Scenario Use in European Software Organizations - Results from Site Visits and Questionnaires*

M. Jarke, K. Pohl, P. Haumer, K. Weidenhaupt
RWTH Aachen

E. Dubois, P. Heymans
FUNDP Namur

C. Rolland, C. Ben Achour, C. Cauvet, J. Ralyté
Université Paris-1 Sorbonne

A. Sutcliffe, N.A.M. Maiden, S. Minocha
City University London

1 Introduction

A pervasive, but not well-studied phenomenon in every-day systems development is the use of examples, scenes, narrative descriptions of context, mock-ups and prototypes. Loosely all these ideas can be called scenario-based approaches, although an exact definition is not easy beyond saying that all these techniques emphasize some description of the real world. Recently such approaches have begun to attract considerable attention in Requirements Engineering (RE), Human Computer Interaction (HCI) and Information Systems (IS) research. While from the research point of view there are now first attempts to provide a comprehensive classification of the *state of the art* in scenario-based approaches [Rolland et al. 97], little is known on the practical relevance of these techniques, i.e. studies on the current *state of the practice* are still missing so far.

In this paper, we report on our insights into today's industrial practice gained during 15 site visits at industrial systems development projects and from a questionnaire disseminated to a larger number of organizations. These efforts have been carried out as part of the long term research ESPRIT project CREWS (Cooperative Requirements Engineering With Scenarios).

Our survey showed that usage of scenarios is widely adopted in industrial practice, but seldom well-defined and exploited systematically. Our study therefore aimed at

- making the implicit, but mostly not well-documented practices more explicit;
- examining how scenario-based techniques fit in the overall development process/current methodologies used in the industrial projects;
- determining the main benefits and problems of scenario-based approaches;
- providing an outlook to the future plans of the organizations and the needs research is expected to fulfill

* This work was in part founded by the European Community under ESPRIT Reactive Long Term Research project 21.903 CREWS

The rest of the paper is organized as follows: in section 2 we give an overview of the research method employed during the state of the practice survey. Next we present the results of the site visits in section 3. In section 4 we evaluate the results of the questionnaire.

2 Research Method

For our survey we pursued a two-pronged strategy for information elicitation. During February to June 1997 we conducted a series of individual site visits. Altogether, we examined 15 projects carried out at twelve organizations in Germany (6 projects), UK, France, and Belgium (each 3 projects). Based on the experiences gained from the first round of site visits, we developed a questionnaire whose purpose was to augment the findings of the site visits with more empirical evidence.

2.1 Site Visits

2.1.1 Selection of Interview Partners

Most of our interview partner were recruited directly or indirectly from the Industrial Steering Comitee (ISC) which accompanies the CREWS project in order to ensure the practical relevance of its project results. The ISC consists of 17 representatives from software companies or independent consultants from 6 European countries which have specific interest in Requirements Engineering in general and the application of scenario-based techniques in particular. The companies participating in the ISC vary considerably in terms of their size, application domain and RE method background, thus providing a broad basis for exploration of current scenario practice in different settings. Through the representatives of the companies we were able to establish a closer contact to our interview partners. The interview partners were project leaders or consultants of projects which already had employed scenario-based approaches and therefore had considerable experiences with such techniques.

2.1.2 Interview Strategy

The CREWS classification framework [Rolland et al. 97] was used to derive a catalogue of questions for scenario characterization according to the four views of form, purpose, content, and life-cycle. Besides these scenario characteristics the interview plan also addressed two other topics

1. *project profile*: to better understand scenario use in a broader context and to identify potential correlations between project and scenario characteristics
2. *experiences*: to elicit main benefits gained through the use of scenarios, and to capture known open problems and future needs.

One of the main problems we faced during our study is that up to now a clear and agreed understanding of the term scenario has not yet emerged, neither in the research communities nor in the industrial practice. Since we wanted to get insights in the whole spectrum of scenario-related techniques we adopted a very broad view of the term scenario, ranging from use cases to prototypes and systems animations to reference examples. Unfortunately, most of our interview partners were somewhat biased towards a specific meaning of the term scenario as inspired by the mainstream object-oriented textbook methods. For example, scenarios were frequently only considered as Use Case Models in the sense of Jacobsons' OOSE [Jacobson et al., 1992] or Message Trace Diagrams according to the OMT notation [Rumbaugh et al., 1991] or the new UML [Rational, 1997]. On the other hand our interview partners were not aware of other usage of what we would call scenario-based techniques like the systematic collection of filled-in forms, reports, lists or the use of prototypes in order to validate requirements. Nevertheless they often employed such techniques in their daily work practice.

Thus, in order to avoid a too narrow view we let the interview partners first talk relative freely about their overall development process and then ask concrete and deeper questions where we assumed usage of scenarios. We also avoided to use the term scenario explicitly but asked more indirect questions concerning what we expect is characteristic for scenarios like:

- How do you *communicate* about the current system and the future system with non-technical people like customers and end-users?
- Which role do *examples* play in the development process?
- How do you *elicit* and *validate* requirements?

By asking these questions we made the scope of our interest clear to the interview partners. We then could proceed with asking more concrete questions concerning the topics mentioned in the interview framework above.

2.2 Questionnaire

In addition to the site visits, a questionnaire was developed based on the first site visit experiences and distributed to a larger number of software organizations.

The questionnaire was structured according to the interview framework used during the site visits, but significantly coarsened to keep it as easy as possible. The question schema was as follows:

1. project profile
   a) In which application domain do you conduct projects?
   b) What is the typical project size?
   c) What overall systems development paradigm do you use?
2. Which scenario types are used?
3. What is the size and number of a typical scenario?
4. Which representation and media is used for scenarios?
5. Which tools and techniques are used for scenario generation, use and management?
6. phases and activities
   a) In which phases are scenario generated and used?
   b) More precisely, for which specific RE tasks are scenarios generated and used?
   c) Is the same scenario reused in different phases?
7. stakeholders
   a) Which stakeholders are involved in scenario generation and use?
   b) How many stakeholders are involved in scenario generation and use?
8. Which are the main problems in scenario generation, use, and management?
9. Which are the main benefits of scenario use?
10. Which are the future plan for scenario use?
11. Which is the own definition of the term scenario?

For questions (2), (4), (6a), (6b), (6c), (7a) we provided a set of default answers, which could be ranked in a scale ranging from 1 to 7, where 1 = no, 2 = rare, 3 = moderate, 4 = significant, 5 = frequent, 6 = very frequent, 7 = constant use, generation or involvement of the default item. The remaining questions were formulated as free-form questions. We will present an evaluation of the answer in section 4.

3 Site Visits Results

In this section, we report on the individual site visits. Table 1 characterizes the projects described in the following subsection in terms of their application domain, project size and if the project was performed in-house, by a software contractor or if we got our insights from a consultant of the project. The project size was classified into the categories *small* (less than 10 person years), *medium* (between 10 and 50 person years) and *large* (greater than 50 person years). According to our interview framework we first briefly sketch for each site visit the project (organization) profile and then highlight the main scenario characteristics as well as the scenario generation and usage process in the project and the experiences made. On request of (most of) our interview partners, we had to anonymize the project summaries.

no	application domain	size and duration	in-house/ contractor/consultant	section
1	medical information system	small	software contractor	3.1
2	network documentation and management	small	software contractor	3.2
3	business information system - banking	large	in-house	3.3
4	business information system - public authorities	large	consultant	3.4
5	business information system - insurance	small	consultant	3.5
6	satellite communication	medium	consultant	3.6
7	air traffic control system	medium	in-house	3.7
8	systems engineering for warships	large	software contractor	3.8
9	medical systems / consumer electronics	large	in-house	3.9
10	c/s applications for government and banks	large	consultant	3.10
11	radio telecommunication	medium	consultant	3.11
12	transportation	medium	software contractor	
13	CASE tools for bank applications	large	software contractor	3.12
14	water invoicing management system	small	software contractor	3.13
15	process engineering	medium	software contractor	3.14

Table 1: Project Characteristics.

3.1 Medical Information System

3.1.1 Project Background

The aim of the first project was to develop a medical information system which was delivered as a supply product for a portable measuring device for diabetes patients. The main functions of the software component were to keep a patient diary, to support the assessment of the patient's medical history and the evaluation of the patient data. The project team consisted of four trained developers. They had close interaction with 6-8 contact persons from the customer side. The customer representatives were mostly engineers of the hardware measuring device and physicians with a solid technical background and good abstraction abilities. No end-users were involved.

3.1.2 Scenario Characteristics

A number of different scenario-related artifacts were used in this projects: informal scripts of sample system applications, table-based usage scenarios in the form of class-responsibility-collaboration (CRC) cards, paper-based user interface forms, and executable user interface prototypes.

The most interesting insights gained from this project were how these various techniques were used in conjunction. On the one hand there was an elicitation chain in which one type of artifact was used to derive the next type of artifact, resulting in a requirements specification. On the other hand there was a validation chain where the former artifacts were reused for validation. Figure 1 shows the intertwined scenario generation and usage process which is described in more detail below.

Scenario Generation and Usage

The project started with a 20 page long project description provided by the customer. Based on the knowledge obtained from this document the project team identified main topics and developed sample usage scenarios for each topic in the form of informal prose texts. These scenario were discussed with the customer representatives in half-day meetings. Based on the feedback the project team revised the sample scenarios and structured them into a table-based representation (CRC cards). Again the CRC cards were discussed with the customer side and then transformed into a class model which was the central part of the requirements specification.

However it turned out that the class model in conjunction with the CRC cards and the informal sample scenarios was not sufficient to give the customer a clear overall impression of the future system. Therefore, user interface forms were developed for most classes which reflected the attributes and associations to other classes in the data fields and the methods for data manipulation in the buttons. Each of the forms was documented by describing the meaning of the data fields (attributes), plausibility checks and the effects of pressing a certain button. In the terminology of the project these form-centered system descriptions were called use cases. Although the use cases provided a much more natural view on the class model the customer still had no clear picture of the overall functionality of the system and its dynamics. The project team therefore developed an operational user interface prototype by which the usage sequence and navigation between different forms was illustrated. The courses of actions defined in the CRC cards were used to validate the prototype. Additionally, the sample usage scenarios developed at the beginning were used as source for concrete sample data.

