A Compositional Knowledge Level Process Model of Requirements Engineering*

Daniela E. Herlea[1], Catholijn M. Jonker[2], Jan Treur[2], Niek J.E. Wijngaards[1,2]

[1] Software Engineering Research Network, University of Calgary,
2500 University Drive NW, Calgary, Alberta T2N 1N4, Canada
Email: danah@cpsc.ucalgary.ca

[2] Department of Artificial Intelligence, Vrije Universiteit Amsterdam,
De Boelelaan 1081a, 1081 HV, Amsterdam, The Netherlands
URL: http://www.cs.vu.nl/~{jonker,treur,niek} Email: {jonker,treur,niek}@cs.vu.nl

Abstract.
In current literature few detailed process models for Requirements Engineering are presented: usually high-level activities are distinguished, without a more precise specification of each activity. In this paper the process of Requirements Engineering has been analyzed using knowledge-level modelling techniques, resulting in a well-specified compositional process model for the Requirements Engineering task. This process model is considered to be a generic process model: it can be refined (by instantiation or specialisation) into a process model for a specific kind of Requirements Engineering process.

1 Introduction

Requirements Engineering (RE) addresses the development and validation of methods for eliciting, representing, analyzing, and confirming system requirements. Requirements Engineering further concerns methods for transforming requirements into specifications for design and implementation. A requirements engineering process is characterised as a structured set of activities needed to create and maintain a systems requirements document (Davis, 1993; Kontonya and Sommerville, 1998; Loucipoulos and Karakostas, 1995; Martin, 1988; Sommerville and Sawyer, 1997). To obtain insight in this process, a description of the activities is needed, the inputs and outputs to/from each activity are to be described. Furthermore, tools are needed to support the requirements engineering process.

* A preliminary and shorter version of this paper was presented at the IEA/AIE'99 conference, see

No standard and generally agreed requirements engineering process exists. In (Kontonya and Sommerville, 1998; Sommerville and Sawyer, 1997) the following activities are considered to be core activities in the process:

- *Requirements elicitation*, through which the requirements are discovered by consulting the stakeholders of the system to be developed.
- *Requirements analysis* and negotiation, through which requirements are analyzed in detail for conflict, ambiguities and inconsistencies. The agreement of the stakeholders on a set of system requirements is essential.
- *Requirements validation*, through which the requirements are checked for consistency and completeness.
- *Requirements documentation*, through which the requirements are maintained and motivated.

Aside from the above, in (Dubois, Du Bois, and Zeippen, 1995) also the activity *modelling* is distinguished. Loucipoulos and Karakostas (1995) distinguish *elicitation*, *specification* and *validation* as the main activities. Other approaches in the literature distinguish activities, like *requirements determination* (Yadav, Bravoco, Chatfield, and Rajkumar, 1988). These activities overlap with some of the activities mentioned above.

Various knowledge modelling methods and tools have been developed, for an overview see (Brazier and Wijngaards, 1997), and applied to complex tasks and domains. The application of a knowledge modelling method to the domain of Requirements Engineering in this paper has resulted in a compositional process model of the task of Requirements Engineering, based on the compositional knowledge modelling method DESIRE (Design and Specification of Interacting Reasoning components); cf. (Brazier, Jonker and Treur, 1998). DESIRE is based on a formal specification language and is supported by graphical tools. For an account of the formal semantics of the underlying language, see (Brazier, Treur, Willems, and Wijngaards, 1999).

In the approach presented in this paper requirements and scenarios are considered equally important; see also (Herlea, Jonker, Treur and Wijngaards, 1999*c*) . Requirements describe, for example, functional and behavioural properties of the

---

(Herlea, Jonker, Treur, and Wijngaards, 1999*b*).

system to be built, while scenarios describe use-cases of interactions between a user and the system; e.g., (Erdmann and Studer, 1998; Weidenhaupt, Pohl, Jarke, and Haumer, 1998). Both requirements and scenarios can be expressed in varying degrees of formality: from informal, to a semi-formal structured natural language description, to a formal description using temporal logic. Another distinction with other approaches to Requirements Engineering modelling is the possibility to introduce levels of process abstraction within the system being designed on the basis of the Requirements Engineering process results. Requirements and scenarios on one level of abstraction, are related to requirements and scenarios at the next lower level of abstraction; the requirements and scenarios at a level of abstraction 'realise' the requirements and scenarios at the next higher level of abstraction. These refinement relations between requirements play an important role in a compositional design process.

The compositional knowledge modelling method DESIRE has been applied to obtain the formal process model of the task of Requirements Engineering. The obtained process model is intended to be a *generic model*. A generic model is generic with respect to processes or tasks and knowledge structures. Genericity with respect to processes or tasks refers to the level of process abstraction: a generic model abstracts from processes at lower levels. A more specific model with respect to processes is a model within which a number of more specific processes, at a lower level of process abstraction are distinguished. This type of refinement is called *specialisation*. Genericity with respect to knowledge refers to levels of knowledge abstraction: a generic model abstracts from more specific knowledge structures. Refinement of a model with respect to the knowledge in specific domains of application, is refinement in which knowledge at a lower level of knowledge abstraction is explicitly included. This type of refinement is called *instantiation*.

In the literature, software environments supporting Requirements Engineering are described, but no knowledge level model is specified in detail. The model introduced here has been specified at an implementation-independent conceptual and logical level. It provides a detailed design for Requirements Engineering processes and for implementation of supporting software environments. A generic process model for the task of Requirements Engineering has the main advantage of *reuse* and *adaptability* to specific circumstances. Reuse as such, reduces the time, expertise and effort needed to construct process models for Requirements Engineering. Which processes and

knowledge structures are applicable in a given Requirements Engineering process depends on the situation. Whether a process can be used immediately, or whether instantiation, specialisation, and/or modification is required, depends on the desired properties of the Requirements Engineering process.

The compositional process model constructed for the Requirements Engineering task is described in detail in Sections 4 to 9. The compositional design method DESIRE is described in Section 2. In Section 3 a case study is introduced that was undertaken to test our ideas on the process model. In Section 4.3 some more details of this case study are discussed. A discussion is presented in Section 10. Appendix A can be used as an index to the paper.

2 Design of Compositional Process Models

The process model specification for requirements engineering, described in this paper, has been developed using the compositional development method DESIRE for single- and multi-agent systems (Design and Specification of Interacting Reasoning components); cf. (Brazier, Jonker, and Treur, 1998). Within this method knowledge of the following three types is distinguished:

- process composition,
- knowledge composition, and
- the relation between process composition and knowledge composition.

The development of a single- or multi-agent system is supported by graphical design software with an underlying formal language. Translation to an operational system is straightforward; in addition to the graphical design tools the software environment includes implementation generators with which specifications can be translated into executable code of a prototype system. Formal semantics can be found in (Brazier et al., 1999). The three types of knowledge are discussed in more detail below.

2.1 Process Composition

Process composition identifies the relevant processes at different levels of process abstraction, and describes how a process can be defined in terms of, or ‘is composed of’ lower level processes.

2.1.1 Identification of Processes at Different Levels of Abstraction

Processes can be described at different levels of abstraction; for example, the process of the multi-agent system as a whole, processes defined by individual agents and the external world, and processes defined by task-related components of individual agents. The identified processes are modelled as *components*. For each process the *input and output information types* are specified, i.e., the allowed type of information that can be used in the interfaces of the process. The identified levels of process abstraction are modelled as *abstraction/specialisation relations* between components: components may be *composed* of other components or they may be *primitive*. Primitive components may be either reasoning components, or components capable of performing tasks such as calculation, information retrieval, optimisation. For primitive reasoning components, based on a knowledge base, within the software environment a sophisticated inference engine is available. The levels of process abstraction provide process hiding at each level.

2.1.2 Composition of Processes

The way in which processes at one level of abstraction are composed of processes at the adjacent lower abstraction level is called *composition*. This composition of processes is described by a specification of the possibilities for *information exchange* between processes (*static view* on the composition), and a specification of *task control knowledge* used to control processes and information exchange (*dynamic view* on the composition).

2.2 Knowledge Composition

Knowledge composition identifies the knowledge structures at different levels of knowledge abstraction, and describes how a knowledge structure can be defined in terms of lower level knowledge structures. The knowledge abstraction levels may correspond to the process abstraction levels, but this is often not the case.

2.2.1 Identification of knowledge structures at different abstraction levels

The two main structures used as building blocks to model knowledge are: *information types* and *knowledge bases*. Knowledge structures can be identified and described at different levels of abstraction. At higher levels details can be hidden. An *information*

type defines an ontology (or lexicon, vocabulary) to describe objects or terms, their sorts, and the relations or functions that can be defined on these objects. Information types can graphically be represented on the basis of conceptual graphs and logically in order-sorted predicate logic. A *knowledge base* defines a part of the knowledge that is used in one or more of the processes. Knowledge is represented by formulae in order-sorted predicate logic, which can be normalised by a standard transformation into if-then rules.

2.2.2 Composition of Knowledge Structures

Information types can be composed of more specific information types, following the principle of compositionality discussed above. Similarly, knowledge bases can be composed of more specific knowledge bases. The compositional structure is based on the different levels of knowledge abstraction distinguished, and results in information and knowledge hiding.

2.3 Relation between Process Composition and Knowledge Composition

Each process in a process composition uses knowledge structures. Which knowledge structures are used for which processes is defined by the relation between process composition and knowledge composition.

3 An Example Case Study

The example domain for the case study is the development of a multi-agent system that keeps its human users informed with respect to their interests and the rapidly changing available information on the World Wide Web. The task of the multi-agent system is to inform each of its users on information available (e.g., papers) on the World Wide Web that is within their scope of interest. The sources of information are the World Wide Web, but also information providing agents that operate on the World Wide Web, for example, agents related to Web sites of research groups, which announce new papers included in their web-site.

To get an impression, the following list shows the initially elicited requirements R1 to R9, including their sub-divisions a-d; this is taken from the requirements document. For traceability, numbers between “{” and “}” refer to parts of the original interview.

