CREWS Report 96-01

A PROPOSAL FOR A SCENARIO CLASSIFICATION FRAMEWORK

C. Rolland, C. Ben Achour, C. Cauvet, J. Ralyté (University of Paris I)
A. Sutcliffe, N.A.M. Maiden (City University)
M. Jarke, P. Haumer, K. Pohl (RWTH Aachen)
E. Dubois, P. Heymans (FUNDP Namur)

1. Introduction

Use of examples, scenes, narrative descriptions of contexts, mock-ups and prototypes have attracted considerable attention in Requirements Engineering, Human Computer Interaction and Information Systems communities. Loosely all these ideas can be called scenario based approaches, although exact definitions are not easy beyond saying these approaches emphasise some description of the real world. Experience seems to tell us that people react to 'real things' and that this helps clarifying requirements. Indeed the widespread acceptance of prototyping in system development points to the effectiveness of scenario based approaches. However, we have little understanding about how scenarios should be constructed, little hard evidence about their effectiveness and even less idea about why they work.

One of the earliest demonstration of effective scenario based design was the IBM voice message system for the Los Angeles Olympics [Gould, 1987]. Since then scenarios have come in a variety of shapes and forms. Interpretation of scenarios seems to depend on their usage and how they are generated. In the HCI community scenarios have been proposed as detailed descriptions of a usage context so design decisions can be reasoned about [Carroll 95] or small scale examples of an existing product which are used to anchor discussion about different design theories [Young et al., 1987]. In software engineering "use cases" have been developed as informal narrative description of use, responsibilities and services within object oriented design [Jacobson et al., 1992], [Jacobson, 1995]. Scenarios in the information systems community have evolved to the concept of a rich picture which gives the social and environmental setting of a required system so arguments can be developed about the impact of introducing technology, and the matching between user requirements and task support provided by the system [Kyng, 1995]. Finally in Requirements Engineering scenario scripts have been proposed as test data for checking dependencies between a requirements specification and the users/environment in which it will have to function [Potts et al., 1994].

The paper is an attempt to explore some of the issues underlying scenario based approaches in Requirements Engineering (RE) and to propose a framework for their classification. Motivations for developing the framework are threefold : (a) to help understanding and clarifying existing scenario based approaches, (b) to situate the industrial practice of scenarios and (c) to relate typical RE situations to classes of scenarios.

The paper is organised in four sections. Section 2 is an overview of the framework which is further detailed in section 3 and applied to three scenario based approaches. Section 4 reviews 8 current scenario based approaches using the framework. Finally, conclusions are drawn in section 5.

2. Framework overview

The proposed scenario framework suggests to consider scenarios along four different views, each view allowing to capture a particular relevant aspect of scenarios. Each specific scenario will be characterised according to these four views.

The form view deals with the expression mode of a scenario. Are scenarios formally or informally described, in a static, animated or interactive form ? - these are the kinds of questions about scenarios which emerge from this view.

The contents view concerns the kind of knowledge which is expressed in a scenario. Scenarios can, for instance, focus on the description of a system functionality or they can describe a broader view in which the functionality is embedded into a larger business process with various stakeholders and resources bound to it.

The purpose view is used to capture the role that a scenario is aiming to play in the requirements engineering process. Describing the functionality of a system, exploring design alternatives or explaining drawbacks or inefficiencies of a system are examples of roles that can be assigned to a scenario.

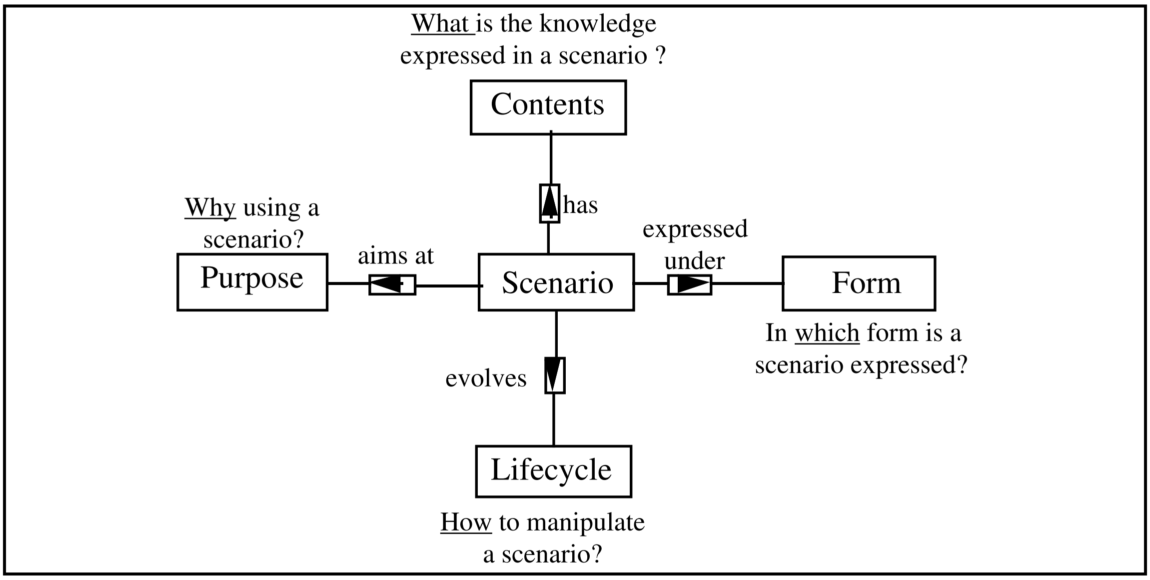

Figure 1 : The four views on scenarios

The lifecycle view suggests to consider scenarios as artefacts existing and evolving in time through the execution of operations during the requirements engineering process. Creation, refinement or deletion are examples of such operations. The question of persistency versus transience is also addressed in this view.

We have adopted a faceted classification method, similar to the one proposed by Prieto-Diaz and Freeman [Prieto-Diaz and Freeman, 1987] for classifying reusable components. To each view is associated a set of *facets*. Facets are considered as viewpoints or dimensions suitable to characterise and classify a scenario according to this view. For example, the *description* facet in the *form* view is a facet helping to classify scenarios according to the medium and the language used for their description.

A facet has a metric attached to it. Each facet is measured by a set of relevant attributes. For instance, the *description* facet is measured with two attributes, namely the *medium* attribute and the *notation* attribute. The former defines the medium used for the description of the scenarios whereas the latter captures the formality level of the notations used for their description. A scenario based approach is positioned in the framework by affecting values to the attributes of each facet. Attribute values are defined within a domain. A domain may be a predefined type (INTEGER, BOOLEAN, ...), an enumerated type (ENUM {x, y, z}), or a structured type (SET or TUPLE). For instance the *notation* attribute is defined on the enumerated type {formal, semi formal, informal} whereas the *medium* attribute is defined on a set type, whose elements belong to the enumerated type {text, graphics, image, video, software prototype}. Thus, a given scenario based approach might be positioned within the *description* facet with the two following (attribute; value) pairs:

(medium; {text, software prototype})

(notation; informal)

In order to validate the framework against existing scenario based approaches, we selected three approaches that we feel, cover a reasonably large part of the spectrum of approaches. In section 4 we shall additionally consider eight complementary approaches.

* The use case approach of OOSE [Jacobson et al., 1992], [Jacobson, 1995] is a good candidate to illustrate the use of scenarios in software engineering.

* Kyng's approach [Kyng, 1995] has been selected because of its attempt to demonstrate that mock-ups, prototypes and scenarios can support all steps of the design process enhancing creative end-user participation. In this approach, we consider that the six design artefacts (summaries, work situation descriptions, use scenarios, mock-ups and prototypes, exploration / requirement scenarios and explanation scenarios) used along the design process can be viewed as some kind of scenario.

* Pott's approach [Potts et al., 1994] is representative of scenario based approaches in Requirements Engineering and proposes a rather complete requirements analysis model supporting documentation, discussion and evolution of requirements.

3. The proposed classification framework

3.1 The form view

Scenarios are described in a variety of media, using more or less formally defined notations. Moreover, the way they are displayed varies from animated presentation supporting dynamic user interaction to static graphical or textual forms. The framework captures this variety of descriptions and presentations within two facets : the *description* and the *presentation* facets, respectively.

3.1.1 The description facet

The terms used to describe scenarios such as software prototypes or mock-ups are formally defined whereas, at the other extreme, scenarios can be concrete descriptions of some excerpt of reality expressed in natural language. The level of formality of the set of notations used for describing scenarios is a first attribute of this facet. The medium used for the description is a second one.

Scenarios have been described in a variety of media. Narrative text is probably the most common, as many authors associate scenarios to narration of "stories" [Erickson, 1995] [Holbrook, 1990]. In the software engineering community, [Jacobson et al., 1992] and [Rumbaugh et al., 1991] express use cases and event traces with structured english with the claim that narrative texts facilitate the capture of requirements. There are a number of formatted variants such as tables [Potts et al., 1994], structured texts [Regnell et al., 1995] or scripts [Rubin and Goldberg, 1992]. Other media have been used to amplify scenarios, graphics [Rumbaugh et al., 1991], images [Gough et al., 1995], sketches and photographs of the application and its environment, and videos [Wood et al., 1994], to capture a system context or to record design scenarios being discussed in meetings.

Scenarios may also be presented as implemented software system in prototypes, mock-ups or simulators. One example is the traffic light scenario used in the ARIES system to simulate the operations of a prototype design [Benner et al., 1992]. Other examples are the Hsia's et al. scenario approach to simulate a user view point through a conceptual machine [Hsia et al., 1994] and animated scenarios to simulate the reaction of a system to external stimuli developed by Lalioti and Theodoulidis [Lalioti and Theodoulidis, 1995].

Besides being vehicled in a certain medium, scenarios are described using more or less formally defined notations. In a number of cases there exists a modelling language to capture scenario descriptions. Scenarios presented in a *tabular form* are an example. Such tables like in [Potts et al., 1994] often encode temporal sequences of actions or events together with the agent responsible for their executions. *Scenario scripts* used in the Object Behaviour Analysis methodology [Rubin and Goldberg, 1992] to capture the services offered by the system objects is another example of descriptions based on semi-formal notations. *Structured representations* such as the use case extension proposed by Regnell et al. [Regnell et al., 1995] fall in the same category. Regnell et al. propose two kinds of structured representations : the use case description and the use case specification. The first one is a structured textual description containing a list of conditions defining a context in which the specific flow of events of the use case can occur. The use case specification expresses through a structured diagram, the temporal ordering of user stimuli, system responses and atomic operations. Both descriptions are based on conventional notations that we quote as semi-formal notations. This form of scenario is illustrated in Figure 2 through the ATM (Automated Teller Machine) example.