The main tools used were the word processor and Visual Basic/Visual C++ for rapid UI prototype generation.

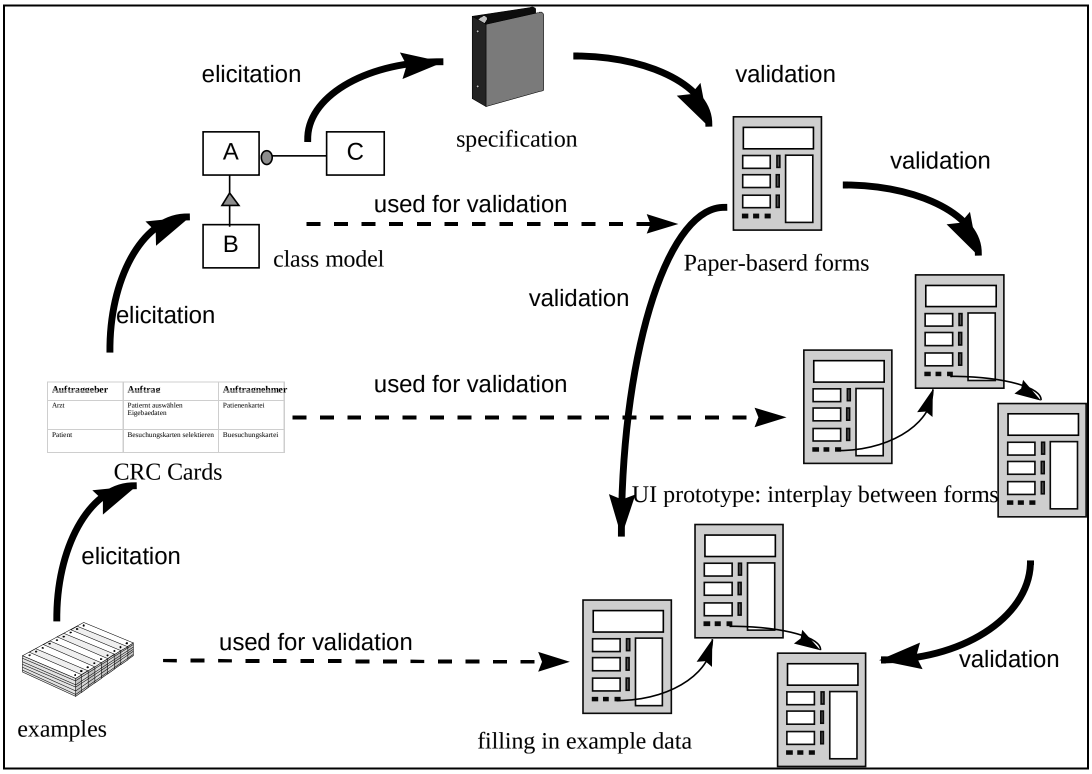

Figure 1: Scenario Generation and Usage Process

3.1.3 Experiences

Our interview partner pointed out that in general the customers mostly have problems to describe the desired system functionality on their own. Rather one should provide as concrete as possible suggestions to which the customer can react. Scenarios in the various forms described above are seen as a very good means to present and communicate such suggestions to the customer. Moreover the prototype was regarded as crucial for giving an impression of the overall functionality of the system and validating the object model.

Most problems with scenario usage faced in this project concern the management and the life-cycle of scenarios. There is a lack of guidelines and decision support on the granularity of a scenario. If the number grows the question arises which scenarios to keep and how to retrieve them for later usage. This is aggravated by the observation that the usage environment of the new system is most often a moving target so that typically only 70 percent of the scenarios developed in the early phases still reflect the reality adequately once the system is deployed. Hence it is questionable that maintaining also the scenarios is worth the effort, especially without sufficient tool support. The problems with change management and consistency between scenario artifacts, requirements specification and final system resulted in the fact, that scenarios loose importance in later phases (e.g. during testing or maintenance), i.e. they were mainly regarded as transient artifacts until the prototype was approved.

Summarizing, the most interesting observation in this project was how different scenario techniques were used in conjunction for both eliciting and validating the class model. First, informal scenarios of sample system usages and CRC cards were progressively used to populate the class model. When the prototype was built for validating the class model, the role of these early artifacts changed in that they acted now as a „guideline“ how to use the prototype for validation. Another important finding was that the class model is too abstract to give an impression of the overall system functionality. UI prototypes served as a user-friendly view on the abstract class model.

3.2 Network Documentation and Management

3.2.1 Project Background

The project described here takes places in the context of the privatization of the telecommunication market in Germany. The project aims at building an administration software for the management, monitoring and billing of the telecommunication services. The background is that the customer, a consortium of large companies, intends to use its network infrastructure for public telecommunication services. This involves a lot of new functionality about which the customer has nearly no prior experiences. Since nearly no processes were known, a mainly data-driven approach was adopted which was centered around the development of an extended ER diagram and its incremental transformation into an operational prototype.

The project team consists of six developers. The customer side is represented by three permanent contact persons and five additional experts. Most of the customer representatives are hardware engineers of the network devices but also end-user of the software system to be built. None of them had prior experience with requirements engineering methods in general, and EER modeling in particular.

3.2.2 Scenario Characteristics

Scenario-based techniques were used both for the elicitation of the data model and its validation. The process started with a first very coarse-grained project specification from the customer, mostly mere product descriptions of the existing hardware components (hubs, routers, communication lines, etc.) In interdisciplinary workshops the customer representatives explained their problems and provided sample data. For harmonizing the terminology a first mapping of the professional language of the customer („a net consists of hubs, routers, communication lines“) into the abstraction world of the developers was defined („a net is composed of edges and nodes“) and documented by example mappings. Based on the workshop results the project team developed an initial EER model. However, experiences from the early project phases that the customers were not to validate EER model since they did not understand the concepts of this language, e.g the semantics of inheritance and or relationship cardinalities.

For enabling a validation of the EER model by the customer the project team decided to adopt an incremental prototyping approach. A prototype generator was developed which takes the EER model stored in a repository as input and transforms it into a set of dialogues. This transformation is based on a formalized styleguide which defines the standard behavior of the generated prototype, e.g. how to display the entities in forms, which standard buttons to provide or how to navigate between the forms along the relationships between entities.

The prototype could now be used to validate the EER model in a more natural form. Typical problems now detected by the customers were for example that they wondered why a certain data field appeared in the data entry form of an entity: they had not understood that this attribute was inherited by a super entity. Another example was that the customers were amazed about a multiple selection window. This was caused by an 1:N relationship the corresponding entity had to another entity in the EER which the customers had not understood. These mistakes in the data model could only be detected by the customer through the prototype but not through the EER model itself.

Besides elemental changes in the data model, deviations from and enhancements of the generated standard functionality of the prototype were discussed with the customer during prototype demonstrations. Interestingly, such deviations were documented in the form of use cases centered around individual dialogues. However, these use cases served only as internal documentation for the developers.

3.2.3 Experiences

After bad experiences with the attempt to talk about the abstract EER data model with the customer directly, scenarios in the form of a prototype are now seen as an ideal means for validating the data model.

The project team tried to acquire scenarios in the form of sample system usage to be used test cases from the customers. However, it turned out as nearly impossible to convince the customers of the need to spend effort on this task. The reluctance of the customer was mainly caused by his vague understanding of the desired system functionality.

Summarizing, the specific feature of this project was that the system was totally new, i.e. did not replace a prior system, and none of the involved stakeholder had considerable experiences. Hence, it turned out that it is difficult to elicit scenarios (sample system usage, test cases) from customer if he has only a very vague impression of the desired functionality. Starting with an abstract data model, an operational prototype derived from this data model quickly became the central medium for communication with the customer. On the one hand the prototype provided a customer-understandable view on the data model for validation, on the other hand it was much easier for the customers to envision new concrete system functionality through the prototype. Such enhancements of the standard functionality of the prototype were documented in the form of use cases.

3.3 Banking Project

3.3.1 Project Background

The project described here is carried out by a service center for a group of about 450 banks in the south-west of Germany. The aim of the project is to develop an integrated office system to support all tasks and services in the banking sector. About 100 developers, mostly with a mainframe background, are involved in the project. It is the first one at the service center to be carried out according to an (inhouse) object-oriented methodology. The overall philosophy of this method is based on the leitmotif of a *well-equipped workplace for expert human work* and uses the design metaphors of *material*, *tools* and *automatons* by which the domain under consideration is structured. The overall chain of software artefacts used in the project is as depicted in figure 2.

3.3.2 Scenario Characteristics

The upper three types of artifacts are mainly used to elicit and communicate requirements with the end user.

Scenarios are the primary communication means between the developer and the user. By scenario a *short prose text* is meant, which describes the *current* situation at a *work place* using the professional language of the application domain. Scenarios are structured into three levels:

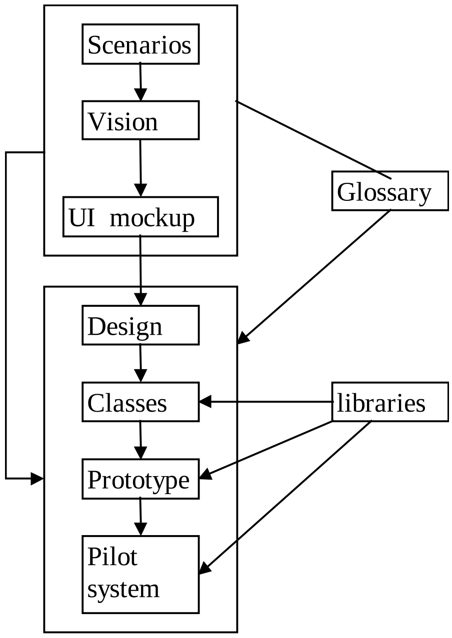

Figure 2: Chain of Artifacts in the Banking Project

- *overall scenarios* provide a coarse-grained picture of the tasks at a certain workplace. They focus on the goals of the tasks and not on the temporal sequences of certain steps within the task.
- *task scenarios* describe an individual task in the form of script. Decision points are used only sparsely, i.e. complex decisions are avoided and the scenario is rather spitted into two.
- *detailed studies of activities* describe in detail how a single activity/step is carried out, e.g. how to find a form in a folder

Task scenarios precise an overall scenario, while detailed studies precise steps occurring in task scenarios. All kinds of scenarios are on the type level.