R1 The system shall service the individual users of the group. {9}

R2 The system shall behave towards a user based on that user's research interests and topics. {9}

R3 The system shall have the task of searching information on the internet. {1, 10}

a. The user shall be able to input a "search topic".

b. The system shall provide search results from the internet.

c. The search results are related to (depends on) the "search topic".

d. Search results are provided after the user inputs a search topic.

R4 The system shall have the task of keeping the user 'aware' of modifications to information on the Internet. {4, 10}

a. The user shall be able to input an "awareness topic".

b. The system shall notify the user about modifications to the information on this awareness topic.

c. The notification shall be done after these modifications (found at c) become available on the internet.

R5 a. The user shall be able to specify times when s/he cannot be disturbed. {13}

b. The system shall not disturb the user at the specified times.

R6 a. The system shall be able to suggest non-requested information {14}

b. Suggestion of non-requested information is based on: learning from overlapping research interests (among users of the System). {14}

R7 a. The user shall be able to constrain the suggested information from the system. {15}

b. The system shall adhere to the constraints when suggesting information.

R8. The system shall be able to save search topics and awareness topics results and to delete results.

R9. The system shall be able to authenticate the users.

In Section 4.3 it will be discussed in more detail how some of these requirements were reformulateed during the Requiurements Engineering process.

4 Composition of Requirements Engineering

An overview of the different processes and their abstraction levels within the process Requirements Engineering is shown in Appendix A; this overview can also be used as an index for the paper. In subsequent sections for each of the composed processes their process composition and knowledge composition, are specified.

Within each of the sections one level of process abstraction is described. The composition of Requirements Engineering is described in Section 4. The composition of Elicitation is described in Section 5. The composition of Manipulation of Requirements and Scenarios is described in Section 6. The composition of Maintenance of Requirements and Scenarios Specification is described in Section 7. The composition of Manipulation of Requirements is described in Section 8. The composition of Manipulation of Scenarios is similar to the composition of Manipulation of Scenarios and therefore not described in detail in this paper. The composition of Reformulation of Requirements is described in Section 9.

The process of Requirements Engineering is described in two phases: first is process composition, then composition of knowledge structures related to this process. The information types identified in the process identification in Section 4.1 are described in detail in the knowledge composition in Section 4.2. Knowledge bases have not been specified; they depend on specific application domains. The reader may already take into account Section 4.3, where, as an ilustration, for some example requirements it is shown how they were reformulated during the process.

4.1 Process Composition of Requirements Engineering

Following the structure shown in Section 2, the process composition of requirements engineering is described by its levels of process abstraction, identification of processes, and composition relation between processes.

The first two levels of process abstraction for requirements engineering are shown in Figure 1. The processes elicitation, manipulation of requirements and scenarios, and maintenance of requirements and scenarios specification are distinguished within the process requirements engineering.

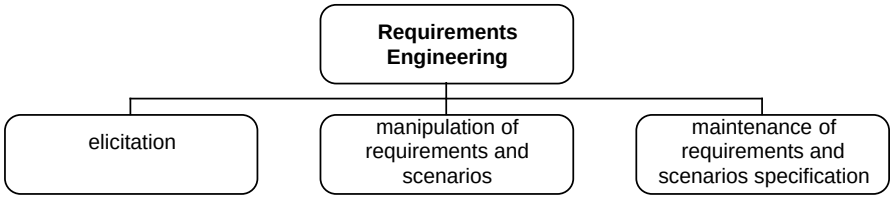

Figure 1. First level of process abstraction within process Requirements Engineering**.**

The process elicitation provides initial problem descriptions, requirements and scenarios elicited from stakeholders, as well as domain ontologies and knowledge acquired in the domain. The process manipulation of requirements and scenarios attempts to resolve ambiguities in requirements and scenarios, and identifies and possibly removes requirements not supported by stakeholders, and inconsistent requirements and scenarios. This process reformulates informal requirements and scenarios, to more structured semi-formal requirements and scenarios, and, if needed, finally to formal requirements and scenarios. It also provides relationships among and between requirements and scenarios. The process maintenance of requirements and scenarios specification maintains the documents in which the information requirements and scenarios are described, including information on traceability.

Each of the processes depicted in Figure 1 can be characterized in more detail in terms of their interfaces (input and output information types), as shown in Table 1.

process	*input information type*	*output information type*
elicitation	• requirements and scenarios information	• elicitation results • elicitation basic material
manipulation of requirements and scenarios	• elicitation results	• requirements and scenarios information

maintenance of requirements and scenarios specification	• elicitation results • requirements and scenarios information • elicitation basic material	• elicitation results • requirements and scenarios information • elicitation basic material

Table 1. Interface information types of direct sub-processes of requirements engineering.

The input and output information types in the interface of the processes described in Table 1 are elaborated below:

- The process elicitation uses input information on the requirements, scenarios, and relations among them (requirements and scenarios information). The process produces as output descriptions of the problem, elicited requirements and scenarios, relations between elicited requirements and scenarios and existing requirements and scenarios, and domain ontologies and domain knowledge (elicitation results), and the elicited material, e.g., stakeholders protocols, underlying the elicitation results (elicitation basic material).
- The process manipulation of requirements and scenarios needs descriptions of the perceived problem from the stakeholders as a result from the elicitation task; elicited requirements and scenarios, relations between elicited requirements and scenarios and existing requirements and scenarios, and domain ontologies and domain knowledge (elicited requirements). The process produces reformulated requirements, scenarios, and relations among them (requirements and scenarios information).
- The process maintenance of requirements and scenarios specification stores information on the requirements, scenarios, and relations among them (requirements and scenarios information), descriptions of the problem, elicited requirements and scenarios, relations between elicited requirements and scenarios and existing requirements and scenarios, and domain ontologies and domain knowledge (elicitation results), and the elicited material underlying the elicitation results (elicitation basic material). The process maintenance of requirements and scenarios specification provides as output information the same information as its input information. Its only function is to store the information. It has no further processing, such as combining or adding information.

The static perspective on the composition relation between the process requirements engineering and its direct sub-processes is shown in Figure 2. Within the component requirements engineering a number of information links are distinguished. The names of these information links reflect which information can be exchanged.

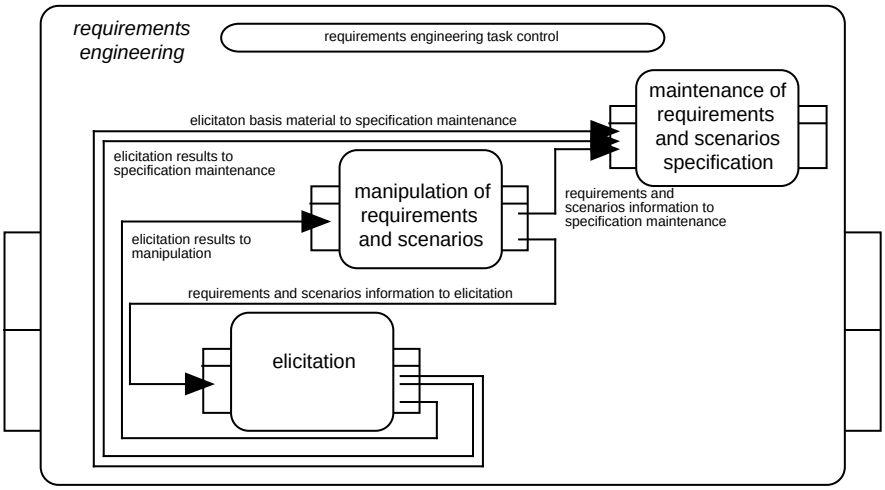

Figure 2. Process composition of requirements engineering: information links

The dynamic perspective on the composition relation specifies control over the sub-components and information links within the component requirements engineering. Task control within requirements engineering specifies a flexible type of control: during performance of each process it can be decided to suspend the process for a while to do other processes in the meantime, and resume the original process later. The task control specifies which sub-component is activated under which conditions.

On startup of requirements engineering, *elicitation* is immediately activated with, of course, no existing requirements and scenarios information in its input interface. Upon termination of elicitation, its results can be processed by manipulation of requirements and scenarios and can be placed in documents by maintenance of requirements and scenarios specification. On termination of manipulation of requirements and scenarios, elicitation can be reactivated. In contrast to its initial activation, this time elicitation can be based on information resulting from the manipulation of previously elicited requirements and scenarios.

4.2 Knowledge Composition of Requirements Engineering

The information types described in the interfaces of the component requirements engineering and its direct sub-components are briefly described in this section. All of these information types specify statements *about* requirements and/or scenarios. In turn a requirement is a statement that some behavioural property is required, expressed by the object-level information types in Figure 3. To be able to express, for example, that a requirement is ambiguous, or that a scenario has been elicited, or that a requirement is a refinement of another requirement, requirements and scenarios expressed as statements on the object level, are terms at the meta-level. The information types on the meta-level of Figure 3 all make use of a meta-description construct specified in the information type requirements meta-descriptions that makes object-terms of the object level statements stating that a certain property is a requirement.

The information types specified in the interfaces of the component requirements engineering and its direct sub-components all refer to. This information type contains a sort REQUIREMENTS which contains all formal, semi-formal, and informal requirements as its objects. The sort REQUIREMENTS has three sub-sorts: FORMAL REQUIREMENTS, SEMI-FORMAL REQUIREMENTS, and INFORMAL REQUIREMENTS. These three sub-sorts are defined in three separate information types which each contain in their respective sort the meta-descriptions of information types containing the actual statements. This is depicted in Figure 3, in which the dashed lines indicate the meta-description-of relation.