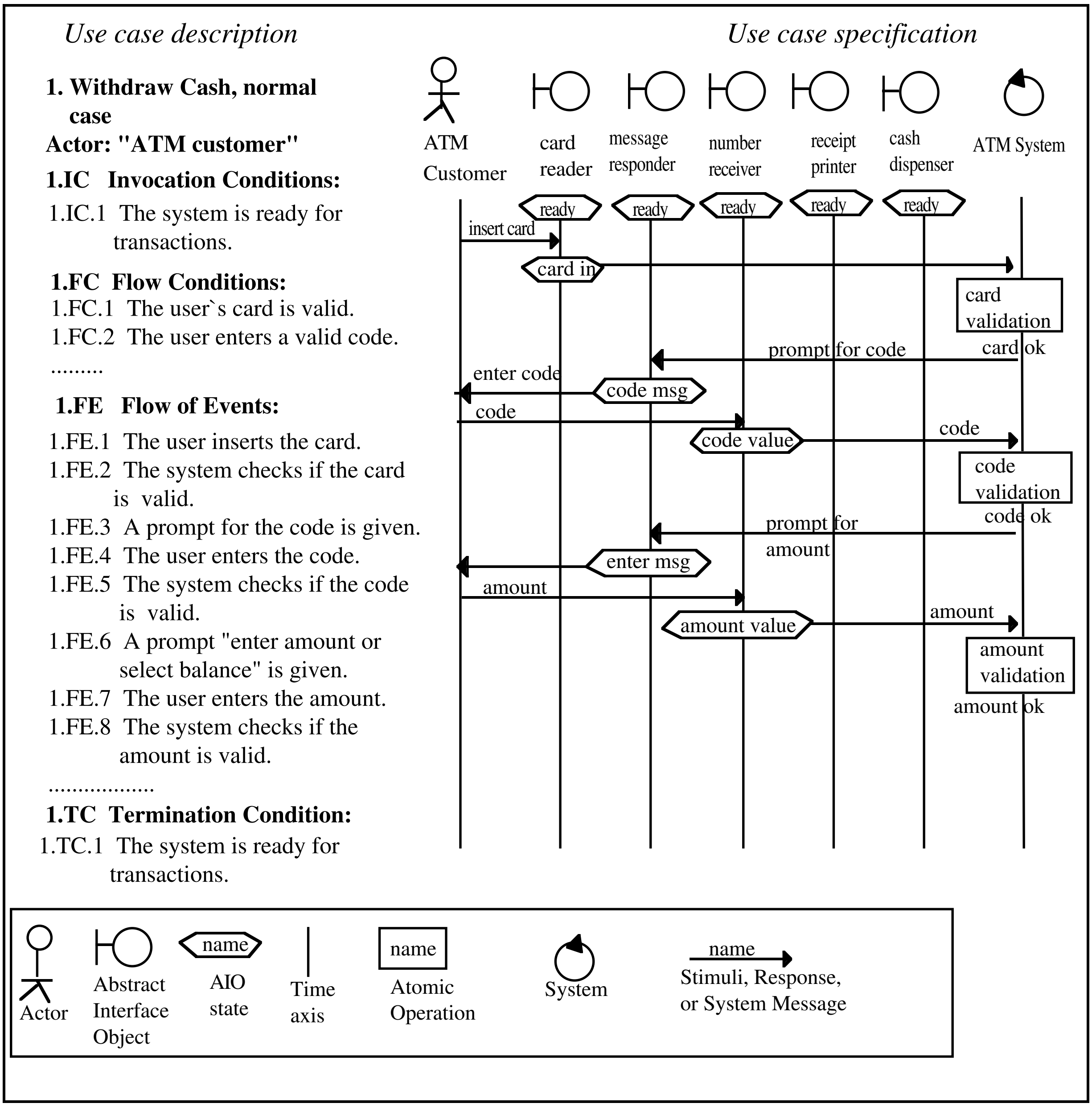

Figure 2 : Use case description and use case specification [Regnell et al., 1995]

Scenarios which are designed systems and are then run as simulations to present a future vision of how the system will behave or how the users might interact with the system need a formally defined modelling language. For example, the underlying language of scenario descriptions proposed by M. Glinz [Glinz, 1995] is a statechart-based model [Harel, 1987]. In the same line, Hsia et al. [Hsia et al., 1994] show that a scenario can be adequately represented by a regular grammar from which a conceptual machine for prototyping may be constructed. Formal descriptions are useful support for validating the requirements specification, reasoning about it and generating cost-effective prototypes.

For stakeholders for whom any kind of formalism is an impediment to comprehension, it may be chosen to present scenarios by only using terms which are taken from their own universe of discourse. Stories descriptions and usage scenarios are two examples which fall in this category. *Stories descriptions,* such as the ones suggested by Erickson [Erickson, 1995] provide a good support for communication and facilitate a shared understanding among participants in the requirements engineering process. *Usage scenarios* describe an envisioned task from the user's perspective making explicit hypotheses and observations about how the user evaluates the importance of objects and relationships constituting the task. The *usage scenario* [Carroll 95] shown in Figure 3. uses for example, a textual sketch to present a situation in which a person interacts with a video information system.

- Harry, a curriculum designer, has just joined a project developing a multi-media information system for engineering education. He browses the project video history. Sets of clips are categorized under major iconically presented headings; under some of these are further menu-driven subcategories.
- He selects the Lewis icon from the designers, the Vision icon from the issues, and an early point on the project time-line. He then selects Play Clip and views a brief scene in which Lewis describes his vision of the project as enabling a new world of collaborative and experience-based education
-...............

Figure 3 : A general narrative scenario example [Carroll 95]

As illustrated in Figure 3, scenario descriptions based on domain-related terms are often expressed using natural language but can use for example domain-related graphical symbols like done in [Benner et al., 1992]. We evaluate them as informal.

In summary, scenarios may be represented in a variety of media, either natural language text, graphics, images, videos or designed prototypes. Furthermore, it may exist a modelling language providing semi-formal / formal notations. Descriptions may also be informal as they are expressed using concrete terms of the reality. Thus, classifying a scenario based approach along the description facet comes down to give its values for the following attributes:

Medium : SET (ENUM {text, graphics, image, video, software prototype})
Notations : ENUM {formal, semi formal, informal}

Note that defining the domain of the *medium* attribute by a set type allows to characterise scenario based approaches using several media.

3.1.2 The presentation facet

Scenarios are displayed in different ways and we make the distinction between those which are visualised in an animated mode and static scenarios displayed as texts or diagrams. Moreover, some allow user interaction, dynamically whereas others don't. The facet, therefore distinguishes scenario approaches according to two facet attributes : animation and interactivity.

Animated presentations benefit from animation techniques to highlight in a natural way the expected behaviour of a system under construction. It becomes for instance, easy to show the impact of events upon the graphical and when appropriated textual - components of the specification. This mode of presentation meets validation requirements as it enhances considerably the communication and understanding of involved parties. The approach for requirements validation proposed in [Lalioti and Theodoulidis, 1995] is an example of animated presentation. The system behaviour is animated and visualised on the screen showing for instance, the chain of impacts of a specific external event to the system internal.

In object-oriented systems, scenario diagrams are a well-known notations for visualising message flows but their dynamic animation has been recently studied [Koskimies and Mossenbock, 1995], [De Pauw et al., 1993]. In [Koskimies and Mossenbock, 1995] for example, scenario diagrams are automatically produced as a side-effect of running the instrumented target system and active facilities are used for understanding the run-time behaviour of object-oriented programs.

Animation through multi-media pictures is another way explored in [Gough et al., 1995]. In this approach, a software tool for managing scenarios is capable of manipulating multi-media 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 annotate the scenario with animation showing how the patient has to be positioned in the treatment room.

On the other extreme, *static presentations* display a scenario as one or several stilted graphic(s) and/or text(s). Such presentations are often used in manual methods.

Besides being static or animated, the presentation of a scenario can have a certain level of *interactivity*. Interactivity relates to the capability offered to the user to control the way the scenario progresses through time. If interactivity is provided, the user is not passive but can act (or be requested to act) at various stages of the scenario presentation.

Basic dynamic features may be provided in terms of functions such as stopping an animation in order to observe a particular situation in detail, starting it again or moving backwards.

Hypertext links inserted in scenarios are another means for planning user interaction when scenarios are used. The possible actions for the user are then, to follow the hypertext links. We can further distinguish among links by identifying :

* traceability links which provide a way to move from one point in the scenario to some other point in another document involved in the requirements engineering process ;
* abstraction / refinement links which make possible observation of some parts of a scenario at different levels of granularity ;
* exploration links which allow users to move from a part of the scenario to another in different ways.

More advanced interaction ways would give to the user the choice to modify for example, the flow of an animation by triggering actions and events.

As a summary, scenario approaches can be classified according to the level of interactivity they provide on the one hand and, the presence or not of animation capabilities on the other hand. Further classification will consist of giving values to attributes from the following domains :

Animation : BOOLEAN
Interactivity : ENUM {None, hypertext-like, advanced}

3.1.3 Applying the framework

Table 1 summarises the position of the three approaches according to the form view by providing the values of the description facet attributes and of the presentation facet attributes.

		Jacobson	Potts	Kyng
DESCRIPTION	Medium	{Text}	{Text}	{Text, Prototype}
	Notation	Informal	Semi formal	Informal
PRESENTATION	Animation	False	False	True
	Interactivity	None	Hypertext-like	Hypertext-like

Table 1 : Classification of three approaches according to the form view

In a standard and manual use of Jacobson's et al. method, scenarios are described with narrative texts which are statically used without any possible planned interaction with the users.

However several extensions of the method have been proposed including more formality [Regnell et al., 1995] or more advanced media to visualise scenarios [Gough et al., 1995].

In the Inquiry Cycle Potts' et al. approach, scenarios are in textual form following some tabular notations. But the RE process is supported by an hypertext tool in which scenarios and requirements are annotated with requirements discussions, rationales and change requests. Therefore, while inspecting a requirement or a scenario fragment, the user can, through hypertext links, retrieve the open questions, responses and arguments that have been posed on this element and the change requests referring to it as well.