System visions describe how the situation of a workplace is supported by a future system. Again, there are three levels:

- general vision provide an overview on the system functionality and describe to what extend tasks described in Scenarios will be dealt with support by the future system
- procedural visions describe how a task will be carried out in future with the aid of tools, automatons and materials
- component vision describe functionality of individual components independent of certain tasks,

Based on system visions, a first UI mockup is built which is discussed with the end-users.

Scenario Generation

The whole project is divided into a number sub projects. For each sub project a working group with the representatives of about 15 banks is established. The representatives are domain experts and end users of the future system. At the working places, interviews are conducted which build the basis for writing scenarios. The scenarios are communicated with the experts and validated in one or two feedback cycles. So far, about 230 scenarios have been developed.

Scenarios are also used in the form of Message Trace Diagrams, but only in the design documentation, i.e. at the technical level, to specify DB transactions between the new system and the legacy system.

Scenario Use

The understanding of the current tasks as described in the scenarios is transformed into system visions where the role of the future system is stressed. Based on the system visions, first UI mock-ups are developed and discussed with the end-users. The consideration of scenarios in this review should help to validate that each task currently supported is also adequately dealt with in the new system. Based on the feedback of the end-user, the system visions/prototypes are adapted. Scenarios, System Vision and UI prototype build the basis for the design.

There exists an interesting bi-directional relationship between a project-wide glossary and the scenarios/visions. When creating a new scenario, the developer has to take the terminology as defined in a project wide glossary into account. Conversely, each glossary term holds a reference to the scenarios in which the term is used. As a result, terms are not only defined by an abstract circumscription but also illustrated by concrete usage examples. This helps all involved stakeholder to adjust their interpretation of the key terms. Technical, the relation between scenarios and glossaries is established by a hypertext infrastructure in a project-wide intranet.

There is currently no systematic use of dedicated CASE tools, mainly because for their development platform (OS/2) suitable ones do not exist. The main tool is therefore the word processor.

3.3.3 Experiences

Scenarios, system visions and UI mock-ups were seen as an ideal and sufficient communication medium with the domain experts in the early stages of the project, especially in conjunction with the glossary.

The amount of scenarios produced was seen as major problem although a three level structuring had been introduced. The project also lacks methodical and technical support to ensure traceability between the different documents. The re-use of scenarios as test cases was seen as interesting opportunity, but is currently not systematically exploited.

Summarizing, the main intention of using scenarios in this project is bridging the gap between the developers' world and the customers'/end users' world by an emphasis on usage of professional language of the domain. Thus, generation of scenarios, i.e. the documentation of the current system behavior, is mainly regarded as a domain knowledge acquisition technique for the developers. Additionally, by reviewing scenarios through the subject matter experts and by the project-wide glossary, it is an instrument to ensure that developers and customers have the same understanding of the work processes to be supported by the system. It was interesting to observe, that in this project a clear separation between current and desired system behavior was made by distinguishing scenarios from system visions. The amount of scenario artifacts produced calls for hierarchically structure scenarios and system visions into three levels of detail.

3.4 Public Authorities Project

3.4.1 Project Background

The aim of the project described here is to build a common software to be used by all authorities of a specific sector in the 16 German States. The project started in 1994 and is expected to be finished by the year 2003. The whole project involves more than 100 people. These people are split up into 25 groups consisting of 3-7 people each. 10 of them deal with technical matters, while the other 15 are in charge of the domain analysis of specific topics like personal data capture, fines etc. Most team members are recruited from the data processing departments of the authorities. Although they mostly have an administrative education, they are not involved in the daily business processes.

The project started with defining a coarse-grained class model. These effort were stopped as it became apparently that the complexity was too high and the business processes were not sufficiently understood. System analysis is now carried out according to an object-oriented in-house methodology which emphasizes the need for careful business process analysis. The methodology, a mixture of Use Cases analysis according to [Jacobson et al, 1992], business object modeling and modeling notations borrowed from OMT/UML [Rumbaugh et al., 1991; Rational, 1997], is written down in central guideline and tailored towards the specific needs of that administration sector. The work on this guideline is still in progress and it evolves as the project progresses. Six kinds of models which are developed more or less sequentially are intended in the guideline: (1) business process model (Use Cases); (2) Resource/Responsibility Models (a preliminary, informal stage of the class model); (3) Class Model (according to OMT, now UML); (4) User Interaction Model (forms, UI prototypes); (5) State Model (StateCharts); (6) Interaction Model (Message Trace Diagrams). For the time being, the project is still in the first phase, i.e. business process modeling.

3.4.2 Scenario Characteristics

In the first phase of the project identifying and writing use cases is the key means to get an understanding of the business processes to be supported. In fact, in this project the terms *use case* and *business process* are used synonymously.

Use cases are written down in natural language text according to a quite rigid template. The template consists of several sections of which the most important ones are *event*, *pre-condition*, *actions*, *exceptions* and *postconditions*. Although the actions can contain iterations and alternatives, this section is normally a sequence of request/response pairs in the form of ”The user does .... - The system does ...”. Hence, the actions section explicitly define the system boundaries. Use case descriptions are on the type level and describe the desired system, not the current system.

A use case description is typically 3-8 pages long of which the half is devoted the actions section. As an overview, use case diagrams are used which can be packaged to use case groups. Up to now, 350 use cases have been developed. In average, each use cases takes 3 man-days to be identified and documented.

Besides use cases, the term *scenario* is used for descriptions of concrete instances of business processes how they are carried out *currently*. Scenarios are documented informally according to a similar template like the use case descriptions. The guideline recommends to identify and document a small number of different scenarios in parallel to capturing and abstracting their essence into a use case. It is highly recommended to augment the scenario with *filled-in* forms, files, lists, reports etc which are used in the scenario.

Scenario Generation

The development of use cases and scenario is as follows. A team member identifies a small number of relevant topics according to his domain knowledge and pre-structures them into business processes but without elaborating a full use case description at this point. He/she then conducts interviews concerning the identified topics with subject matter experts in the offices. During the interview the team member collects basic information concerning the topics (workflow descriptions, filled-in forms, print-out of reports, ...), sketches scenarios together with the expert and assigns this information to possible use cases. Based on this information the team member elaborates a use case description for the business process observed in the office. This use case description is primarily reviewed by subject matter experts in the office. As a means for validation, the scenarios of the current work flows are used to ensure that all functionality provided by the current system is also considered in the future system. Additionally, so-called communication partners in each of the 16 states in order to ensure that the use case is valid in each state (important because of differences in laws). A quality assurance group checks formal conformance of the use case description to the guideline (but not the content).

Each team develops its use case models on its own. Reuse is supported by a special reuse team which maintains a library of all published and preliminary use cases. Publishing and changing use case models can only be done according to a formal policy which is enforced among the different topic teams by the reuse team. In order to harmonize the interfaces between use cases shared by different teams, monthly bi/trilateral developer conferences take place when the need for clarifying their interfaces is identified.

Scenario Use

Writing use cases is seen as the central activity to determine the functional requirements of the system. The guideline explicitly mentions that everything missing in the use case model will not be considered in later development phases. The integration of the 350 individual use cases into a coherent model is still to be done.

The use case model is the basis for the Resource/Responsibility model (an informal pre-stage to the class model) and the guideline provides advice how to transform a use case into such a model. However, since the project is not yet in this stage, there are no experiences available.

Up to now, the CASE tool Paradigm Plus in conjunction with Word is in use. Rational Rose is currently evaluated.

3.4.3 Experiences

After having made bad experiences with starting directly with the class model, use case models are now seen as a viable way to manage the complexity of the domain. End users in the offices liked to talk about use case descriptions, while it was sufficient for managers to reason about the graphical use case diagrams in order to get an overview. Although intended by the guideline, concrete scenarios played a minor role. End user were mostly able to understand the type-level UC descriptions of the future system.

Traceability is explicitly mentioned in the standard as a key factor for maintainability of the scenarios, the existing artifacts, the UC models, their later usage and their different relationships. Yet, there is no technical means for ensuring traceability. Hence, the standard as well as the QA group appeal explicitly to the discipline of the developer teams.

Summarizing, in this project the terms use case and business process were used interchangeably (but considered at a fine-granular level). Use case modelling is therefore mainly seen as a problem understanding activity to make the developers familiar with the business processes to be supported by the future system. It was explicitly seen by our interview partner as an informal technique which is only necessary in complex domains; in simpler situations it was recommended to start with data model from scratch. However, in this project use cases act as main documentation of requirements and thus take a very central role in the RE process as basis for all later models.

3.5 Life Insurance Project

3.5.1 Project Background

The aim of the project is to replace the old software for life insurances by a new, modern system. The general process model is the V-model (official model of the German authorities). The project originally started with a business process analysis supported by a Petrinet-based tool. These efforts failed since petrinets were neither accepted by the developers nor by the end-users so it was stopped after six months. The business process analysis is now carried out by identifying and writing use cases. In contrast to public authorities project, the further development method is not object-oriented, but structured (SERM). The project team consists of 2-4 (changing) people. 70 % of the man power comes from the data processing department of the insurance company while 30 % comes from experts of the several departments.

