CREWS Report Series 97–05

Modelling Contextual Information about Scenarios

Klaus Pohl and Peter Haumer

RWTH Aachen, Informatik V, Germany

Ahornstraße 55, 52056 Aachen

e-mail: {pohl,haumer}@informatik.rwth-aachen.de

appeared in 'Third International Workshop on Requirements Engineering: Foundation for Software Quality RESFQ',
June 16-17, 1997, Barcelona, Spain

Modelling Contextual Information about Scenarios

Klaus Pohl and Peter Haumer
RWTH-Aachen, Informatik V, Germany
e-mail: {pohl,haumer}@informatik.rwth-aachen.de

Abstract: *Scenario-based approaches have proven useful for requirements elicitation, validation and negotiation. Besides the direct and indirect stated requirements current scenario-based approaches capture also contextual information about the existing or future system, but lack in a systematic support for representing and reasoning about this information.*

Based on a literature survey we define a comprehensive set of concepts needed to represent contextual usage knowledge of scenarios. In contrast to existing approaches, we propose to relate contextual knowledge not only to the whole scenario, but also to the scenario components, e.g. single or sets of interactions between the system and the user of the system. Consequently, we propose two contextual models, a scenario context model (SCM) and an interaction context model (ICM).

1 Introduction

Requirements engineering itself can be understood as a process of *establishing a overall system vision in context* [12]. Thus, RE is not an indirect analysis process, but a mutual learning process in which the various stakeholder involved have to reach an agreement about the requirements for the future system [17].

Scenario-based approaches have proven useful to support the various stakeholders in reaching such an agreement. Due to the grounding of the indirect or direct expressed requirement on concrete examples, e.g. a concrete application (use) case, scenarios help in bridging the gap between the various stakeholders and the requirements engineers, i.e. they support a common understanding of current and future requirements. Moreover, they provide ideal means to elicit and validate the stakeholders needs as well as to get aware of their knowledge about the current system and the context the system is going to operate in (e.g. [20; 10; 6; 27]).

Although the need for expressing contextual knowledge (e.g. organizational information, social settings, goals) in scenarios is widely recognized, e.g. [14; 16], a comprehensive approach for expressing contextual knowledge and relating this knowledge to the requirements expressed in the scenarios is missing.

In this paper we propose to model contextual knowledge at two levels: at the scenario level and at the interaction (task) level. We first provide a structure for the context of information systems (Sect. 2) and discuss the concepts and their relations used to represent contextual knowledge in current scenario-based approaches (Sect. 3). We then define a conceptual model for representing the contextual information about a whole scenario, called *scenario context model*, and a more fine-grained model for capturing the contextual information about the interactions expressed within a scenario, called *interaction context model* (Sect. 4). We finally discuss their relations and provide an outlook on future work (Sect. 5).

2 Structuring the Context of Information Systems

2.1 Three Types of Scenario-Approaches

Each system is embedded in some context. To express the degree to which a scenario addresses this context we first distinguish between three main types of scenarios (Fig. 1):

A. *System internal scenarios* focus mainly on the system itself, i.e. the scenario does not consider the context the system is embedded in. Such scenarios are for example used to represent interactions between system objects, e.g. object method calls:

B. *Interaction scenarios* represent knowledge about the interaction of the system with its context. This may include interactions with stakeholders and/or other systems. A common notation used to represent such scenarios are message sequence (trace) diagrams;

C. *Contextual scenarios* represent, in addition to the direct interactions between the system and its context, also information about the context of the system. For example business goals are stated and related to the services provided by a system, system external relations of stakeholders are represented, the use of the information obtained by a service of the system may be expressed, or organizational policies may be stated.

The differentiation between type B and C scenarios corresponds to the classification proposed by Kyng [14], who calls type C scenarios *rich* scenarios, and type B scenarios *narrow* scenarios.

The classification of 20 scenario-based approaches using to the three categories introduced above is depicted in Fig. 2. As the figure indicates, all scenario-based approaches consider knowledge about system-

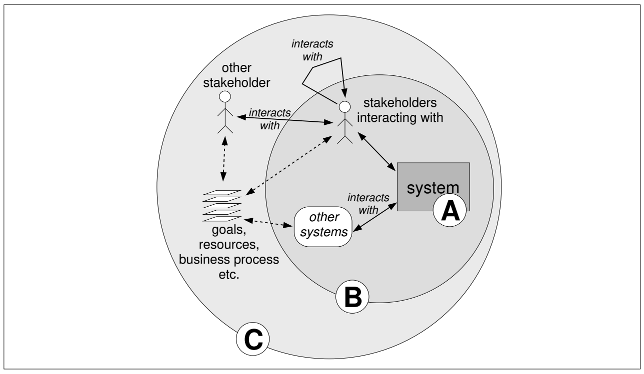

Fig. 1 Three types of scenarios

user or/and system-system interactions and thus fall under category B. In addition, more than half of the classified approaches somehow model information about the context the system is going to operate in whereas only few approaches propose to model system internal interactions.

