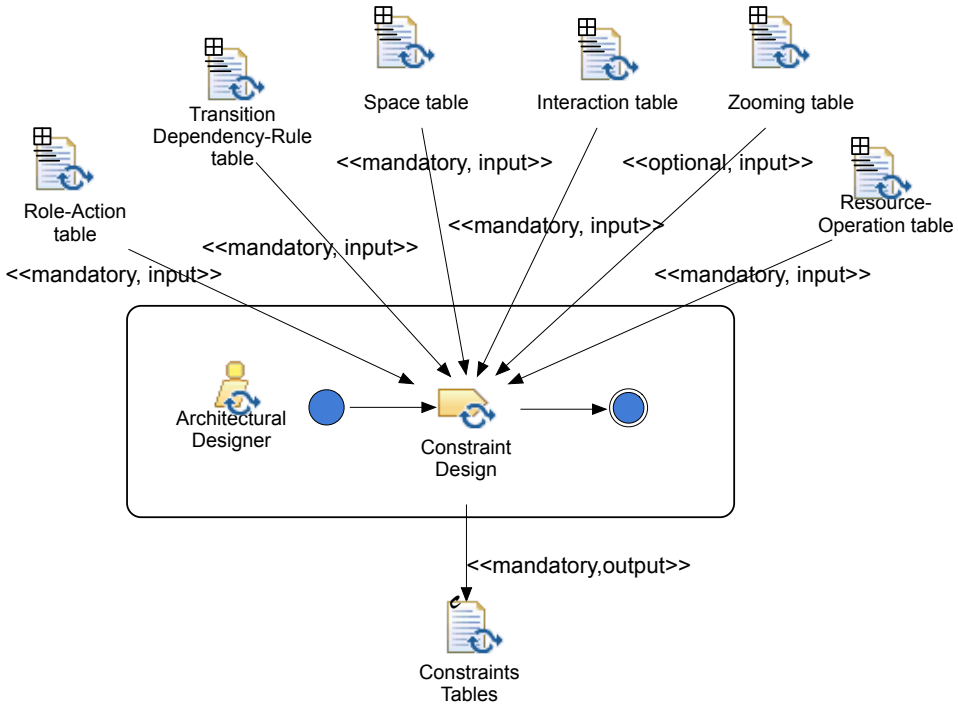


Figure 2.64: The flow of tasks in the Constraint Design activity

Interaction Design activity

The Interaction Design activity is composed by the following tasks:

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 Figure 2.65.

Space Design activity

The Space Design activity is composed by the following tasks:

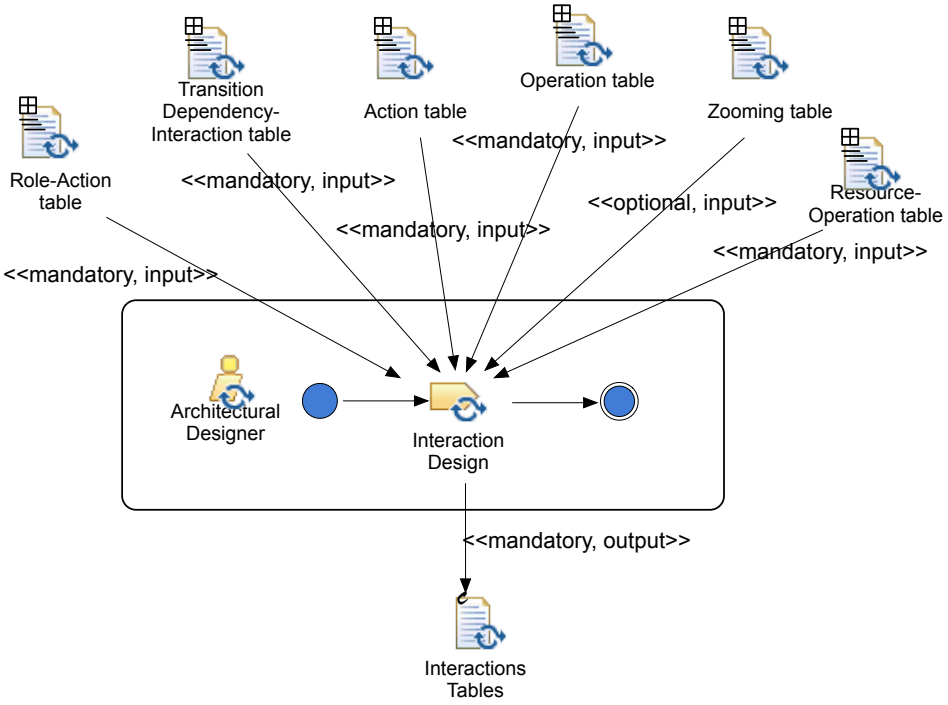


Figure 2.65: The flow of tasks in the Interaction Design activity

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 Figure 2.66.

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 2.67 reports the relationships among the work products of this step and the MMMElements of the Architectural Design step. In Figure 2.67 are also reported the relationships among the Zooming table and the MMMElements of the Architectural Design—see Subsection 1.3.1 for details.