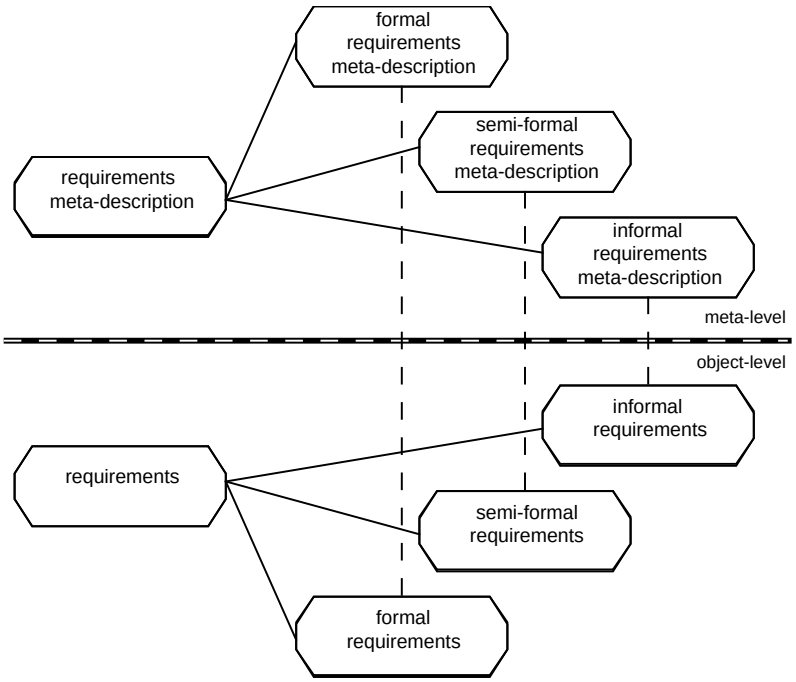

Figure 3. Information types and meta-levels related to meta-description of requirements

The construction of the information type containing the meta-description of scenarios is similar to the construction of the information type containing the meta-description of requirements. The sorts SCENARIOS, INFORMAL SCENARIOS, SEMI-FORMAL SCENARIOS, and FORMAL SCENARIOS are specified.

The information type requirements and scenarios information is based on three information types: requirements information, scenarios information, and relations between requirements and scenarios, as shown in Figure 4. In turn, the information type requirements information is based on three information types: current requirements, clusters of requirements, and relations among requirements. The information type scenarios information is based on three similar information types: current scenarios, clusters of scenarios, and relations among scenarios.

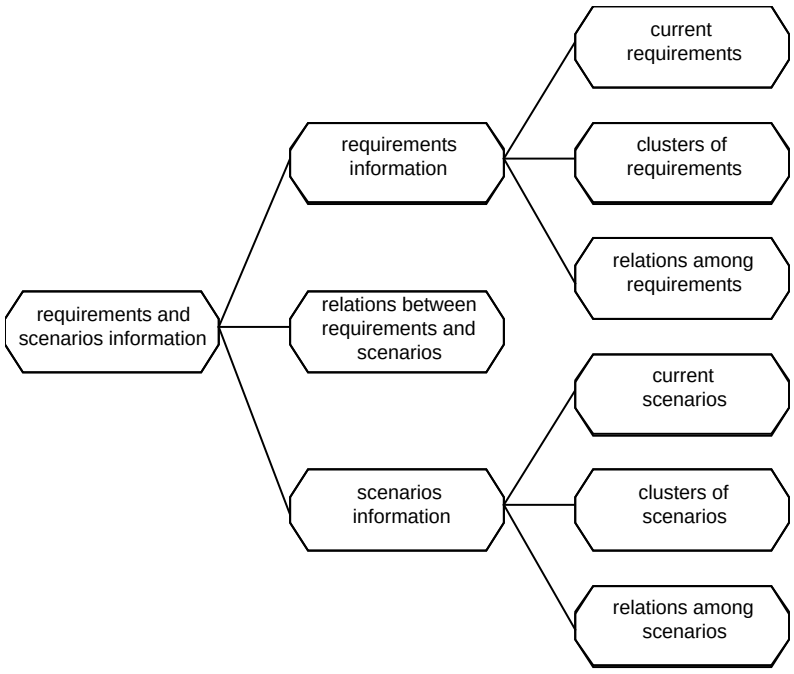

Figure 4. Information type requirements and scenarios information.

Examples of relations defined in these information types are shown in Table 2. In this table, for example, requirement_in_cluster: REQUIREMENT * REQ-CLUSTER-ID expresses that requirement_in_cluster is a binary relation on the product set REQUIREMENT * REQ-CLUSTER-ID.

Information type	*Examples of relations*	
current requirements	current_requirement:	REQUIREMENTS
current scenarios	current_scenario:	SCENARIOS
clusters of requirements	requirement_in_cluster:	REQUIREMENT * REQ-CLUSTER-ID
clusters of scenarios	scenario_in_cluster:	SCENARIO * SCEN-CLUSTER-ID
relations among requirements	req_is_more_precise_than_req:	REQ-CLUSTER-ID * REQ-CLUSTER-ID
	req_refines_req:	REQ-CLUSTER-ID * REQ-CLUSTER-ID
	/* refinement across one level of process abstraction within the requirements and scenarios */	
relations among scenarios	scen_is_more_precise_than_scen:	SCEN-CLUSTER-ID * SCEN-CLUSTER-ID

	scen_refines_scen:	SCEN-CLUSTER-ID * SCEN-CLUSTER-ID
	/* refinement across one level of process abstraction within the requirements and scenarios */	
relations between requirements and scenarios	scen_illustrates_req:	SCEN-CLUSTER-ID * REQ-CLUSTER-ID
	scen_satisfies_req:	FORMAL-SCEN-CLUSTER-ID * FORMAL-REQ-CLUSTER-ID

Table 2. Information types and examples of relations defined in the information types related to requirements and scenarios information.

The relations describing relationships between and among requirements and scenarios specify the smallest relationships possible; e.g., transitive closures of 'chains of relationships' are not specified.

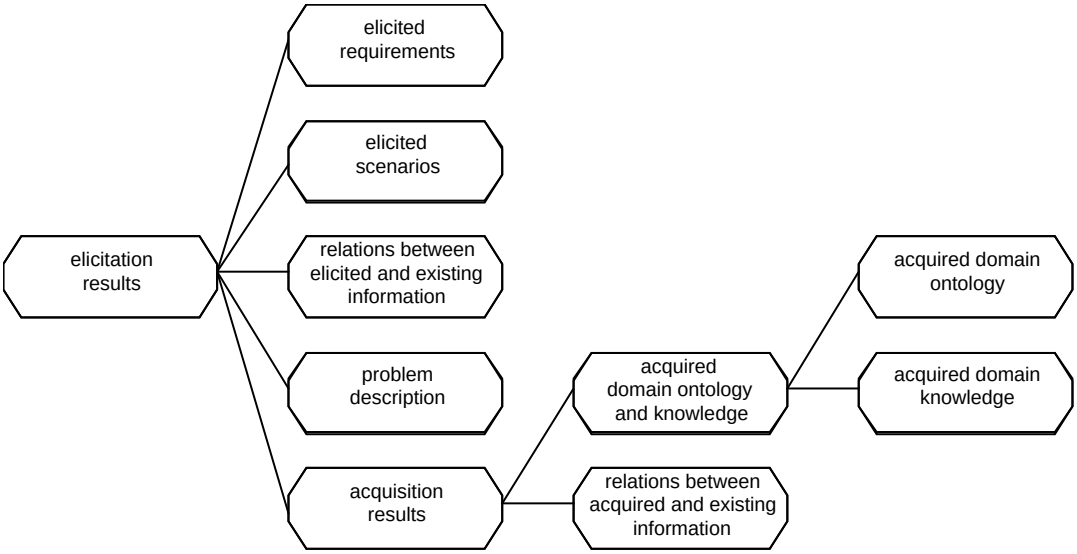

Figure 5. Information type elicitation results.

Figure 5 shows the composition of the information types elicitation results, acquisition results and acquired domain ontology and knowledge.

Examples of relations defined in these information types are shown in Table 3.

Information type	*Examples of relations*

elicited requirements	elicited_requirement:	INFORMAL REQUIREMENTS
elicited scenarios	elicited_scenario:	INFORMAL SCENARIOS
relations between elicited and existing information	req_based_on:	INFORMAL-REQ-CLUSTER-ID * REQ-CLUSTER-ID
	scen_based_on:	INFORMAL-SCEN-CLUSTER-ID * SCEN-CLUSTER-ID
problem description	identified_problem:	PROBLEM DESCRIPTION
acquired domain ontology	acquired_ontology:	DOMAIN ONTOLOGY
acquired domain knowledge	acquired_knowledge:	DOMAIN KNOWLEDGE
relations between acquired and existing information	acquired_based_on:	DOMAIN ONTOLOGY AND DOMAIN KNOWLEDGE * CLUSTER-ID

Table 3. Information types and examples of relations defined in the information types related to elicitation results.

The relation acquired knowledge represents both knowledge about and models of the domain.

4.3 Illustrations from the case study

For the example application, first a list of nine, rather imprecisely formulated initial requirements was elicited. As an example, the elicited requirement on 'keeping aware' is discussed below.

4.3.1 Elicitation

Example of an informal initial requirement:

L0.R1 The user needs to be kept 'aware' of relevant new information on the World Wide Web.

Requirement L0.R1 is based on the information elicited from the interview with the stakeholder. The following scenario was elicited from the stakeholder as well:

L0.Sc1

1. user generates an **awareness scope** : AS1
2. user is waiting
3. **new information** is made available on the World Wide Web
4. user receives **results for awareness scope** AS1: ASR1

4.3.2 Manipulation

The requirement L0.R1 was analysed and reformulated into a more precise requirement.

Reformulation from informal to semiformal

In the (reformulated) scenarios and requirements, terminology is identified, relevant for the construction of domain ontologies (words in bold-face are part of the domain ontologies being acquired).

Example of a reformulation of a requirement at top level:

L0.R1.1 The user will be notified of **new information** (on the World Wide Web) on an **awareness scope**
after the user has expressed the **awareness scope** and
just after this **new information** becomes available on the World Wide Web,
unless the user has retracted the awareness scope (**awareness scope retraction**).