Before discussing the concepts used by the various approaches for defining contextual information in detail, we provide a structure for the context of information systems and discuss the classification of scenarios along these structure.

Approach	**(A) System Internal Scenario**	**(B) Interaction Scenario**	**(C) Contextual Scenario**
Armour et al. [1]	x	x	x
Bäumer et al. [2]		x	x
Benner et al. [3]	x	x	x
Bertreaud et al. [4]		x	
Cockburn [5]		x	x
Glinz [6]		x	
Gough et al. [7]		x	
Holbrook [8]		x	x
Hsia et al. [9]		x	
Jacobson et al. [10]		x	
Koskimies et al. [13]		x	
Kyng [15]	x	x	x
Leite et al. [16]		x	x
Potts et al. [20]	x	x	x
Rawthorne [21]	x	x	x
Regnell et al. [22]	x	x	x
Rubin et al. [23]		x	
Rumbaugh [24]		x	
Scalzo [25]	x	x	x
Some et al. [26]	x	x	

Fig. 2 Classification of existing scenario approaches.

2.2 *Structuring the Context of Information Systems*

Each type of system has its own typical context. For example, in the case of information systems the output is typically used by humans to fulfill a task better whereas in the case of embedded systems the output of the software system is used by other systems, e.g. in an airplane the output of the navigation system is used by the auto pilot system to adjust the course. Due to these differences the context depends on the type of system to be built. In the following we focus on information systems which are increasingly becoming an integral part of our everyday lives.

An information system can be described in analogy to a sharable telescope through which a user community observes a domain of interest more effectively than without it [11]. The domain of interest may or may not overlap with the user community itself and it may or may not be changeable by the user community. But it makes sense to distinguish, from a cognitive as well as a social viewpoint, between the *usage* world, the *subject* domain world, and the *system* world, and to describe their relationships (see [12; 18] for details). A fourth world, the *development* world, has the basic task of assisting the vision holder in realizing the vision in the context of the other worlds. In addition, the development world must consider its internal development context of people, methods, experiences, and tools. It is the development world in which the RE process takes place.

Thus, the context of an information system can be structured by the usage, subject, and system world. In the following we discuss the contextual information which might be represented for the system, subject, and usage world in more detail.

Context information about the usage world: The usage world comprises stakeholders who are owners and direct/indirect users of the system. The relationships between users and owners can vary widely, but might be defined in the organizational structure [18]. Scenarios which capture the interaction between the users and the system fall under category B. If in addition to the direct user interactions also information about the relations between the users, their role, the reasons (goals) for interacting with the system etc. are stated, the scenario belongs to category C.

Context information about the system world: The system to be built may have relations to the other systems, or the development of the system may be constraint by, e.g. standards or the technical infrastructure. For example, the system may exchange data with other systems or may receive events from other systems. A scenario can deal with this interactions (relations), similar to the user interactions, at both levels (B or C). A scenario which only represents the data, events exchange with other systems fall under category B. If the interactions between other systems and their consequences are considered the scenario will be ranked in category C. Similar, if the restrictions of the infrastructure or the standard to be followed are defined, the scenario will be ranked as category B scenario. If the rationale behind choosing a particular standard or infrastructure are expressed, the scenario may be ranked as category C scenario. Similar, if the new system requires a change in, e.g. the technical infrastructure, and the influences of this change are expressed in the scenario, the scenario will fall under category C.

Context information about the subject world: In addition to the above, a scenario may focus on the relations to the subject world. A regular information system is intended to maintain information about some subjects of the real world. A scenario will be classified as category B scenario, if the direct relation to these subjects is considered within the scenario, e.g. information update (the way how changes of the subjects are communicated to the system) or the correctness of representation (e.g. by checking the represented information periodically with a domain expert). If relations between different subjects are captured by the scenario, it will fall under category C.

3 Representing Contextual Information: State of the Art

3.1 Concepts used for Representing Contextual Information

In this section we briefly describe the concepts used in existing scenario approaches (classified as category C) for representing contextual knowledge. Existing approaches typically concentrate on the representation of contextual knowledge about the usage world. Contextual information about the system world is rarely represented. Contextual information about the subject world is almost neglected.