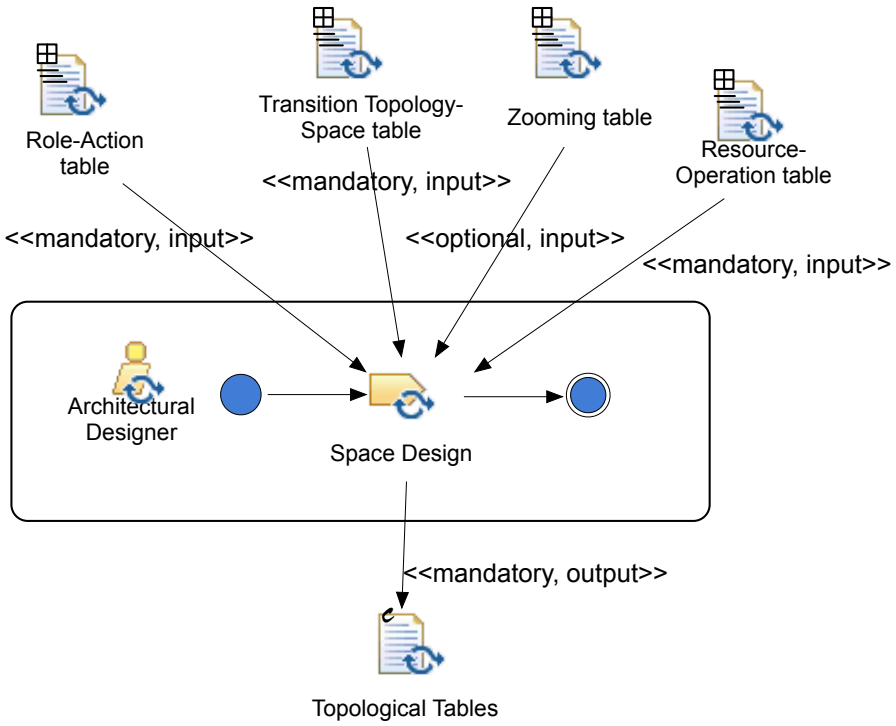


Figure 2.66: The flow of tasks in the Space Design activity

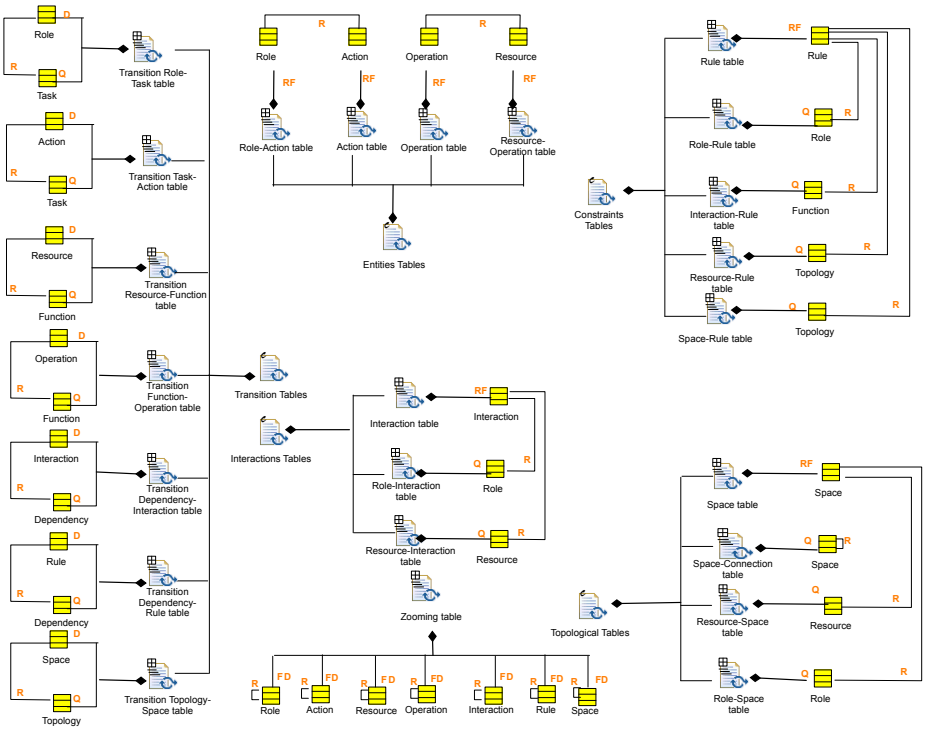


Figure 2.67: The Architectural Design work products

Kinds of work products

Table 2.3 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.

Table 2.3: 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 ($(L)TDI_t$)	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 action 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
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

Example of work products

Figure 2.68 depicts an example of the structure of the Transition Tables:

- *Transition Role-Task table*($(L)\,TRT_t$) specifies the mapping between tasks and roles
- *Transition Task-Action table*($(L)\,TTA_t$) specifies the mapping between tasks and actions
- *Transition Resource-Function table* ($(L)\,TRF_t$) specifies the mapping between functions and resources
- *Transition Function-Operation table* ($(L)\,TFO_t$) specifies the mapping between functions and operations
- *Transition Dependency-Interaction* ($(L)\,TDI_t$) specifies the mapping between dependencies and interactions

Role	Task
role name	*task names*

Task	Action
task name	*action names*

Resource	Function
resource name	*function names*

Function	Operation
function name	*operation names*

Dependency	Interaction
dependency name	*interaction names*

Dependency	Rule
dependency name	*rule names*

Topology	Space
topology name	*space names*

Figure 2.68: Transition Tables, in top-down order: $(L)\,TRT_t$, $(L)\,TTA_t$, $(L)\,TRF_t$, $(L)\,TFO_t$, $(L)\,TDI_t$, $(L)DRu_t$, $(L)\,TTopS_t$

- *Transition Dependency-Rule table ($(L)\,TDI_t$)* specifies the mapping between dependencies and rules
- *Transition Topology-Space table ($(L)\,TTopS_t$)* specifies the mapping between topologies and spaces

The following Figures present some examples of the Transition Tables for the conference management case study. Figure 2.74 depicts an example of the structure of the Entities Tables:

- *Action table($(L)A_t$)* specifies the actions that roles may be able to execute, and describes them

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

Figure 2.69: Transition Role-Task table $(C)\,TRT_t$

Resource	**Function**
People DB	management user, management reviewer
Paper DB	management paper, management review, management partitioning, management assignment
Process DB	management process
WebService	webSite

Figure 2.70: Transition Resource-Function table $(C)\,TRF_t$

Dependency	**Rule**
RegSubDep	RegSubRule
RegAssDep	RegAssRule
PartAssDep	PartAssRule
AssRevDep	AssRevRule
UserInfDep	UserInfRule
ReviewerInfDep	ReviewerInfRule
PaperInfDep	AuthorInfRule
PartInfDep	MatchRule
SubInfDep	SubInfRule
AssInfDep	AutRevRule, ReviewRule
ReviewInfDep	AuthorInfRule
WebAccessDep	WebAccessRule

Figure 2.71: Transition Dependency-Rule table $(C)\,TDRu_t$

Dependency	**Interaction**
UserInfDep	UserInfInteraction
ReviewerInfDep	ReviewerInfInteraction
PaperInfDep	AuthorInfInteraction
PartInfDep	PartInfInteraction
SubInfDep	SubInfInteraction
AssInfDep	AssInfInteraction
ReviewInfDep	ReviewInfInteraction
WebAccessDep	WebAccessInteraction

Figure 2.72: Transition Dependency-Interaction table $(C)\,TDI_t$

Topology	**Space**
place	S-place

Figure 2.73: Transition Topology-Space table $(C)\,TTTop_t$

Action	Description
action name	*description*

Operation	Description
operation name	*description*

Role	Action
role name	*action names*

Resource	Operation
resource name	*operation names*

Figure 2.74: Entities Tables, in top-down order: $(L)A_t$, $(L)O_t$, $(L)RA_t$, $(L)RO_t$

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*

Figure 2.75: Action table $(C)A_t$

- *Operation table ($(L)O_t$)* specifies the operations that resources may provide, and describes them
- *Role-Action table ($(L)RA_t$)* specifies the list of the actions that a specific role is able to do
- *Resource-Operation table ($(L)RO_t$)* specifies the list of the operations that a specific resource is able to provide

Figure 2.75 presents an example of the Entities Tables for the conference management case study. Figure 2.76 depicts an example of the structure of the Interactions Tables:

- *Interaction table($(L)I_t$)* specifies the interactions, and describes them
- *Action-Interaction table ($(L)AcI_t$)* specifies the list of the interactions where actions are involved
- *Operation-Interaction table ($(L)OpI_t$)* specifies the list of the interactions where operations are involved

Interaction	Description
interaction name	*description*

Action	Interaction
action name	*interaction names*

Operation	Interaction
operation name	*interaction names*

Figure 2.76: Interactions Tables, in top-down order: $(L)I_t$, $(L)AcI_t$, $(L)OpI_t$

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

Figure 2.77: Interaction table $(C)I_t$

Figure 2.77 presents an example of the Interaction Tables for the conference management case study. Figure 2.78 depicts an example of the structure of the Constraints Tables:

- *Rule table*$((L)I_t)$ specifies the rules, and describes them
- *Interaction-Rule table* $((L)IRu_t)$ specifies the list of the rules where interactions are involved
- *Resource-Rule table* $((L)ReRu_t)$ specifies the list of the rules where resources are involved
- *Role-Rule table* $((L)RoRu_t)$ specifies the list of the rules where roles are involved
- *Space-Rule table* $((L)SRu_t)$ specifies the list of the rules where spaces are involved

Figure 2.79 presents an example of the Constraints Tables for the conference management case study. Figure 2.80 depicts an example of the structure of the Topological Tables:

- *Space table*$((L)S_t)$ specifies the spaces, and describes them

Rule	Description
rule name	*description*

Interaction	Rule
interaction name	*rule names*

Resource	Rule
resource name	*rule names*

Role	Rule
role name	*rule names*

Space	Rule
space name	*rule names*

Figure 2.78: Constraints Tables, in top-down order: $(L)Ru_t$, $(L)IRu_t$, $(L)ReRu_t$, $(L)RoRu_t$, $(L)SRu_t$

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 and modify only his own information*
ReviewerInfRule	*reviewer can access and modify only his own information*
AuthorInfRule	*author can access and modify only the public information of his 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 his own papers*
WebAccessRule	*access to the system must be authorised*

Figure 2.79: Rule table $(C)Ru_t$

Space	Description
space name	*description*

Space	Connection
space name	*space names*

Resource	Space
resource name	*space names*

Role	Space
role name	*space names*

Figure 2.80: Topological Tables, in top-down order: $(L)S_t$, $(L)SC_t$, $(L)ReS_t$ and $(L)RoS_t$

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

Figure 2.81: Space table $(C)S_t$

- *Space-Connection table ($(L)SC_t$)* specifies the list of the spaces connected with the specific space
- *Resource-Space table ($(L)ReS_t$)* specifies the list of the spaces where resources are allocated
- *Role-Space table ($(L)RoS_t$)* specifies the list of the spaces that roles can perceive

Figure 2.81, Figure 2.82, Figure 2.83, Figure 2.84 present the Topological Tables for the conference management case study.

Space	**Resource**
S-place	People DB, Paper DB, Process DB, WebService

Figure 2.82: Resource-Space table $(C)ReS_t$

Space	**Connection**

Figure 2.83: Space-Connection table $(C)SC_t$

Role	**Space**
Conference Secretary	S-place
Chair	S-place
Author	S-place
Reviewer	S-place
PC-member	S-place

Figure 2.84: Role-Space table $(C)RoS_t$

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. In this phase we design the system in terms of agents, societies, artifacts, compositions, aggregates, workspaces, uses, linkedTo, manifests, and speakTo.

Agent — is an autonomous entity able to play several roles

Society — is a group of interacting agents and artifacts when its overall behaviour is essentially an *autonomous*, proactive one.

Artifact — is a group of interacting agents and artifacts when its overall behaviour is essentially a *functional*, reactive one.

Aggregate — is defined as the abstraction responsible for a collection of artifacts

Workspace — is a conceptual locus in the environment

Use — the act of interaction between agent and artifact: agent uses artifact

SpeakTo — the act of interaction among agents: agent speaks with another agent

Manifest — the act of interaction between artifact and agent: artifact manifests itself to agent

LinkedTo — the act of "interaction" among artifact: artifact is linked to another artifact

Figure 2.85 presents the Detailed Design process, while Figure 2.86 presents the flow of activities, the involved roles and the work products

2.4.1 Process roles

One role is involved in the Layering pattern: the Detailed Designer.

