CREWS Report Series 97-08

Scenario Integration into Requirements Engineering Methods

Jolita Ralyté, Camille Ben Achour

Université Paris 1 Sorbonne
CRI, 17 rue de Tolbiac
75013 Paris
{ralyte,camille}@univ-paris1.fr

Proceedings of the Workshop
on the Many Facets of Process Engineering (MFPE'97),
Gammarth, Tunis, September 22-23, 1997

Scenario Integration into Requirements Engineering Methods [1]

Jolita Ralyté, Camille Ben Achour

Université Paris 1 Sorbonne
CRI, 17 rue de Tolbiac
75013 Paris
{ralyte,camille}@univ-paris1.fr

Abstract :

Scenario Based Requirement Engineering (RE) techniques have for main motivation the opening of Requirement Engineering processes to neophytes. By combining the methodological approach of requirement analysis to informal descriptions, they increase productivity into classical RE techniques. In this context, an important problem is the integration of scenario approaches into RE methods. This paper presents a knowledge base of different scenario chunks which allows to identify the scenarios appropriate to the context of the RE project, and to integrate them into any RE methods. The identification of scenario chunks is partly based on the internal properties of scenarios which are extensively described in a framework for scenario classification.

Keywords : *Requirements Engineering, Scenario, Use case, Scenario classification, Knowledge Base, Situational Method Engineering.*

Résumé :

Les techniques d'Ingénierie des Besoins (RE) basées sur les scénarios visent à faire participer les utilisateurs et les néophytes de la conception aux processus d'Ingénierie des Besoins. En combinant l'approche méthodologique d'élucidation des besoins à des descriptions informelles, ils apportent une plus grande productivité par rapport aux techniques classiques de RE. Dans ce contexte, un problème important est l'intégration des approches par scénario dans les méthodes de RE. Cet article présente une base de connaissances des différents modules de scénario. Elle permet d'identifier des scénarios appropriés au contexte du projet de RE et de les intégrer dans les méthodes de RE. L'identification du module de scénario est partiellement basée sur les propriétés internes des scénarios qui sont décrites dans un cadre de référence pour la classification des approches basées sur les scénarios.

Mots clés : *Ingénierie des besoins, Scénario, Cas d'utilisation, Classification des scénarios, Base de connaissance, Ingénierie situationele des méthodes.*

1. Introduction

The area of method engineering has emerged in response to an increasing feeling that methods are not well-suited to the needs of their users. Methods are often of a generic nature and can't be followed literally, they are adapted from one business situation to another. The situational method engineering discipline aims at defining Information Systems Development methods by reusing and assembling different existing method fragments. This approach allows to construct modular methods which can be modified and augmented to meet the requirements of a given situation. A method is viewed as a collection of method fragments [Rolland 96], [Harmsen 94]. Thus, method fragments are the basic building blocks which allow to define methods in a modular way. In this paper we are interested in specific fragments of method, namely scenario based approaches, that we call *scenario chunks*.

Use of examples, scenes, narrative descriptions of contexts, mock-ups and prototypes have attracted considerable attention in Requirements Engineering (RE), Human Computer Interaction (HCI) and Information Systems (IS) communities. Loosely all these ideas can be called scenario based approaches, although exact definition is not easy beyond saying these approaches emphasise some description of the real world. Within the long term research ESPRIT project CREWS, we have proposed a framework for classifying scenarios as a way to explore the issues underlying scenario based approaches in RE. Applying the framework on several scenario approaches [Rolland 97] proved the existence of the large variety of products and practices of scenarios. This work shows also that one has little understanding about how scenarios should be retrieved, constructed, and appended to different methods.

---

[1] This work is partly funded by the Basic Research Action CREWS (ESPRIT N°21.903). CREWS stands for Cooperative Requirements Engineering With Scenarios.

Our objective is to develop an approach for integrating different kinds of scenarios as method fragments into usual RE methods. Our proposal is to apply two method engineering principles, namely the reuse of scenario chunks and the situational approach to integrate scenario chunks in traditional requirements engineering methods. To achieve this goal has raised three issues : firstly the representation and organisation of scenario chunks in a base of scenarios, secondly the definition of an approach for reusing and retrieving relevant scenario chunks for the situation at hand and thirdly the definition of mechanisms to adapt existing methods for supporting the usage of scenarios in the requirements engineering process.

We propose to organise the base of scenario chunks at two levels : the scenario knowledge level and scenario meta-knowledge level (Figure 1). The scenario knowledge is represented in the scenario base in the form of scenario chunks. These chunks are available at different levels of granularity. The scenario meta-knowledge level stores the information about the situation in which the scenario chunks are useful and supports the search of the scenario chunks in the scenario base. Therefore each scenario chunk is coupled to a description of its context of use and to guidelines to integrate it into existing methods.

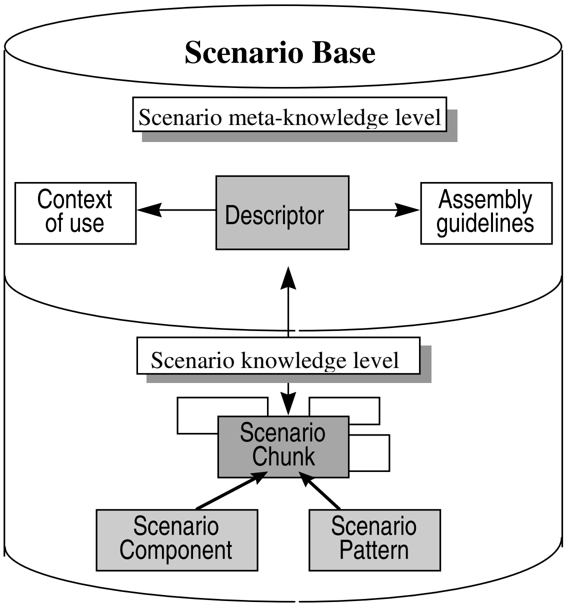

Figure 1 : Organisation of the base of scenario chunks.