M. Kyng proposes to base the whole design process on scenarios, mock-ups and prototypes. Therefore, depending of the process step, scenarios can be in textual forms, concrete mock-ups or software prototypes. In addition, hypermedia facilities can be used with the final prototype system.

3.2 The contents view

The content of scenarios may vary from behavioural information (actions, events, activities) to objects (entities, data, attributes..) or episodes of events and event histories. Other typical contents are organisational information (structure of company, groups, departments, agents, etc.) and stakeholder information (characteristics of people, their views and aspirations and wishes, aims and objectives). We refer to these as the *coverage* facet of scenario based approaches.

Contents tends to delineate "scenarios in the wide" which describe chunks of the world including the social level [Kyng, 1995] from "scenarios in the narrow" which contain more detailed description of behaviour and event dependencies of a system where the design intent has already been established [Jackson, 1996]. In addition to the contextual information, argumentation about what is described can be expressed in a scenario e.g. the reasons why certain agents perform certain actions. These aspects are respectively associated to the *context* and *argumentation* facets.

Another distinction which might be drawn is between concrete and more abstract descriptions e.g. instance scenarios versus type scenarios (the *abstraction* facet).

Finally, in order to capture the variety of aspects related to the contents view, the framework introduces four facets, the *coverage facet*, the *context facet*, the *argumentation facet* and the *abstraction facet* , respectively.

3.2.1 The abstraction facet

The content of a scenario can be concrete, abstract or a mix of different degrees of abstraction or concreteness. 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. The three attributes of the facet, namely *Instance, Type* and *Mixed* allow us to measure the level of abstraction or concreteness suggested in the contents description of a scenario based approach.

The experience seems to tell us that people react to 'real things' and that this helps eliciting requirements. Therefore, many current scenario based approaches define the scenario contents at the instance level. *Instance scenarios* [Potts et al., 1994] also called *concrete scenarios* [Carroll 95] refer to specific agent names or events with concrete argument values. Young et al. [Young et al., 1987] 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". Unlike in object-oriented approaches, instances referred to in scenario descriptions are not seen as instances of a class but as existing by themselves. Their existence is 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. Potts et al. [Potts et al., 1994] add another utility of concrete scenarios, i.e. to reduce ambiguities. In situations dealing with individuals (see for example Figure 4) i.e. where entities such as agents of the same type interact, or when several entities of one type interact with one entity of another type, it is more useful to have entity instances to avoid confusion. Figure 4 is an example of concrete scenario which illustrates this point. In this case an initiator, Esther, has scheduled a particular meeting that requires the attendance of Annie (active), Kenji (important) and Colin (ordinary). The meeting must be held next week, but Kenji and Colin can attend only on days when Annie is out of town. Dealing with instances helps understanding why no meeting is feasible.

Another value of instance scenarios is that application-specific information such as a label or name of something or someone brings further, implicit application information with it. In the library case it might be well-known that borrower "John Smith" is a suspected thief or that "Jill Brown" is a difficult person liable to ask lots of useless questions during a transaction. Implicit information enriches the scenario and makes it more useful during requirements acquisition and validation.

Agent	Action
Esther	Creates new meeting
Esther	Determines that Kenji is an important participant
Esther	Determines that Annie will be presenting
Esther	Determines that Colin is an ordinary participant
Esther	Types meeting description
Esther	Sets data range to be Monday - Friday next week (It's Wednesday p.m. now)
Esther	Determines drop-dead date is Friday noon
Scheduler	Sets timeout to be Fri 9am
Scheduler	Sends boilerplate message to Colin requesting constraints
Scheduler	Sends boilerplate message to Kenji requesting constraints and preferred location
Scheduler	Sends boilerplate message to Annie requesting constraints and equipment requirements

Figure 4 : Concrete scenario "Annie out of town" [Potts et al., 1994]

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.

However, introducing instances into scenarios might be costly. Potts et al. [Potts et al., 1994] for example, mention that instance scenarios have the drawback of doubling their size and possibly introducing irrelevant details (they found 51 actions in the scenario for "Annie Out Of Town" as compared with 21 actions in the scheduling episode). But concrete scenarios may make requirements discussion easier and help to solve conflicts more quickly.

Type scenarios also called *abstract scenarios* are more abstract than instance scenarios. The entities described are not entity instances, like specific agent names, but entity types, like subscriber or customer. Their use is typical of the Software Engineering Community. In OOSE [Jacobson et al., 1992], each execution of a use case is an instance scenario with the concrete actor names, event parameters, states and conditions; for example Peter Reid requests loan of one copy of "How to be a Great Football Manager" by Bill Shankley to Sheila Webster. But use-cases and associated actors are expressed at the type level. Similarly, [Hsia et al., 1994] make 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 instanciation of a scenario schema.

Abstractions in type scenarios may be at different levels. OOSE for example, promotes scenario reuse from abstract use cases, which are higher level abstractions than use-cases. Abstract use cases include scenario descriptions shared by different use cases. As a result one or more use cases can reuse an abstract use case. For example, in the ATM system the abstract use case "Card transaction", which describes the card and personal identification number checks is reused by the concrete use-cases "Cash Withdrawal", "Transfer funds" and "Deposit funds"

M.Rosson and J.Carroll [Rosson and Carroll, 1995] 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 a given design project.

The question of determining the levels of element abstraction for an entire scenario is a vexing one. One consequence of informal scenarios is that multiple levels of abstraction can be expressed in one scenario due to inclusion of elements at different levels of abstraction. Another issue is complex situations which have different levels of abstraction. Consider categories of exception situations so important in scenarios. Exception situations are often complex and can be categorised at numerous levels of abstraction, for example a machine failing due to power loss might be described in a scenario as: a machine failure, a power failure, a local power failure (as opposed to the national grid), a blown fuse or a disconnected socket. This demonstrates that the notion of element abstraction is more complex, even for single elements such as exception situations.

Given all of these factors it is possible, or even probable, that an individual scenario will contain different levels of abstraction. For example, one might imagine to describe interaction with the system when a well-known individual "John" uses the system to retrieve information about all student borrowers and when the system has a general power-failure. This simple scenario includes information described at both the instance and type level, and application-specific labels about a known individual.

Each scenario in the following classification is classified according to the abstraction facet using three attributes derived from above discussion:

Instance : BOOLEAN
Type : BOOLEAN
Mixed : BOOLEAN

The three attributes may be valued "True" for a given approach if this approach accepts scenario instances, scenarios types and scenarios containing both instance and type information.

3.2.2 The context facet

Previous attempts [Kuutti, 1995] to categorise scenarios tend to delineate "scenarios in the wide" which describe the system context including organisational information, social settings, goals, etc. from "scenarios in the narrow" which contain more focused descriptions (behaviour, event dependencies) of a system [Jackson, 1996]. The context facet aims at classifying scenario approaches according to the amount of contextual information they capture.

Inspiring ourselves from Kuutti [Kuutti, 1995] and Iivari [Iivari, 1989], we delineate between the *internal behaviour of the system*, the *system interactions with its environment,* the *organisational context of the system*, and furthermore, we address the *organisation environment,* foreseeing that still in a wider perspective, the organisation is itself influenced by external factors (history, legislation, economics, etc.). These are the four attributes attached to 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 its interactions with the environment, they usually address the problem of describing narrow scenarios.

Use cases [Jacobson et al., 1992], [Jacobson, 1995], [Regnell et al., 1995], message trace diagrams [Rumbaugh 91] and many other scenario based approaches centered on behavioural requirements [Holbrook, 1990], [Wexelblat, 1987], [Johnson et al., 1995] 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.* Jacobson's et al. use cases for instance rely clearly on "the boundary between the system and its environment". As recommended in Jacobson's et al. guidelines, neither organisational information, nor inside system structure or system function that might not be seen by the user is to appear.

In the example of the Scheduler (Figure 5), Potts et al. take a broader perspective by including in scenarios, both *internal system* information and system interactions. The "schedule meeting" activity, for example, does not involve any system/agent interaction. However this action, which proceeds a confirmation message broadcast to all participants, is from Potts' et al. point of view relevant in the scenario, and therefore part of the script.

No	Agent	Action
1	Initiator	Request meeting of a specific type with meeting info. (for example agenda/purpose) and date range
2	Scheduler	Add default (active/important) participants, and so on
3	Initiator	Determine three participants
4	Initiator	Identify one presenter as active participant
5	Initiator	Identify initiator's boss as important participant
6	Initiator	Send request for preferences
7	Scheduler	Send appropriate mail messages to participants (including additional requests to boss and presenter)
8	Ordinary participant	Respond with exclusion and preference sets
9	Active participant	Respond with exclusion and preference sets and equipment requirements
10	Scheduler	Request required equipment
11	Important participant	Respond with exclusion and preference sets and possibly location preference
12	Scheduler	Schedule meeting on the basis of responses, policies, and room availability
13	Scheduler	Send confirmation message to all participants and meeting initiator

Figure 5 : Scenario for the meeting scheduler (no conflicts) [Potts et al., 1994]

While the application concepts are defined in narrow scenarios by the services provided by the system to support agents in their organisational activities, the description in scenarios of the *organisational context* can be seen as a part of the explanation of the application domain knowledge [Iivari, 1990]. 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, 1995]. 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.

The excerpt of a use scenario taken from [Kyng, 1995] which is depicted in Figure 6 illustrates how a use scenario contains contextual information about the stakeholders (secretaries who have the responsibility of registering correspondence) as well as a description of an interaction with the system (hypermedia system allowing input of the material and stimulating the establishing of public links between the material).

A Journal system is an integrated part of the hypermedia. The secretaries currently responsible for registering correspondence become responsible for entering letters, faxes, change requests, and non conformance reports into the hypermedia network. As a supplement and partly a substitute for registering keywords, they establish initial sets of public links between new and existing materials in the hypermedia. When, for example, an incoming letter is a response to a letter already stored in the network, a "Refer-to"-link is established between them.

Figure 6 : Use scenario [Kyng, 1995]

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 EEC legislation or policies, for instance, entail business process reengineering of European industrial enterprises and re-thinking of the use of technology.

In summary, we have defined four types of contextual information that might be included in the content of scenarios in a non exclusive way. This leads to the following attributes for characterising scenario based approaches along the context facet :

System internal : BOOLEAN
System interaction : BOOLEAN
Organisational context : BOOLEAN
Organisational environment : BOOLEAN

3.2.3 Argumentation facet

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, we adapt the well known IBIS model [Conklin and Begemann, 1988], [Conklin and Burges Yakemovic, 1991] and propose 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.