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

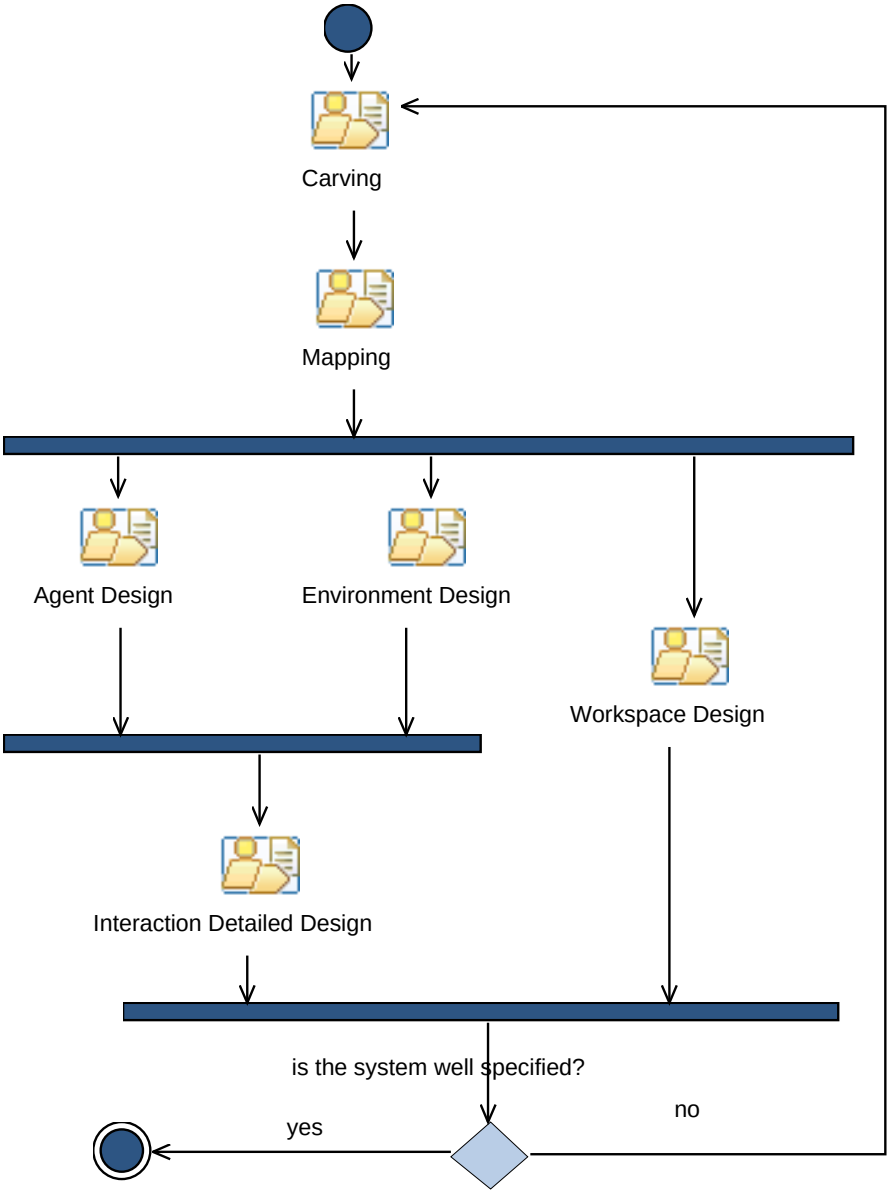


Figure 2.85: The Detailed Design process

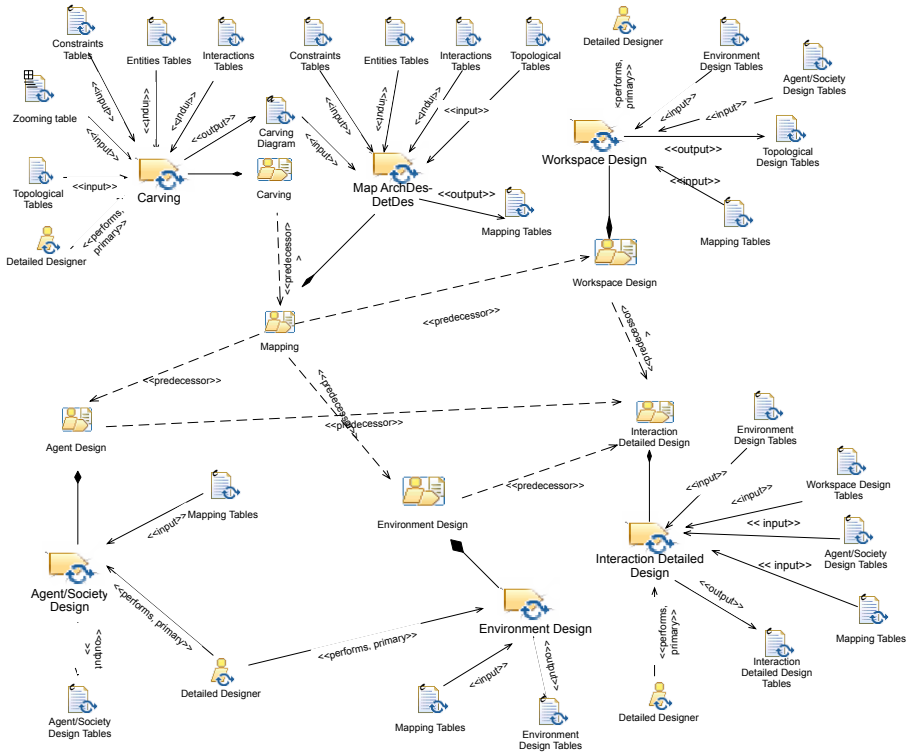


Figure 2.86: The Detailed Design flow of activities, roles and work products

- 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
- 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 by the following tasks:

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 Figure 2.87.

Mapping activity

The Mapping activity is composed by 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 Figure 2.88.

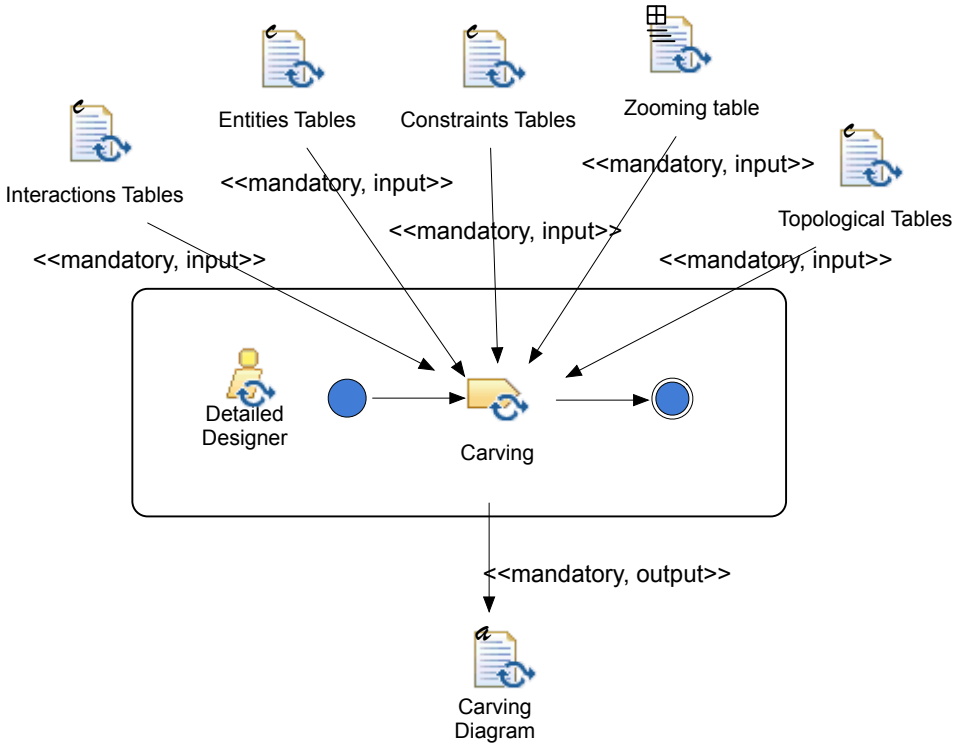


Figure 2.87: The flow of tasks in the Carving activity

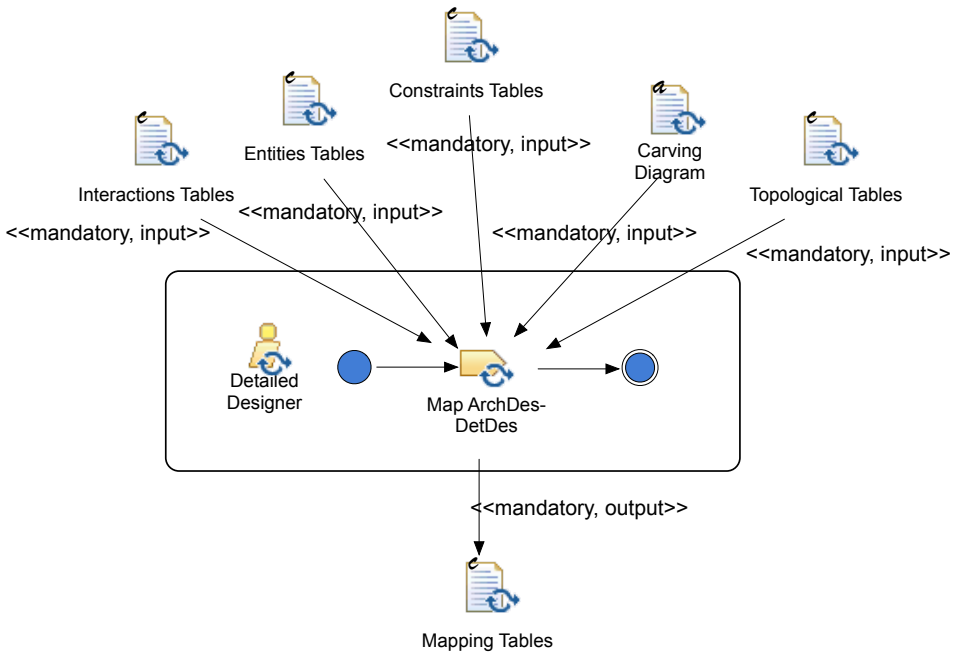


Figure 2.88: The flow of tasks in the Mapping activity

Agent Design activity

The Agent Design activity is composed by 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 Figure 2.89.