3.5.2 Scenario Characteristics

Scenario Generation

As also mentioned by other interviewees, the hardest part is the identification of the Ucs. In this project, the identification of UCs did not start from scratch but could rely on more than 100 workflow identified earlier using the petrinet-based method. Since these workflows were regarded as too fine-grained, they have been coarsened to 27 business processes. Up to now, 5 business processes have been fully specified and documented in a single 40-50 page long document. Such a document consists of a UC description (similar to the one of public authorities project, but less structured) and about five instance-level scenarios for the most likely threads through a use case (also written as narrative text). At this stage it was pointed out that scenario exploration should be focussed on *normal* behavior in order to reduce complexity. The domain expert involved in the team was asked for these typical threads. A critique mentioned by interviewee is, that at the moment it is not always clear if the use cases/scenarios describe the current or the future system.

Scenario Use

Together with the domain expert each use case is checked against the scenarios. Then, for each use case a set of DFD models is derived. More precisely, each instance-scenario is transformed into a single DFD which is quite straightforward. Using the techniques of McMenamin/Palmer [McMenamin & Palmer, 1984] the different DFDs for each scenario are integrated with each other and their essence is extracted. In parallel, extended ER models are gained from the UC description/scenario descriptions. So far, 5 business processes have been documented in this way taking 3-6 months each. The models produced are only integrated for a single use case but not across use cases, but re-use and integration will increase as the team is more trained and has aquired more domain knowledge. The scenarios are not thrown away, but will be used later as test cases.

The Innovator tool is used which supports the generation and documentation of all models currently needed.

3.5.3 Experiences

Like the public authorities project, the project is in an early stage, so there are no definite experiences yet. Up to now, it can be said that the informal UC models of the business processes are much better suited to model, comprehend and communicate requirements than the petrinet models used in the first try.

Our interviewee pointed out that UC models provide enough structure to manage the complexity but are informal enough to be used as a communication medium with all stakeholders.

Summarizing, the use of scenarios in the project was quite similar to the public authorities project described in section 3.4. However, although normally seen in the context of object-oriented methods, use case modelling is here used as „front-end“ for a process employing a structured analysis method.

3.6 Satellite Communications

3.6.1 Project Background

The site visit described here took place at a large German service provider and consultant firm. Our interview partner works as consultant and has applied a use case driven approach in different domains like energy suppliers, public authorities, financial information systems, control system, and telecommunication. Although the talk mainly dealt with their scenario use in general, he mostly referred to a satellite communications project when talking about use case examples. This project was a medium-sized one (15 person years).

3.6.2 Scenario Characteristics

Scenarios are used in the form of Jacobson use cases. The generation process is as follows: At first during half-day interdisciplinary workshops a coarse-grained use case diagram is developed by identifying the potential system usages. Again it was pointed out that finding a use case and determining the right granularity is the most difficult task. The initial use case diagram is communicated with the management of the customer and use cases are grouped into logical use case groups. Moreover the use cases are classified as so-called primary or secondary use case. While the former ones are essential for fulfilling the actual purpose of the system, the latter ones are considered as supplementary. An example for a primary use case in the satellite communication project was „provide voice link“ while the use case „billing“ was regarded as a secondary one. This distinction proved to be very useful for prioritizing efforts to define coarse-grained project plans and for contract purposes to define deliverable stages.

After an initial set of use cases has been defined, for each one a use case description is elaborated in close collaboration with domain experts (interviews and review sessions). Use case descriptions are structured according to a template providing sections for name, goal, event, actor, input, output, pre/postcondition, action etc. The most important use cases are augmented with additional user interface forms/prototypes.

Although seen as a very important task, our interview partner pointed out that this phase should be kept considerably short (3-4 weeks at most). In particular he recommended to concentrate on the most likely threads through a use case since most customers tend to have hundreds of exceptional cases in mind but do not see the essential characteristics of a specific use case. As a thumb of rule, a use case description should be restricted to five pages.

In later phases of the project, the description of system-internal interactions are described with use cases as well. Depending on the project context, use cases are transformed into different, more formal notations, for example message trace diagrams or data flow diagrams. It was interesting to observe that the use of use cases did not depend on the overall modelling paradigm used in the project.. Rather it was regarded as a universal front end for varying methods.

The main tools used is Word. Software Through Pictures and Rational Rose are also in use although they suffer from some restrictions and limitations.

3.6.3 Experiences

A general experience was that the usage-oriented decomposition of the overall system functionality provides a natural way to communicate requirements with the customer. It was pointed out that most customers explicitly preferred to talk about use cases rather than about ER, DFD or MTD models. Moreover the dichotomy between the graphical use case diagram and the more detailed use case description was seen as advantageous. While the use case diagram provides enough information for the management to get a rich enough picture of the system, the use case descriptions serve as communication medium for eliciting and validating requirements with the domain experts. Another interesting observation was the universal applicability of use case modelling regardless of the problem domain or overall system development paradigm.

As a vision for future system development our interview partner envisaged a component market of executable software components which are directly related to typical use cases of a problem domain, thereby shortening the long chain of different models from the requirements phase to the implementation phase.

3.7 Air Traffic Control Project

3.7.1 Project Background

The aim of the this project is to define and implement an integrated air traffic control system across the 35 countries of the European continent which makes national differences between regional and national control systems invisible to airspace users such as pilots. The project is a complex long-term one with a projected completion date in 2015. Stakeholders include national governments who have to fund the project, national bodies such as the UK's Civil Aviation Authority, national and international pilot associations, balloonists, microglider users and even meteorologists. The project is still in its earliest stages. It has produced documents outlining the basic concepts and high-level 'wish-list' requirements for the system. However no thorough requirements engineering process has taken place.

Two companies from UK and France have been contracted by EuroControl to undertake a prototype first-pass requirements acquisition task. The purpose of this task is two-fold. First it will produce a more complete and correct requirement specification, although this will still be too high-level to enable system specification and design. Second it will provide feedback about how to undertake EuroControl's complex and political requirements acquisition process. The two companies are using scenarios to undertake this requirements acquisition. The one is developing scenarios which the other will use to acquire user and system requirements over a period of 3 months, and in particular to populate a large and complex object model, identifying the required class operations and attributes.

3.7.2 Scenario Characteristics

Each scenario begins at a high level but is decomposed and to include strategic, operational and tactical tasks, and even behavioral requirements for the software system. One example is a scenario which explores how and when restricted (e.g. military) airspace is made available to non-restricted access by commercial aircraft. The scenario includes details of national policies and procedures for making such airspace available, procedures to make airspace available on a weekly or daily basis and tactical procedures to make airspace available on request from an aircraft in flight. This scenario also includes specific behavior requirements for the required system, software or otherwise. At the moment the coverage of these scenarios is ad hoc, due to the prototypical nature of the project. However there will more serious issues of scenario coverage in the future stages of the project.

Scenario Generation

In this prototype study between 10-15 scenarios have been developed by. These scenarios were developed during brainstorming meetings by the scenario development team, then extended using available sources of information from documents. Each scenario is described at first using natural language, then transformed into OOSE use-case models and finally UML message sequence diagrams.

Scenario Use

Walkthroughs of the scenarios themselves will be conducted to improve the object models, and it is possible that some later walkthroughs will be undertaken with pilots. Thus, in near future scenarios will be used to “illuminate”, validate and improve a complex domain-specific object model currently containing 100s of class objects for air traffic control. One interesting use is that scenarios are being used in conjunction with the Open Distributed Processing (ODP) approach which has been adopted by the whole project. The consequence is that each scenario will be designed to explore one or more of the 5 viewpoints on system modeling advocated in ODP. By cutting across different viewpoints, thus each scenario can act as a device for integrating viewpoints and detecting inconsistencies, conflicts etc. Although at present there is little intention to use the scenarios with large numbers of diverse stakeholders, the potential of scenarios acting as easy-to-understand shared artifacts to facilitate negotiation was acknowledged.

The scenarios are developed, used and managed using the UK's SELECT use-case and requirements management toolkit.

3.7.3 Experiences

One of the major problems to be addressed in the full project is the large number of diverse stakeholders, most of whom will have conflicting requirements. It is hoped that scenarios can help resolve such conflicts.

One reason for the current pilot study is the acknowledged difficulties in acquiring requirements for such a complex system. Scenarios are recognized as one part of the solution, however scenario use on such a scale introduces its own problems. The contractors to this study are exploring and learning about these problems at the moment. We hope to report more on their experiences in the near future.

Summarizing, this project is an example for scenario usage in the context of very a large and complex control system. Although the project is in its earliest stage it exhibits some interesting characteristics. There are two separate groups who develop scenarios (French company) and use scenarios (British company); this stresses the fact that scenario documents must be communicable. The scenario documents produced undergo a typical transformation from informal narrative text to use case models to more formal interaction diagrams. Most interestingly, the use of scenario is envisaged as a vehicle for integrating and negotiating viewpoints. The lack of suitable methods and tools for scenario generation and use in large scale is expected to become a major problem in future.

3.8 Systems Engineering for War Ships

3.8.1 Project Background

The project described here is carried out at a large ship builder which develops large warships including the future UK aircraft carriers and support vessels. One of the major problems for the systems engineering teams is to determine the requirements for these ships from the requirements documents provided to them once the building contract has been secured.

Requirements specification for new platforms takes place in the context of the company's reusable logical model of platform environment and behaviors. In essence, these models, developed using Yourdon notations, include large amounts of data which are reusable when specifying different platforms. The premise here is that there are many similarities between the requirements for different platforms, and these similarities allow for extensive reuse during requirements specification. The interviewees estimated that over 80-person years of effort had gone into the development of these models.

Requirements specification and management and reuse from the generic domain models takes place using the IME toolkit. The toolkit contains models of generic behavior, including libraries of reusable behavior models and reusable equipment models.

Models developed with the IME toolkit can be divided into those which take a 'static' view of the whole system and those which take a 'dynamic' view of the system. Static views are encompassed in the environment model, behavioral model and system function model. Dynamic views are encompassed in scenario modeling, operational and performance modeling and systems effectiveness performance. The typical sequence of model development based on the generic domain models is: (1) environment model (static), (2) scenario model (dynamic), (3) behavioral model (static), (4) operational/performance model (dynamic), (5) system function model (static), (6) systems effectiveness performance (dynamic) and so on.