In the next section, we briefly present the scenario knowledge level. Section 3 introduces the scenario meta-knowledge level and explains how to select a necessary scenario chunk according to the RE method. Finally, conclusions are drawn in section 4.

2. Scenario knowledge Level

The scenario knowledge is contained in the scenario chunks corresponding to the different chunks of scenario based approaches like the use cases proposed in the OOSE methodology [Jacobson 92], scenario scripts [Potts 94], interaction diagrams [Rumbaugh 96], abstract usage views [Regnell 95], etc.

In the scenario knowledge level, we view scenario chunks as artefacts with a product part and a process part. The product part includes products or subparts of the products to be delivered by a scenario chunk. The process part represents the stages, activities and tasks to be carried out in order to produce the product. The descriptions of the product and process parts are respectively represented by a product and a process meta-model. Figure 2 and Figure 3 present elements of the product and process meta-models for Potts et al. scenario chunk [Potts 94], described with the notation used in [Rolland 94] and [Tempora 94].

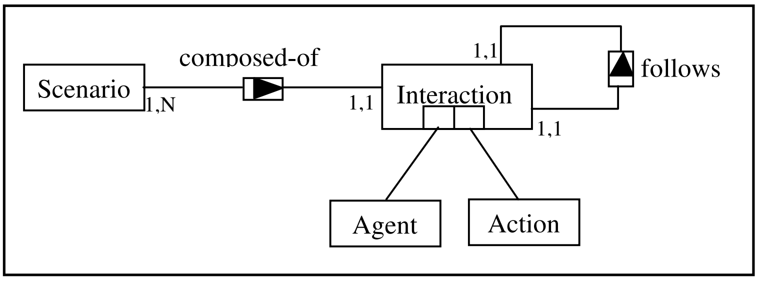

Figure 2 : Fragment of the product model of the scenario chunk based on the Potts' approach [Potts 94].

Whereas most often the scenario approaches are rather centred on the product, we consider that the process dimension is necessary to propose a scenario base in which scenario chunks are related to methodological concerns such as motivations, situations they capture, alternative ways of working and so on. Moreover scenario approaches seldom use an explicit model to describe scenarios. Therefore, to integrate adequately scenario chunks in RE methods, each model is retrieved, explicitly described, and added to our knowledge base of scenarios.

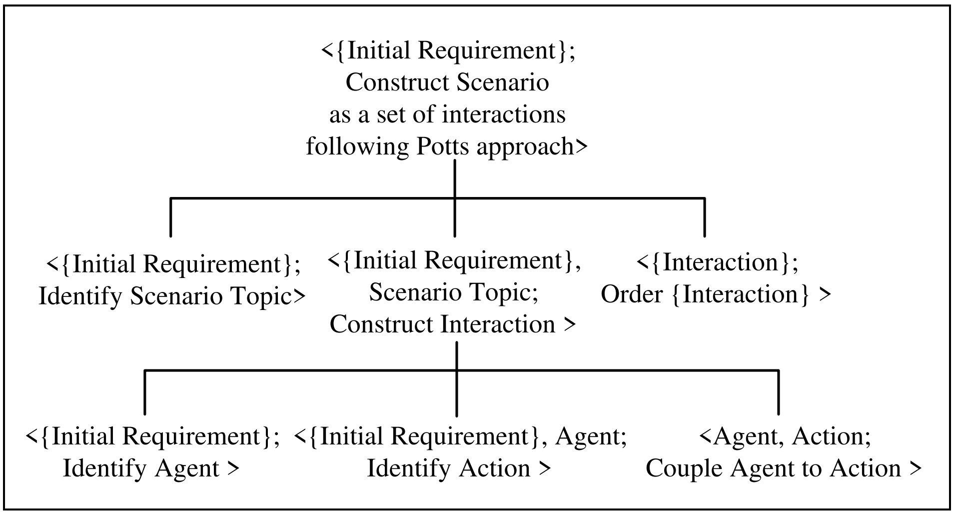

Figure 3 : Process model of the scenario chunk based on the Potts' approach [Potts 94].

3. Meta-knowledge level

The knowledge about when and how to reuse scenario fragments is described and stored in the scenario base in association with each scenario fragment. We call this knowledge the meta-knowledge on scenarios. The meta-knowledge contains the specification of chunks of scenarios, whereas the knowledge level only includes the realisation of the scenario chunk. In the process of integrating a scenario approach in an existing requirements engineering method, these two levels of knowledge serve in separate steps. The meta-knowledge supports the retrieval and integration steps whereas the scenario knowledge is the material effectively reused and integrated in the existing method. How to retrieve and integrate chunks of scenarios is defined at the meta knowledge level as contextual processes defined with the help of *descriptors*.

Our proposal is thus to exploit and extend a contextual approach to define descriptors for each scenario chunk. The meta knowledge on scenario chunks provided by the descriptors aims at facilitating the use of the scenario base. If we keep in mind that the scenario base is used for reusing scenario chunks within existing methods, then the descriptors must help in categorising the situation of project for which a scenario chunk is relevant. Besides the situation of the projects, the intention the user of the scenario base has is mind is determinant in retrieving the good scenario chunks. Each of these aspects is respectively defined in the contextual, internal, and intentional parts of the descriptors.

3.1 Contextual Part

The contextual part of the descriptors describes the external view of scenario chunk and allows to select the necessary scenario chunks. This process of retrieval is performed according to the RE situation in which the chunk should be integrated and to RE intention which should be satisfied. In the contextual part scenario chunks can be presented as black boxes with a scenario in input or in output or used as an intermediate product (Figure 4). Input products are part of the situations in which the scenario chunks may be applied. Output products describe the results obtained by using scenario chunk. Intermediate products are needed only for internal calculations or transformations. The key property of a scenario chunk is its name which represents the RE intention supported by this chunk.