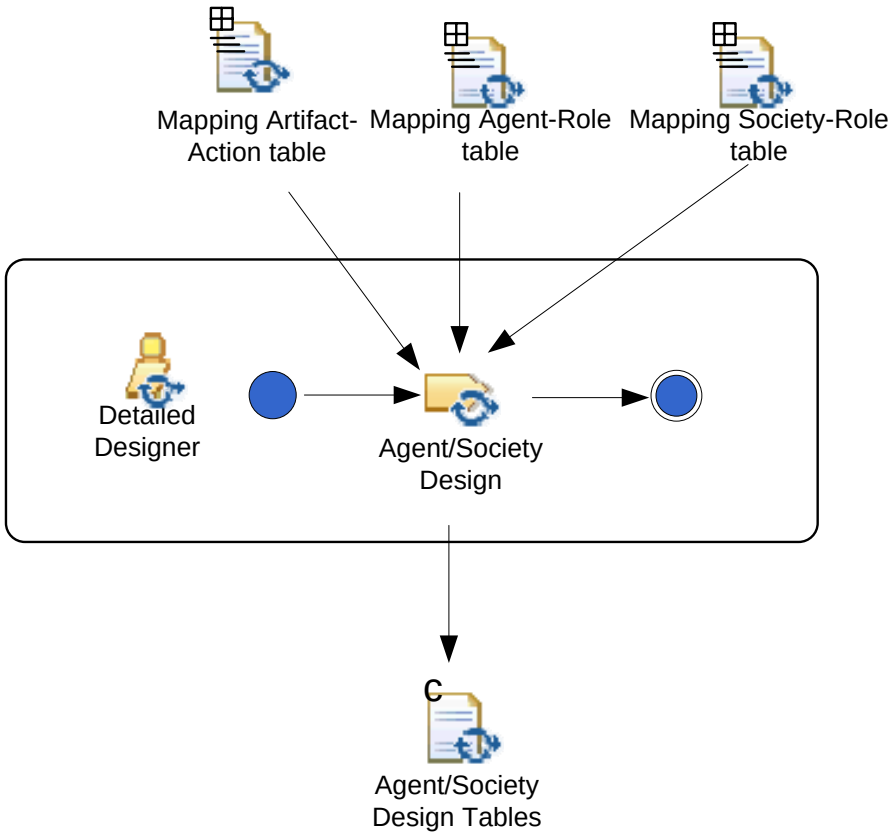


Figure 2.89: The flow of tasks in the Agent Design activity

Environment Design activity

The Environment Design activity is composed by 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 Figure 2.90.

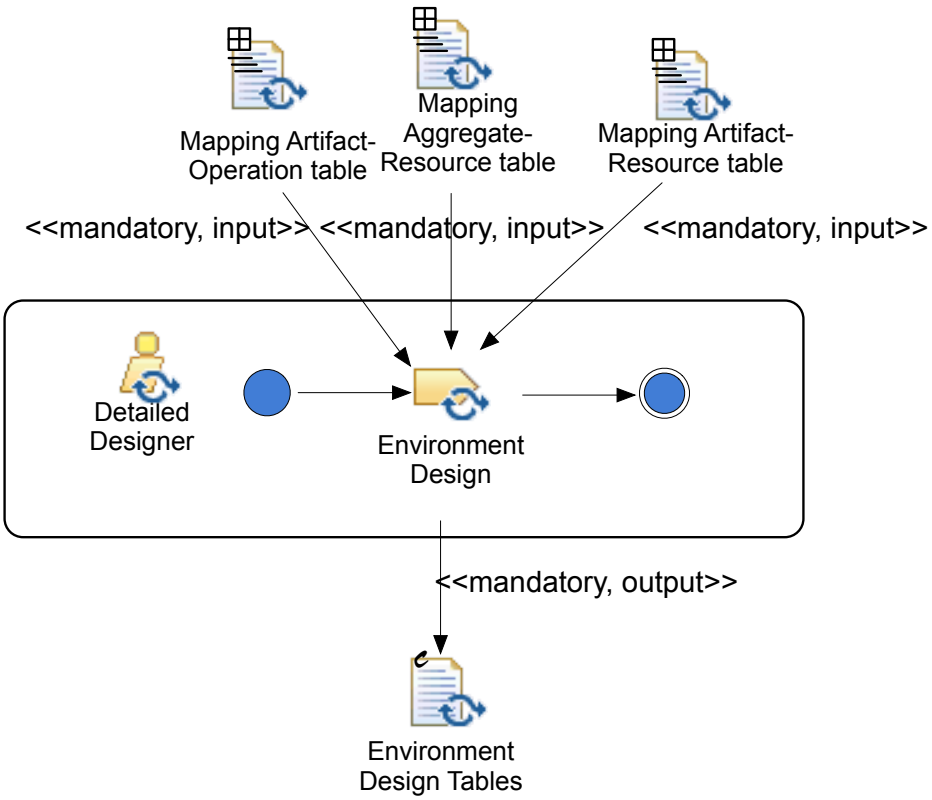


Figure 2.90: The flow of tasks in the Environment Design activity

Interaction Detailed Design activity

The Interaction Detailed Design activity is composed by 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 Figure 2.91.

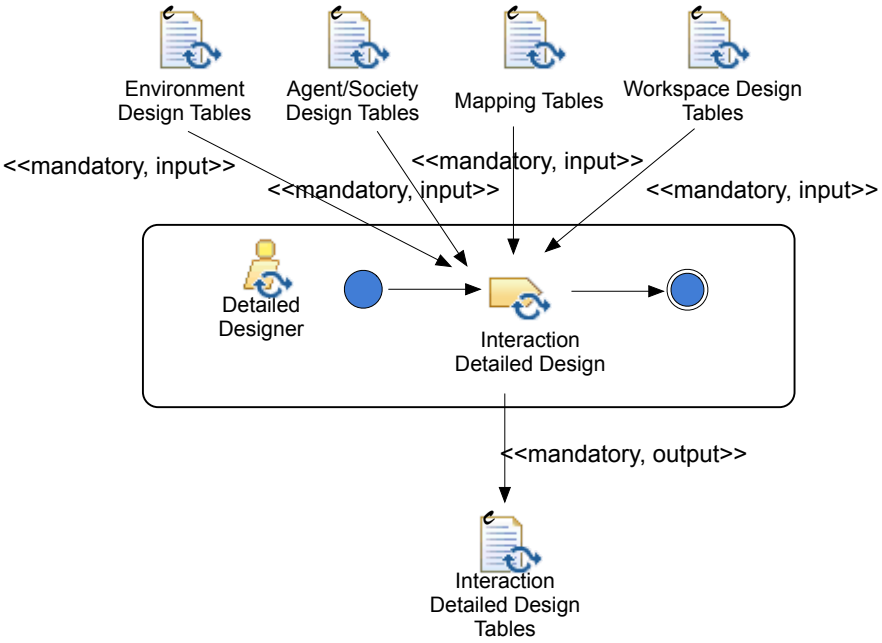


Figure 2.91: The flow of task in the Interaction Detailed Design activity

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 Figure 2.92.