However the organization has identified a need to improve the dynamic modeling side of their model development. For example current use of scenarios for requirements identification is small despite its prominent position in the development process above.

3.8.2 Scenario Characteristics

Current problems with scenario development and use have led to a prototype tool called the operational concept demonstrator. The aim of this tool to enable the development and animation of complex scenarios of naval warfare to gather requirements for new platforms. These scenarios can be developed for different levels of 'system'. At the highest level the platform is treated as a black box within its environment (the task force, sea, enemy attacks etc) and the scenarios provide a basis for exploring requirements for the entire platform. These scenarios are referred to as either operational scenarios or operational situations. At another level this platform can be treated as a white box and scenarios are developed to explore interactions between subsystems (e.g. propulsion, weapons etc) on the platform. It is anticipated to use the same approach to decompose each of these subsystems to individual pieces of equipment and their use by personnel.

Let us consider development of the operational scenarios in more detail. Operational scenarios are developed against the background of rich domain models containing information about all aspects of the environment, for example different types of friendly and hostile ships, submarines and aircraft, and their permissible and typical behaviours, typical task force formations, environmental information and so on. Information in this domain model enables the user to define and animate ship behavior in a scenario.

The demonstrator provides a map-like visualization of the scenario which the user can manipulate directly with the place or move ships, enter new incoming missiles or remove any element from the scenario. The user has control over time in the scenario and is able to speed up, slow down, freeze or reverse the animation as needed to elicit requirements.

The organization intends to use this prototype demonstrator with sales staff and systems analysts to capture platform requirements directly into the IME toolkit.

A prototype white-box demonstrator has also been developed to explore requirements for main platform sub-systems. The main focus of this prototype is to animate and simulate communications between the sub-systems and explore critical message highways on the platform under various external conditions, for example how the satellite and weapons systems communicate under various circumstances. Again these scenarios can be developed through selection of existing equipment and system models from the generic domain models. This enables rapid animation without excessive input from the user in the first place. It is also possible to use such a scenario to explore, for example, impact analyses on changes and upgrades to equipment on the platform. These scenarios are often intended to provide a basis for further exploration of requirements away from the scenario itself.

3.8.3 Experiences

One important requirement mentioned is the need for guidance embedded in process models for scenario development and use. There is currently a lack of such guidance. Another requirement is a single repository for requirements, scenarios and the complex links between requirements and scenarios. This indicated the need for a classification of scenarios to inform design of such a repository. Such a classification would also provide a basis for developing multiple views on single scenarios.

Another important requirement is the need for decision support to assist stakeholders to use the complex scenarios. In particular there is a problem in making tradeoffs between decisions about requirements and features of a scenario which is in need of method and software tool support. This led on to a discussion of the importance of risk attached to requirements which should be considered during a scenario analysis.

Summarizing, scenario use in this project is an example for very large, rich and complex executable scenarios of naval warfare. Their purpose is mainly validating performance requirements. A specific feature is the reliance on domain models to develop and execute scenarios and the close integration with structured analysis methods.

3.9 Medical Systems/Consumer Electronics.

3.9.1 Project Background

The company sketched in this section uses scenarios in two application domains: medical systems and consumer electronics. The medical systems department has been using scenarios and use cases to develop software-intensive systems since 1993. Scenarios are used to develop systems in two main application areas. The first is medical systems, such as image acquisition and viewing systems for x-ray machines. The second is consumer electronic products such as videos, televisions and so on. Since 1993 there has been a steady increase in the use of scenarios. Scenarios are now seen to be the one of the mainstream development techniques. One reason for this is their fit with the OMT method which has superceded use of structured analysis notations in the application areas mentioned.

3.9.2 Scenario Characteristics

The terms scenario and use case are used interchangeably to cover the range of scenarios developed at Philips. Most are procedural scenarios containing structured text, images, tables and pictures. The top-level scenarios are quite large - one that we studied was 14 pages of small-font print A4. The structure of the scenario was defined using a predefined template. This template ensured the inclusion of features such as a lexicon for the scenario as well as references to other models such as object models. A typical large scenario would include between 5 and 10 main procedures. Each procedure would in turn include on average about 40-50 individual user functions, therefore a typical scenario can include on average about 250 user functions. For highly integrated systems this number is often larger.

A typical template structure for a scenario is as follows, although this structure is not always adhered to:

1) Change History
2) References
3) Brief Description
4) Flow of events - pre & post conditions
    - normal flow
    - exceptions
    - parallel events
5) Other requirements
6) Constraints
7) Remarks
8) Object descriptions in the object model.

One of the most interesting findings from the site visit was the blurring of the difference between scenario development and scenario use and the stakeholders involved in both phases. Different stakeholders often take different views on the scenario, and impose different needs on it. For example commercial stakeholders are less interested in the formal flow through the scenario than are technical developers. These different views have important implications for scenario development, use and management.

Scenario Generation

The scenarios themselves were always developed by trained systems developers, in conjunction with other stakeholders, mostly based on a commercial requirements specification arising from market analysis. Typically the development of a scenario would start when a developer sits down with 4 or 5 other stakeholders. In medical systems these stakeholders are often subject matter experts. An informal, initial version of the scenario would be developed through the use of other acquisition techniques such as brainstorming. This implies a considerable overlap between scenario development and requirements acquisition. The developer(s) then works alone to produce a more complete and precise version of the scenario. The scenario undergoes internal developer reviews before review by other stakeholders. For medical system applications these reviewers are subject matter experts. For electronic consumer products the reviewers are from the commercial side of Philips, however the normal maximum number of reviewers in one session is 5. Sometimes product managers or marketing experts become involved later in the process, although this is not normal practice.

Scenario use

Scenarios are mainly used to derive and explain requirements for new system features. As was reported already, requirements acquisition takes place during scenario development activities and requirements validation takes place during scenario review activities. As a result the resulting 'complete' scenario is more a systems specification for system development than a further requirements engineering document, as implied by the close links between the scenario and the OMT object model.

In this situation the scenarios are used to validate the object model as proposed by current object-oriented analysis methods, and scenario and object model development take place in parallel. At the end of the development process most scenarios are archived for future reference during testing, maintenance and documentation. For example, scenarios for future television use are kept for reuse when future new televisions are needed. Sometimes the scenarios is used for end-user training when the system is complete and available for use.

Software Tool Support

A number of software tools to develop and use scenarios are in use. The current main tool is StP (Software Through Pictures). StP provides important links to other tools such as Framemaker (which in turn provides useful capabilities for inclusion of hypermedia documents), MS Word (for consumer electronic products) and Rational's Objectory tool. They make strong use of templates in products such as Framemaker and Word to structure documents to standards, however these facilities are not used to structure scenarios themselves.

3.9.3 Experiences

A number of important current problems were identified and requirements to solve these problems were expressed. The most important problem was that scenario development was seen as a craft without systematic guidance. More guidelines, heuristics and manual procedures to guide the development process are needed. These guidelines must be built on a sound process model for scenario development. Current processes such as that from Jacobson's OOSE are too flexible and open to (mis)interpretation.

Furthermore, since requirements acquisition is interleaved with scenario development, there is a lack of knowledge about which acquisition questions to ask at what point during the scenario development process. Yet another problem is determining the most effective size or granularity of each scenario? Hence, the process model must also improve the coverage of scenarios to ensure that all requirements are covered at some point during the scenario analysis without developing the scenarios into too much detail. Guidelines for prioritising these requirements, and there corresponding user functions in the scenario, are needed.

Furthermore the process model must also guide the developer to link the scenario with other important artefacts from the development process such as user interface designs and testing documents. These fundamental problems are exacerbated by a lack of useful software tools with which to develop and manage scenarios. This scenario management problem is compounded by current difficulties identifying and managing inconsistencies with OMT object models developed for the problem domain.

As a consequence the need for a cohesive framework to manage scenarios was identified which would facilitate better forward and backward traceability throughout the process. This framework should provide intelligent assistance for structuring and tracking scenarios. It was seen that the application of more formalization to scenarios would provide the foundations for such a framework. In the framework it is perceived that scenarios will provide the glue linking other models arising from the systems development process. Another requirement is the need to promote scenario reuse across application domains as well as in vertical domains such as televisions. This could be achieved through such a framework. The use of the intranet is seen as an important potential tool for developing the framework.

A wider question was arose, and which the interviewees wanted an answer to, was whether the current scenario methods were usable and integratable with current working practices.

Summarizing, scenarios in the form of use cases is mainly used as structuring device for requirements so no clear distinction between use case and the requirement specification can be made. Use case description are large and complex documents predefined structure, parts etc. The scenario development and use process stresses the involvement of domain experts. The lack of heuristics and guidelines for scenario generation and use is seen as a main problem.

3.10 Client-Server Business Applications

3.10.1 Project Background

Most of the applications developed at company described here are Client-Server, distributed (Intranet) business applications. The customers are typically bank companies or government administrations. The usual size of a project is between 100 and 2000 men-days. The development teams are not distributed.

3.10.2 Scenario Characteristics

From a methodological point of view, the company works on the basis of a methodology established by the DSDM consortium (of which they are part). This general methodology (called SystemsCRAFT) has an object-oriented software development methodology component. The main phases of this component are :

1. Initial Study : this has to be done in a short time (4 days to a few weeks) and involves writing user requirements (URs) and system requirements (SRs). For both types of requirements natural language, obeying a PSS-05-like standard, is used (SRs also include use cases) and traceability matrices for checking completeness of SRs wrt. URs. It must also produce as an output a planning and estimation of costs for the development phases to come.
2. Global Business Analysis: here OMT is applied to identify domain objects which are linked to use cases which themselves result from refinement of use cases of the previous phase refined with *extends* and *uses* links. This results in Object Interaction Diagrams. (OMT is planned to be replaced by UML in the future.)
3. Global System Design : the constraint requirements (as opposed to capability requirements) identified in the first phases are here used to put the domain objects of phase (2) in a concrete environment. The output of this phase is a system architecture.
4. Detailed design and Coding : here, the work is divided by subsystem. For each subsystem, a RAD lifecycle approach is adopted i.e. subsystems are made by incremental prototyping. Build plan and test plans are also part of this phase.
5. Integration Testing
6. Acceptance Testing (reusing use cases of (1) and (2))

Hence, both informal (semi-structured text) and semi-formal (Use Cases and Object Interaction Diagrams) scenarios are used. In later phases also incremental prototyping techniques are employed.