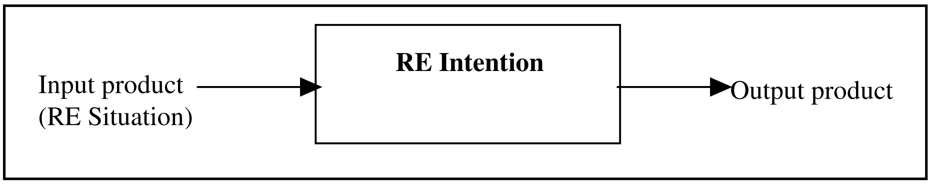

Figure 4 : Context of a scenario chunk .

Each scenario chunk should satisfy an RE intention, like for instance to capture functional requirements by using scenarios, to transform informal requirements into an analysis model. This RE intention generalise one or several scenario intentions, such as to capture scenarios, to integrate scenarios into scenario families or to specialise scenario. The complexity of scenario approaches leads to make the distinction between two types of scenario chunks : atomic and compound ones. An atomic scenario chunk corresponds to a single RE intention and one corresponding scenario intention. For example, in the scenario chunk based on the Potts' approach [Potts 94] the RE intention "capture functional system requirements" corresponds to the scenario intention "capture scenario describing functional system requirements" (Figure 5).

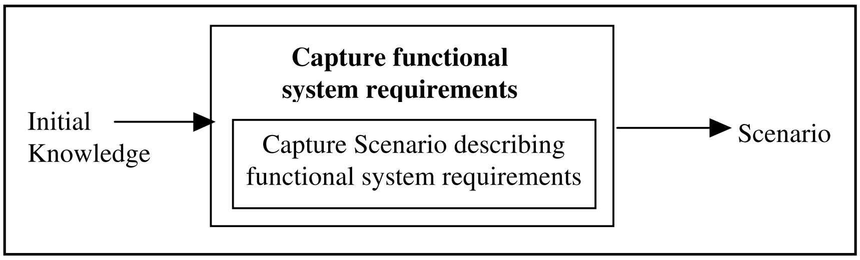

Figure 5 : Example of atomic scenario chunk based on the Potts' approach [Potts 94].

Compound scenario chunks are composed of several atomic chunks and thus based on several intentions. For example, in the scenario chunk based on the Rumbaugh methodology [Rumbaugh 91] the RE intention "construct system dynamic model by using scenario" uses two scenario intentions "capture scenario" and "transform scenario into dynamic model" (Figure 6).

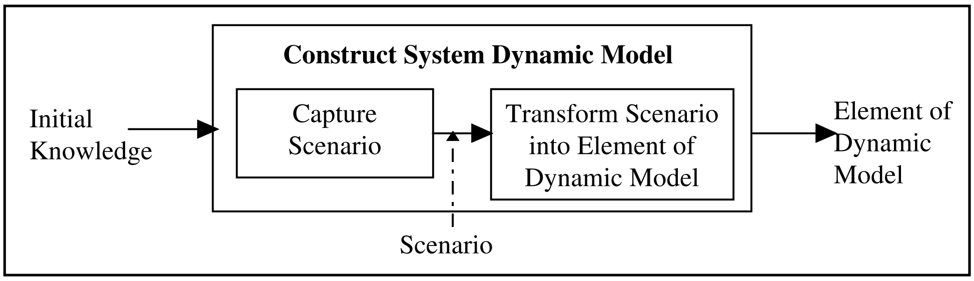

Figure 6 : Example of compound scenario chunk.

3.2 Internal Part

The contextual part of descriptors is not sufficient to discriminate scenario chunks. Many scenario chunks can be used in similar situations and satisfy the same intentions but involve different kinds of information, different ways of working, etc. The information on the scenario formality, scenario abstraction, the nature of interactions, the covered requirements or the scenario life cycle are proved necessary to select more precisely the adequate scenario chunks. Indeed, this additional information is the most relevant to discriminate between scenario approaches, and we find necessary to define carefully and precisely these internal properties. The internal part of the descriptors is inspired by the framework for scenario classification proposed in [Rolland 97]. This framework allows to identify the relevant properties of scenarios and to discriminate between scenario chunks. Three views of the framework, namely the form, content, lifecycle of scenarios are used to identify the internal properties of the scenario chunk descriptors. Each of them is defined and illustrated in the following sub sections.

3.2.1 Form view

The *form* view deals with the mode of expression of scenarios. This view, addresses scenarios as a communication medium between stakeholders, and two facets are associated to it : the *description* and *presentation*. The description facet identifies the *media* - or notations - used for describing scenarios (textual, graphical, video or software), and the level of *formality* at which these descriptions are done (formal, semi-formal, informal). Scenarios are displayed in different ways .The presentation facet makes the distinction between those which are presented through *animation*, and static scenarios displayed as inert texts or diagrams. Moreover, some scenarios allow interaction with the user. *Interactivity* relates to the capability offered to the user to control the way the scenario progresses through time. Basic interactivity features are provided by scenarios allowing only to stop an animation, start it again, or move forward or backward. Hyperlinks (like abstraction/refinement, traceability, exploration links) are an other support for interactive presentations of scenarios. The most advanced interactivity features identified in the framework are those which allow the user to modify the flow of an animation by triggering actions and events.

3.2.1.1 The description facet

Text is probably the most widespread medium used as many authors associate mainly scenarios to narration of "stories" ([Erickson 95], [Holbrook 90] [Kyng 95] [Jacobson et al.,92]). There is however a number of formatted variants such as tables [Potts 94], structured texts [Regnell 95] or scripts [Rubin 92]. Graphics [Rumbaugh 91], images [Gough 95], interaction diagrams [Jacobson et al.,92], sketches and photographs of the application and its environment, are grouped under the graphical value. As well as videos [Wood 94], they are used to capture a system context or to record design scenarios discussed in meetings. Some scenario approaches also recommend the use of several media, like text and graphics in [Holbrook 90].