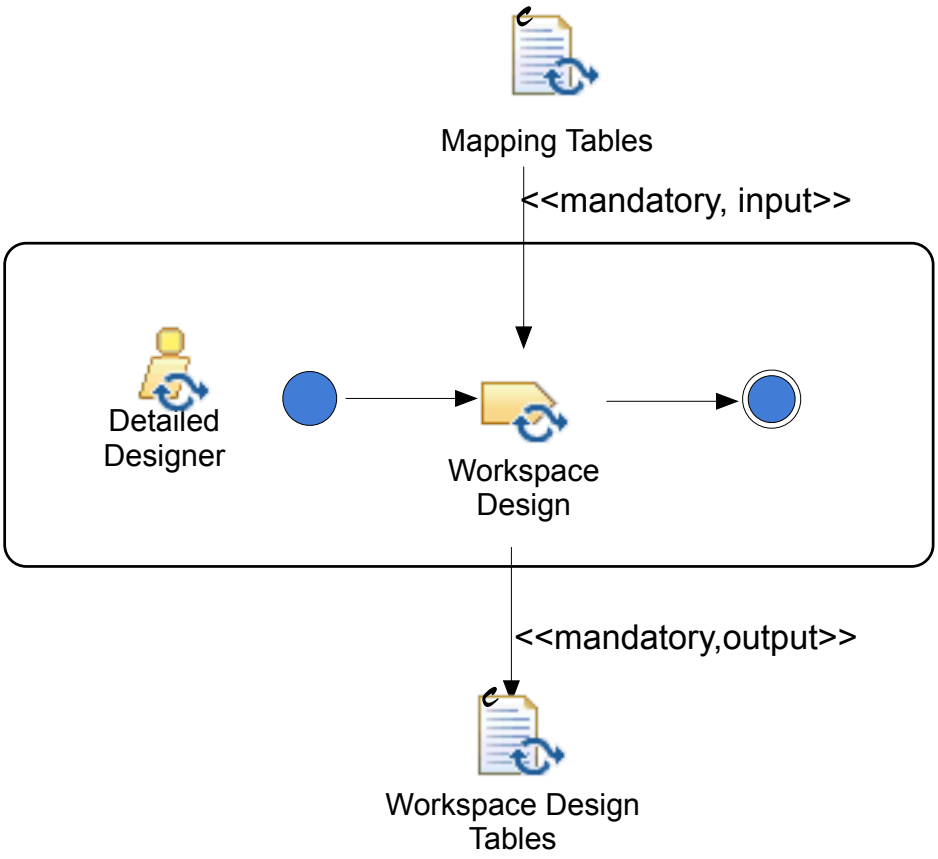


Figure 2.92: The flow of task in the Workspace Design activity

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 2.93

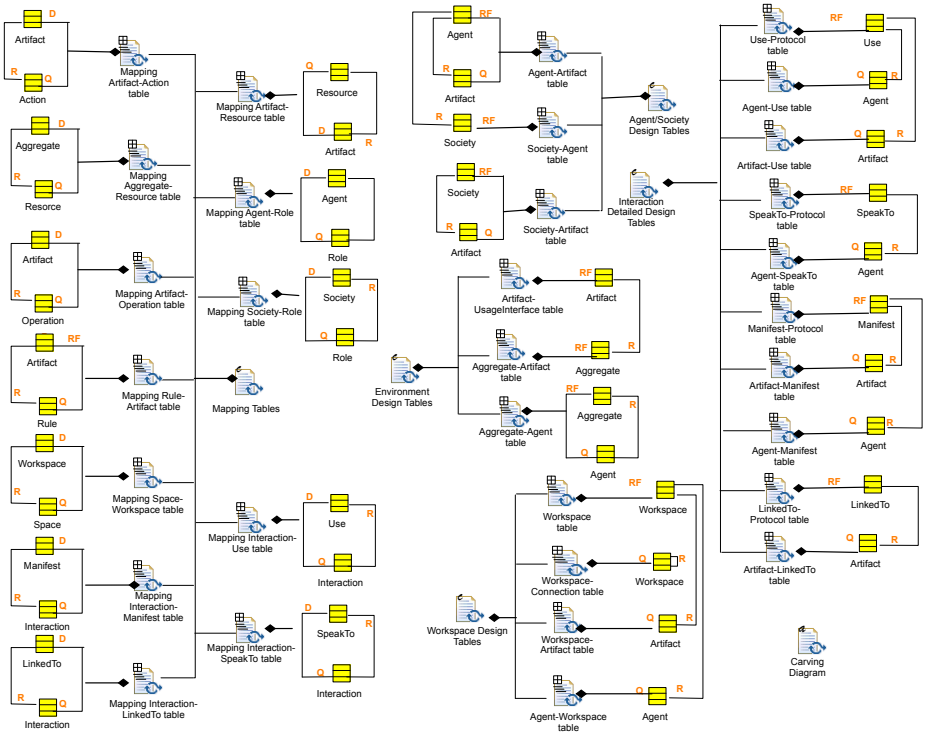


Figure 2.93: The Detailed Design work products

reports the relationships among the work products of this step and the MMMElements of the Detailed Design step.

Kinds of work products

Table 2.4 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.

Table 2.4: 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 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
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

Example of work products

An example of the Carving Diagram is reported in Figure 2.94, where the hypothesis that the Architectural Design phase outlined roles R1, R2 at the core layer C, roles R4, R5, and the projection of R2 at layer $C+1$ is made, along with roles R6, R7, R8 and R9 at layer $C+2$. Turning this conceptual view into a real design view means choosing one representation level (i.e., zoom) for each entity, starting from the core layer: so, for instance, we could decide to zoom only R1, keeping R2 at the basic (core) representation level. Moreover, we could choose to further in-zoom R4 as the set of (sub)roles R6, R7, while keeping R5 as is. The result can be graphically expressed by *carving out* the roles R1, R2, R4, R5, R6, and R7 from the Architectural Design view, as shown with the curbed line in Figure 2.94 (centre).

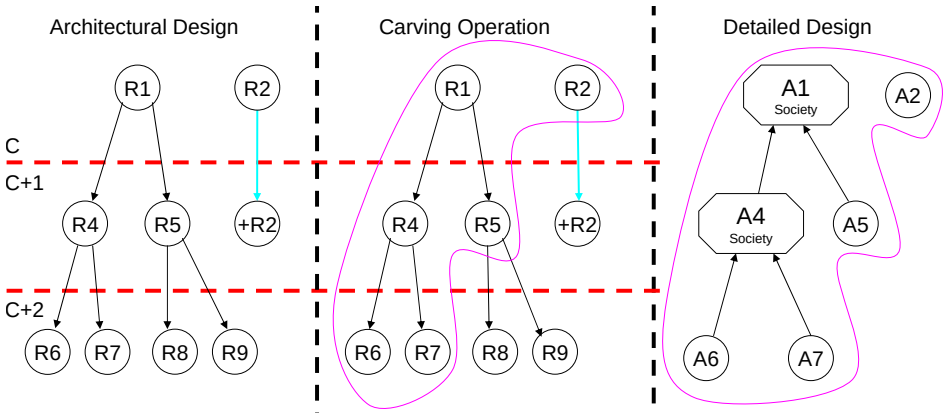


Figure 2.94: Design steps and Carving Operation

A similar approach is adopted for the environmental entities: the unzoomed resources identified in the previous step are now mapped onto suitable artifacts (intended as entities providing some services), and in-zoomed resources are mapped onto aggregates of artifacts. This "carving operation" represents the boundary between Architectural Design – expressed in terms of roles, services, resources, and workspaces – and Detailed Design—expressed in terms of agents, agent societies, artifacts, and aggregates. In the case of our example (see Figure 2.94 (right)), role R1 is to be mapped

onto an agent society (A1), while role R2 is to be mapped onto an individual agent (A2). Going further, the agent society A1 is composed of two entities, representing roles R4 and R5: again, the first is to be mapped onto an agent society (A4), since role R4 is in-zoomed in the carving, while R5 is to be mapped onto an individual agent (A5); the same process applies to A4, which turns out to be composed of agents A6 and A7, mapping roles R6 and R7, respectively.

An example of the carving for the conference management system is reported in Figure 2.95.