The scenarios produced are mainly used to specify requirements and to validate them with customer in an explanatory way. Scenarios are also seen useful to explore possible alternative system requirements in phase 1. There is not really use of scenarios for elicitation but there is interest in that one of the main problems presented by the company is “ How to ask the right questions to customers ? ”.

Tool support is provided by the Select Entreprise CASE tool which is a competitor of Rational Rose. The advantage of Select Entreprise wrt. Rational Rose seems to be a better guidance to its usage and a better compatibility with their methodology. The main functionalities of the tool are repository management, process management, editing of object models, use cases, interaction diagrams, etc. An associated traceability tool (Select Tracer) is currently being studied for future use.

3.10.3 Experiences

Scenarios is the form of both Use Cases and Message Sequence Charts were regarded as good communication medium between different stakeholders. However the importance of choosing appropriate names for actions and actors was stressed. In the context of their methodology they were most useful to check completeness of object models, i.e. determining missing objects.

Estimating the complexity of use cases appears to be a difficult task. Its utility is to guess the time it will take to process them through the development cycle. A formula is used for that which uses such information as the number of involved objects, presence of subsystem interaction, the number of actions. While the use of scenarios to validate object models is sufficiently well understood in this company, they see the need of better scenario-based elicitation techniques to be satisfied by research.

Summarizing, use cases drive the company’s development methodology. They are regarded very useful as a way to communicate with the customers and for improving object models. Interestingly, use cases are created using templates including complexity estimation metrics (for planning purposes). Scenario-based elicitation techniques are currently the main contribution they expect from research to improve their process.

3.11 Radio Telecommunication / Transportation Project

3.11.1 Project Background

Our interview described here has focused on two projects for which the organization of our interview partner has been a consultant. The first one is a 50 man-year mobile radio telecommunication project (communication between base stations and mobiles) where several network topologies had to be supported (star, linear, ...). This project has now been stopped.

The second project is in the transport sector and deals with the development of (re-)configurable train networks. It is a 10 man-year project. The basic problem is that each train has its own information system made on the basis of an internal network. A protocol must be found that allows separate systems to be coupled, and, conversely, a global one to be separated. For example, the basic steps of the protocol for grouping two trains/networks is (1) fritting (2) „ who’s there ? “ (3) „ the two networks decided to make only one “ and (4) „ the two networks elect a master “.

3.11.2 Scenario Characteristics

The scenarios that were used in telecommunication project were represented with message sequence charts annotated with natural language and were typically one page long. They were mainly used to specify the interfaces between subsystems which were themselves specified with SA/RT. There was a team located is Paris which was responsible for the system and a team in Charleroi in charge of the infrastructure. The scenarios helped Paris team in making clear to the Charleroi team the BISS 1200 protocol which, until then, had only been loosely defined. Scenarios are mainly felt as a good way for the developers to communicate. It was estimated that, in this project, scenarios represented approximately half of the work performed by the system engineers.

In the transportation project, the use of scenarios is supported by a tool, namely Verilog's Geode. Scenarios are formal ones, represented with message sequence charts, and Geode enables them to be checked against an SDL specification and to be animated. The scenarios in this project are very complex and the use of the Geode tool is regarded as being of great help in managing this complexity. When problems are identified with the tool, the specification is requested to be changed as soon as possible but no traceability information is maintained.

In both projects, only behavioural scenarios (expressing functional and non-functional temporal requirements) are used and their purpose is to specify subsystem interaction. In the telecommunications project, scenarios have an explanatory and descriptive role and serve both for elicitation and validation (reuse of scenarios as test cases). On the other hand, in the transportation project focus is on validation and scenarios have mainly an exploratory role, namely the analysis of possible solutions and their costs.

3.11.3 Experiences

Our interview partner pointed out that scenarios in the form of message sequence charts serve as good (graphical) communication tool, but here mostly used for facilitating communication between developers. Hence scenarios are used more formally than in most other projects interviewed. They proved efficient in managing complexity and allows very complex systems to be delivered in reasonable time, provided that suitable tool support (like the Geode tool) is available. Furthermore, it was pointed out that message sequence charts serve as basis for validation tests.

As main problem it was experienced that scenarios are not easy to manipulate without tools. Moreover, for requirements analysis more generic scenarios required than testing. Going from generic to specific scenarios is not a trivial task. Moreover the integration with other aspects of requirements is unclear, e.g. with object models. In particular, they look for a way to summarize, as a „ state " what has happened until a given point in a scenario. They further expressed the wish for a mechanism to impose different views on scenarios according to who is examining them.

Summarizing, both projects described in this section have a strong technical emphasis. It interesting to observe that (in contrast to more business application oriented projects) scenarios here are use in a more formal way and less for communication purposes with the customer. Rather they are used internally as communication means between developers. In the case where large systems are decomposed into several subsystems specified by different teams, the use of scenarios (message trace diagrams) was proven useful for specifying the interfaces among systems developed by different teams. When very complex systems are developed, mastering and analysing complex scenarios cannot be made without a supporting tool.

3.12 CASE Tool for Banking Applications

3.12.1 Project Background

The project presented here is that of a CASE (computer aided software engineering) tool for the generation of Human Computer Interfaces of the client part of client / server bank applications. In this three years and a half project it was decided to adopt a way of working involving the client (a French bank) in the design validation process in addition to the requirement gathering tasks.

3.12.2 Scenario Characteristics

The approach is thus based on scenarios under the form of an incremental prototype. This prototype was first developed by two engineers for six months. In order to get a comparative idea of the size, the first version of the end-product was delivered at the end of the first year of the project.

The prototype is presented as a sequencing of windows built with NSDK. It is intended to give a rough idea of the interface of the end product. The prototype is issued after several meetings with technical managers representing the client. Before the prototype is created rough MCDs sketches (one may translate literally MCD into design data model in English; MCDs are design models of the Merise method) are produced. Both MCDs and the prototype evolve through time. Several other meetings are done in order to validate the prototype and the MCDs, and to explore emerging requirements. The design documents and the prototype are judged complementary. Indeed, the prototype has the advantage of showing up some aspects of the final product which would not be written in Natural Language or with MCDs or with Message Trace Diagrams, like ergonomy (e.g.: the fact that 3 windows may be opened at the same time, or the aspect of a window). On the other side, MCDs ensure the quality of the requirements design which can not be done with the only prototype.

Consistency between the prototype and design documents is managed manually. Considering that it happens even that the prototype is modified during meetings, this risk has been taken in order to improve cost, flexibility and response time. However such choice of project management would not be taken in the frame of larger projects.

When judged enough complete and valid, the prototype was used as such, as the interface of the end product.

3.12.3 Experiences

The choice of designing a prototype was done as a way to involve the client in the tasks of validation and exploration of the requirements engineering process. However, it was seen as a mistake afterwards not to have involved end users since the earlier phases of the prototype validation and exploration.

Advantages of prototype are evolutivity, ability to express the ergonomic aspects which can not be done with NL or MTDs, and ability to involve both technical managers, and end users in the requirements engineering process. This way of working has also the advantage of being adapted to junior design teams.

The main problem is that it is empirical. The need of formal ways of working was thus expressed in order to improve efficiency, productivity and reusability of the process and experience.

3.13 Water Invoicing Management System

3.13.1 Project Background

The project described in this section deals with the re-engineering of the water invoicing process. The project size is about 40,000 days-man, its duration was of four years. Scenarios were used due to the evolution of the client vision of his own process, and of the desired system. Because of the huge size of the project, the project management may be qualified of complex, involving co-operatively subcontractors and the customer himself in addition to software development organization. The customer was represented by several types of users in the project : customer responsibles, computer scientists, and end-users of the old system. All of them were involved either in the creation, exploration, validation and agreement steps of the scenarios which were used during the project.

3.13.2 Scenario Characteristics

The first document issued from interviews with the customers describes in natural language the management rules of the invoicing system. This document which was written for one year and a half is about five hundred pages long. The text which covers functional, structural and some organisational requirements is completed by some informal diagrams and illustrations, as a way to facilitate the customer comprehension. In addition to the writing of this document, several models describing abstractly the flows of exchanges of the system were defined.

After that Entity Relationship models were built to verify and control the validity and consistency of the requirement description documents, a prototype was created in Visual Basic. This prototype is a partial view of the system interface. It contains about twenty per cent of the screens of the end system. This prototype which is thrown out after use was intended to serve a user validation purpose. As a side effect, it was also a way for the customer to "sell" the designed system to the future internal end-users. One can thus say that it was as well important from the point of view of the customer’s policy of project management as in the help it provides to make users validate the system. Once the prototype and specification documents validated, OMT design models were written with the STP/Unix tool, and a mock-up was built. The mock-up, made with Visual C++ takes back the agreed prototype interface and completes it (a thousand screens in all). Contrarily to the prototype, the mock-up is not thrown away, but constitutes the interface of the end product.

The OMT models are completed with use cases hand written manually in natural language. These uses cases mainly describe the sequences of events which happen when an object action is performed. They also include statements such as goals, triggering conditions of events, and final state of the concerned object. They focus in priority on exceptions.

The size of the use cases goes from ten to two hundred pages per use cases. About two hundred use cases were associated to the design object models. The use cases were associated to the design object models in order to keep legibility at a detailed level of the specifications. Indeed, legibility of the design models was a project management constraint required by the customer. Use cases were validated, commented, corrected and agreed by the user. After validation and agreement were achieved, use cases where used together with design documents for the technical study and the code generation.