For stakeholders to whom any kind of formalism is an impediment to comprehension, informal descriptions, like Kyng's scenarios, Jacobson's use cases, or usage scenarios (text in [Rosson 95] graphical in [Benner 92]) provide a good support for communication and facilitate a shared understanding among participants in the requirements engineering process. Scenarios presented in a tabular form are examples of semi-formal notations. For instance, Potts [Potts 94] encodes sequences of actions or events together with the agent responsible for their executions into structured tables of texts. Regnell's et al. [Regnell 95] use case specifications, as Jacobson's interaction diagrams, expressed through a structured diagram, the temporal ordering of interactions between the user and the system. Scenarios which are designed systems and are run as simulations to present a future vision of the system behaviour are very often described under a formal modelling language. For example, the underlying language of scenario descriptions proposed by Glinz [Glinz 95] is a statechart based model [Harel 87]. In the same line, Hsia et al. [Hsia 94] show that a scenario can be adequately represented by a conceptual machine expressed with a regular grammar to validate the requirements specifications.

3.2.1.2 The presentation facet

The animated mode of presentation meets validation requirements (e.g. [Lalioti 95]) as it enhances considerably the communication and understanding of involved parties. Animation through multimedia pictures is explored in [Gough 95]. In this approach, a software tool for managing scenarios is capable of manipulating multimedia assets and to assign them to various parts of the scenario. In a scenario describing treatment operations of a patient using a radiotherapy machine, the tool is for example, able to show by animation how the patient has to be positioned in the treatment room. In object-oriented systems, scenario diagrams are a well-known notations for visualising message flows but their dynamic animation is recent [Koskimies 95], [De Pauw 93]. As a conclusion, it should be noted that scenarios are most often static, but provide sometimes hypertext interactivity like [Holbrook 90].

3.2.2 Contents view

The *contents* view defines the kind of knowledge which is expressed in a scenario. Indeed, scenarios can focus on the description of the functionalities of the designed system as well as describe a broader picture in which the functionality is embedded into a large business process with various stakeholders and resources bound to it. The four facets of the contents view (*context*, *coverage*, *abstraction* and *argumentation*) tend to delineate "scenarios in the wide" which describe abstract chunks of the world including the political and social levels [Kyng 95] from "scenarios in the narrow" which contain more detailed descriptions of behaviour and event dependencies [Jackson 96].

Inspired from Kuutti [Kuutti 95] and Iivari [Iivari 89], the context facet delineates between the *internal* of the system, the *system interactions with its environment,* the *organisational* context of the designed system, and furthermore, addresses the *organisational environment,* foreseeing that still in a wider perspective, the organisation is itself influenced by external factors (history, legislation, economics, etc.).

The coverage facet, based on the classical classification used in RE and system development, makes the distinction between *functional*, *non functional* and *intentional* aspects. As widely accepted in the Information System world [Iivari 91], [Olle 92], the functional aspects are identified by the *structure*, *behaviour* and *function*. The list of possible types of non-functional requirements is adapted from the Requirement Specification Model (RSM) which subsumes the non-functional requirements stated in 21 international standards and guidelines [Pohl 96]. Based on several goal driven approaches (Enterprise Modelling [Bubenko 94], *i** [Yu 94], KAOS [Dardenne 93]), we propose to identify the intentional aspects of scenarios within the following set of concepts : goal, problem, responsibility, opportunity, cause, and goal dependency .

With the abstraction facet, it is possible to identify the need in *concrete*, *abstract* or *mixes* of different degrees of abstraction or concreteness in the scenarios. Concrete scenarios or instance scenarios concentrate on details of individual agents, events, stories and episodes with little or no abstraction. Type scenarios describe facts in categories and abstractions derived from experience, and mixed scenarios contain these two types of concepts.

In addition to the contextual information, argumentation about certain system features, agents, roles or activities can be expressed within a scenario, e.g. the reasons why an agent performs certain actions, or elaborates different alternative solutions before making his decision. To classify scenarios according to the argumentation knowledge stated, the well known IBIS model [Conklin 88] was adapted. The framework proposes to distinguish between:

- positions : descriptions of alternative solutions to a problem,
- arguments : arguments for objecting or supporting a given position,
- issues : descriptions of problems or conflicts and,
- decisions : choices of a particular position.

3.2.2.1 The context facet

As information systems and software engineering communities are very much focused on the structure and functions of the designed system and in some way on their interactions with the environment, they mainly address the problem of describing scenario in the narrow. Use cases [Jacobson 92], [Jacobson95], [Regnell 95], message trace diagrams [Rumbaugh91] and many other scenario based approaches centred on behavioural requirements [Holbrook 90], [Wexelblat 87], [Johnson 95] define scenarios as transactions mainly composed of system / agent message passing. The agents are external entities, human users or any external system playing a role in the use of the designed system. They see a system as a black box observed by its users and, therefore focus on the description of the system interactions with its environment.

While the application concepts are defined in narrow scenarios as the services provided by the system to support agents in their organisational activities, the description of the organisational context can be seen as a part of the explanation of the application domain knowledge [Iivari 90]. Roughly, the organisational context has the same flavour as a rich scenario. It does not aim at a narrowly focused task description, but at a "broad picture of how the work gets done" [Kuutti 95]. This includes knowledge on the stakeholders (as well users as owners, legal regulators, other people impacted by the system, etc.) like their motivations, goals, social relationships, membership in groups, responsibilities (as defined in enterprise modelling). But this rich vision may also deal with the structure of the organisation, the relations of the system to business processes (business rules or policies of the organisation), and to resources.