Next, in the process to semiformal and formal reformulations, for the informally specified requirement L0.R1.1, the following reformulation steps have been made:

At any point in time
The user will receive on its input **results for awareness scope** , i.e., **new information** on an **awareness scope**
after the user has generated on its output the **awareness scope** and
just after this **new information** becomes available as output of the World Wide Web ,
unless by this time the user has generated on its output an **awareness scope retraction**.

At any point in time,
if at an earlier point in time the user has generated on its output an **awareness scope,** and
since then the user has not generated on its output an **awareness scope retraction** referring to this **awareness scope**, and
just before **new information** within this **awareness scope** becomes available as output of the World Wide Web ,
then the user will receive on its input this **new information** within the **awareness scope** .

Based on these reformulation steps the following semi-formal structured requirement has been specified:

L0.R1.2 At any point in time,
if
at an earlier point in time

user output :	an **awareness scope,** and
since then	
not user output :	**retraction** of this **awareness scope**, and
just before	
World Wide Web output:	**new information** within this **awareness scope**
then	
user input:	**new information** within this **awareness scope**

The interplay between requirements elicitation and analysis and scenario elicitation and analysis plays an important role. To be more specific, it is identified which requirements and scenarios relate to each other; for example, L0.R1.2 relates to L0.Sc1.2. If it is identified that for a requirement no related scenario is available yet (isolated requirement), then a new scenario can be acquired.

L0.Sc1.2

1.	user output:	**awareness scope**
2.	user is waiting	
3.	World Wide Web output:	**new information**
4.	user input:	**results for awareness scope**

Reformulation from semiformal to formal

To obtain formal representations of requirements, the input and output ontologies have to be chosen as formal ontologies. The domain ontologies acquired during the reformulation process for the example application were formalised; part of the domain ontologies related to the focus on requirements and scenarios is shown below:

ontology element:	*explanation:*
SCOPE	a sort for the search scopes and awareness scopes
USER	a sort for the names of different users
PERSISTENCE_TYPE	a sort to distinguish between persistent and incidental scopes
INFO_ELEMENT	a sort for the result information
result_for_scope	a binary relation on INFO_ELEMENT and SCOPE
persistent, incidental	objects of sort PERSISTENCE_TYPE corresponding to the difference in persistence between an awareness scope and a search scope
input:	
is_interested_in	a ternary relation on USER, SCOPE and PERSISTENCE_TYPE

output:

result_for_user a ternary relation on INFO_ELEMENT, USER and SCOPE

In addition, the temporal structure, if present in a semi-formal representation, has to be expressed in a formal manner. Using the formal ontologies, and a formalisation of the temporal structure, a mathematical language is obtained to formulate formal requirement representations. The semantics are based on compositional information states which evolve over time. An *information state* M of a component D is an assignment of truth values $\{true, false, unknown\}$ to the set of ground atoms that play a role within D. The compositional structure of D is reflected in the structure of the information state. A formal definition can be found in (Brazier, Treur, Willems, and Wijngaards, 1999). The set of all possible information states of D is denoted by $IS(D)$. A *trace* $\mathcal{M}$ of a component D is a sequence of information states $(M^t)_{t \in \mathbf{N}}$ in $IS(D)$. Given a trace $\mathcal{M}$ of component D, the information state of the input interface of component C at time point t of the component D is denoted by $state_D(\mathcal{M}, t, input(C))$, where C is either D or a sub-component of D. Analogously, $state_D(\mathcal{M}, t, output(C))$, denotes the information state of the output interface of component C at time point t of the component D. These formalised information states can be related to statements via the formally defined satisfaction relation |=. Behavioural properties can be formulated in a formal manner, using quantifiers over time and the usual logical connectives such as not, &, $\Rightarrow$.

Examples of formal representations of top level requirements:

L0.R1.2 is formalised by L0.R1.3: The first part of this requirement addresses the case that information relating to an awareness scope is already present, whereas the second part addresses the case that the information becomes available later.

L0.R1.3:

$\forall\ \mathcal{M}, t$

$[\ state_S(\mathcal{M}, t, output(U)) \models is\_interested\_in(U:USER, S:SCOPE, persistent)\ \&$

$state_S(\mathcal{M}, t, output(WWW)) \models result\_for\_scope(I:INFO\_ELEMENT, S:SCOPE)\ ]$

$\Rightarrow\ \exists t' > t$

$state_S(\mathcal{M}, t', input(U)) \models result\_for\_user(I:INFO\_ELEMENT, U:USER, S:SCOPE)$

&

$\forall M , t1, t2>t1$

$state_s(\mathcal{M}, t1, output(U))$ |= is_interested_in(U:USER, S:SCOPE, persistent) &

$state_s(\mathcal{M}, t2, output(WWW))$ |= result_for_scope(I:INFO_ELEMENT, S:SCOPE) &

$\forall t'$ [$t1 < t' < t2$ $\Rightarrow$

[not $state_s(\mathcal{M}, t', output(WWW))$ |= result_for_scope(I:INFO_ELEMENT, S:SCOPE) &

not $state_s(\mathcal{M}, t', output(U))$ |= not is_interested_in(U:USER, S:SCOPE, persistent)]

$\Rightarrow \exists t3 > t2$

$state_s(\mathcal{M}, t3, input(U))$ |= result_for_user(I:INFO_ELEMENT, U:USER, S:SCOPE)

Example of a formal representation of a top level scenario

The following formal scenario representation relates to the second formal requirement representation expressed above. Note that point at time point 2 nothing happens, which corresponds to the waiting of the user, of course in another (but similar) scenario the waiting could take more time.

L0.Sc1.3:

$state_s(M, 1, output(U))$ |= is_interested_in(U:USER, S:SCOPE, persistent)

$state_s(M, 3, output(WWW))$ |= result_for_scope(I:INFO_ELEMENT, S:SCOPE)

$state_s(M, 4, input(U))$ |= result_for_user(I:INFO_ELEMENT, U:USER, S:SCOPE)

5 Composition of Elicitation

Following Section 2, the process of elicitation is described in two phases: first is process composition, then composition of knowledge structures related to this process.

5.1 Process composition of elicitation

In Figure 6 the first two levels of process abstraction for elicitation are shown. The processes problem analysis, acquisition of domain ontology and knowledge, and elicitation of requirements and scenarios are distinguished within the process elicitation.

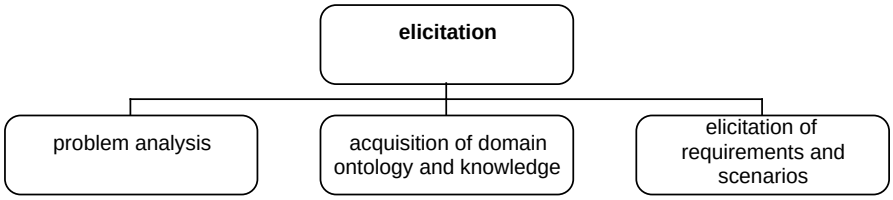

Figure 6. First level of process abstraction within process elicitation.

The three sub-processes of elicitation, as depicted in Figure 6, are closely intertwined. The process problem analysis extracts the perceived problem from the stakeholders. It can

also determine that requirements and scenarios are needed for another level of process abstraction. The process acquisition of domain ontology and knowledge acquires from stakeholders ontologies and knowledge of the domain, possibly related to existing requirements and scenarios. The process elicitation of requirements and scenarios elicits requirements and scenarios from stakeholders on the basis of identified problems, and existing requirements and scenarios.

Each of the processes depicted in Figure 6 can be characterized in terms of their interface information types, as shown in Table 4.

process	*input information type*	*output information type*
acquisition of domain ontology and knowledge	• requirements and scenarios information • problem description	• acquisition results
problem analysis	• requirements and scenarios information • acquisition results	• problem description
elicitation of requirements and scenarios	• requirements and scenarios information • acquisition results • problem description	• elicited requirements • elicited scenarios • relations between elicited and existing information

Table 4. Input and output information types of the direct sub-processes of the process elicitation.

The input and output information types in the interface of the process described in Table 4 are elaborated below:

- The process acquisition of domain ontology and knowledge uses as input information on requirements, scenarios, and relations among them (requirements and scenarios information), and descriptions of identified problems (problem descriptions). The process produces acquired domain ontologies and knowledge (acquisition results).
- The process problem analysis uses information on the requirements, scenarios, and relations among them (requirements and scenarios information), and acquired domain ontology and knowledge (acquisition results). The process has as output a description of identified problems (problem descriptions).

- The process elicitation of requirements and scenarios requires information on requirements, scenarios, and relations among them (requirements and scenarios information), acquired domain ontology and knowledge (acquisition results), and a description of identified problems (problem descriptions). The ouput of the process consists of the elicited requirements (elicited requirements), the elicited scenarios (elicited scenarios), and relationships between elicited requirements and scenarios and existing requirements and scenarios information (relations between elicited and existing information).

The static perspective on the composition relation between the process requirements engineering and its direct sub-processes is shown in Figure 7.

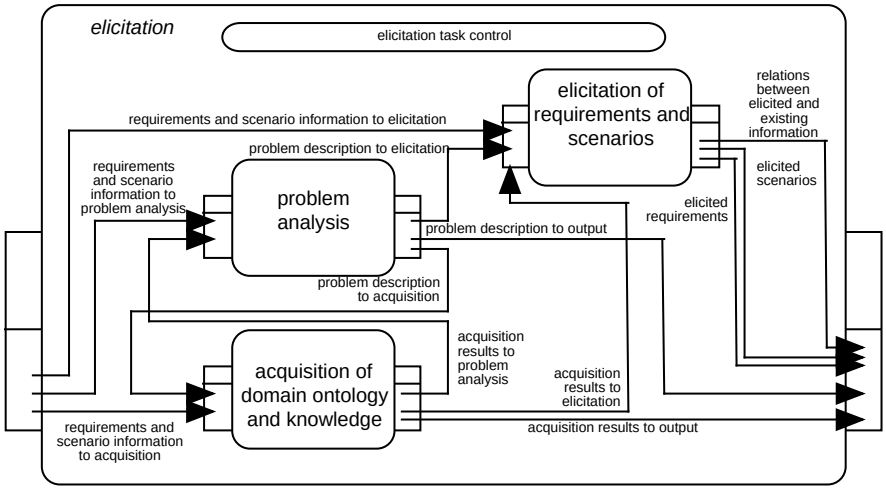

Figure 7. Process composition relation of elicitation : information links

Within the component elicitation a number of private and mediating information links is distinguished. The names of these information links reflect which information can be exchanged through the information link between the two processes.