Fig. 3 depicts the most common concepts used by existing approaches for representing contextual information:

Agents/Users: In most approaches an agent can be a human or a system. The concept of agent is mostly used to represent system-user and/or system-system interaction [6; 10; 9; 13; 23]. In addition, some approaches [3; 5; 8; 20; 25] propose to represent also interactions between the agents and between the agents and other stakeholders and/or systems which are not directly interacting with the system to be build.

Roles or Actors or user types represent classes of users. They are used to classify users or other systems interacting with the system with respect to their roles they play or the responsibilities they have [1; 2; 5; 16; 20; 22].

Goals: For representing the reasons why a particular action is performed within a scenario the concept of goal is used. In most approaches goals are related to the whole scenario (e.g. [16]). In addition, some approaches propose to relate goals to the agents involved in a scenario (e.g. [22] or even to a subset of the scenario [5]).

Location: This concept is used to represent that the execution of a scenario is bound to some geographical location, e.g. a particular kind of office[5; 16; 25]. Mostly a location is identified by a name which might be further defined in some kind of dictionary [16].

Resources: The concept of resources is used to represent other systems or agents, information, money, time, or any other material/product which are crucial for the performance of a scenario [2; 16]. In addition, some approaches propose to model the providers of resources [25].

Pre-/Postconditions: Preconditions are used to describe the conditions to initiate the execution of a given scenario [4; 7; 16; 22; 23; 26]. Postconditions are used to indicate the states of the system and the environment after scenario termination [4; 7; 22; 23]. Some approaches propose to express the pre- and postconditions as object states of agents and/or resources [23]. Others represent pre- and postconditions using

Approach	**Agent/ User**	**Org. Role**	**Goals**	**Location**	**Resource**	**pre-cond.**	**post-cond.**
Armour et al. [1]		X				X	X
Bäumer et al. [2]		X					
Benner et al. [3]	X						
Cockburn [5]		X	X	X		X	X
Holbrook [8]	X		X				
Kyng [15]							
Leite et al. [16]		X	X	X	X	X	
Potts et al. [20]	X	X					
Rawthorne [21]	X		X				
Regnell et al. [22]	X	X	X			X	X
Scalzo [25]	X		X	X	X	X	X

Fig. 3 Concepts used to represent contextual knowledge.

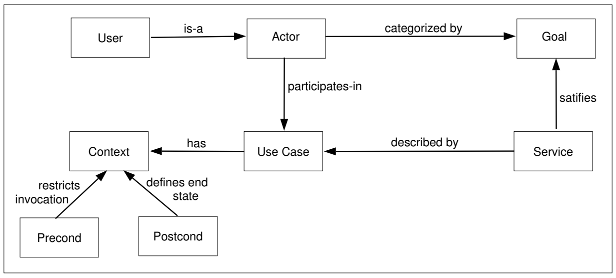

Fig. 4 Conceptual model for representing conceptual information of Regnell et al. [22].

prosa text [7; 25] or using a restricted grammar [16].

3.2 Modelling Contextual Knowledge: Two Examples

In the following we present two “comprehensive” approaches which explicitly define a set of concepts and relations needed to represent contextual knowledge about scenarios.

3.2.1 Capturing contextual knowledge about use cases (Regnell et al.[22])

Regnell et al. [22] define scenarios as instances of use cases. A use case describes a usage situation of the system. In addition to other use case approaches, they propose a set of concepts and relations for describing the usage context of use cases. A conceptual model of the proposed concepts is depicted in Fig. 4.

The concept *context* is used to express *pre- and postconditions* which constrain the scope of the use case. A precondition is defined as properties of the system (target system) and its environment (host system) that need to be fulfilled in order to invoke the use case. A postcondition describes the state of the host and target system after the use case has terminated.

A *user* belongs to the environment the system is going to operate in. A user can be either a human or another software and/or hardware system.

A *service* is described by a set of use cases. It subsumes a package of functional entities (features) the environment (user) expects from the system. A service is invoked by an actor to satisfy a goal or a set of goals the actor wants to achieve (the actor has).

An *actor* (or user type) represents a set of users that have some common characteristics with respects of why and how they use the system (set of services).

Goals are used to categorize users into actors, i.e. an actor (user type) is defined by the goals s/he is related to. In addition, the goals represent the objectives an actor wants to achieve when requesting a certain service.

To model this information they propose a so-called environment model where the actors and services are defined and related to the use cases. Although the need for representing goals and their relations to actors and services was stated, these relations are not discussed in detailed. Consequently no concepts for representing goals and their relations are provided in the environment model. Similar, no concepts for representing pre- and postconditions are provided in the proposed environment model.