Despite the lack of examples of scenario approaches involving the organisational environment, we believe that such information may become important within scenarios. One of the real world examples supporting our belief is the increased internationalisation of activities and therefore, information systems. New European Economical Community legislation or policies, for instance, entail business process reengineering of European industrial enterprises and re-thinking of the use of technology. An other concrete example is the failure in automating the London Ambulance Service, due to lack of such kind of requirements [Finkelstein 96] [Macaulay 96].

3.2.2.2 The coverage facet

Most scenario models focus on descriptions of behaviours (e.g. [Firesmith 94], [Glinz 95] [Rawsthorne 96], [Rubin 92], [Jacobson 92], [Some 96]), either of the system under design or at the level of interactions with the environment. It is more and more accepted by the IS community that system design requires some understanding of the organisation's objectives, intentions and goals. Goal driven approaches such as Enterprise Modelling [Bubenko 94] or various requirements engineering models (*i** [Yu 94], KAOS [Dardenne 93]) have been developed with this purpose. Obviously in "rich scenarios" it is clear that intentions, wishes and goals of the various stakeholders are important elements of the scenario contents (for instance [Potts 94] or [Holbrook 90]).

Some scenario approaches address risk evaluation (e.g. time constraints in UML [Rumbaugh96]) or foresee some possible non-functional extensions [Kyng 95]. Again, only a few scenario models give explicit and extended guidelines about what kind of non-functional requirement should one express, and how to express them. Non-functional coverage could be completed, as defined in our framework with performance, time constraints, cost constraints, user support, documentation, examples, backup / recovery, maintainability, flexibility, portability, security / safety, design constraints and error situations.

3.2.2.3 The abstraction facet

The experience seems to tell us that people react to 'real things' and that this helps eliciting requirements. Many current scenario based approaches define the scenario contents at the instance level. Instance scenarios [Potts 94] also called concrete scenarios [Carroll 95] refer to specific agent names to reduce ambiguities in situations of many individuals of the same type. Young et al. [Young 87] for example, view a scenario as "an idealised but detailed description of a specific instance of a human-computer interaction". Carroll [Carroll 95] says "scenarios seek to be concrete, they focus on describing particular instances of use and on a user's view of what happens, how it happens, and why". Their existence is then motivated by the need of describing a particular incident or a specific instance of a human-computer interaction, thus it is not relevant to relate them to a particular class.

Instance-level information plays therefore, two different roles (i) default labels which just add realism to the scenario; (ii) application-specific labels/names bring extra contextual information to the scenario.

Type scenarios also called abstract scenarios are typically used in the Software Engineering Community. Hsia [Hsia 94] makes the distinction between a scenario schema which is a sequence of event types that accomplishes a functional requirement and a scenario instance which is an instantiation of a scenario schema. In OOSE [Jacobson 92], each execution of a use case is an instance scenario with the concrete actor names, event parameters, states and conditions, but the use cases and their associated actors are expressed at the type level and are intended to reuse. M.Rosson and J.Carroll [Rosson 95] go beyond in proposing six abstract usage situations for usage scenarios. Their hope is that these generic situations can be used as heuristics for developing specific scenarios that will be useful in performing several design projects.

3.2.2.4 Argumentation facet

As often, neither Jacobson’s use cases nor Pott’s tables include arguments above the proposed solutions. Holbrook, on the contrary, proposes to investigate several design solutions to a single problem, and to complete them by an argumentation to select between the alternative possibilities. The form and content of these arguments is however unclear, and the method focuses more on the selection of the design solution. Kyng’s approach is one of the most preoccupied in expressing argumentation. Kyng’s use scenarios explore alternative solutions, and include explicit pro and con arguments, without stating any particular decision.

3.2.3 The lifecycle view

How scenarios are captured, evolve and are improved is defined in the *life cycle* view. This view suggests to consider scenarios as dynamic artefacts existing and evolving in time through the execution of operations. In other terms, the framework takes the classical object-oriented position of looking upon scenarios both from a static and a dynamic perspective. The dynamic perspective being the one followed in this view.

First of all, many approaches, consider explicitly or implicitly scenarios as disposable artefacts. The *lifespan* facet deals with *transient* versus *persistent* scenarios. Transient approaches use scenarios as temporary items. In this case, scenarios are intended to be the support of a bounded subpart of the requirements engineering or design activities. As they are not to be used anymore, one can consider they may be disposed immediately after use. In persistent approaches, on the contrary, scenarios persist over the RE process time span. Here, the notion of persistence has to be differentiated from the classical database persistence. Two reasons can justify to keep scenarios persistent. Either they are considered as part of the requirement specifications, or the project documentation management strategy imposes to keep track of the scenarios used (e.g. for later reuse).

Second, as for any RE artefacts, scenario based approaches define ways of working based on several operations. The operation facet defines if and how scenarios are *captured*, *refined*, *integrated*, *expanded*, and *deleted*.

3.2.3.1 The lifespan facet

Object-oriented methods use often scenarios as temporary supports facilitating the requirement elicitation activity [Robertson 95], or requirements capture activity, like OMT [Rumbaugh 91]. In OMT, the use of scenarios is limited to the support of state transition diagrams (STD) construction. Themselves, STDs have no value after they have been used. Likewise, [Lalioti 95] [Hsia 94] and [Benner 92] have a transient approach to scenario lifespan. In these approaches, scenarios are set to facilitate only the validation step of the requirements specification process. However they do not survive after the checks are made and the specifications proved to be correct.

On the contrary, Kyng's stepwise approach [Kyng 95] is an example in which scenarios are used along the all RE process. Indeed, in this approach the RE specifications are incrementally constructed by continuously expanding scenarios. The Inquiry Cycle approach [Potts 94] is an other illustration of persistent approach. There, scenarios are a communication support for acquiring, eliciting and validating requirements. They differ from requirements but have to be stored, as they may be accessed at any time through the hypertext facilities provided by the tool environment. Similarly in OOSE [Jacobson 92] use-case models are supports in the purpose of building design and implementation models, but as any model are part of the reusable project documentation.