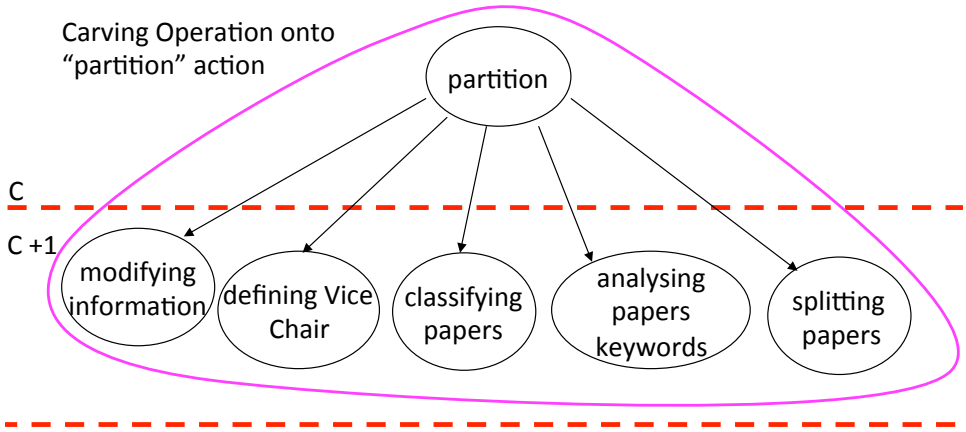


Figure 2.95: Carving Operation in the conference management system

Figure 2.96 depicts an example of the structure of the Mapping Tables:

- *Mapping Agent-Role table (MAR_t)* maps Roles onto Agents
- *Mapping Society-Role table (MSR_t)* maps Roles onto Societies
- *Mapping Artifact-Action table ($MAAc_t$)* maps Actions onto Individual Artifacts
- *Mapping Artifact-Resource table ($MArR_t$)* maps Resources onto Artifacts
- *Mapping Aggregate-Resource table ($MAggR_t$)* maps Resources onto Aggregate
- *Mapping Artifact-Operation table ($MArOp_t$)* maps Operation onto Environmental Artifacts
- *Mapping Artifact-Rule table ($MArRu_t$)* maps the Rules onto the artifacts that implement and enforce them
- *Mapping Artifact-Operation table (MSW_t)* maps Spaces onto Workspaces

- *Mapping Interaction-Uses table (MIU_t)* maps Interactions onto Use
- *Mapping Interaction-Manifests table (MIM_t)* maps Interactions onto Manifest
- *Mapping Interaction-SpeakTo table ($MISp_t$)* maps Interactions onto SpeakTo
- *Mapping Interaction-LinkedTo table (MIL_t)* maps Interactions onto LinkedTo

The following figures present some examples of the Mapping Tables for the conference management case study. Figure 2.100 depicts an example of the structure of the Agent/Society Design Tables:

- *Agent-Artifact table ($(L)AA_t$)* specifies the (individual) artifacts related to agents
- *Society-Agent table ($(L)SA_t$)* specifies which agents work in the society
- *Society-Artifact table ($(L)SAr_t$)* specifies the artifacts related to societies

Figure 2.101 presents an example of the Agent/Society Design Tables for the conference management case study. Figure 2.102 depicts an example of the structure of the Environment Design Tables:

- *Artifact-UsageInterface table ($(L)AUI_t$)* specifies the operations provided by artifacts
- *Aggregate-Artifact table ($AggArt_t$)* lists the artifacts belonging to a specific aggregate
- *Aggregate-Agent table ($AggAge_t$)* lists the agents belonging to a specific aggregate

Figure 2.103 presents an example of the Environment Design Tables for the conference management case study. Figure 2.104 depicts an example of the structure of the Interaction Detailed Design Tables:

- *Use-Protocol table (UP_t)* details the protocols for each "uses" interaction
- *Agent-Use table ($UAge_t$)* specifies the "uses" where each agent is involved
- *Artifact-Use table ($ArtU_t$)* specifies the "uses" where each artifact is involved

Agent	Role
agent name	*role names*

Society	Role
society name	*role name*

(Individual) Artifact	Action
artifact name	*action names*

(Environmental) Artifact	Resource
artifact name	*resource names*

Aggregate	Resource
aggregate name	*resource name*

(Environmental) Artifact	Operation
artifact name	*operation names*

Rule	Artifact
rule name	*artifact names*

Workspace	Space
workspace name	*space names*

Interaction	Use
interaction name	*use names*

Interaction	Manifest
interaction name	*manifest names*

Interaction	SpeakTo
interaction name	*speak names*

Interaction	LinkedTo
interaction name	*linked names*

Figure 2.96: Mapping Tables in top-down order: MAR_t, MSR_t, $MAAc_t$, $MArR_t$, $MAggR_t$, $MArOp_t$, $MArRu_t$, MSW_t, MIU_t, MIM_t, $MISp_t$, MIL_t

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

Figure 2.97: Mapping Agent-Role table MAR_t

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

Figure 2.98: 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

Figure 2.99: Mapping Artifact-Rule table $MArRu_t$

Agent	(Individual) Artifact
agent name	*artifact names*

Society	Agent
Society name	*agent names*

Society	Artifact
society name	*artifact names*

Figure 2.100: Agent/Society Design Tables in top-down order: AA_t, SA_t, SAr_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

Figure 2.101: Agent-Artifact table AA_t

Artifact	Usage Interface
artifact name	*list of operations*

Aggregate	Artifact
aggregate name	*artifact names*

Aggregate	Agent
aggregate name	*agent names*

Figure 2.102: Environment Design Tables in top-down order: AUI_t, $AggArt_t$, $AggAge_t$

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

Figure 2.103: Artifact-UsageInterface table AUI_t

- *SpeakTo-Protocol table (SP_t)* details the protocols for each "speakTo" interaction
- *SpeakTo-Agent table ($SpAge_t$)* specifies the "speakTo" where each agent is involved
- *Manifest-Protocol table (MP_t)* details the protocols for each "manifest" interaction
- *Manifest-Artifact table ($MArt_t$)* specifies the "manifests" where each artifact is involved
- *Agent-Manifest table ($AgeM_t$)* specifies the "manifests" where each agent is involved
- *LinkedTo-Protocol table (LP_t)* details the protocols for each "linkedTo" interaction
- *Artifact-LinkedTo table ($LArt_t$)* specifies the "linkedTo" where each artifact is involved

Figure 2.105 presents an example of the Interaction Detailed Design Tables for the conference management case study. Figure 2.106 depicts an example of the structure of the Workspace Design Tables:

- *Workspace table ($(L)\,W_t$)* describes the workspaces
- *Workspace-Connection table ($(L)\,WC_t$)* shows the connections among the workspaces
- *Workspace-Artifact table ($(L)\,WArt_t$)* shows the allocation of the artifacts within workspaces
- *Workspace-Agent table ($(L)\,WA_t$)* lists the workspaces that each agent can perceive

Figure 2.107 presents an example of the Workspace Design Tables for the conference management case study.

Use	Protocol
use name	*protocol description*

Agent	Use
agent name	*use names*

Artifact	Use
artifact name	*use names*

SpeakTo	Protocol
speak name	*protocol description*

Agent	SpeakTo
agent name	*speak names*

Manifest	Protocol
speak name	*protocol description*

Artifact	Manifest
artifact name	*manifest names*

Agent	Manifest
artifact name	*manifest names*

LinkedTo	Protocol
linked name	*protocol description*

Artifact	LinkedTo
artifact name	*linked names*

Figure 2.104: Interaction Detailed Design Tables in top-down order: UP_t, $AgeU_t$, $ArtU_t$, SP_t, $AgeSp_t$, MP_t, $ArtM_t$, $AgeM_t$, LP_t, $ArtL_t$

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

Figure 2.105: Use-Protocol table UP_t

Workspace	Description
workspace name	*description*

Workspace	Connection
workspace name	*workspace names*

Workspace	Artifact
workspace name	*artifact names*

Agent	Workspace
agent name	*workspace names*

Figure 2.106: Workspace Design Tables, in top-down order: $(L)\,W_t$, $(L)\,WC_t$, $(L)\,WArt_t$ and $(L)\,WA_t$

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

Figure 2.107: Workspace-Artifact table WA_t

3
Work Products Dependencies

The diagram in Figure 3.1 describes the dependencies among the different composite work products.