3.2.2 Capturing contextual knowledge about scenarios (Leite et al.[16])

Similar to Regnell et al. [22] the concept of *actor* is used to define a person or an organization structure that has a role in the scenario. In contrast to Regnell et al. no differentiation between actor (user type) and a concrete user is made.

The concept of *goal* is used to describe an objective to be achieved in the macrosystem, e.g. business goal, personal goal etc. The scenario itself defines how the stated goal can be achieved. In contrast to Regnell et al. a goal is only related to a scenario but not to the actor (user type). Whereas Regnell et al. propose to categorize actors based on the set of goals which are associated to the actor, the categorization of actors in Leite et al. is an open questions.

In addition to Regnell et al. Leite et al. propose to model the *resources* required for performing a scenario. The availability of a certain resource can be restricted by a textual constraint.

The concept *context* is used to define the geographical location (setting) of a scenario and to express the required initial state (precondition). In comparison with Regnell et al. no postconditions are represented but, in addition, the definition of the geographical location is requested. A location is defined by a name which can be further described in a dictionary. The concepts of location is used to express that the scenario can only be performed in a given (geographical) location, e.g. in the office of the secretary of the vice president.

To describe the proposed contextual usage information, Leite et al. sug-

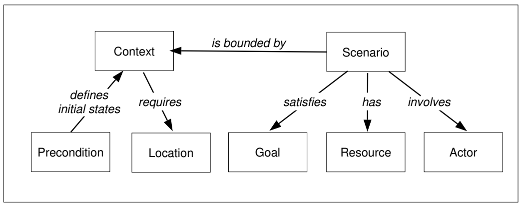

Fig. 5 Conceptual model for representing contextual information of Leite et al. [16].

gest to use structured text which is, in comparison with the environmental model proposed by Regnell et al., more expressive, but, on the other hand, lacks a graphical notation.

4 The Scenario Context Model and Interaction Context Model

We first discuss the needs to represent contextual knowledge at two levels: the scenario and the interaction level (Sect. 4.1). We then propose a conceptual model for representing contextual information at the scenario-level which subsumes the concepts suggested for representing contextual information in other scenario-based approaches and allows the stakeholders to specify the context of complete scenarios (Sect. 4.2).

The more fine grained model for representing contextual information for the interactions captured within a scenario is described in section 4.3. This model enables the stakeholders to express contextual knowledge about parts of a scenario and thus supports the understanding of a particular (sequence of) interaction(s). Finally we discuss the relations between the scenario context and the interaction context models (Sect. 4.4).

4.1 Representing Contextual Information

Most scenario-based approaches relate the contextual information to a top level concept, e.g. scenario (e.g. [16]), use case or service (e.g. [22]). Although most of them provide concepts for structuring scenarios (e.g. episodes, interactions, events, views on message sequence diagrams, ...) they do not provide any means for relating the contextual information presented at the top level to lower level constructs.

This is a bit surprising since the contextual information expressed at the scenario level is mostly build up from more detailed description or observations. For example, an agent related to a scenario is normally involved in an interaction with the system or performs some kind of task within the scenario. Similar, each role (actor, user type) defined for a scenario appears in one or more interactions or other statements within the scenario. Furthermore, the resources required to perform a scenario should be equal with the set of resources required for performing the interactions and tasks defined in the scenario. In other words, the contextual information of a scenario is typically defined by collecting the information about the context expressed in the scenario description or by observing the interactions and tasks subsumed by the scenario. Briefly, the contextual information related to a scenario subsumes the contextual information expressed for the various tasks and interactions the scenario consists of.

Representing contextual information on the lower level (e.g. for each interaction and/or task performed) enables the stakeholders to understand why a certain interaction or a set of interactions should be performed in the expressed manner or why the performance of a particular task is important for the organization. Thus, contextual knowledge should also be expressed at a lower level, i.e. for the interactions and tasks performed.

As a consequence we propose to defined contextual information at two levels, namely at the scenario (or use case) level and at the interaction (task) level.

4.2 Scenario Context Model

The *Scenario Context Model* (SCM) defines the concepts and their relations needed to capture and structure contextual knowledge of a scenario. The model is depicted in Fig. 6. The concepts and their relations are described below.

Preconditions: As in most approaches, preconditions are used to restrict the invocations of a scenario. They state the prerequisites for a scenario. A precondition may be defined in textual form or may be expressed as a constraint about the other concepts introduced in the SCM. For example, a precondition can express that a certain type of resource must be available.