3.2.3.2 The operation facet

Capture operations deal with the generation of scenarios. Capture *by reuse* operations concern approaches in which scenarios are created on the basis of some reusable knowledge. This is the case of Rosson and Caroll [Rosson 95], who propose six generic scenarios to instantiate in specific RE situations. Other approaches propose more or less explicitly to capture scenarios by classical enquiry RE techniques, or by translation from already existing schemes. They are then considered to perform capture *from scratch*.

Refinement operations aim at restructuring scenarios to make them easy to understand or more reusable. Refinement operations do not increase the contents of scenarios. For instance, Jacobson [Jacobson 92] proposes the "uses" association as a way to factorise pieces of descriptions shared by different concrete use cases into abstract use cases. Similarly, the "extends" association is an enhancement mechanism to build an advanced use case from several basic ones.

Some authors propose to think of scenarios as stories [Crowley 82], [Erickson 95] which are fragmented in nature. If initially produced as sketchy pieces of details, scenarios can be integrated. Potts et al. [Potts 94], for example, propose to gather scenarios into families of scenarios including various sub-cases of a given scenario. Similarly Regnell's et al. [Regnell 95] extension to Jacobson's et al. [Jacobson 92] use case approach propose to integrate fragmented scenarios into a single usage view per actor.

Expansion operations aim at adding new types of knowledge into scenarios. They may thus be directly opposed to refinement operations. Expansion is necessary in stepwise scenario development approaches. For example in Kyng's multi model approach [Kyng 95], initial scenario descriptions of working situations are expanded, to sketch possible computer support solutions and simulate the future workplace.

Delete operations terminate the scenarios life span. Deletion happens when the scenarios are used temporarily, in which case they can be deleted immediately after use. On the contrary, when the scenarios are part of the requirements specification, or part of the documentation, their life span is the one of the requirements specification, and both are deleted at the same time.

3.3 Intentional Part

As described in the section 3.1, the reuse of scenario chunks is based on an intention driven process. Each scenario approach plays a role in the RE process in which it is embedded. The *purpose* view of our framework partly captures this role and helps to retrieve the scenarios, and this in complement to the intention identified in the RE process. Typical roles assigned to scenarios are : exploration of design alternatives, explanation of the drawbacks or inefficiencies of a system, or more simply description of the functionalities of a system

In the software engineering community scenarios aim mainly at supporting the capture of system requirements [Rumbaugh 91], [Jacobson 92], [Potts 94] whereas in the HCI community, scenarios became almost a standard to explain the way users interface with systems, detailing complex sequences of events [Wright 92], [Carroll 95], [Nardi 92]. In business process re-engineering, scenarios are means to investigate alternative design solutions [Campbell92]. Along the purpose view, scenarios are classified according to the role(s) they play in the RE process. The scenario classification framework identifies three core purposes : *descriptive, exploratory* and *explanatory*. In descriptive scenarios, the analyst and user walk through a process to understand its operations, involved agents, triggering events and the like. Descriptive scenarios primarily serve the purpose of capturing requirements as one or more end to end transactions involving the system and its environment. Exploratory scenarios are useful when several different possible solutions for satisfying a given system requirement have to be explored and evaluated to support an argumented choice. Explanatory scenarios are useful in situations which raise issues and require explanations about the rationale of these issues. The role of such scenarios is to provide detailed illustrations of these situations and their rationale.

In the software engineering community, most of scenario based approaches are descriptive scenarios used to capture behavioural requirements [Anderson 93], [Jacobson 92], [Rubin92], [Regnell 95], [Holbrook 90], [Wexelblat 87]. These scenario often take the point of view of external users or observers who do not know the internal of the system but are able to describe abstract sequences of events that might satisfy the requirements of their task. Descriptive scenarios are also useful to investigate the opportunities for process improvements or to investigate the impacts of a new system and therefore for business process reengineering.

Exploratory scenarios make explicit the link between solutions and requirements. Holbrook [Holbrook 90] proposes to use exploratory scenarios for conducting requirements elicitation. Wirfs-Brock in [Wirfs-Brock 95] admits that it is easy for engineers to lose sight of what and why their design practices follow, when they focus on technical details, and feels that alternatives should be explicitly encouraged early in the design process.

A typical example of explanatory approach is the one of a scenario describing the current operations or desired features of the system by referring to concrete or real incidents. Explanatory scenarios are intended to support explanation and argumentation about drawbacks, inefficiencies or lacks of system performance, putting emphasis on incident causality. Contrarily to descriptive and exploratory scenarios which must be investigated by means of a deliberate, inquiry process, explanatory scenarios are given more spontaneously. The 'Kegworth' scenario presented by P.Wright [Wright 92] which describes a particular aircraft accident happened in the UK is an example of scenario which includes explanations and reasons of the accident. It can be used to identify commonly recurring specific events such as pilot error, equipment failure, or bad design.

In Figure 7 we present an example of the internal and intentional parts of the descriptor. In this example we use the scenario chunk described in Figure 2 and Figure 3. The corresponding contextual part of its descriptor is presented in Figure 5.

Form			
	Description		
		Media :	Text (NL)
		Formality :	Semi-formal (Table)
	Presentation		
		Animation :	False
		Interactivity :	Hypertext-like
Contents			
	Context :		System Internal, System Interaction
	Coverage		
		Functional :	{Structure, Functions, Behaviour}
		Non functional :	{ }
		Intentional :	{ }
	Abstraction :		Type, Instance
	Argumentation :		{ }
Lifecycle			
	Lifespan :		Persistent
	Operation :		Capture (from scrach), Integration
Purpose :			Descriptive

Figure 7: Internal and Intentional parts of the descriptor for the scenario chunk of Potts' approach.

The descriptor shows what corresponding scenario chunk uses tabular form scenarios which describes functional system requirements in natural language. The links between scenarios may be expressed through the hypertext links. Moreover, the descriptor also shows what scenarios may be used in type or instance level according the necessary level of detail. The obtained scenarios are used along the all RE process and they are considered as a part of the requirements specification document.