Consequently, we define the argumentation facet with the following four attributes :

Positions : BOOLEAN
Arguments : BOOLEAN
Issues : BOOLEAN
Decisions : BOOLEAN

The use scenario of M.Kyng [Kyng, 1995] depicted below (Figure 7) explores two alternative solutions (positions) for dealing with incoming materials: to scan or to photocopy. For both positions, pro and con arguments are provided, e.g. scanning requires less resource than photocopying. The scenario does not state a particular decision, i.e. it is not clear if the material should now be scanned.

Instead of having secretaries photocopy incoming materials for manual distribution and filling in several local archives, the entry of material into the hypermedia (e.g., by scanning) should imply automatic notification to personnel subscribing to that type of material. This procedure requires less photocopying, but probably more printers for enabling people to get hard copies quickly. Photocopies of certain materials may be made for a few persons who have to print anyway.

Figure 7 : "Archiving" use scenario example [Kyng, 1995]

Most of the provided examples are free of any argumentation information.

3.2.4 The coverage facet

The coverage facet aims at classifying scenario based approaches according to the kind of information they capture. Our proposal is based on the classical classification used in both requirements engineering and system development which makes the distinction between *functional*, *non functional* and *intentional* aspects. These are the three attributes of the coverage facet. They are further refined by using existing typologies for each of the three perspectives. For instance, the *functional* aspect is further decomposed into *structure*, *behaviour* and *function* as widely accepted in the Information System world [Iivari, 1991], [Olle et al., 1992]

The coverage facet must be regarded as orthogonal to the context facet. This means that the Functional, Non-Functional and Intentional coverage may be applied either to the context or to the internal of the system under construction or both. A scenario describing the structure of the organisation will be classified as *contextual* (Organisational context :true) from the context point of view and *structural* (functional : {structure}) from the coverage facet viewpoint. A scenario focusing on the intentional perspective of the information system will be ranked as *system internal* in the context facet and classified *intentional* in the coverage facet.

The functional attribute

The Information System (IS) community has established as a standard that *functional aspects* can be captured with three design artefacts : *structure, function* and *behaviour* [Iivari, 1991], [Olle 92]. The functionality attribute of the coverage facet is based on this classification.

Most scenario models focus on descriptions of behavioural aspects (e.g. [Firesmith, 1994], [Glinz, 1995], [Rawsthorne, 1996], [Rubin and Goldberg, 1992], [Some et al., 1996]) of either the system under design or the interactions with the users. The extract underneath, adapted from [Jacobson et al., 1992], clearly contains structural descriptions of a user interface (items, unique numbers, tables), functional (... choose to view either all places...) and behavioural (if we have selected an item, ...) aspects of the usage of the interface (Figure 8). Jacobson's et al. use cases address the three aspects of the functionality attribute but only at the system interactions level.

Potts' et al. scenarios [Potts et al., 1994], intended to extract behavioural and functional information on the system, also cover the organisational structure aspect. Figure 5 illustrates clearly that the organisation structure outside the system is an important input for the scenario. Jacobson's et al. and Potts' et al. scenario approaches are thus, comparable from the functional coverage point of view, but differ from the context point of view.

The intentional attribute

It is more and more accepted by the Information Systems community that system design requires some understanding of the organisation's objectives, intentions and goals. Goal driven approaches such as Enterprise Modelling [Bubenko, 1994] or various requirements engineering models (*i** [Yu and Mylopoulos, 1994], KAOS [Dardenne et al., 1993]) 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.

Initialization

. . .

(3) The items can be ordered in a number of ways. This is selected with the ORDER menu. The following orders are possible: (a) alphabetical order, (b) index order (each item has a unique number), (c) turnover of the items, (d) storing order.

(4) In the 'Form place' table we might choose to view either all places in the current warehouse or, if we haven't celected an item, the places where that item exists.

. . .

Planning

. . .

(2) The planning should minimize the use of trucks on condition that all delivery dates should be held and the trucks should be compatible with any delivery requirements for the items (for example, insize). This should be done by adjusting existing, already planned redistributions to take account of the new redistribution requirements.

Figure 8 : Jacobson's et al. warehouse use case [Jacobson et al., 1992]

These justify the *intentional* attribute of the coverage facet. Based on the analysis of the mentioned goal driven approaches, we propose to further refine the intentional attribute using the following set of concepts : *Goal, Problem, Responsibility, Opportunity, Cause, Goal dependency* .

But intentional models are seldom included in scenario approaches. For example, in [Jacobson et al., 1995], Jacobson et al. state the importance of customer's expectations in the business process reengineering of a company. However, Jacobson's et al. use cases do not include explicitly the actors' goals. They are so to speak, implicitly underlying the interface between the reegineered company and its environment.

The non-functional attribute

Some scenario approaches address risk evaluation (e.g. time constraints in UML [Rumbaugh 96]) or foresee some possible non-functional extensions [Kyng, 1995]. 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 is further refined 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. These types are adapted from the Requirement Specification Model (RSM) which subsumes the non-functional requirements stated in 21 international standards and guidelines [Pohl, 1996].

Kyng's scenario shown in Figure 7 contains a number of non-functional aspects. Especially in the first paragraph one can find user support, security, performance, and design constraints requirements. None of these is explicitly stated as required in the approach but Kyng's guidelines encourage their expression.

Based on the above discussion, the three attributes of the coverage facet are associated to the following value domains :

Functional : SET (ENUM {Structure, Function, Behaviour })
Intentional: SET (ENUM {Goal, Problem, Responsibility, Opportunity, Cause, Goal dependency})
Non Functional : SET (ENUM {Performance, Time constraints, Cost constraints, User support, Documentation, Examples, Backup / recovery, Maintainability, Flexibility, Portability, Security / Safety, Design constraints, Error handling })

3.2.5 Applying the contents view

Table 2 presents the classification of the three selected approaches according to the contents view. Each approach is measured with the attributes previously introduced in the description of the four facets of the view.

In *Jacobson's et al. approach*, use cases are described at the type level and what is refered to as a scenario is a use-case instance. Designers using this approach develop use-case models which are collections of use-case classes (i.e. use case types) and actors. Scenarios (i.e. instance scenarios) only exist when the system is in operation and they are instances of use-case classes. Consequently this approach combines type and instance scenarios but used at different stages of the system life cycle. *Potts' et al. approach* supports instance scenarios. Instances are used (cf. Figure 4) to avoid confusion in describing certain situations whereas most situations are expressed at the type level. *Kyng's approach* describes scenarios at both the type and the instance levels. Instance scenarios are used to express some specific details of described situations; as an example, the scenario described in Figure 9 uses given names for users (U1, U2) and text component (C1); this makes more understandable the lock problem arising when there is a conflict among two *different* users accessing the *same* text component.

		Jacobson	Potts	Kyng
ABSTRACTION	Instance	True (scenario)	True	True
	Type	True (use-case)	True	True
	Mixed	False	True	True
CONTEXT	System Internal	False	True	True
	System Interaction	True	True	True
	Org. Context	False	True	True
	Org. Environment	False	False	False
ARGUMENTATION	Position	False	False	True
	Arguments	False	False	True
	Issues	False	False	True
	Decision	False	False	False
COVERAGE	Functional	{S, F, B}	{S, F, B}	(S, F, B}
	Intentional	{ }	{ }	(goal, responsability}
	Non Functional	{ }	{ }	{user support, security, performance, design constraints}

Table 2 : Scenario based approaches classification according to the contents view

According to the context facet, *Jacobson's et al.* scenarios are classified *system interaction* as they are clearly centered on the description of the system interactions with its users. Internal system aspects are not modelled. Use cases do not capture neither information about organisational context nor information about organisational environment. In *Pott's approach*, because the system can be considered as an actor, internal actions can be represented. For instance in Figure 5, the scheduler which corresponds to the system executes the "schedule" action (action No 12) of the meeting on the basis of responses, policies and room availability. The detail of internal actions can even be described through more detailed scenarios. Figure 6 has shown that *use scenarios* in *Kyngs's approach* could capture information on the organisation and the system interaction with the environment. Another kind of scenario (so-called Explanation Scenarios) allows to capture internal behaviour of the system. None of these three approaches mention or provide examples with information about the organisation environment.

U1 and U2 have started a session on the same hypertext H1. U1 has opened the text component C1 with read lock, and U2 has opened it with a write lock. They both set the immediate update preference on C1. U2 is making changes to C1, but hasn't committed any changes, hence U1's view of C1 has not been updated for a while. U1 attempts to create...

Figure 9 : Requirement scenario [Kyng, 1995] using given names

In *Jacobson's et al. approach*, the use case model is not tailored to express different viewpoints or arguments on a design solution. In *Pott's approach,* scenarios analysis raises questions which are subject to discussions in the requirements analysis process. Although a scenario can be annoted with its rationale or assumptions, it does not itself capture argumentation. *Kyng's approach* is more powerful in expressing argumentation. Figure 7 (which is a use scenario) shows that the description of future possibilities for work organisation can be related and compared to the existing situation. The approach also proposes a specific type of scenario (called Exploration / Requirement scenarios) with the aim of keeping trace of the design rationale. For instance, the requirement scenario in Figure 9 reflects a situation in which the design of a locking mechanism for support of co-operation on hypermedia documents is required. This scenario provides a "pointer to the situations that inspired the requirements in this way constitute some of the reasons for the requirement" [Kyng, 1995].

As stated in 3.2.4 *Jacobson's et al. approach* scenarios cover system functional requirements at the system interactions level. Use case descriptions do not contain neither intentional aspects nor non-functional requirements. In *Potts' et al. approach*, scenarios themselves only cover functional requirements of the system. A scenario describes a functionality of the desired system. System behaviour is expressed through actions initiated by the system, for instance in the scenario "slow responder" in the meeting scheduling problem, the action "Scheduler recognises that the time-out has expired and reminds late participant" deals with the system behaviour. Scenarios do not capture neither non functional goals nor intentions. However the authors of this approach have investigated the derivation of goals from scenarios [Takahashi et al., 1996] [Anton]. In *Kyng's approach*, use scenarios cover functional and structural aspects of requirements for work organisation whereas requirement scenarios express constraints imposed by the future software system (non-functional requirements). For instance, the use scenario "archiving" in Figure 7 highlights new tasks and responsibilities of the secretaries when using the hypermedia system. In the same approach the requirement scenario partially shown in Figure 9 expresses a system functional requirement, namely the one of a mechanism for cooperative work with document sharing. The approach takes into account interface requirements (non functional requirements) through mock-ups and prototypes which are presented and discussed with users during workshops.