The dynamic perspective on the composition relation specifies control over the sub-components of the component elicitation. Task control within elicitation specifies which sub-component is activated under which conditions. The three sub-components of elicitation can all be activated in parallel: results obtained by a sub-component can be used by another sub-component for interaction with stakeholders. At startup of requirements engineering, elicitation does not have any information in its input interface,

and therefore its sub-components do not have any information in their input interfaces. This results in elicitation of an initial perception of the problem, initial acquisition of domain ontology and knowledge, and initial elicitation of requirements and scenarios. In subsequent activations, information is available at the input interface of elicitation, which can be used to influence interactions with stakeholders.

An alternative to the current approach described above is a sequential approach, in which first problem analysis is activated, after its termination acquisition of domain ontology and knowledge is activated, and after the latter termination, elicitation of requirements and scenarios is activated. After termination of elicitation of requirements and scenarios a choice exists: terminate internal activities for elicitation, or activate problem analysis.

5.2 Knowledge composition of elicitation

The information types described in the interfaces of the component elicitation and its direct sub-components have already been described in Section 4.2.

6 Composition of Manipulation of Requirements and Scenarios

In Section 6.1 the process composition of manipulation of requirements and scenarios again is described, and in Section 6.2 the composition of knowledge structures.

6.1 Process composition of manipulation of requirements and scenarios

Figure 8 shows the first two levels of process abstraction for manipulation of requirements and scenarios. The processes manipulation of requirements, manipulation of scenarios, and identification of relationships between requirements and scenarios are distinguished within the process manipulation of requirements and scenarios.

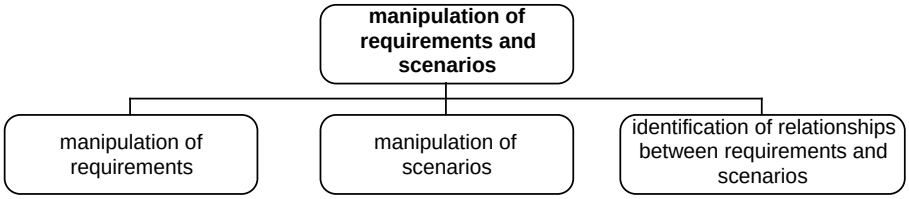

Figure 8. First level of process abstraction within process manipulation of requirements and scenarios.

The process manipulation of requirements is responsible for removing ambiguities, resolving requirements not fully supported by stakeholders, and resolving inconsistencies, while striving for progressive formalisation of requirements. This process also produces the relationships among requirements. The process manipulation of scenarios is similar to the process manipulation of requirements. The process identification of relationships between requirements and scenarios establishes which requirements are related to which scenarios, and vice versa.

Each of the processes depicted in Figure 8 can be characterized in terms of their interface information types, as shown in Table 5.

process	*input information type*	*output information type*
manipulation of requirements	• elicited requirements • relations between elicited and existing information • acquisition results • isolation information	• requirements information
manipulation of scenarios	• elicited scenarios • relations between elicited and existing information • acquisition results • isolation information	• scenarios information
identification of relationships between requirements and scenarios	• requirements information • scenarios information	• relations between requirements and scenarios • isolation information

Table 5. Interface information types of the processes within manipulation of requirements and scenarios.

The input and output information types in the interface of the process described in Table 5 are elaborated below:

- The process manipulation of requirements requires as input requirements elicited from stakeholders (elicited requirements), relations between elicited requirements and scenarios and existing requirements and scenarios information (relations between elicited and existing information), results of acquisition of domain ontology and knowledge (acquisition results), and isolated requirements and scenarios (isolation information). The process produces as ouput requirements, clusters of requirements, and relations among requirements (requirements information).
- The process manipulation of scenarios requires scenarios elicited from stakeholders (elicited scenarios), relations between elicited requirements and scenarios and existing requirements and scenarios information (relations between elicited and existing information), results of acquisition of domain ontology and knowledge (acquisition results), and isolated requirements and scenarios (isolation information). The process has as ouput scenarios, clusters of scenarios, and relations among scenarios (scenarios information).
- The process identification of relationships between requirements and scenarios performs its task in two steps: first the relations between requirements and scenarios are determined, then it identifies isolated requirements (i.e., requirements for which no scenario exists) and isolated scenarios (i.e., scenarios for which no requirement exists). As input the process needs information of two types:
  - requirements information: requirements, clusters of requirements, and relations among requirements,
  - scenarios information: scenarios, clusters of scenarios, and relations among scenarios.

  The process identification of relationships between requirements and scenarios produces output information of two types:
  - relations between requirements and scenarios: relations between requirements, scenarios, clusters of requirements, and clusters of scenarios and
  - isolation information: isolated requirement and isolated scenarios.

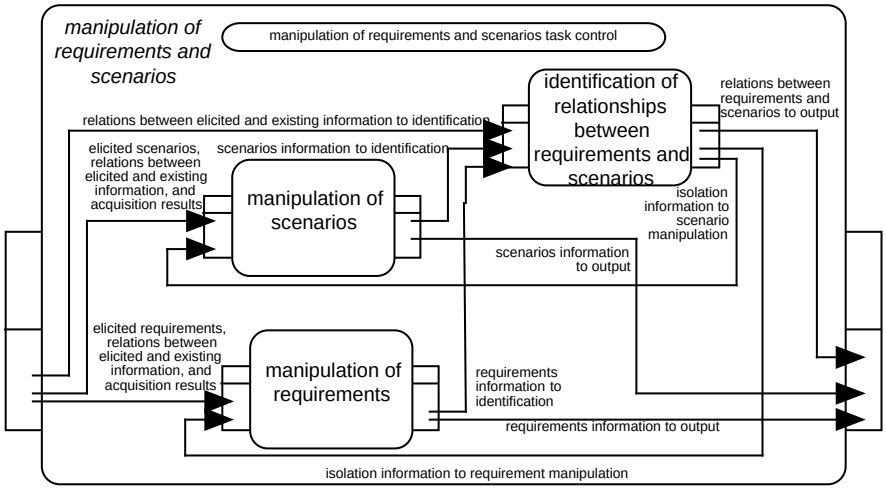

Figure 9. Process composition of manipulation of requirements and scenarios: information links.

The static perspective on the composition relation between the process requirements engineering and its direct sub-processes is shown in Figure 9. Information exchange within the component manipulation of requirements and scenarios is possible through a number of information links.

The dynamic perspective on the composition relation specifies control over the sub-components of the component manipulation of requirements and scenarios. Task control within manipulation of requirements and scenarios specifies which sub-component is activated under which conditions. On the basis of the dependencies in information links, the processes manipulation of scenarios and manipulation of requirements need to finish before the process identification of relationships between requirements and scenarios is able to finish. A number of alternative task control descriptions can be constructed which adhere to this observation. In a purely sequential approach first manipulation of requirements becomes active, then manipulation of scenarios, and finally identification of requirements- and scenarios- relationships.

6.2 Knowledge composition of manipulation of requirements and scenarios

The information types described in the interfaces of the component manipulation of requirements and scenarios and its direct sub-components have been described in Section 4.2. The information type isolation information is newly introduced in the sub-

components: it consists of two information types: isolated requirements, and isolated scenarios.

7 Composition of Maintenance of Requirements and Scenarios Specification

As before, first the process composition for the process of maintenance of requirements and scenarios specification is described (in Section 7.1), then composition of knowledge structures related to this process (in Section 7.2).

7.1 Process composition of maintenance of requirements and scenarios specification

The first two levels of process abstraction for maintenance of requirements and scenarios specification are shown in Figure 10. The processes maintenance of requirements and scenarios documents, and maintenance of traceability links are distinguished within the process maintenance of requirements and scenarios specification.

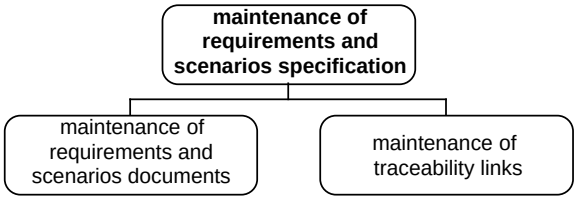

Figure 10. First level of process abstraction within process maintenance of requirements and scenarios specification.

The process maintenance of requirements and scenarios documents represents the information on requirements and scenarios in a number of documents. The process maintenance of traceability links creates the hyperlinks within and between documents.

Each of the processes depicted in Figure 10 can be characterised in terms of their interface information types, as shown in Table 6.

process	*input information type*	*output information type*
maintenance of requirements and	• requirements and scenarios	• requirements and scenarios

scenarios documents	information • elicitation results	information • elicitation results
maintenance of traceability links	• traceability relations	• traceability relations

Table 6. Input and output information types of the direct sub-processes of the process maintenance of requirements and scenarios specification.

The input and output information types in the interface of the process described in Table 6 are elaborated below:

- The process maintenance of requirements and scenarios documents uses as input information on the requirements, scenarios, and relations among them (requirements and scenarios information), and descriptions of the problem, elicited requirements and scenarios, relations between elicited requirements and scenarios and existing requirements and scenarios, and domain ontologies and domain knowledge (elicitation results). The process produces as output information its input information: no information is changed.
- The process maintenance of traceability links stores all information regarding traceability, therefore, it needs and produces information of type traceability relations without changing that information. The information type consists of references to the information types requirements information, scenarios information, relations between elicited and existing information, and relations between requirements and scenarios.