4. Conclusion

Recently, Requirement Engineering, Information System, Software Engineering and Human-Computer Interaction communities have been considerably attracted by the use of scenarios. These communities propose a large variety of scenario based approaches emphasising a more use-oriented perspective in developing computer systems. The aim of this paper is to introduce the reuse of these approaches within the frame of situational method Engineering. To perform this objective, we propose a scenario base composed of different chunks of scenarios at the knowledge level, and descriptors for each of these scenario chunk at the meta knowledge level. The descriptors allow to retrieve scenario chunks appropriate to the project context, and provide guidelines to integrate them. Most of the paper focuses on an extensive description of the descriptor structure which is based on the views and facets of the scenario classification framework presented in [Rolland 97]. Numerous examples extracted from the literature are used to illustrate our purpose. The integration of the scenario chunks into RE methods are not aborted in this paper.

Experience in construction of scenario chunks has raised difficulties linked to the lack of formal descriptions such as meta models of the scenario approaches. Moreover, the approaches seldom include the process dimension of scenarios. Research issues emerging from these remarks are triple. First, to complete the scenario base by different reusable scenario chunks is necessary to allow scenario usage in any RE situations and to satisfy more RE intentions. Therefore, more variety should be addressed in the scenario application fields. The second problem is to define a process for retrieving the scenario approach the most appropriate to the context of the RE project in which it will be used. Our third issue stands in the need of guidelines to integrate scenarios into existing RE methods.

5. References

[Anderson 93] J.S. Anderson, B. Durney, "Using Scenarios in Deficiency-driven Requirements Engineering", IEEE Int. Symposium on Requirements Engineering, RE'93, San Diego, pp 134-141, 1993.

[Benner 93] K.M. Benner, M.S. Feather, W.L. Johnson, L.A. Zorman, "Utilizing Scenarios in the Software Development Process", International Federation for Information Processing (IFIP), 1993.

[Bubenko 94] J. Bubenko, "Enterprise Modelling", Ingenierie des systèmes d'information, Vol 2, No 6, 1994.

[Campbell92] R. L. Campbell, "Will the Real Scenario Please Stand Up?", SIGCHI Bulletin, Vol 24, No 2, pp 6-8, 1992.

[Carroll 95] J. M. Caroll, "The Scenario Perspective on System Development", in Scenario-Based Design: Envisioning Work and Technology in System Development, Ed J.M. Carroll, 1995.

[Crowley82] D. J. Crowley, "Understanding Communication: The Signifying Web", New York: Gordon and Breach Science Publisher, 1982.

[Conklin 88] J. Conklin and M. L. Begemann, "gIBIS: A Hypertext Tool for exploratoy Policy Discussion". ACMTOOIS, 6(4):303-331, 1988.

[Dardenne 93] A. Dardenne, A. V. Lamsweerde, S. Fickas, "Goal-directed Requirements Acquisition", Science of Computer Programming 20, pp 3-50, Elsevier, 1993.

[De Pauw 93] W. De Pauw, R. Helm, D. Kimelman, J. Vlissides, "Visualizing the Behavior of Object-Oriented Systems", Proc. of OOPLA'93, SIGPLAN Notices 28(10), October 1993.

[Erickson 95] T. Erickson, "Notes on Design Practice: Stories and Prototypes as Catalysts for Communication", in Scenario-Based Design: Envisioning Work and Technology in System Development, Ed J.M. Carroll, 1995.

[Finkelstein 96] A. Finkelstein, J. Dowell. "A Comedy of Errors : the London Ambulance Service Case Study". Proceedings of the Eighth International Workshop on Software Specification and Design, Germany, 1996.

[Firesmith 94] D. G. Firesmith, "Modeling the dynamic Behaviour of Systems, Mechanisms, and Classes with Scenarios", In Software DevCon '94, pages 73-82. SIGS Publications, NY, 1994.

[Glinz 95] M. Glinz, "An integrated formal Model of Scenarios based on Statecharts", Lecture Notes in Computer Science '95 ,pages 254-271, 1995.

[Gough 95] P. A. Gough, F. T. Fodemski, S. A. Higgins, S. J. Ray, "Scenario - an industrial Case Study and Hypermedia Enhancements", Second IEEE International Symposium On Requirements Engineering, 1995.