The GBL organisation is characterized by extensive use of computers to support different aspects of work........The computer systems do not, however, support horizontal information flow (except for e-mail), sharing of materials, and planning/monitoring of activities in daily work. The horizontal information flow and sharing of materials are currently accomplished by circulating multiple hard copies of documents among the personnel who needs access to the documents.......

Figure 10 : Initial study summary [Kyng, 1995]

Some intentions can be captured in initial study summaries about the work of the end users. For instance, Figure 10 presents a summary about the situation regarding computer use at GBL (Great Belt Link Ltd). It summarises an important shift in focus from the so-called vertical systems to the intention for improving horizontal support.

3.3 The purpose view

In the software engineering community scenarios aim mainly, at supporting the capture of system requirements [Rumbaugh et al., 1991], [Jacobson et al., 1992], [Potts et al., 1994] whereas in the HCI community, scenarios became almost a standard to explain the way users can interface with systems detailing complex sequences of events [Wright, 1992], [Carroll, 1995], [Nardi, 1992]. In business process re-engineering, scenarios are means to investigate alternative design solutions [Campbell, 1992]. Apparently, there exists some core purposes for using scenarios that the purpose view tries to identify. Along the *purpose view,* scenarios are classified according to the role they aim to play in the requirements engineering process. We identify three of them : *descriptive, exploratory* and *explanatory.* Although such a categorisation might prove to be useful in orienting people to their needs of different scenarios, it can be the case that the same scenario will be used for several purposes. This is tackled in the facet metric by associating a three-tuple attribute to the facet.

In *descriptive scenarios*, the analyst and user walk through a process to understand its operations, involved agents, triggering events and the like. They primarily serve the purpose of capturing requirements. Potts' et al. approach is a typical example [Potts et al., 1994]. Requirements are derived from descriptions of one or more end to end transactions involving the system and its environment. In most cases of scenario based approaches in software engineering, descriptive scenarios have been used to capture behavioural requirements [Anderson and Durney, 1993], [Rubin92], [Regnell et al., 1995], [Holbrook, 1990], [Wexelblat, 1987]. The scenario takes the point of view of an external user or observer who does not know the internal of the system but is able to describe an abstract sequence of events that might satisfy the requirements of his 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. The use of scenarios in the marketing domain [Campbell, 1992] is an example.

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. Such scenarios make explicit the link between solutions and requirements. Capt. H.Holbrook III proposes to use exploratory scenarios for conducting requirements elicitation [Holbrook, 1990]. In the scenario generation phase the designer formulates different design models (design models can be regarded as requirements specifications) allowing to fulfil goals identified for a future system. Different design models can be developed using varying configurations of resources and/or processing algorithms. Based on the perceived behaviour of these models, scenarios are then developed to choose among the alternative designs. For example, in specifying the requirements for a Library System, the designer could have formulated two design models for satisfying the system goal "*Maintain the Check out of the Books".* Both use an on-line database. One model uses scanners to read barcodes put on books and library cards whereas the second does not. A scenario is associated to each design model. These two scenarios help exploring various solutions and selecting the most appropriated one. Exploratory scenarios might also be regarded as a strategy encouraging requirements engineers for systematically looking for alternative decisions and enlarging the vision they have of the problem to solve. Wirfs-Brock in [Wirfs-Brock, 1995] 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.

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. A typical example is the one of a scenario describing the current operations or desired features of the system by referring to concrete or real incidents. They are intended to support explanation and argumentation of drawbacks, inefficiencies or lacks of system performance, putting emphasis on incident causality. Contrary 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, 1992] which describes a particular aircraft accident happened in the UK is an example of scenario which includes explanations and reasons of the accident (Figure 11). It is an exploratory kind of scenario.

One of the engines on the aircraft became damaged when a turbofan blade fractured. The pilots, in attempting to contain the fault, shut down one of the engines. What they did not realize at the time was that they had shut down the healthy engine by mistake. Later in the flight they carried out manoeuvres for their final approach and the increased load on the damaged engine led it to fail entirely. The pilots were unable to restart the healthy engine they had mistakenly shutdown and the aircraft crashed onto a motorway just short of the runway threshold.

Figure 11 : Explanatory scenario [Wright, 1992]

Scenarios like the "Kegworth" one, can be used to identify commonly recurring events such as certain kinds of pilot error, equipment failure, or bad design. Figure 12 shows the possible ways in which - according to P. Wright - a "Kegworth" scenario can be used.

A 'Kegworth' scenario can be used for:
* the analysis of pilot error and failures of decision making,
* the analysis of the inability of the pilots to reprogram the flight management interface,
* the improvement of the cockpit instruments and warning system,
* the pilot training so as to help pilots avoid the making the same mistakes.

Figure 12 : The possible ways of the 'Kegworth' scenario usage

In summary, a scenario based approach will be classified according to the role it aims to play by three boolean values associated to the three following attributes :

Descriptive : BOOLEAN
Exploratory: BOOLEAN
Explanatory : BOOLEAN

3.3.1 Applying the purpose view

The classification of the three selected scenario based approaches using the metric defined for the role facet of the purpose view is summarised in Table 3.

		Jacobson	Potts	Kyng
ROLE	Descriptive	True	True	True
	Exploratory	False	False	True
	Explanatory	False	False	False

Table 3 : Scenario based approaches classification according to the purpose view

Potts' et al. [Potts et al., 1994] and Jacobson's et al. [Jacobson et al., 1992] approaches rely exclusively upon descriptive scenarios. Potts' et al. scenarios are end-to-end sequences of activities performed by several roles to achieve some goal. Jacobson's et al. scenarios describe the required system funcionalities through the interactions between the system and its environment (actors).

Kyng [Kyng, 1995] proposes five types of scenarios among which four are classified descriptive and one is exploratory. *Summaries*, *work situation descriptions* , *use scenarios* and *explanation scenarios* are descriptive scenarios, whereas *exploration scenarios* are exploratory ones. Kyng defines work situation descriptions as "descriptions of relevant, existing situations within the users' workplace", whereas "use scenarios describe future possibilities". Explanation scenarios, despite their name are also descriptive of new possibilities but "they are more abstract and more detailed than use scenarios" and "they are produced later in the design process than use scenarios". Exploration scenarios "are intended to adequately reflect situations that we want our software to support, and then to functions as discussion tools in the process of understanding and developing the concepts of the software design, not primarily of the work situations". As several exploration scenarios can be constructed reflecting for example, different choice of resources these scenarios serve to explore different design models and therefore are classified as exploratory.

3.4 The lifecycle view

How scenarios are captured, evolve and are improved is the concern of *the life cycle view*. This view suggests to consider scenarios as artefacts existing and evolving in time through the execution of operations. In other terms, the framework takes the position of looking upon scenarios from both a static and a dynamic perspective. The dynamic perspective is the one followed in this view. Although most of existing scenario based approaches deal with the static aspects of scenarios, some are concerned by their dynamic aspects. The life-cycle view helps classifying scenario based approaches depending of the way they handle these dynamic aspects. First of all it seems that most approaches, explicitly or implicitly consider scenarios as disposable artefacts. The *lifespan facet* deals with transient versus persistent scenarios. Secondly, reasoning by analogy with object oriented approaches raises the question of which operations are eventually performed on scenarios. This question is the concern of the *operation facet* of the life cycle view.

3.4.1 The lifespan facet

As living artefacts, scenarios have a lifespan. The framework distinguishes between *transient scenarios* used as temporary items from *persistent scenarios* which persist over time.

Transient scenarios are meant to be a support for some requirements engineering or design activity and are thus, thrown after being used. In object-oriented methods, scenarios are typically used for capturing requirements [Rumbaugh et al., 1991], in other approaches they are facilitators for requirements elicitation [Robertson, 1995] whereas in some others they are a temporary support for requirements validation [Lalioti and Theodoulidis, 1995], [Hsia et al., 1994]. In OMT [Rumbaugh et al., 1991] for example, scenarios are instruments supporting the construction of state transition diagrams which have no value after been used. In [Lalioti and Theodoulidis, 1995] and [Benner et al., 1992] scenarios are set to facilitate the validation of the requirements specification but don't survive after the checks are made and the specification proved to be correct.

Despite transient scenarios which have a short life span, *persistent scenarios* exist longer as long as the documentation of the project they belong to exists. There are two reasons for scenarios to be persistent. The first one is when scenarios are considered part of the requirements specification, the second one is when the project documentation keeps track of the scenarios used. The stepwise approach proposed by Kyng [Kyng, 1995] is an example of the first case. Indeed, in this approach the RE specification is incrementally constructed by integrating scenarios. The Inquiry Cycle approach [Potts et al., 1994] is an illustration of the second case. Scenarios are there, a communication support for acquiring, eliciting and validating requirements; they differ from requirements but are stored as artefacts that can be accessed at any time through the hypertext facilities provided by the tool environment. Similarly in OOSE [Jacobson et al., 1992] a use-case model serves the purpose of building design and implementation models but as a model is part of the project documentation.

According to this facet, scenarios are classified into transient and persistent scenarios. This facet is described as an attribute whose type is an enumerated set composed of the two values, transient and persistent.

Lifespan : ENUM{transient, persistent}

3.4.2 The operation facet

As any dynamic artefact, scenarios are created, transformed and deleted through the execution of operations. The operation facet aims at classifying scenarios according to the kinds of operations carried out on them. Thus, this facet is concerned with how scenarios are captured, evolve and are eventually transformed during the RE process.

Operations on scenarios can be classified in the following types:

* capture operations,
* refinement operations,
* integration operations,
* expansion operations,
* delete operations.

Capture operations deal with the generation of scenarios**.** Most of the existing approaches seldom propose any alternative to the creation of scenarios from scratch. However, scenario capture by reuse has already been proposed. For instance, Rosson and Caroll [Rosson and Caroll, 1995] propose six generic scenarios to be used in specific situations.