The static perspective on the composition relation between the process requirements engineering and its direct sub-processes is shown in Figure 11.

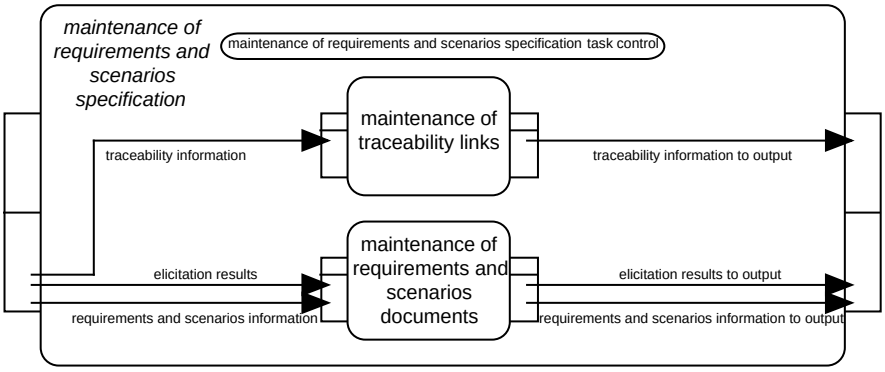

Figure 11. Composition relation between the process of maintenance of requirements and scenarios specification and its direct sub-processes.

Within the component maintenance of requirements and scenarios specification a number of mediating information links is distinguished. The names of these information links reflect which information can be exchanged through the information link between the two processes.

The dynamic perspective on the composition relation specifies control over the sub-components of the component maintenance of requirements and scenarios specification. Task control within maintenance of requirements and scenarios specification specifies which sub-component is activated under which conditions.

7.2 Knowledge composition of maintenance of requirements and scenarios specification

The information types described in the interfaces of the component maintenance of requirements and scenarios specification and its direct sub-components have been described in Section 4.2.

8 Composition of Manipulation of Requirements

The composition of manipulation of scenarios is similar to the composition of manipulation of requirements. The difference lies it the subject of manipulation: scenarios; this distinction is reflected in the names of the sub-processes of manipulation of scenarios and in the names of information types related to these sub-processes.

8.1 Process composition of manipulation of requirements

The first level of process abstraction within manipulation of requirements is shown in Figure 12. The processes reformulation of requirements, validation of requirements, detection of ambiguous and non-fully supported requirements, detection of inconsistent requirements, and identification of functional clusters of requirements are distinguished within the process manipulation of requirements.

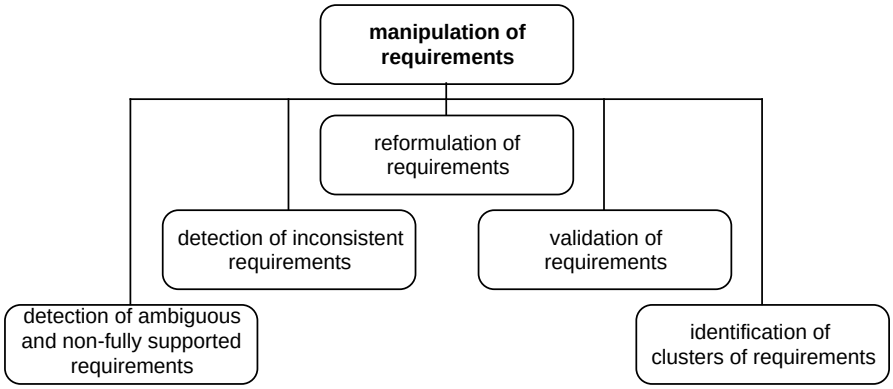

Figure 12. Processes at different abstraction levels in process manipulation of requirements.

The process detection of ambiguous and non-fully supported requirements analyses the requirements for ambiguities and the extent of non-supportedness of requirements by stakeholders. The process detection of inconsistent requirements analyses the requirements for inconsistencies among requirements. The process reformulation of requirements plays an important role within manipulation of requirements: problematic requirements are reformulated into less problematic requirements by adding more and more structure to requirements: from informal to semi-formal to formal. The process validation of requirements has interaction with stakeholders to establish the supportedness of a requirement in relation to a stakeholder, and whether pro and con arguments exist for a requirement. The process identification of clusters of requirements identifies clusters of requirements on the basis of clustering criteria.

The process manipulation of scenarios has a structure similar to manipulation of requirements.

Each of the processes depicted in Figure 12 can be characterized in terms of their interface information types, as shown in Table 7.

process	*input information type*	*output information type*
detection of ambiguous and non-fully supported requirements	• elicited requirements • current requirements • validated requirements information • non-formalisable requirements	• ambiguity information • unsupportedness information
detection of inconsistent requirements	• current requirements • relations among requirements	• inconsistency information
reformulation of requirements	• elicited requirements • ambiguity information • inconsistency information • validated requirements information • isolated scenarios	• current requirements • relations among requirements • requirement alternatives • non-formalisable requirements
validation of requirements	• requirement alternatives • unsupportedness information	• validated requirements information
identification of clusters of requirements	• current requirements • relations among requirements	• clusters of requirements

Table 7. Interface information types of processes within manipulation of requirements.

The input and output information types in the interfaces of the processes described in Table 7 are elaborated below.

- The process detection of ambiguous and non-fully supported requirements requires as input the following information types: requirements elicited from stakeholders (elicited requirements), the current requirements (current requirements), validations of requirements (validated requirements information), and indications of which requirements are non-formalisable (non-formalisable requirements). The process outputs requirements which have an ambiguity (ambiguity information) or are non fully supported by the stakeholders (unsupportedness information) .
- The process detection of inconsistent requirements requires as input the current requirements (current requirements), and relations among these requirements (relations among requirements). The output of the process consists of groups of requirements which together are inconsistent (inconsistency information).

- The process reformulation of requirements requires one or more of the following information types: groups of requirements which have an ambiguity (ambiguity information), groups of requirements which have an inconsistency (inconsistency information), a validation of the requirements (validated requirements information), and isolated requirements and scenarios (isolation information). There are two possible reasons for a requirement to be isolated. The first is that a scenario still has to be formulated for this requirement, the second is that the requirement itself is not correct: it has to be removed or strongly reformulated. Isolated requirements are input of reformulation of requirements to validate them on correctness. Isolated scenarios are input, because on the basis of these scenarios some new requirements may be formulated. The formulation of scenarios for isolated requirements and the validation of isolated scenarios are performed within the process reformulation of scenarios. The process reformulation of requirements produces the current requirements (current requirements), relations among these requirements (relations among requirements), alternative options and trade-offs for requirements (requirement alternatives), and indications of which requirements are non-formalisable (non-formalisable requirements).
- The input of the process validation of requirements consits of alternative options and trade-offs for requirements (requirement alternatives). The process's output consists of validations of requirements in terms of supportedness by stakeholders and arguments pro and con alternatives (validated requirements information).
- The process identification of clusters of requirements requires the current requirements (current requirements), and relations among these requirements (relations among requirements). It produces clusters of requirements (clusters of requirements).

The static perspective on the composition relation between the process manipulation of requirements and its sub-processes is shown in Figure 13.

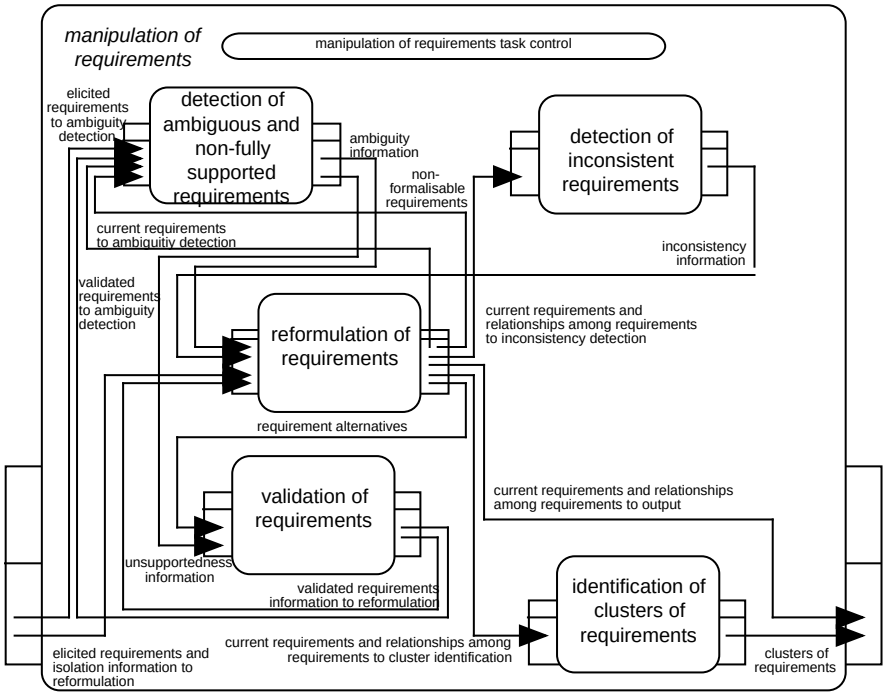

Figure 13. Process composition of manipulation of requirements: information links.

Within the component manipulation of requirements a number of private and mediating information links is distinguished. The names of these information links reflect which information can be exchanged through the information link between the two processes.

The dynamic perspective on the composition relation specifies control over the sub-components of the component manipulation of requirements. Task control within manipulation of requirements specifies which sub-component is activated under which conditions. A sequential description of control over the sub-components is given. Task control with varying degrees of parallelism are also possible, but not described here.

On activation of manipulation of requirements, detection of ambiguous and non-fully supported requirements is activated, and elicited requirements are transferred to that process. On termination of detection of ambiguous and non-fully supported requirements the process reformulation of requirements is activated, and information on ambiguity of requirements is transferred to that process. On termination of reformulation of requirements a number of conditions exist, which may result in parallel activation of sub-components:

- After termination of detection of ambiguous and non-fully supported requirements, and resolution of ambiguities, if any, by reformulation of requirements, detection of inconsistencies is activated.
- If requirement alternatives are produced, then validation of requirements is activated.
- If reformulation of requirements is considered to have produced interesting results, then detection of ambiguous and non-fully supported requirements is activated.
- If reformulation of requirements is considered to be finished, then identification of clusters of requirements is activated.