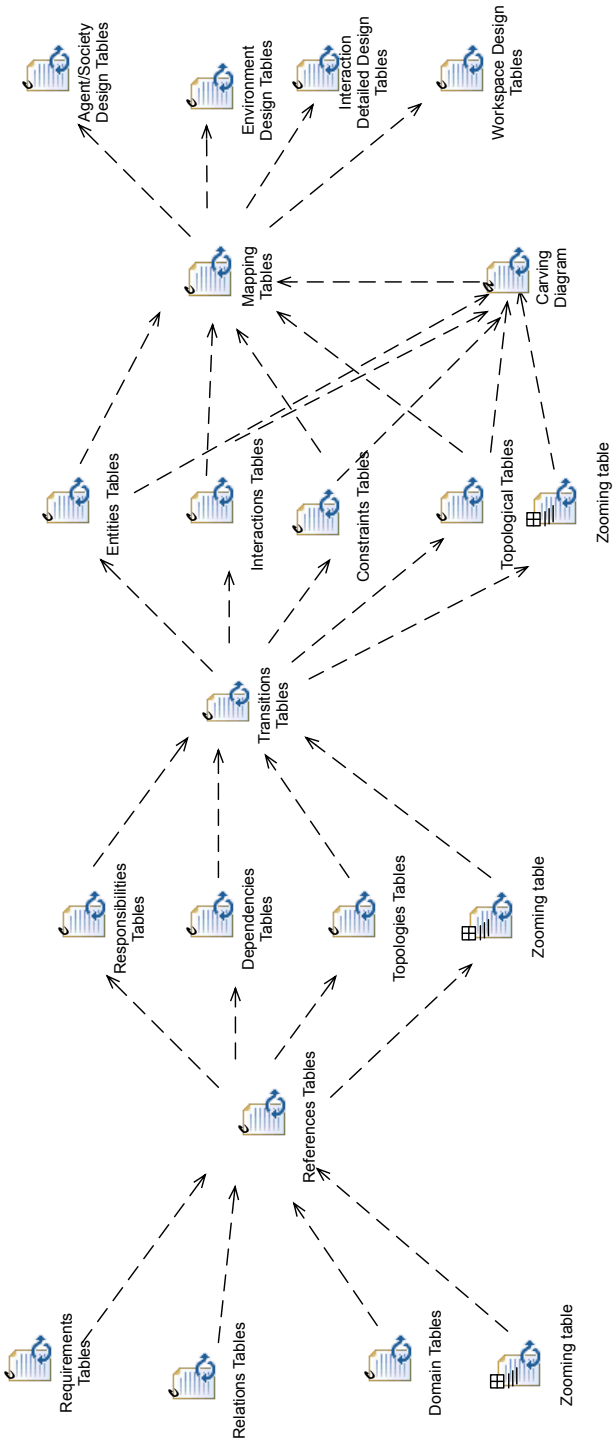


Figure 3.1: The work products dependencies

Bibliography

[1] aliCE Research Group. SODA home page. http://soda.alice.unibo.it.

[2] Ambra Molesini, Enrico Denti, and Andrea Omicini. Agent-based conference management: A case study in SODA. *International Journal of Agent-Oriented Software Engineering*, 4(1):1–31, 2010.

[3] Ambra Molesini, Elena Nardini, Enrico Denti, and Andrea Omicini. Advancing object-oriented standards toward agent-oriented methodologies: SPEM 2.0 on SODA. In Matteo Baldoni, Massimo Cossentino, Flavio De Paoli, and Valeria Seidita, editors, *9th Workshop "From Objects to Agents" (WOA 2008) – Evolution of Agent Development: Methodologies, Tools, Platforms and Languages*, pages 108–114, Palermo, Italy, November 2008. Seneca Edizioni.

[4] Ambra Molesini, Elena Nardini, Enrico Denti, and Andrea Omicini. Situated process engineering for integrating processes from methodologies to infrastructures. In Sung Y. Shin, Sascha Ossowski, Ronaldo Menezes, and Mirko Viroli, editors, *24th Annual ACM Symposium on Applied Computing (SAC 2009)*, volume II, pages 699–706, Honolulu, Hawai'i, USA, 8–12 March 2009. ACM.

[5] Ambra Molesini and Andrea Omicini. Documenting SODA: An evaluation of the process documentation template. In Andrea Omicini and Mirko Viroli, editors, *WOA 2010 – Dagli oggetti agli agenti. Modelli e tecnologie per sistemi complessi: context-dependent, knowledge-intensive, nature-inspired e self-**, volume 621 of *CEUR Workshop Proceedings*, pages 95–101, Rimini, Italy, 5-7 September 2010. Sun SITE Central Europe, RWTH Aachen University.

[6] Ambra Molesini, Andrea Omicini, Enrico Denti, and Alessandro Ricci. SODA: A roadmap to artefacts. In Oğuz Dikenelli, Marie-Pierre Gleizes, and Alessandro Ricci, editors, *Engineering Societies in the Agents World VI*, volume 3963 of *LNAI*, pages 49–62. Springer, June 2006. 6th International Workshop (ESAW 2005), Kuşadası, Aydın, Turkey, 26–28 October 2005. Revised, Selected & Invited Papers.

[7] Ambra Molesini, Andrea Omicini, Alessandro Ricci, and Enrico Denti. Zooming multi-agent systems. In Jörg P. Müller and Franco Zambonelli, editors, *Agent-Oriented Software Engineering VI*, volume 3950 of *LNCS*, pages 81–93. Springer, 2006. 6th International Workshop (AOSE 2005), Utrecht, The Netherlands, 25–26 July 2005. Revised and Invited Papers.

[8] Object Management Group. Software & Systems Process Engineering Meta-Model Specification 2.0. `http://www.omg.org/spec/SPEM/2.0/PDF`, April 2008.

[9] Andrea Omicini. SODA: Societies and infrastructures in the analysis and design of agent-based systems. In Paolo Ciancarini and Michael J. Wooldridge, editors, *Agent-Oriented Software Engineering*, volume 1957 of *LNCS*, pages 185–193. Springer, 2001. 1st International Workshop (AOSE 2000), Limerick, Ireland, 10 June 2000. Revised Papers.

[10] Andrea Omicini. Formal ReSpecT in the A&A perspective. *Electronic Notes in Theoretical Computer Sciences*, 175(2):97–117, June 2007. 5th International

Workshop on Foundations of Coordination Languages and Software Architectures (FOCLASA'06), CONCUR'06, Bonn, Germany, 31 August 2006. Post-proceedings.

[11] Andrea Omicini, Alessandro Ricci, and Mirko Viroli. *Agens Faber*: Toward a theory of artefacts for MAS. *Electronic Notes in Theoretical Computer Sciences*, 150(3):21–36, 29 May 2006. 1st International Workshop "Coordination and Organization" (CoOrg 2005), COORDINATION 2005, Namur, Belgium, 22 April 2005. Proceedings.

[12] Valeria Seidita, Massimo Cossentino, and Salvatore Gaglio. Using and extending the spem specifications to represent agent oriented methodologies. In Michael Luck and Jorge J. Gómez-Sanz, editors, *Agent-Oriented Software Engineering IX, 9th International Workshop, AOSE 2008, Estoril, Portugal, May 12-13, 2008, Revised Selected Papers*, volume 5386 of *Lecture Notes in Computer Science*, pages 46–59. Springer, 2009.

[13] Ian Sommerville. *Software Engineering 8th Edition*. Addison-Wesley, 2007.