[Harel 87] D. Harel, "Statecharts: a Visual Formalism for Complex Systems", Sci. Computer Program, 8, pp. 231-274, 1987.
[Harmsen 94] F. Harmsen, S. Brinkkemper, H. Oei, "Situational Method Engineering for Information System Project Approaches". Methods and Associated Tools for the Information System Life Cycle, A.A. Verrijn-Stuart and T.W. Olle (Eds), Elsevier Science, 1994.
[Holbrook 90] C. H. Holbrrok_III, "A Scenario-Based Methodology for Conducting Requirement Elicitation", ACM SIGSOFT Software Engineering Notes, 15 (1), pp.95-104, 1990.
[Hsia 94] P. Hsia, J. Samuel, J. Gao, D. Kung, Y. Toyoshima, C. Chen, "Formal Approach to Scenario Analysis", IEEE Software, pp. 33-41, 1994.
[Iivari 89] J. Iivari, "Levels of abstraction as a conceptual Framework for an Information System", in E. D. Falkenberg & P. Lindgren (eds), Information System Concepts: An In-depth Analysis, North Holland, pp 323-352, 1989.
[Iivari 90] J. IIvari, "Object-Oriented Information System Analysis: Comparative Analysis of six Object-Oriented Analysis Methods", IFIP Transactions: Methods and Associated Tools for the Information System Life cycle, A. A. Verrijn-Stuart & T. W. Olle (Eds) North-Holland, 1990.
[Iivari 91] J. Iivari, "Object-Oriented design of information systems: the design process", in Proc. of the IFIP TC8/WG8.1 Working Conference on the Object Oriented Approach in Information Systems, Canada, Edited by F. Van Assche, B. Moulin, C. Rolland, North Holland, 1991.
[Jackson 96] M. Jackson, "Software Requirements and Specifications", Addison Wesley, 1996.
[Jacobson 92] I. Jacobson, M. Christerson, P. Jonsson, G. Oevergaard, "Object Oriented Software Engineering: a Use Case Driven Approach", Addison-Wesley, 1992.
[Johnson 95] P. Johnson, H. Johnson, S. Wilson, "Rapid Prototyping of User Interfaces driven by Task Models", In John M. Carroll, editor,Scenario-Based Design: Envisioning Work and Technology in System Development, pp 209-246 John Wiley and Sons, 1995.
[Koskimies 95] K. Koskimies, H. Mossenbock, "Scene: Using Scenario Diagrams and Active Text for illustrating Object-Oriented Programs", Proc. of ICSE-18, pp 366-375, 1995.
[Kuutti 95] K. Kuutti, "Work processes: Scenarios as a Preliminary Vocabulary", In John M. Carroll, editor, Scenario-Based Design: Envisioning Work and Technology in System Development, pp 19-36. John Wiley and Sons, 1995.
[Kyng 95] M. 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.
[Lalioti 95] V. Lalioti and B. Theodoulidis, "Use of Scenarios for Validation of Conceptual Specification", Proceedings of the Sixth Workshop on the Next Generation of CASE Tools, Jyvaskyla, Finland, June 1995.
[Macaulay 96] L.A. Macaulay. "Requirements Engineering". Springer Verlag London Limited, 1996.
[Nardi 92] B. A. Nardi, "The Use of Scenarios in Design", SIGCHI Bulletin, 24(4), 1992.
[Olle 92] T. W. Olle, J. Hagelstein, I.G. MacDonald, C. Rolland, H.G. Sol, F.J.M. Van Assche, A.A. Verrijn-Stuart, "Information Systems Methodology : a Framework for Understanding". Addison-Wesley Publishing Company, Second Edition, 1992.
[Pohl 96] K. Pohl. "Process Centered Requirements Engineering". J. Wiley and Sons Ltd., 1996.
[Potts 94] C. Potts, K. Takahashi, A. I. Anton. "Inquiry-based Requirements Analysis". IEEE Software,11(2):21-32, 1994.
[Rawsthorne 96] D. A. Rawsthorne, "Capturing functional Requirements through Object Interactions", In Proceedings of ICRE '96, pages 60-67. 1996.
[Regnell 95] B. Regnell, K. Kimbler, A. Wesslen, "Improving the Use Case Driven Approach to Requirements Engineering", I. C. S. Press, Eds., Second IEEE International Symposium On Requirements Engineering, (York, England), pp. 40-47, March 1995.
[Robertson 95] S.P. Robertson, "Generating Object-Oriented Design Representations via Scenarios Queries", in Scenario-Based Design: Envisioning Work and Technology in System Development, Ed J.M. Carroll, pp 279-30895, 1995.
[Rolland 94] C. Rolland. "A Contextual Approach to Modelling the Requirements Engineering Process". SEKE'94, 6th International Conference on Software Engineering, Vilnius, Lithuania, 1994.
[Rolland 96] C. Rolland, N. Prakash. "A Proposal For Context-Specific Method Engineering". IFIP TC8 Working Conference on Method Engineering, Atlanta, Georgie, USA, 1996.
[Rolland 97] C. Rolland, A. Sutcliffe, M. Jarke, E Dubois, et al. "A Proposal for a Scenario Classification Framework". ESPRIT 4th Framework Programme CREWS 21.903, Deliverable D1-I1.1, 1997.
[Rosson 95] M.B. Rosson, J.M. Carroll, "Narrowing the Specification-Implementation Gap", in Scenario-Based Design: Envisioning Work and Technology in System Development, Ed J.M. Carroll pp 247-278, 1995.
[Rubin 92] K. S. Rubin and A. Goldberg, "Object Behaviour Analysis", Communications of the ACM, 35(9):48-62, 1992.
[Rumbaugh 91] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen, "Object-Oriented Modelling and Design", Prentice Hall, 1991.

[Rumbaugh 96] J. Rumbaugh, G. Booch, "Unified Method, Notation Summary" Version 0.8, Rational Software Corporation, 1996.
[Some 96] S. Some, R; Dassuli, J; 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), 1996.
[Tempora 94] Tempora ESPRIT Project : final report, 1994.
[Wexelblat 87] A. Wexelblat, "Report on Scenario Technology", MCC Technical Report STP-139-87, Microelectronics and Computer Technology, Corporation, Austin, Texas, 1987.
[Wirfs-Brock 95] R. Wirfs-Brock, "Designing Objects and their Interactions: A Brief Look at Responsability-driven Design" in Scenario-Based Design: Envisioning Work and Technology in System Development, Ed J.M. Carroll, 1995.
[Wood 94]D.P. Wood, M. G. Christel, S. M. Stevens, "A Multimedia Approach to Requirements Capture and Modelling", Proc. ICRE'94, Colorado Springs, 1994.
[Wright 92] P. Wright, "What's in a Scenario", SIGCHI Bulletin, Volume 24, Number 4, October 1992
[Young 87] M. R. Young, P. B. Barnard, "The Use of Scenarios in Human-Computer Interaction Research: Turbocharging the Tortoise of Cumulative Science", CHI + GI 87 Human Factors in Computing Systems and Graphics Interface, Toronto, 1987.
[Yu 94] E. Yu, J. Mylopoulos, "Using goals, Rules, and Methods to support Reasoning in Business Process Reengineering", Proc. 27th Hawaii Int. Conf. System Sciences, Maui, Hawaii, Vol IV, pp 243-243, 1994.