On termination of detection of inconsistencies, reformulation of requirements is activated. On termination of validation of requirements, reformulation of requirements is activated. On termination of identification of clusters of requirements, manipulation of requirements terminates itself.

8.2 Knowledge composition of manipulation of requirements

The information types described in the interfaces of the component manipulation of requirements and its direct sub-components are briefly described in this section.

The information types ambiguity information, inconsistency information, and non-formalisable requirements express statements about requirements: whether a requirement is ambiguous, whether a group of requirements is inconsistent, and whether a requirement is not formalisable.

The information type validated requirements information is based on two information types: annotated requirements, and critiqued requirements, as shown in Figure 14. The information type annotated requirements contains relations expressing whether a requirement is supported by a stakeholder, or not. The information type critiqued requirements is based on two information types: pro arguments, and con arguments. The information types pro arguments and con arguments contain relations expressing pro and con arguments for or against (respectively) requirement alternatives.

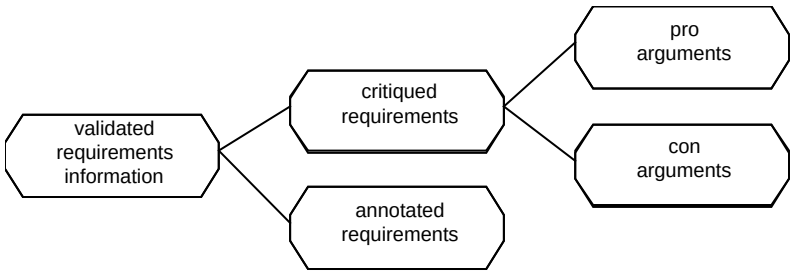

Figure 14. Information type validated requirements information.

The information type requirements alternatives is based on two information types: requirement options, and requirement trade-offs, as shown in Figure 15. The information type requirements options contains relations expressing alternatives for a requirement. The information type requirements trade-offs specifies arguments for and against requirement alternatives; it is based on the information types pro arguments and con arguments.

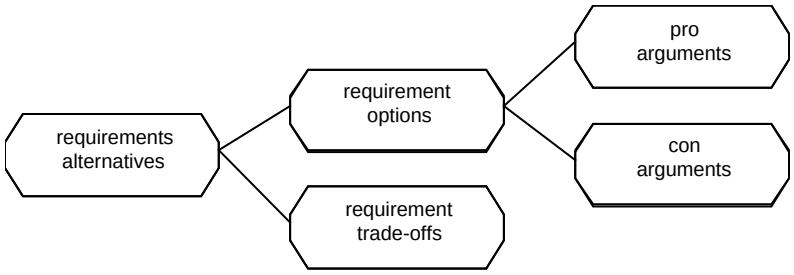

Figure 15. Information type requirements alternatives.

The information type ambiguity information (not shown) contains relations expressing which groups of requirements are ambiguous. The information type inconsistency information contains relations expressing which groups of requirements are inconsistent. The information type non-formalisable requirements contains relations expressing requirements that are not formalisable: either these requirements are informal requirements and cannot be reformulated into semi-formal requirements, or these requirements are semi-formal requirements that cannot be reformulated into formal requirements.

9 Composition of Reformulation of Requirements

In Section 9.1 the process composition of reformulation of requirements is discussed; then, in Section 9.2, the composition of knowledge structures related to this process is discussed.

9.1 Process composition of reformulation of requirements

The first two levels of process abstraction for reformulation of requirements are shown in Figure 16. The processes reformulation into informal requirements, reformulation into semi-formal requirements, and reformulation into formal requirements are distinguished within the process reformulation of requirements.

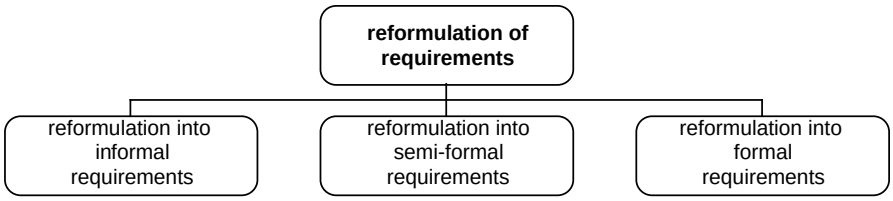

Figure 16. First level of process abstraction within process reformulation of requirements.

The process reformulation into informal requirements reformulates informal requirements in (other) informal requirements. The process reformulation into semi-formal requirements reformulates information and semi-formal requirements into semi-formal requirements. The process reformulation into formal requirements reformulates informal, semi-formal, and formal requirements into formal requirements. All of these reformulation processes keep track of reformulation relations among requirements.

Each of the processes depicted in Figure 16 can be characterized in terms of their interface information types, as shown in Table 8.

process	*input information type*	*output information type*
reformulation into informal requirements	• elicited requirements • ambiguity information • inconsistency information • validated requirements	• informal requirements information

	information	
reformulation into semi-formal requirements	in addition to the above: • informal requirements information	• semi-formal requirements information • non-formalisable requirements
reformulation into formal requirements	in addition to the above: • semi-formal requirements information	• formal requirements information • non-formalisable requirements

Table 8. Input and output information types of the direct sub-processes of the process reformulation of requirements.

The input and output information types in the interface of the process described in Table 8 are elaborated below:

- The process reformulation into informal requirements requires input of one or more of the following information types: groups of requirements which have an ambiguity (ambiguity information), groups of requirements which have an inconsistency (inconsistency information) and a validation of the requirements (validated requirements information). The process outputs information on informal requirements (informal requirements information).
- In addition to the abovementioned input, the process reformulation into semi-formal requirements requires the information type for information on informal requirements (informal requirements information). The process produces output information on semi-formal requirements (semi-formal requirements information), and indications of which requirements are non-formalisable (non-formalisable requirements).
- In addition to the abovementioned input, the process reformulation into formal requirements requires as input the information type for information on semi-formal requirements (semi-formal requirements information). The process produces output information on formal requirements (formal requirements information), and indications of which requirements are non-formalisable (non-formalisable requirements).

The static perspective on the composition relation between the process reformulation of requirements and its direct sub-processes is shown in Figure 17.

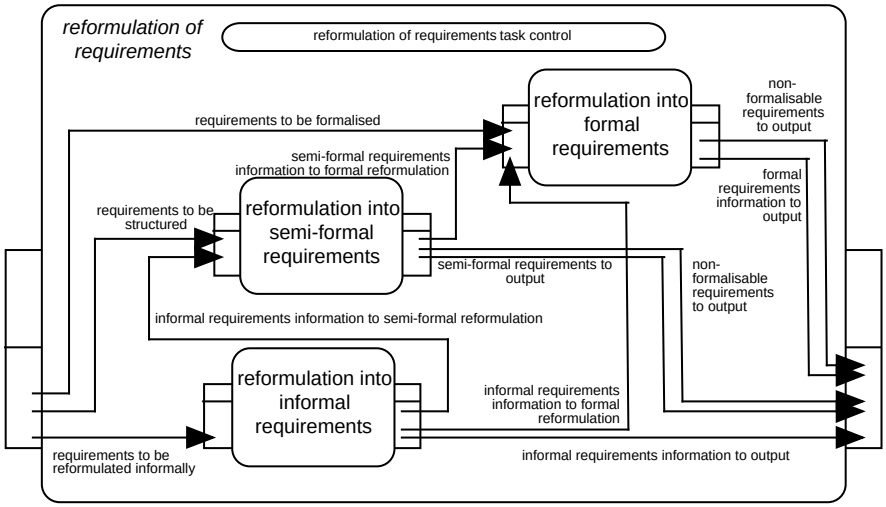

Figure 17. Composition relation between the process of reformulation of requirements and its direct sub-processes.

Within the component reformulation of requirements a number of private and mediating information links is distinguished. The names of these information links reflect which information can be exchanged through the information link between the two processes. The information links requirements to be formalise, requirements to be structured, and requirements to be reformulated informally link the following information types: elicited requirements, ambiguity information, inconsistency information, validated requirements information, and isolation information.

The dynamic perspective on the composition relation specifies control over the sub-components of the component reformulation of requirements. Task control within reformulation of requirements specifies which sub-component is activated under which conditions. Although some dependencies in terms of information flow exist between these sub-components, sequential or more parallel approaches to task control can be equally well employed.

9.2 Knowledge composition of reformulation of requirements

The information types described in the interfaces of the component reformulation of requirements and its direct sub-components are briefly described in this section.

The information types informal requirements information, semi-formal requirements information, and formal requirements information are all constructed in a similar fashion. The information type informal requirements information is based on three information

types: current informal requirements, relations among requirements, and informal requirement alternatives, as shown in Figure 18. The information type semi-formal requirements information is based on three information types: current semi-formal requirements, relations among requirements, and semi-formal requirement alternatives. The information type formal requirements information is based on three information types: current formal requirements, relations among requirements, and formal requirement alternatives.

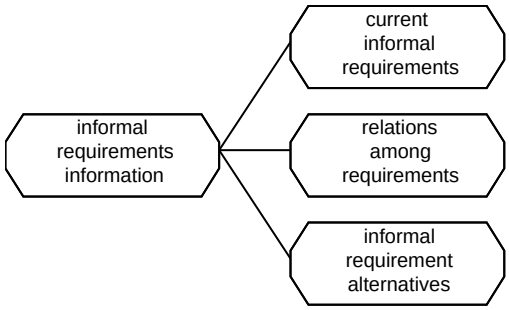

Figure 18. Partial view on information type informal requirements information

10 Discussion

The compositional knowledge modelling method DESIRE has been applied to the task of Requirements Engineering. The resulting compositional process model has been presented in some detail in this paper. The process model has been constructed on the basis of studies of available literature, and a real-life case study in Requirements Engineering: analysis and design of a Personal Internet Assistant (Herlea, Jonker, Treur, and Wijngaards, 1999a). An overview of the overall composition is depicted in Appendix A.