Postconditions: A postcondition is used to express the result of a scenario,

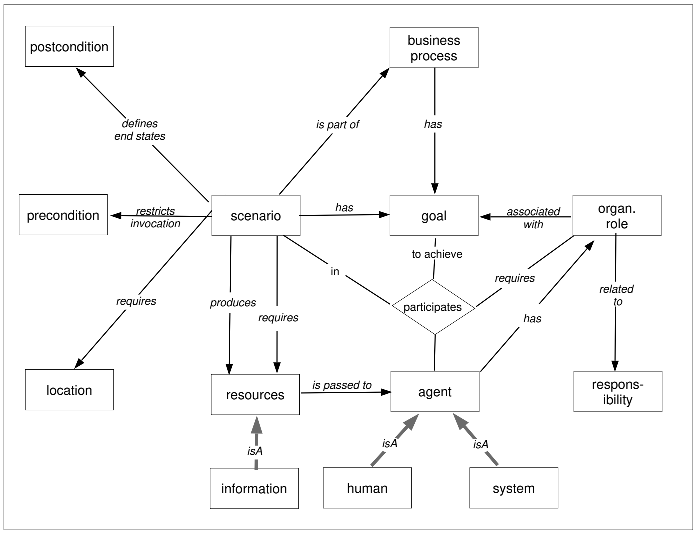

Fig. 6 The scenario context model

again in textual form, or/and the states of the captured context of the scenario and/or the system itself. For example, a postcondition can name the products (information) produced and/or the state a certain agent is in after the scenario has terminated.

Location: Similar to Leite et al. a location expresses geographical constraints about the scenario.

Resources: As in other approaches, the concepts of resources is used to represent information, money, time, other systems or agents, or any other material/product which are crucial for the performance of a scenario. In addition to other approaches we propose to model the resources (information) produced and required in a scenario as well as the fact that a resource (information) was passed to an agent which must not be involved in the scenario itself. This enables us to represent the use of resources outside the scenario. Among others, such contextual knowledge is useful if the scenario is going to be changed or if the produced information (resource) is not required anymore.

Agent: As in most approaches an agent can be a human or another system. An agent can play different (organizational) roles within a scenario. The *participates* relationship expresses the involvement of an agent in a scenario, states the goals to be achieved and the organizational roles required for a particular participation. An agent need not to have a participates relationship. Thus agents which are not directly involved in the scenario can be represented, e.g. agents which obtain information produced in the scenario from an agent participating in the scenario.

Organizational role: We propose to explicitly define the roles an agent has within an organization instead of defining a set of user types or actors (e.g. like in [22]) to differentiate between the various roles of a system user. In addition we suggest to relate the organization roles to the goals which must be achieved and to the responsibilities which are associated with the role. Of course an organizational role can be associated to more than one agent. Since for most organizations an organizational structure exists, we think it is much more effective to reuse the concepts defined in these structure instead of defining new ones. In addition, misunderstandings are avoided since an existing terminology is used.

Goal: Similar to most other approaches we suggest to represent the overall goal of a scenario. In addition we propose to represent, via the *participates* relation, the goals a particular agent has and the interactions by which s/he tries to satisfy these goals. Moreover, an organizational roles can be associated with a set of goals. Thus, if an agent is assigned to an organization role (via the *has role* relationship), the agent has to achieve the goals related to the role.

Responsibilities are defined for each organizational roles. By assigning an organizational role to a certain agent (via the *has role* relationship), the agent automatically “inherits” the responsibilities related to the role.

Business process: We suggest to explicitly relate the scenario to the business process (or set of business process) it contributes to. Thereby the role of the scenario within the usage world can be represented.

4.3 Interaction Context Model

The *Interaction Context Model* (ICM) defines the concepts and their relations needed to capture and structure contextual knowledge about the interactions represented in a scenario. To be able to relate contextual information to the interaction one has to assume some kind of representations for the interactions.

We have decided to base our contextual interaction model on the widely used message sequence (trace) diagrams (MSDs) which are used to represent the interaction between the (target) system and its environment. An interaction is represented as simple object association between the *system* and *agents* (see Fig. 7). An agent (human or other system) can either initiate an agent-system interaction by requesting a particular information or task of the system. Vice versa, the system can request an interaction with an agent

For representing contextual information about the interactions we suggest to extend MSDs by the concepts described below (see Fig. 7).

Task: When asking a stakeholder to describe interactions with the system he typically explains the tasks (part of a business process) he has to perform. To capture a broader context of the scenario at hand we suggest to represent the *business tasks* the agent performs. In addition, each interaction the agent is involved in should be related to the business task for which it was performed (part-of relation between agent/system interaction and business task; not depicted in Fig. 7).