Refinement operations aim at transforming scenarios to make them easy to understand or more reusable. They re-structure scenarios without increasing their contents. For instance, Jacobson [Jacobson, 1995] proposes the "uses" association as a way to factorise a scenario description shared by different use cases. These shared descriptions are called abstract use cases whereas the original use cases are called concrete use cases. Similarly, the "extends" association [Jacobson, 1995] is an enhancement mechanism to build an advanced use case from several basic ones.

Several authors propose to think of scenarios as stories [Crowley, 1982], [Erickson, 1995] which are fragmented in nature. If initially produced as fragmentary pieces of details, scenarios can be *integrated.* Potts et al. [Potts et al., 1994], for example, propose to gather scenarios into families of scenarios in order to assemble in the same family the various subcases of a given scenario. Similarly, Regnell et al. [Regnell et al., 1995] propose an extension to Jacobson's et al. [Jacobson, 1992] use case approach. The extension consists in integrating different fragmented scenarios into a single usage view per actor. Glinz [Glinz, 1995] goes further on from the simple idea of integrating scenarios by proposing four composition templates: sequence, alternative, iteration and concurrence.

An *expansion operation* aims to add new knowledge in a scenario description. Expansion operations are necessary in stepwise approaches for developing scenarios such as the one proposed by Kyng [Kyng, 1995]. In this approach, initial scenarios descriptions of working situations are expanded to sketch possible computer support solution and simulate the future workplace.

Delete operations terminate the scenario lifespan. There are two kinds of situations in which they occur, when the scenario is used to support the discovery of requirements, it can be thrown immediately after use, when the scenario is part of the requirements specification, its lifespan is the one of the requirements specification.

The operation facet is composed by the five types of operations on artefacts presented above. The operation of capture is defined by an attribute taking its value in the enumerated set corresponding to the two approaches already identified : {from scratch, by reuse}. The four remaining types of operations are declared as existing or not in the different existing approaches.

Operation

Capture : ENUM{from_scratch, by_reuse}
Integration : BOOLEAN
Refinement : BOOLEAN
Expansion : BOOLEAN
Deletion : BOOLEAN

3.4.3 Applying the lifecycle view

In this section, we classify Jacobson's et al. [Jacobson et al., 1992], Potts' et al. [Potts et al., 1994] and Kyng's [Kyng, 1995] scenario based approaches according to the lifecycle view. In Table 4 the positions of the three approaches are given using attributes of the lifespan and operation facets.

		Jacobson	Potts	Kyng
LIFESPAN		persistent	persistent	persistent
OPERATION	Capture	from_scratch	from_scratch	from_scratch
	Integration	False	True	False
	Refinement	True	False	False
	Expansion	False	False	True
	Deletion	False	True	False

Table 4 : Scenario based approaches classification according to the lifecycle view

Jacobson's et al. use cases are persistent because they form the use case model which is part of the RE document. Similarly, Potts' et al. scenarios are stored in the artefact document and therefore, are persistent. Kyng's scenarios are fully integrated to the requirements specification and are thus, persistent.

In the three approaches scenarios are constructed from scratch. Jacobson proposes the "use" and "extend" associations as means for use case refinement whereas Potts' et al. and Kyng's approaches don't refer to any refinement operation. On the other hand, scenarios of Potts' et al. approach can be integrated in families whereas they are fragmented in the two other approaches. Finally, Kyng's scenario based approach includes the expansion operation whereas in Potts' et al. and Jacobson's et al. approaches, scenarios may only be transformed into other models.

3.5 The process perspective on scenarios

The classification framework described above is intended to classify models of scenarios. As presented in Figure 1, our view on scenario models is typically a product-oriented one. This includes the purpose, the form, the contents, and the lifecycle aspects of scenario models. However, both the information systems and software engineering communities have pointed out the complementarity of the product and the process aspects of artefacts. The process dimension of scenarios is seldom considered in the literature. This justifies that we did not include in the framework the process aspect of scenario based approaches. However what approaches offer to model the process of generating scenarios is important and this section is a preliminary discussion on this matter.

Industrials' scenario practice has already reported this problem of poor process definition in the literature [CREWS ISC 96]. However some methodological guidelines for constructing scenarios are provided. Jacobson et al. [Jacobson et al., 1995], for example, recommend a sequence of tasks to construct a use case (find actors, find use cases, priotorize use cases, describe use cases, select metrics, review). Even if "rules of the thumb" (Figure 13) are provided to guide in the performance of these tasks, many methodological concerns such as motivations, situations, alternative ways of working, etc. are not tackled.

We recommend that the first step should be to try to identify the actors in the business...
A hint on how to verify that all of the actors in the business have been found is to study a set of examples of real people with whom the company has contact...
To verify that all of the responsibilities of the business are actually included in the use-case model, you should make sure for each responsibility it is always possible to point to at least one use case in the model...
You should not divide up the use cases in the business too much. It is normal to have 10-20 use cases in a business. If you find more than 30 use cases, you should suspect that too much detail has been put into the model; perhaps you have forgotten that use cases are to be complete courses of events...

Figure 13 : Task description for BPR by Jacobson et al. [Jacobson et al., 1995]

Moreover the required level of formalisation of the scenario model to support the description of complete guidelines is seldom achieved. Only a few approaches formally define the concepts underlying the notion of scenario i.e. provide a meta-model ([Nardi 95], [Regnell et al., 1995], and UML [Rumbaugh and Booch, 1991]).

How the scenario usage can drive the RE process in a reactive manner is an important but unresolved question. What is necessary is to develop strategies that based on scenarios will play a reactive role in guiding the RE process, learning from what has already been done, suggesting what is missing, supporting discussion, debate and agreement. Holbrook [Holbrook 95] proposes two main scenario generation strategies corresponding to two different types of situations. Holbrook claims that reactive scenario generation is required in the situation of wicked, ill-structured problems. In opposition, enactive strategies are possible in the case where the situation is familiar to the designer. Here, scenarios are formulated "to force the user to articulate essential system characteristics or select between various options".

Enactive strategies seem to be usual among the requirements engineering community. Small improvements (small in the sense that they do not require long reengineering tasks, neither any completely new design approach) suggested by Jacobson et al. as a policy for business process reengineering can be regarded as an enactment strategy.

In opposition, Potts et al. [Potts et al., 1994] use scenarios as a discussion "catalyst" helping to capture reactively the stakeholders' views on the designed system. However the reactiveness is not defined as such but just suggested as part of the role that the RE facilitator should play in the process.

Another aspect is *"how to envision the RE process based on scenario usage"* . Kyng [Kyng, 1995] addresses the problem by organising the full RE process in five steps, each of them being based on a specific type of scenario. Even if this method provides some answer to the general question it justifies neither the stepwise organisation of the process nor the use of specific types of scenarios.

A last, but not least interesting concern is how classes of scenarios relate to specific RE activities. Rosson and Caroll [Rosson and Caroll, 1995] propose a taxonomy of six types of situations for HCI development. When the project in hand matches one of the situations predefined in the taxonomy, the corresponding type of scenario is developed. The taxonomy is thus used as an input to methodological guidelines for the selection and description of scenarios.

This approach has demonstrated that the problem of situating the usage of scenarios is relevant and requires identification of fine grained situations. It converges also with more general works on RE process definition [Rolland and Grosz, 1995] stating that a contextual approach linking situations to intentions could help structure the process within a decision oriented framework.

4. Review of scenario based approaches

In addition to the three approaches used to illustrate the proposed framework, we propose in this section a review of eight other approaches in order to situate the current literature on scenarios. Table 5 summarises the classification of the eight approaches.

		Benner	Gough	Holbrook	Hsia	Regnell	Rumbaugh	Scalzo	Somé
Descri-	**Medium (1)**	{T, I, P}	{T, I, P}	{T, I}	{T}	{T, I}	{T, G}	{T, G}	{T}
ption	**Notation (2)**	Any	I	I	F	S-F	S-F	S-F	S-F
Presen-	**Animation (3)**	Any	T	T	F	F	F	F	F
tation	**Interactivity**	Any	Hypertext	Hypertext	None	None	None	None	None
Abstra-	**Instance**	T	F	T	T	F	F	T	F
ction	**Type**	T	T	T	T	T	T	T	T
	Mixed	T	F	T	F	F	F	T	F
Context	**System Internal**	T	F	F	F	T	F	T	T
	System Interaction	T	T	T	T	T	T	T	T
	Org. Context	T	F	T	F	F	F	T	F
	Org. Environment	T	F	F	F	F	F	F	F
Argumen-	**Position**	T	F	F	F	F	F	T	F
tation	**Arguments**	T	F	T	F	F	F	T	F
	Issues	T	F	T	F	F	F	T	F
	Decisions	T	F	T	F	F	F	F	F
Coverage	**Functional (4)**	{S, F, B}	{S, F, B}	{F, B}	{B}	{S, B, F}	{S,F,B}	{S,F,B}	{S, B}
	Intentional (5)	{}	{}	{G, GD}	{}	{G, R}	{}	{G, R, O}	{}
	Non-Functional (6)	{}	{}	{}	{}	{}	{}	{ (7) }	{TC}
Role	**Descritpive**	T	T	T	T	T	T	T	T
	Exploratory	T	F	T	F	F	F	T	F
	Explanatory	T	F	F	F	F	F	F	F
LifeSpan (8)		?	Persistent	Transient	Persistent	T & P	Transient	?	Persistent
Operation	**Capture**	Reuse	Scratch	Scratch	Scratch	Scratch	Scratch	Reuse	Scratch
	Integration	?	F	F	F	F & T	T	T	T
	Refinment	?	T	F	F	F	T	F	F
	Expansion	?	F	F	F	T	F	F	T
	Deletion	?	F	T	F	T & F	T	?	T

(1) T: Text, I: Image, G: Graphics, P: Prototype
(2) Any: any accepted value, I: Informal, S-F: Semi-Formal, F: Formal
(3) T: True, F: False
(4) S: Structure, F: Function, B: Behaviour
(5) G: Goal, GD: Goal decomposition, R: Responsibilities, O: Opportunity
(6) TC: Time Constraints
(7) {Performance, Time/Cost Constraints, User Support, Flexibility, Error Handling}
(8) "?" is used for indeterminable values

Table 5 : Classification of eight scenario approaches

[Benner,1992]

Rather than describing a particular scenario-based approach, [Benner, 1992] mentions the functionalities of three tools making use of scenarios and illustrates with the help of a traffic-control application four roles scenarios can have during the Requirements Engineering tasks. Benner et al. also give a definition of a scenario by highlighting various scenario characteristics, which results in some kind of informal meta-model of the notion of scenario.
Benner et al. do not address explicitely the issue of the various forms a scenario can take. Nevertheless examples are given which use a variety of media (text, image, prototype), a variety of notation formality degrees (from informal texts to formally specified scenarios) and a variety of presentations (static to animated, non interactive to advanced interactivity).