It is interesting to note that, there is a difficulty to qualify the one or the other of the earliest documents as scenarios. Indeed, contrarily to the prototype and use cases, these documents have only a descriptive and exploratory purpose and could be found at the basis of most of requirements engineering project.

3.13.3 Experiences

In contrast to other projects, one of the conclusions drawn from this project is that Use Cases are most useful to involve the customer in *advanced* steps of the requirements engineering process like exploration, agreement, or validation. Two kinds of scenarios were used in the project. First a disposable prototype which presents some aspects of the designed system interface. Such a prototype is important in the selling aspect of the project. It is also practical as it is low cost, easily modified and accessible by non-computer scientist customers. Second, the use cases associated with object oriented design models are considered as necessary means of behaviour descriptions when object oriented modelling is adopted.

Problems arise from the trace. Indeed all the documents issued from a project can be used until maintenance, but no tool supported means is known at the moment to relate the different documents the one to the other, and furthermore to relate them retrospectively with the logical level. A solution which is envisaged is to use an hypertext-like environment to fulfil this traceability requirement.

3.14 Process Engineering for IS Development

3.14.1 Project Background

The project described here is carried out at an international Information Technology company in Systems Integration, Consultancy and Outsourcing which operates in the fields of defence, banking and finance, energy, industry, commerce and services, the public sectors and telecommunications. The company has developed an approach which focuses on processes, people and technology. A specific project is established which aims at eloborating this approach. It proposes a process describing the tasks of project management from the requirements elicitation, analysis and design to the Information System (IS) deployment. Thus, the aim of the project is to assist the process of IS development. It is not proposed as a CASE or CARE tool, but it provides complementary assistance through a large set of "helper" files (which constitutes a five hundred pages hypertext document). For the sake of compatibility, the helper files supports the description of concepts with the UML (Unified Modelling Language) design model. Each helper file describes a different task of the project management process. Helper files define situations of use, steps of the local task, advantages and risks, and is completed with examples. Beyond these files, some are devoted to the definition and use of scenarios.

3.14.2 Scenario Characteristics

Two main types of scenarios are defined in the project.

On the one hand, scenarios are structured natural language descriptions of typical interactions of the system domain (at the organisational environment context level), actors, organisational and skill objects. Such scenarios provide an organisational vision of the system. However their effective life span is short within the scope of a project, these scenarios are intended to serve for the reuse activities. It is interesting to notice that the structure of these scenarios is a hypertextual one with refinement links. Indeed there is no imposed level of detail; it is only required to structure scenario from a broader level to more detailed descriptions.

On the other hand, scenarios are descriptions of user/system interactions under the form of message trace diagrams. These scenarios, issued from the specifications, serve inside the project to complete the requirements, for unitary user validation, and for software qualification. Contrarily to the previous ones, these scenarios are user oriented. They do not intend to cover all the aspects of the designed system (e.g. all the exception cases are not addressed), but to make use of a user knowledge view of the intended system to improve its quality.

Previous experiences with scenarios has led to two remarks concerning their size. First, for the reasons of exhaustivity and consistency, each scenario should not be more than ten pages sized, and in average there should not be more than ten scenarios per requirements engineer. Beyond these average size and quantity, it becomes costly to ensure their consistency, their evolution, and the correspondence with design models. Second, scenarios should not be used within large scale projects (involving more than ten requirements engineers). Indeed experience has proved that beyond this size, and given the above average data on scenarios size, it is difficult to ensure that all the representative information are included in the scenarios. However scenarios will be used (at a slightly increased cost) if the project can be subdivided into smaller teams attached to sub parts of the designed system.

3.14.3 Experiences

The experiences of scenario use are regarded as globally positive. Scenarios have the advantage of being usable at any moment of the project management process, in particular during the steps of Design, Testing and Software Qualification.

Although scenarios provide an efficient support in most of the cases they are judged dangerous and costly in the frame of large scale projects. It is foreseen to extend their use, but a topical preoccupation concerns the definition of writing rules. From the research point of view another topical expectation is related to the traceability of the requirements through scenarios.

3.15 Summary

Table 2 summarizes the main scenario characteristics found in the various projects according to the four views of the CREWS classification framework (form, content, usage/purpose, management/life-cycle). To characterize the relevance of the table entries, we use three attribute values: „X“ means significant relevance, „O“ moderate relevance, and „–“ no relevance of that aspect. In the following, we sketch some main observations on the results in the table.

3.15.1 Content

Three categories of scenario content were found. System context refers to descriptions of the broader environment in which the system is embedded, system interaction covers how the system interacts with its environment, and system internal refers to internal interactions between components of systems.

3.15.2 Form

The form of scenarios seems to be correlated with the content. Five basic representation types dominated in the projects. While some projects used narrative text without any structure, most preferred structured textfollowing a more or less rigid template or table-structure. In any case, natural language was mostly used for context and interaction scenarios. More formal diagrammatic notations, such as object interaction or message trace diagrams, and animations play a major role in internal scenarios while images (e.g. screendumps of user interface forms) are used for illustration mostly in context and interaction scenarios.

project/scenario facet		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
form	narrative text	X	O	X	O	O	O	O	X	O	–	–	–	X	–	O
	structured text	X	X	O	X	X	X	X	X	X	O	–	X	X	–	X
	diagrammatic notations	–	–	O	X	–	O	X	X	–	X	X	X	X	–	X
	images	X	X	O	O	O	O	–	O	O	–	–	–	O	–	–
	animations/simulations	–	–	–	–	–	–	–	–	X	–	X	–	–	X	–
content	system context	O	O	O	O	–	–	–	X	X	–	–	–	X	–	X
	system interaction	X	X	X	X	X	X	X	X	X	–	O	X	X	X	X
	system internal	–	–	O	–	–	–	–	O	X	X	X	–	X	–	–
purpose/ usage	concretization of abstract models	X	O	X	X	X	X	X	X	X	–	–	X	O	O	X
	scenarios instead of abstract models	–	–	–	X	X	–	–	–	–	–	–	–	–	–	–
	scenario use with prototypes	X	X	X	X	O	O	–	–	O	–	–	X	X	X	–
	complexity reduction	X	X	X	X	X	X	O	X	O	X	O	X	X	X	X
	agreement and consistency	X	X	X	X	X	X	X	X	X	X	O	X	X	X	O
	scenario use with glossaries	–	–	X	O	O	–	O	–	–	–	–	–	–	–	–
	reflection on static models	X	X	X	X	X	X	X	X	–	–	–	X	X	X	X
management/ lifecycle	partial views	O	O	X	X	X	X	O	X	X	X	X	O	X	–	O
	distributed scenario development	O	–	X	X	X	O	O	X	X	O	–	–	X	X	O
	scenario reviews	X	X	X	X	X	X	X	O	O	O	O	O	X	–	X
	traceability	X	O	X	X	X	O	O	X	X	O	O	O	X	O	O
	basis for test cases	O	–	O	O	O	O	O	–	O	O	O	O	O	O	O
	evolution	X	O	X	X	X	X	X	O	X	O	X	X	X	O	X

Table 2: Summary of Scenario Characteristics.

3.15.3 Usage/Purpose

Beyond their expected purpose as means for requirements elicitation and validation (not mentioned in the table), lots of other usage aspects play a major role. For example, concretizing requirements by lowering the abstraction level was seen as an important purpose of scenarios in 13 projects; two of them even changed their process to a scenario-based approach after encountering severe problems with the traditional development of abstract models. The role of scenario as means for reaching partial agreement and consistency of the requirements specification was stressed in nearly all projects. Another prevalent use of scenarios was complexity reduction of the requirements engineering task and enforcing an interdisciplinary development process. Somewhat surprisingly, 12 projects emphasized the reflection of dynamic scenarios on static models, i.e. the population and validation of object models. Some not widespread, but interesting interactions of scenarios were observed with prototypes and glossaries.

3.15.4 Management/Life-cycle

The management issues which played (in at least some projects) a major role were: how to impose partial views on a scenario, how to handle distributed scenario development, how to enable quality assurance through scenario reviews, how to make scenarios traceable along the whole process, how to re-use scenarios as test cases, and how to evolve scenarios.

3.15.5 Resumée

Summarizing, the diversity of scenario usage observed in the projects was much greater than one would expect from the UML-incited understanding of scenarios as instances of use cases. About half of the projects claimed that they followed a use case approach but all of them had needed to extend the textbook version significantly to make it work for them. A more detailed elaboration on the conclusions concerning purpose and management across all 15 projects can be found in [Weidenhaupt et al., 1997].

4 Questionnaire Results

The questionnaire was sent to representatives and project leaders of altogether 253 software companies (215 German, 15 British, 12 French, and 11 Belgian organizations). Among these were about 35 companies from which we knew that they used scenario-related techniques, while from additional companies we had no prior knowledge (mostly German). 26 (10,3 per cent) of the questionnaires were returned and only 4 of them came from the additional companies. This might indicate that in general industrial usage of scenario-related techniques is currently not yet wide-spread.

In the following, we highlight some results of the questionnaire evaluation concerning the project profile of the respondents' organizations, the used scenario types, their representation and media, the phases in which scenarios are generated and used, the stakeholders involved, and the main problems and benefits.

4.1 Project Profiles

business IS	control software	telecommunications	medical systems	software engineering tools
13	8	3	2	2

Table 3: Application Domains.

Table 3 indicates that scenario techniques are most prevalent in projects for business information systems (both in the financial and/or governmental sector). They also play an important role for control software, i.e. in more technical settings often exhibiting significant real-time characteristics. Three respondents used scenarios in telecommunication projects, two for medical systems. Two respondents were suppliers of CASE tools supporting and applying scenario based techniques.

small (less than 10 person years)	medium (between 10-50 person years)	large (greater than 50 person years)	undetermined (keine Angabe)
18	3	2	3

Table 4: Project Size.