Goal: The interactions and tasks performed in a scenario serve a specific purposes, i.e. the interaction are performed to attain a goal the agent has. To be able to understand the purpose of an interaction we suggest to relate each interaction to the goal which should be achieved by the interaction. Similar, each task should be related to a higher level goal the agent wants to achieve by performing the task.

Resource: To be able to define the resources required for a scenario on the

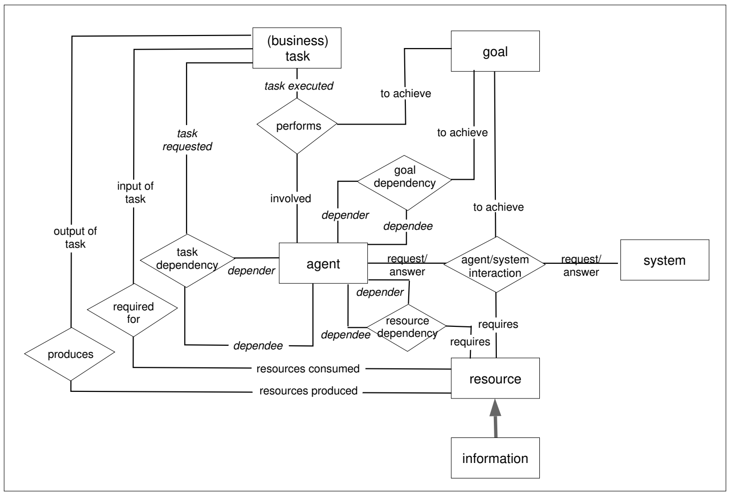

Fig. 7 The interaction (task) context model

top level, we suggest to model the *resources required for the agent/system* interaction. In the case of an information system the resources used and produced within a scenario are most often information or data. Thus, for each agent/system or system/agent interaction the data/information exchanged has to be defined. In addition, the fact that a task executed in a scenario *produces* resources (information) or *requires* resources has to be represented (see Fig. 7).

Agent/Agent interaction: Interactions between external agents and agents interacting with the system are quite common. An agent may pass information obtained by the system to another agent, or request the performance of a task by an external agent. Moreover, an agent may receive the information required for a system interaction from an external agent. The interaction which we have observed in concrete scenarios can be classified using the three types of agent dependencies proposed by the I* model [28; 29]. *Goal dependency:* In a *goal dependency* an agent depends on another agent to make a condition in the world come true. A goal dependency is used if the agent who has received the request can decide how to fulfill the request. In a *task dependency*, an agent depends on another agent to perform a task. In contrast to a goal dependency, in a task dependency the requesting agent specifies a particular course of actions which describe the task. In a *resource dependency* an agent depends on another agent for the availability of an resource, e.g. a particular information required for an interaction (see Fig. 7).

To ease the modelling task we propose to introduce a *view* concept which aggregates a set of agent-system and/or system-agent interactions and relate the contextual concepts defined above to this view concept in the same way they are related to single interactions (not shown in Fig. 7). Through this extension it can be represented that, e.g., a particular goal is achieved through a sequence of actions or that a task requires the performance of a particular set of interactions.

For representing a particular goal, task, resource, agent, etc. we propose to relate each of the concepts and relations defined above to a hypertext node. By specializing our hypertext model and tool described in [19] according to the above defined conceptual model, knowledge base support for navigating in the hypertext and for enabling selective retrieval of model components can be achieved.

The representation of such contextual knowledge at the interaction level provides an invaluable source for representing the context of the whole scenario. Most of the contextual information for the interactions (or sequence of interaction) is easy to access (e.g. can be observed) and thus the contextual interaction information provides a solid basis for defining the context information of the whole scenario; it is much more difficult to express the context of a scenario without such a basis.

In the next section we discuss the relations of the interaction context model and the scenario context model and describe how a scenario context model can be defined based on the context interaction (sequences) model.

4.4 Relations of the SCM and ICM Models

First of all a ICM model must be related to the corresponding SCM model using a consist-of relation which subsumes the set of interactions which are described in a given scenario. In addition, a set of constraints can be defined between the SCM model and the associated set of interactions:

Agent: The agents defined in the SCM model should subsume the agents defined in the ICM model. If an agent is involved in an agent/system or system/agent interaction at the ICM level, a *participates* relation should be defined at the SCM level. Vice versa, external agents, i.e. agents which do not directly interact with the system, must not have a participates relationship at the SCM level.

Goal: In the SCM model agents are associated to goals through their organizational roles and via the *participates* relationship. In the ICM model goals are stated for the tasks performed and the interactions. The relations between the goals expressed at the ICM level and the SCM level typically differ in their granularity. We thus propose to introduce a *subgoal* relationship for representing that a given goal is a subgoal of another one. The subgoal relations have to be defined by the requirements engineer. However, on could request that each goal represented at the ICM level is either part of the set of goals defined at the SCM level or related to a goal at the SCM level via a subgoal relation.