The processes have been described at different levels of process abstraction, with descriptions of their interfaces, a static composition relation specifying possibilities for information exchange, and a dynamic composition relation: 'control flow'. The static composition relation does not prescribe a particular task control through the process composition. The task control is formulated in terms of conditions which trigger particular activities. Some control can be formulated which is generic: irrespective of sequences of activities of specific requirement engineering processes. However, mostly task control will depend on *how* the requirements engineering process is tailored for a

particular product and organisation. This will reflect in, e.g., the amount of flexibility and iterative nature of sub-processes of the requirements engineering process.

The compositional process model presented in this paper has been formally specified and provides more details and structure for the requirements engineering process than process models described in the literature on requirements engineering. For example, in (Kontonya and Sommerville, 1998; Sommerville and Sawyer, 1997) the following activities are considered core activities in the requirements engineering process: 'requirements elicitation', 'requirements analysis and negotiation', 'requirements documentation', and 'requirements validation'. The first three of these core activities form the top level composition of the process model introduced in this paper. In contrast to the references mentioned, in the model introduced here a detailed specialisation of these three main processes is added. In the process model introduced the fourth main activity, 'requirements validation' is considered an integrated part of the manipulation processes both for requirements and scenarios, and is modelled within these processes: detection of inconsistent requirements, detection of inconsistent scenarios, validation of requirements, validation of scenarios.

More extensive processes relating to stakeholders, such as described, for example in (Maiden, Rugg, and Patel, 1999) and (Berztiss, 2000), have not been fully incorporated. If a model is desired where these aspects, such as, for example, stakeholder identification, format selection for requirements gathering (cf., (Berztiss, 2000)), the model will have to be refined for this. The compositional nature of the model supports such refinement processes.

Another aspect not covered in detail is how to assess requirements on cost. If a specific approach for risk analysis involving cost is desired to be part of the model, this also can be added by refining the model.

The compositional process model presented in this paper is a generic process model for Requirements Engineering. It covers many of the process models as described in literature: see above. Due to its compositional structure, the generic process model can easily be refined or modified into a more specific process model for Requirements Engineering, suitable to the situation at hand. If, for example, scenarios are not considered of any importance in a situation, then processes concerning scenarios can be omitted. Likewise, if a particular method is employed to validate requirements, this

method can be added by refining appropriate processes: by instantiation and/or specialisation.

To further investigate the applicability of this compositional process model, additional requirements engineering experiments will be conducted. The formally specified compositional process model for the task of requirements engineering can be employed in the design of automated tools for requirements engineering (e.g., (Dubois, 1998; Dubois, Du Bois, and Zeippen, 1995)), supporting the activities of requirement engineers on the basis of an agreed shared model of the requirements engineering task. In further research the integration of the process model for requirements engineering introduced here, with the design model for compositional systems described in (Brazier, Jonker, Treur and Wijngaards, 1998), will be addressed.

References

Berztiss, A.T. (2000). A flexible requirements process. *Proc. of the 11th Internat. Workshop on Database and Expert Systems Applications*, 2000, pp. 973-977.

Brazier, F.M.T., Dunin-Keplicz, B.M., Jennings, N.R., and Treur, J. (1995;1997). Formal Specification of Multi-Agent Systems: a Real World Case In: Lesser V (ed.) *Proceedings First International Conference on Multi-Agent Systems ICMAS'95* (1995) pp. 25-32 MIT Press. Extended version in: Huhns M and Singh M (eds.) *International Journal of Co-operative Information Systems IJCIS* Vol. 6 No 1 (1997) pp. 67-94, (Special issue on Formal Methods in Co-operative Information Systems: Multi-Agent Systems).

Brazier, F.M.T., Jonker, C.M., and Treur, J. (1998). Principles of Compositional Multi-agent System Development. In: J. Cuena (ed.), *Proceedings of the 15th IFIP World Computer Congress, WCC'98, Conference on Information Technology and Knowledge Systems, IT&KNOWS'98*, 1998, pp. 347-360. To be published by IOS Press.

Brazier, F.M.T., Jonker, C.M., Treur, J. and Wijngaards, N.J.E. (1998). The Role of Abilities of Agents in Re-design. In: Gaines, B. and Musen, M. (eds.). *Proc. of the 11th Knowledge Acquisition Workshop, KAW'98.* Banff. University of Calgary. URL: http://ksi.cpsc.ucalgary.ca/ KAW/KAW98/brazier2.

Brazier, F.M.T., Treur, J., Wijngaards, N.J.E., and Willems, M., (1999). Temporal semantics of compositional task models and problem solving methods. *Data and Knowledge Engineering*, vol. 29(1), 1999, pp. 17-42.

Brazier, F.M.T., and Wijngaards, N.J.E. (1997). An instrument for a purpose driven comparison of modelling frameworks. In: Plaza, E., and Benjamins, R. (eds.). *Proceedings of the 10th European Workshop on Knowledge Acquisition, Modelling, and Management (EKAW'97)*. Sant Feliu de

Guixols, Catalania, Lecture Notes in Artificial Intelligence, vol. 1319, Springer Verlag, 1997, pp. 323-328.

Davis, A. M. (1993). *Software requirements: Objects, Functions, and States*, Prentice Hall, New Jersey, 1993.

Dubois, E. (1998). ALBERT: a Formal Language and its supporting Tools for Requirements Engineering.

Dubois, E., Du Bois, P., and Zeippen, J.M. (1995). A Formal Requirements Engineering Method for Real-Time, Concurrent, and Distributed Systems. In: *Proc. of the Real-Time Systems Conference*, RTS'95, 1995.

Erdmann, M. and Studer, R. (1998). Use-Cases and Scenarios for Developing Knowledge-based Systems. In: *Proc. of the 15th IFIP World Computer Congress, WCC'98, Conference on Information Technologies and Knowledge Systems, IT&KNOWS'98* (J. Cuena, ed.), 1998, pp. 259-272.

Herlea, D., Jonker, C.M., Treur, J. and Wijngaards, N.J.E. (1999). *A Case Study in Requirements Engineering: a Personal Internet Agent*. Technical Report, Vrije Universiteit Amsterdam, Department of Artificial Intelligence, 1999. URL: http://www.cs.vu.nl/~treur/pareqdoc.html

Herlea, D.E., Jonker, C.M., Treur, J., and Wijngaards, N.J.E. (1999*b*) A Formal Knowledge Level Process Model of Requirements Engineering. In: Imam, I., Kodratoff, Y., El-Dessouki, A., and Ali, M. (Eds.). *Multiple approaches to intelligent systems Proceedings of the 12th International Conference on Industrial and Engineering Applications of AI and Expert Systems*, IEA/AIE'99. Lecture Notes in AI, vol. **1611**, Springer Verlag, pp. 869-878.

Herlea, D.E., Jonker, C.M., Treur, J., and Wijngaards, N.J.E. (1999*c*). Integration of Behavioural Requirements Specification within a Knowledge Engineering Methodology. In: D. Fensel, R. Studer (eds.), *Proceedings of the 11th European Workshop on Knowledge Acquisition, Modelling and Management*, EKAW'99. Lecture Notes in AI, Springer Verlag. vol. **1621**, pp. 173-190.

Kontonya, G., and Sommerville, I. (1998). *Requirements Engineering: Processes and Techniques*. John Wiley and Sons, New York, 1998.

Maiden, N.A.M., Rugg, G. Patel, P. (1999). Guidelines for better scenarios: supporting theories and evidence. *Proc. of the 10th Internat. Workshop on Database and Expert Systems Applications*, 1999, pp. 352-356.

Loucipoulos, P. and Karakostas, V. (1995). *System Requirements Engineering*. McGraw-Hill, London, 1995.

Martin, C. (1988). User-centered Requirements Analysis. Prentice Hall, Englewood Cliffs, NJ.

Sommerville, I., and Sawyer P. (1997). *Requirements Engineering: a good practice guide*. John Wiley & Sons, Chicester, England, 1997.

Weidenhaupt, K., Pohl, M., Jarke, M. and Haumer, P. (1998). Scenarios in system development: current practice. In *IEEE Software*, pp. 34-45, March/April, 1998.

Yadav, S., Bravoco, R., Chatfield, A., and Rajkumar, T.M. (1988). Comparison of analysis techniques for information requirements determination. *Communications of the ACM*, vol. **31**(9), pp. 1090-1097, 1988.

Appendix A Overview of all components of the model

In this table, which provides a global overview of the model, it is indicated in which section which composed process is described.

Section	*Processes*
4	requirements engineering
5	1 elicitation
	1.1 problem analysis
	1.2 elicitation of requirements and scenarios
	1.3 acquisition of domain ontology and knowledge
6	2 manipulation of requirements and scenarios
8	2.1 manipulation of requirements
	2.1.1 detection of ambiguous and non-fully supported requirements
	2.1.2 detection of inconsistent requirements
9	2.1.3 reformulation of requirements
	2.1.3.1 reformulation into informal requirements
	2.1.3.2 reformulation into semi-formal requirements
	2.1.3.3 reformulation into formal requirements.
	2.1.4 validation of requirements
	2.1.5 identification of clusters of requirements
	2.2 manipulation of scenarios
	2.2.1 detection of ambiguous and non-fully supported scenarios
	2.2.2 detection of inconsistent scenarios
	2.2.3 reformulation of scenarios
	2.2.3.1 reformulation into informal scenarios
	2.2.3.2 reformulation into semi-formal scenarios
	2.2.3.3 reformulation into formal scenarios
	2.2.4 validation of scenarios
	2.2.5 identification of clusters of scenarios

	2.3 identification of relationships between requirements and scenarios
7	3 maintenance of requirements and scenarios specification
	3.1 maintenance of requirement and scenario documents
	3.2 maintenance of traceability links