Even though the terms used are not exactly the same, the problem of abstraction is explicitely mentioned and the utility of having instance level, type level and mixed level scenarios depending on the purpose, the intended content etc. is discussed. Considering context, little is said about having information with a broader perspective than system internal and system interaction. Yet, sparse comments on some scenario examples suggest that there is an interest in having considerations that go beyong the strict application domain : organisational context level and even organisational environment level are implicitely addressed. Also implicitely, argumentation is present in the majority of the scenario examples and it is possible to find exerpts for all the four categories of statements we use in our framework (positions, arguments, issues and decision). Considering scenarios as being mainly partial descriptions of behaviours arising in restricted situations, Benner et al. focus on the functional nature of scenarios, neglecting the interest an intentional and/or non-functional content could have. Nevertheless, we have to admit that in some scenario examples they give for the traffic-control application, issues related to uncovering non-functional requirements, like safety, are raised as well as other issues related to intentional aspects. Coming back to the functional attribute, we have to precise that stress is put on behaviour and structure (situation) but, implicitely, the notion of function appears in examples.

The purpose scenarios can be used for are refered to as roles. Benner et al. identify four roles : (i) describing and clarifying the relevant properties of the application domain, (ii) uncovering system requirements, (iii) evaluating design alternatives and (iv) validating designs. (i) and (ii) are close to what we call the descriptive role of a scenario. (iii) is the equivalent to our exploratory role, while scenarios used for (iv) seem to be mainly explanatory and/or exploratory scenarios. The account for role is therefore considered explicit even if the terms used are not strictly the same as ours.

The issue of the lifespan of scenarios is not mentioned neither explicitely nor implicitely. And, among operations, only capture is given some attention by proposing to (re-)use a formalized body of domain knowledge for easing the task of capturing scenario descriptions. This knowledge would be made of building blocks classified according to their content (with the help of a terminological knowledge base).

[Gough et al. 1995]

Gough et al. present an extension of Jacobson's et al. [Jacobson et al., 1992] OOSE method based on use-cases. There were two principal extensions: (i) visual representation of the scenarios showing the temporal sequence and work flows between events and actions; and (ii) a hypertext visualisation of the problem domain. These two extensions are reflected in the classification. Scenarios can be described using text, images and prototypes. Scenarios can also be animated and interacted with using the hypertext links.

These two extensions reflect the changes to the original OOSE method. The rest of the classification is the same as for the original OOSE method.

[Holbrook, 1990]

Holbrook proposes a methodology to conduce requirements elicitation with the help of scenarios. This methodology is based on a conceptual architecture linking scenarios to goal and issue sets. It was implemented in an SBRE (Scenario Based Requirements Elicitation) tool supporting the hypertextual architecture proposed to help the stakeholders refine their understanding of the requirements. Holbrook's scenarios may as well be textual narratives illustrated by system interface snaphots as animated simulations.

A scenario is defined by Hollbrook as "a behavioural specification in that it captures how the system reacts to its environment". This narrow definition is enriched by the links to the issue set which give also some informations on the intended organizational context (issues, arguments, decisions), and to the goal set which extends the coverage from functional and behavioural aspects to refined goals. The definition leaves any consideration on the abstraction level out. We consider that scenarios can be at the type or instance levels. However in the Library System example, narratives and images give a mix of abstractions.

As stated in Section 3.3, Hollbrook's scenarios are mainly used in an exploratory purpose. They have to describe some aspects of the system design from which they are constructed, and form the document used to explore the specifications. After having been analysed and discussed, Holbrook's scenarios are deleted and thrown away. Only the issues which have emerged will be used for an other lap in the process cycle.

[Hsia et al. 1994]

Hsia et al. [Hsia et al., 1994] provide a traditional, formal approach to scenario generation and analysis. The scenarios are described using a formal language, hence their given medium and notation. However, unlike other formal approaches to scenario analysis (e.g. Albert [Dubois et al., 1994]), Hsia's et al. formal scenarios cannot be executed, although more traditional verification techniques are applicable to check that the scenario is consistent. The formal scenario modelling language enables the contents of the scenario to be both at the instance-level or the type-level, however the formal analyses applicable to these scenarios might exclude inclusion of instance-level and type-level information in the same scenario. The context of the scenario is a traditional one, focusing on the system interaction.
The scenario modelling language precludes description of relevant system internal and organisational knowledge. Argumentation structures are not included for the same reason. The principle coverage of the scenario is behaviour. As a result each scenario describes interaction between the system user rather than explains it or explores it. With regard to lifespan Hsia et al. provide a high-level process model demonstrating the role of their scenarios from requirements acquisition through to acceptance testing. Hence the scenarios are persistent. However the authors provide no explanation of how the scenarios can be managed during the requirements engineering process.

To summarise, Hsia et al. present a formal approach to scenario analysis. It is intended to provide assitance through the lifecycle of the requirements, although how this might occur is not explored. The focus of the scenarios is on behaviour during user-system interaction. However, the formal scenarios are executable, thus limiting their potential role for requirements exploration and validation.

[Regnell et al., 1995]

Regnell's et al. approach is an extension of the use case driven approach proposed by I. Jacobson [Jacobson et al., 1992]. By comparing to Jacobson's approach, several differences can be identified.

The approach proposes two levels of use cases. Use case descriptions give a more formal notation than Jacobson's use cases. Use case specifications are used to refine the use case descriptions. Events, conditions and problem domain objects are concepts underlying use case descriptions. Use case specifications use also time, atomic operations and abstract interface objects. Structured text and graphics are medium for describing respectively use case descriptions and use case specifications.

As in Jacobson's approach, Regnell's et al. use cases focuses on the system interaction with the environment. However, in use case specifications, system internal is captured through atomic operations.

In regard to the coverage facet, Regnell's et al. and Jacobson's et al. approaches are similar. However, Regnell's et al. use case descriptions capture explicitely actors' goals and responsabilities.

Use case descriptions have transient lifespan. They are only used to produce use case specifications. Use case specifications may serve the purpose of building a design model, so we classify use case specifications as persistent.

Use case descriptions are expanded to use cases specifications, so, the approach supports use cases expansion. The approach also supports the integration of use case specifications into a single usage view per actor. A usage view gethers all the use cases involving the same actor.

[Rumbaugh et al., 1991] (OMT)

In OMT [Rumbaugh et al., 1991] scenarios generated in textual form are prescribed to be a sequence of events. They are transformed in an event graph keeping a semi-formal representation; animation and interactivity are not supported.
In [Rumbaugh, 1994] Rumbaugh differentiates use cases from scenarios to include choices, iterations, parameters etc., whereas scenarios are particular case, a specific path or instance of the use case. Despite this differentiation, the involved entities of the use cases or the scenarios are described as roles and not individuals objects or actors. Therefore we classify OMT scenarios to type abstraction.
Rumbaugh et al. define scenarios just to describe interaction from the outside world with the system (setting the system boundary is the first step in his rules for scenario usage [Rumbaugh, 1994]). No system internal, as well as organisational context are described.
There are no argumentational aspects in Rumbaugh's et al. scenarios or use cases.
In 1994 Rumbaugh extends the usage of scenarios,which where so far used to build dynamic models, to “provide a first cut at the functionality in an application“. In [Koskimies et al., 1996] OMT scenarios are also used to create basic object models. We classify OMT scenario to cover

structural, functional and behavioural aspects of functional requirements. No support is given for intentional or non-functional contents.

The approach aims to use scenarios for complete descriptions of the existing or desired user-system interactions. Therefore it supports descriptive scenarios.

OMT scenarios have a transient lifespan. In [Rumbaugh, 1994] Rumbaugh even doubts in contrast to Jacobson the usefulness of scenarios for other software phases than the requirements phase. There's no reuse of scenarios described in Jacobson publications, but ways of integrating, refining and expanding scenarios by the means of use case association types.

[Scalzo, 1995]

B. Scalzo's approach [Scalzo, 1995] supports different semi-formal representations as well. She notates scenarios in predefined templates, in a *structured English* notation and so called user behaviour diagrams (same as OMT's event traces). Animation and interactivity are not mentioned.

Scalzo's scenarios are defined as interview forms which are filled out by an analyst questioning stakeholders who describe their particular activities or their roles in the organisation. This allows the conclusion that an instance or mixed level of abstraction can be used in Scalzo scenarios. Later when the sescenarios are reenginered by the analyst use BPR rules and principles, one can assume (although it is not stated explicitly) that not individual user, but their organisation roles (current or redefined role) are used.
The system boundary is not set in Scalzo's approach, therefore system internal information can be described in the scenarios. The templates of the interview forms hold a lot of attributes which are concerned to capture the scenarios context. Examples are: providers of work and resources, resources, preconditions, post-conditions etc. It is not discernible that the organisational environment is explicitly captured through specific attributes.
In addition the templates contain attributes which can be considered to be argumentational. The user is asked to state new ideas and how he thinks his task can be performed more effectively. Those kinds of statements will not be given without raising issue, giving positions (which are mainly the proposed ideas) and argumentation why the user thinks his propositions are better.
Scalzo describes how she creates structural, functional and behavioural models from scenarios. For the intentional attribute one can find again fitting attributes in the interview forms and non-functional requirements - like error trapping and prevention for the non-functional requirements concerning error handling - can be stated there as well.

The approach aims to use scenarios for complete descriptions of the existing or desired user-system interactions.Therefore it supports descriptive scenarios. The approach also uses scenario for reengineering activities. The results are newly created scenarios which will be shown to the user to discuss changes and optimisation issues. Those scenarios are clearly classified as exploratory.

Scalzo applies BPR principles and rules directly to task scenarios gaining new scenarios out of it, which can be classified as scenario reuse. Scenarios are integrated in a so called *scenario feature table*. Except of the reuse of scenarios for reengineering no other ways of refinement or expansion are described.

[Somé et al., 1996]

[Somé et al., 1996] proposes an approach that parses scenarios written in structured natural language and automatically integrates them as parts of a specification expressed with timed automata. At the moment, scenarios can only be seen through their textual representation but a prototype generator is under study that will allow to experience the behaviour they describe in a more animated and interactive way. The notation used is semi-formal. Indeed, it has a formal syntax but it is not always possible to interpret unambiguously a structured natural language input in terms of timed automata. In such cases, additional information is requested from the user.