Resource: There are three types of resources: (1) resources which are required at the beginning of the scenario; (2) resources which are produced by the scenario, i.e. which are available after termination; and (3) resources which are used and consumed by the scenario. The resources defined at the ICM level have to be classified according to this three categories at the SCM level. Whereas resources of the type (1) are expressed in the precondition of the scenario, resources of the type (2) are expressed in postcondition. All three types are subsumed by the concept resource at the SCM level.

Business process: In the ICM model business tasks are defined. In the SCM model a scenario is related to business processes. If a business process model exist it can be required that for each task represented at the ICM level the corresponding business process is defined at the SCM level. If there is no business process model either the tasks can be related to the scenarios at the SCM level or one has to identify and define the business processes the tasks are part of.

5 Conclusions

It is widely accepted that representing contextual information about scenarios is important for the understanding and the correct use of the scenarios during the system development process. To access the degree to which contextual information is represented in existing scenario-based approaches we distinguished between three types of scenarios (system internal, interaction and contextual scenarios) and classified 20 existing scenario-based approaches using the three categories.

Based on a more detailed analysis of the scenarios classified as *contextual scenarios* we discussed the concepts and relations provided by those approaches for representing contextual information. Given the four worlds of information systems it is interesting to observe that current approaches mainly focus on the usage world and pay less attention to the subject and system worlds.

In contrast to all existing scenario-based approaches we argued that contextual information has to be represented at two levels, namely at the scenario and at the interaction (task) level. Consequently, we proposed two conceptual models: The *scenario context model* subsumes the concepts and relations for representing contextual information used by existing scenario-based approaches and allows the stakeholders to specify the context of complete scenarios. The *interaction context model* enables the stakeholders to express contextual knowledge about parts of a scenario to specifically support the understanding of these parts for stakeholders. A part can be an individual interaction of a scenario or a particular set of interactions grouped together using the View concept. Views can be used to introduce several levels of granularity for contextual information related to scenarios (e.g., goals for a set of interactions, sub-goals related to sub-sets of this set, sub-goals related to individual interactions of the sub-sets).

Finally, we outlined a first set of rules and constraints for ensuring consistency between scenario context and interaction context models.

Future research is concerned with an elaboration of those rules and the development of appropriate tool support. In addition we will extend the proposed models to capture relevant contextual information about the subject and system world.

Acknowledgments The authors like to thank our master student Andreas Kriehn for many fruitful discussions and comments on early versions of our models. This work is founded by the European Community under ESPRIT Reactive Long Term Research 21.903 CREWS.

References

[1] Frank Armour, Lorrie Boyd, and Monica Sood. Use case modeling concepts for large business system development. *OOPSLA '95, Workshop on Use Cases*, 1995.

[2] Dirk Bäumer, Rolf Knoll, Guido Gryczan, and Heinz Züllighoven. Large scale object-oriented software-development in a banking environment – an experience report. In *Proceedings of Object oriented programming: 10th European conference (ECOOP '96), Linz, Austria, Juli 8 – 12*, pages 73–90, 1996.

[3] K.M. Benner, M.S. Feather, W.L. Johnson, and L.A. Zorman. Utilizing scenarios in the software development process. In C. Roland N. Prakash and B. Pernici, editors, *IFIP WG 8.1 Conference on Information System Development Process, Como, Italy*, pages 117–134. Elsevier B.V. (North Holland), 9 1993.

[4] Regis Berteaud and Jean Bezivin. Requirements modeling in the osmosis workbench. *OOPSLA '95, Workshop on Use Cases*, 1995.

[5] Alistair Cockburn. Structuring use cases with goals. Technical report, Human and Technology, 7691 Dell Rd, Salt Lake City, UT 84121, HaT.TR.95.1, http://members.aol.com/acockburn/papers/usecases.htm, 1995.

[6] Martin Glinz. An integrated formal model of scenarios based on statecharts. *Lecture Notes in Computer Science '95*, pages 254–271, 1995.

[7] P.A. Gough, F.T. Fodemski, S.A. Higgins, and S.J. Ray. Scenarios - an industrial case study and hypermedia enhancements. *IEEE Requirements Engineering '95*, pages 10–17, 1995.

[8] Hilliard Holbrook. A scenario-based methodology for conducting requirements elicitation. *ACM SIGSOFT*, 15(1):95–104, 1 1990.

[9] Pei Hsia, Jayjaran Samuel, Jerry Gao, Yasufumi Toyoshima, and Chris Chen. Formal approach to scenario analysis. *IEEE Software*, pages 33–41, 3 1994.