Scenarios are at the type level : they refer to classes of real-world entities, and to their attributes and to the operations they perform. No particular object-oriented modelling technique is used but it is planned to do so in the future. The approach restricts itself to modelling the internal behaviour of a system and the interactions taking place with its immediate environment. Organisational or broader perspectives are not considered and no support is given to express argumentation neither. Clearly, internal behaviour and interactions are mainly adressed at the functional level. Nevertheless, scenarios or parts of scenarios are never viewed in terms of the functions they allow to achieve. Somé et al. borrow from [Benner,1992] the view that scenarios are meant to represent partial behaviours (possible successions of operations) taking place in restricted situations (states). Therefore, only structure and behaviour are considered. Intentional issues are not addressed and the only non-functional constraints treated are time constraints.

Somé's et al. scenarios are used for capturing requirements. They thus only serve a descriptive purpose.

>From the lifecycle point of view, the approach proposes persistent scenarios since these are supposed to be integrated in the requirements specification. No alernative to generating scenarios from scratch is given but other operations are supported : (i) scenario expansion is required when insufficient information is found during the parsing and the places where the information is missing are indicated by the tool ; (ii) pre-tracability links present in the timed automata support the deletion of scenarios by isolating the parts of the specification a specific scenario is responsible for ; (iii) integration is also supported by proposing three composition operators on scenarios : sequential, alternative and parallel compositions. Refinement is not supported.

Conclusion

Recently, Requirement Engineering, Information System, Software Engineering and Human-Computer Interaction communities have been considerably attracted by scenario usage. These communities propose a large variety of scenario based approaches emphasizing a more use-oriented perspective in developping computer systems. In this paper we have proposed a framework for the classification of scenario based approaches. Suggesting to consider scenarios as artefacts, the framework defines four views : the form, contents, purpose and lifecycle of scenarios. These views are further refined by a set of facets. The affectation of values to each of these facets helps characterize the position of the different approaches in the framework.

Applying the framework on eleven scenario based approaches has highlighted some frequent characteristics. To many authors, scenario should be used to describe a system behaviour through the interactions it has with its environment, and this by the means of a textual representation medium. The classification brought also some gaps up : often, scenario based approaches do not capture the goal dimension of RE. As well, argumentation on alternative opinions is seldom explicitely tackled in the approaches. Moreover, the approaches usually forget to define the process dimension of scenarios.Experience in applying the framework has raised difficulties linked to the lack of formal descriptions such as scenario meta models of the approaches.

Research issues emerging from these remarks are triple. First, the formalization of scenario approaches is an objective that should be aimed at. This would be at the same time profitable in the understanding and in the positioning of the different approaches. Again, as in RE, formal definition of usage processes could clearly emerge from such formal definitions. Second, more variety should be adressed in the application fields. Getting inspired by some common design approaches (e.g. goal oriented design) might make emerge approaches stretching the usual narrow view as above. The third issue stands in the need of practical evaluation of scenarios. The CREWS project is curretly addressing this issue. An industrial committee, fluent in the practice of scenarios is currently using the framework presented in this article in order to clarify and improve their practice of scenarios. At the same time, this experiment aims at a validation of our framework. Other research issues tackled by CREWS are : formalize and and automate the interpretation of textual scenarios, improve the use of original media such as video, enrich the context with a large coverage domain knowledge, and propose validation processes at a mixed level of abstraction.

References

[Anderson and Durney, 1993] 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.

[Anton et al., 1994]A. I. Anton, W. M. McCracken, C. Potts, "Goal Decomposition and Scenario Analysis in Business Process Reengineering", Proc. of the 6th Conference on Advanced Information Systems Engineering pp 94-104, Utrecht, The Netherlands, 1994.

[Benner et al., 1992] K. M. Benner, S. Feather, W. L. Johnson, L. A. Zorman, "Utilising Scenarios in the Software Development Process", IFIP WG 8.1 Working Conference on Information Systems Development Process, pp. 117-134, December, 1992.

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

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

[Caroll and Rosson, 1992] J. M. Carroll, M. B. Rosson, "Getting Around the Task-Artefact Cycle: How to Make Claims and Design By Scenario", ACM Transactions on Information Systems, 10 (2), pp. 181-212, April 1992.

[Carroll, 1995] 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.

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

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

[Conklin and Burges Yakemovic, 1991] J. E. Conklin and K. C.Burges Yakemovic, "A Process oriented Approach to design Rationale", Human Computer Interaction, 6(3-4):357-391, 1991.

[CREWS ISC, 1996] Crews Esprit project, Industrial Steering Commitee meeting slides, October, 1996.

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

[De Pauw et al., 1993] 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.

[Dubois et al., 1994] E. Dubois, P. Du Bois, F. Dubru, M. Petit, "Agent-Oriented Requirements Engineering a Case Study using the ALBERT Language", Proceedings of the Fourth International Working Conference on Dynamic Modelling and Information Systems - DYNMOD'94, Noordwijkerhout (the Netherlands), September 1994.

[Erickson, 1995] 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.

[Firesmith, 1994] 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, 1995] M. Glinz, "An integrated formal Model of Scenarios based on Statecharts", Lecture Notes in Computer Science '95 ,pages 254-271, 1995.

[Gough et al., 1995] 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.

[Gould, 1987] J. D. Gould, "How to design usable Systems", In Proc. Interact'87, H.J. Bullinger & B. Shackel (Eds), North-Holland, Amsterdam, 1987.

[Harel, 1987] D. Harel, "Statecharts: a Visual Formalism for Complex Systems", Sci. Computer Program, 8, pp. 231-274, 1987.

[Holbrook, 1990] C. H. Holbrrok_III, "A Scenario-Based Methodology for Conducting Requirement Elicitation", ACM SIGSOFT Software Engineering Notes, 15 (1), pp. 95-104, 1990.

[Hsia et al., 1994] P. Hsia, J. Samuel, J. Gao, D. Kung, Y. Toyoshima, C. Chen, "Formal Approach to Scenario Analysis", IEEE Software, pp. 33-41, 1994.

[Iivari, 1989] 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.

[Iivari, 1990] 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, 1991] 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, 1996] M. Jackson, "Software Requirements and Specifications", Addison Wesley, 1996.

[Jacobson et al., 1992] I. Jacobson, M. Christerson, P. Jonsson, G. Oevergaard, "Object Oriented Software Engineering: a Use Case Driven Approach", Addison-Wesley, 1992.

[Jacobson, 1995] I. Jacobson, "The Use Case Construct in Object-Oriented Software Engineering", In John M. Carroll, editor,Scenario-Based Design: Envisioning Work and Technology in System Development, pp 309-336. John Wiley and Sons, 1995.

[Jacobson et al., 1995] I. Jacobson, M. Ericsson, A. Jacobson, "The Object Advantage, Business Process Reengineering with Object Technology", Addison-Wesley Publishing Company, 1995.

[Johnson et al., 1995] 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 and Mossenbock, 1995] K. Koskimies, H. Mossenbock, "Scene: Using Scenario Diagrams and Active Text for illustrating Object-Oriented Programs", Proc. of ICSE-18, pp 366-375, 1995.

[Koskimies et al., 1996] K. Koskimies, T. Mannisto, T. Systa, and J. Tuomi. "On the Role of Scenarios in Object-Oriented Software Design". Technical report, Series of Publications A (A-1996-1), Department of Computer Science, Univerity of Tampere, Finland, 1 1996.

[Kuutti, 1995] 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, 1995] 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 and Theodoulidis, 1995] 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.

[Nardi, 1992] B. A. Nardi, "The Use of Scenarios in Design", SIGCHI Bulletin, 24(4).

[Olle et al., 1992] 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, 1996] K. Pohl. Process Centered Requirements Engineering.J. Wiley and Sons Ltd., 1996.

[Potts et al., 1994] C. Potts, K. Takahashi, A. I. Anton. "Inquiry-based Requirements Analysis". IEEE Software,11(2):21-32, 3 1994.

[Prieto-Diaz and Freeman, 1987] R. Prieto-Diaz, P. Freeman, "Classifying Software for Reusability", in IEEE Software, Vol 4 No 1, 1987.

[Rawsthorne, 1996] D. A. Rawsthorne, "Capturing functional Requirements through Object Interactions", In Proceedings of ICRE '96, pages 60-67. IEEE, 1996.

[Regnell et al., 1995] 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, 1995] 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 279-308, 1995.

[Rolland and Grosz, 1995] C. Rolland, G. Grosz, "A General Framework for Describing the Requirements Engineering Process", Intl. Conf. on Systems, Man and Cybernetics, San Antonio, Texas, USA, October 1994.

[Rosson and Carroll, 1995] 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 and Goldberg, 1992] K. S. Rubin and A. Goldberg, "Object Behaviour Analysis", Communications of the ACM, 35(9):48-62, 9 1992.

[Rumbaugh et al., 1991] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen, "Object-Oriented Modeling and Design", Prentice Hall, 1991.

[Rumbaugh, 1994] J. Rumbaugh. Getting started.Journal of Object-Oriented Programming, 7:8-23, 9 1994.

[Rumbaugh and Booch, 1991] J. Rumbaugh, G. Booch, "Unified Method", Notation Summary" Version 0.8, Rational Software Corporation, 1996.

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

[Somé et al., 1996] S. Somé, R. Dssouli, 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) pp 1110-1132, 1996.

[Takahashi et al., 1996] K. Takahashi, C. Potts, V. Kumar, K. Ota, J. D. Smith, "Hypermedia Support for Collaboration in Requirements Analysis", In Proc. of ICRE'96, 1996.

[Wexelblat, 1987] A. Wexelblat, "Report on Scenario Technology", MCC Technical Report STP-139-87, Microelectronics and Computer Technology, Corporation, Austin, Texas, 1987.

[Wirfs-Brock, 1995] 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 et al., 1994]D.P. Wood, M. G. Christel, S. M. Stevens, "A Multimedia Approach to Requirements Capture and Modelling", Proc. ICRE'94, Colorado Springs, 1994.

[Wright, 1992] P. Wright, "What's in a Scenario", SIGCHI Bulletin, Volume 24, Number 4, October 1992

[Young et al., 1987] 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 and Mylopoulos, 1994] 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.