[10] Ivar Jacobson, Magnus Christerson, Patrik Jonsson, and Gunnar Oevergaard. *Object-Oriented Software Engineering: A Use Case Driven Approach*. Addison-Wesley, 1992.

[11] Matthias Jarke. Daida: Conceptual modeling and knowledge-based support of information systems developement processes. *Technique et Science Informatiques*, 9(2):121–133, 1990.

[12] Matthias Jarke and Klaus Pohl. Establishing visions in context: Towards a model of requirements processes. *Proc. of the 14th Int. Conf. on Information Systems, Dec. 5-8, Orlando, Florida, USA*, 1993.

[13] Kai Koskimies, Tatu Männisto, Tarja Systä, and Jyrki Tuomi. On the role of scenarios in object-oriented software design. Technical report, Series of Publications A (A-1996-1), Department of Compuer Science Univerity of Tampere, Finland, 1 1996.

[14] Kari Kuutti. Work processes: Scenarios as a premliminary vocabulary. In John M. Carroll, editor, *Scenario-Based Design: Envisioning Work and Technology in System Development*, pages 19–36. John Wiley and Sons, 1995.

[15] Morten Kyng. Creating contexts for design. In John M. Carroll, editor, *Scenario-Based Design: Envisioning Work and Technology in System Development*, pages 85–107. John Wiley and Sons, 1995.

[16] Julio C.S. do Prado Leite, Gustavo Rossi, Federico Balaguer, Vanesa Maiorana, Gladys Kaplan, Graciela Hadad, and Alejandro Oliveros. Enhancing a requirements baseline with sceanrios. In *Third IEEE International Symposium On Requirements Engineering (RE'97), Annapolis, Maryland*, pages 44–53. IEEE Computer Society Press, 1 1997.

[17] K. Pohl. The three dimensions of requirements engineering: A framework and its application. *Information Systems*, 3(19):243–258, 6 1994.

[18] K. Pohl. *Encyclopedia of Computer cience and Technology*, chapter Requirements Engineering. Marcel Dekker, Inc., N.Y., 1996.

[19] Klaus Pohl and Peter Haumer. Hydra: A hypertext model for structuring informal requirements representations. In Klaus Pohl and Peter Peters, editors, *Proc. of the 2nd Int. Workshop on Requirements Engineering: Foundation of Software Quality (REFSQ 95), Jyvaeskulae, Finnland*, pages 118–134. Augustinus, Aachen, Germany, 1995.

[20] Colin Potts, Kenji Takahashi, and Annie I. Anton. Inquiry-based requirements analysis. *IEEE Software*, 11(2):21–32, 3 1994.

[21] Daniel A. Rawsthorne. Capturing functional requirements through object interactions. In *Proceedings of ICRE '96*, pages 60–67. IEEE, 1996.

[22] Bjoern Regnell, Michael Andersson, and Johan Bergstrand. A hierarchical use case model with graphical representation. In *Proceedings of ECBS'96, IEEE Second International Symposium of Computer-Based Systems*. IEEE, 3 1996.

[23] Kenneth S. Rubin and Adele Goldberg. Object behaviour analysis. *Communications of the ACM*, 35(9):48–62, 9 1992.

[24] J. Rumbaugh. Modelling models and viewing views. *Journal of Object Oriented Programming*, 7(2):14–29, 1994.

[25] Betsy Scalzo. UPROAR - User processes reveal objects and requirements. *OOPSLA '95, Workshop on Use Cases*, 1995.

[26] Stephane Some, Rachida Dssouli, and Jean Vaucher. Toward an automation of requirements engineering using scenarios. *Journal of Computing and Information, Special issue: ICCI'96, 8th International Conference of Computing and Information, Waterloo, Canada*, 2(1):1110–1132, 1996.

[27] Alistair Sutcliffe. A technique combination approach to requirements engineering. In *Third IEEE International Symposium On Requirements Engineering (RE '97), Annapolis, Maryland*, pages 65–74. IEEE Computer Society Press, 1 1997.

[28] E. Yu and J. Mylopoulos. From e-r to a-r – modelling strategic actor relationships for business process reengineering. In *Proceedings of the 13th International Conference on The Entity-Relationship Approach, ER'94, Manchester, UK*, pages 548–565, 12 1994.

[29] Eric S.K. Yu. Towards modelling and reasoning support for early-phase requirements engineering. In *Third IEEE International Symposium On Requirements Engineering (RE '97), Annapolis, Maryland, USA*, pages 226–235. IEEE Computer Society Press, 1 1997.