Most projects referred to in the questionnaires required an effort less than 10 person years. This indicates that also smaller projects can benefit from the use of scenarios.

object-oriented	function-driven / structured	data-driven	evolutionary, iterative	other or undetermined
13	6	3	3	7

Table 5: Overall Systems Development Paradigm.

Most project employed an object-oriented methodology or a mixture of structured and object-oriented techniques.

4.2 Scenario Types

Figure 3 shows the frequency of various scenario types. A bit surprisingly, the scenario type most frequently used are system test cases which has an average of 4.5 (significant to frequent use). Acceptance test cases also play an important role (significant use). The same holds for prototypes although only few projects employ a pure prototype-based evolutionary. An

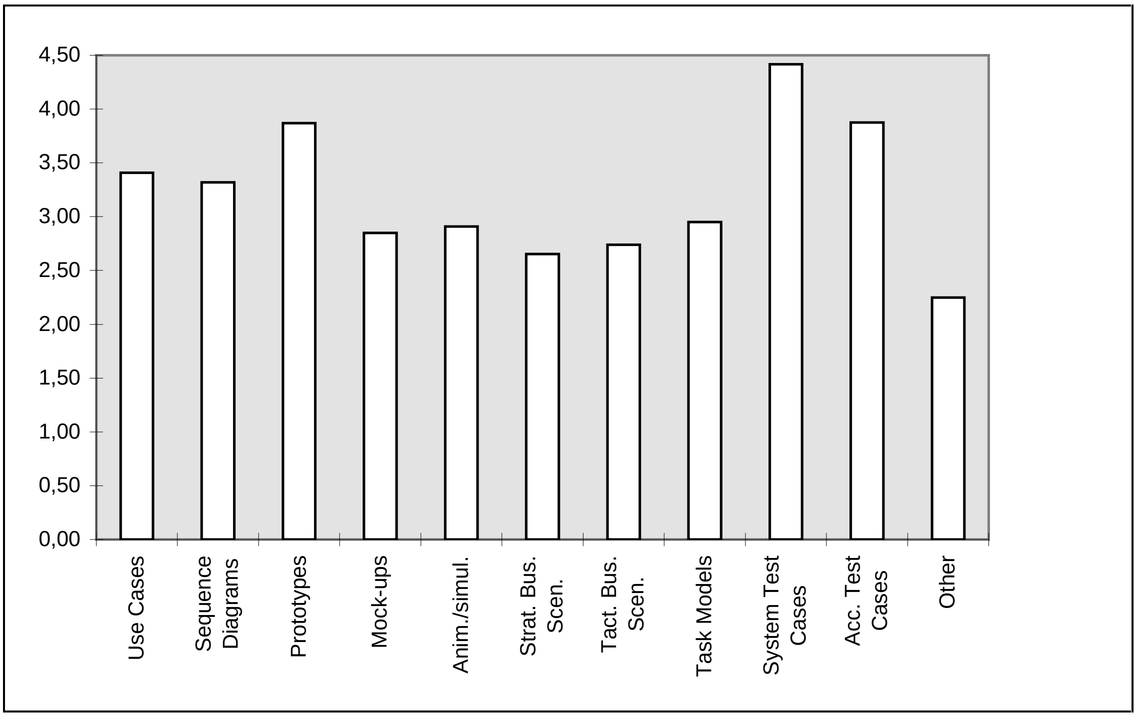

Figure 3: Frequency of Scenario Types.

interesting observation concerning use cases in conjunction with the overall system development paradigm can be made: although having become popular in the context of Jacobsons' object-oriented software engineering methodology, no significant difference between projects employing object-oriented and structured methods can be observed. This confirms an observation from the site visits that a use-case driven approach is not per se object-oriented.

4.3 Representation and Media of Scenarios

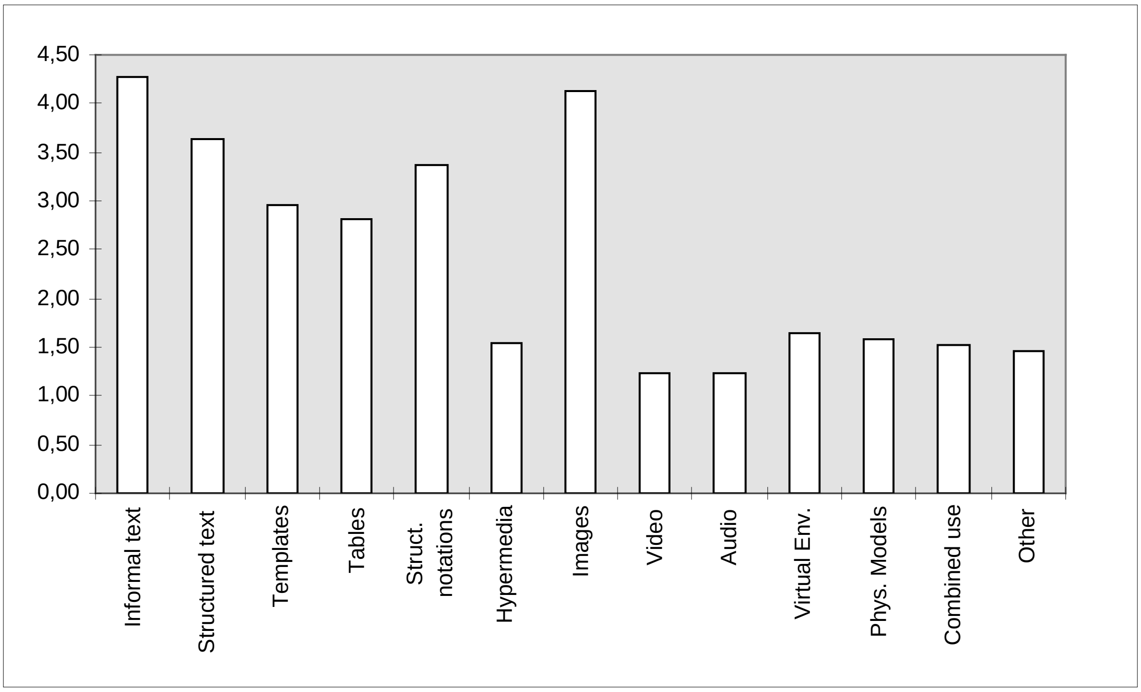

Figure 4: Frequency of Scenario Representations and Media.

As indicated by Figure 4 the by far most frequently used representation formats for scenario is informal narrative text and pictorial representation (significant to frequent use). Of medium importance are semi-structured notations like structured text, templates, tabular representation, and graphical modelling notations. All other representation formats and media are only sparsely used.

4.4 Phases and Activities

Figure 4 depicts how frequent scenarios are generated (white bars) and used (black bars) during certain software development phases and activities. The most prevalent use of scenarios is during requirements engineering (frequent). From the usage perspective scenarios play a major role also in design and system and acceptance testing. It is interesting to observe that in later phases the use of scenarios dominates their generation, especially during system and accepting testing. This indicated that it is aspired in many projects to re-use scenarios developed in earlier phases coherently through the whole lifecycle.

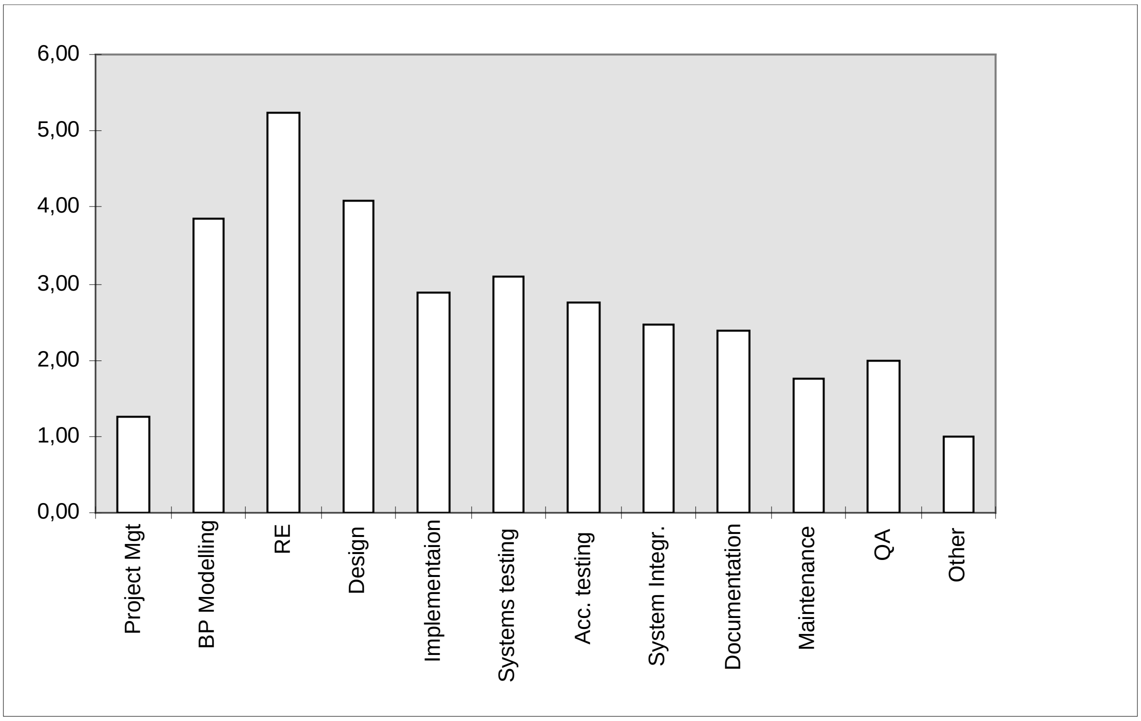

Figure 5: Frequency of Scenario Generation and Use during Different Phases.

Figure 6 depicts more precisely how often scenarios are generated and used for certain tasks within the requirements engineering phase itself. Interestingly, scenario themselves are most often generated and used with the purpose of modelling requirements. This indicates that in practice there is no clear distinction between scenarios and requirements. Another prevalent use is for requirements validation and communication. The significant gap between generation

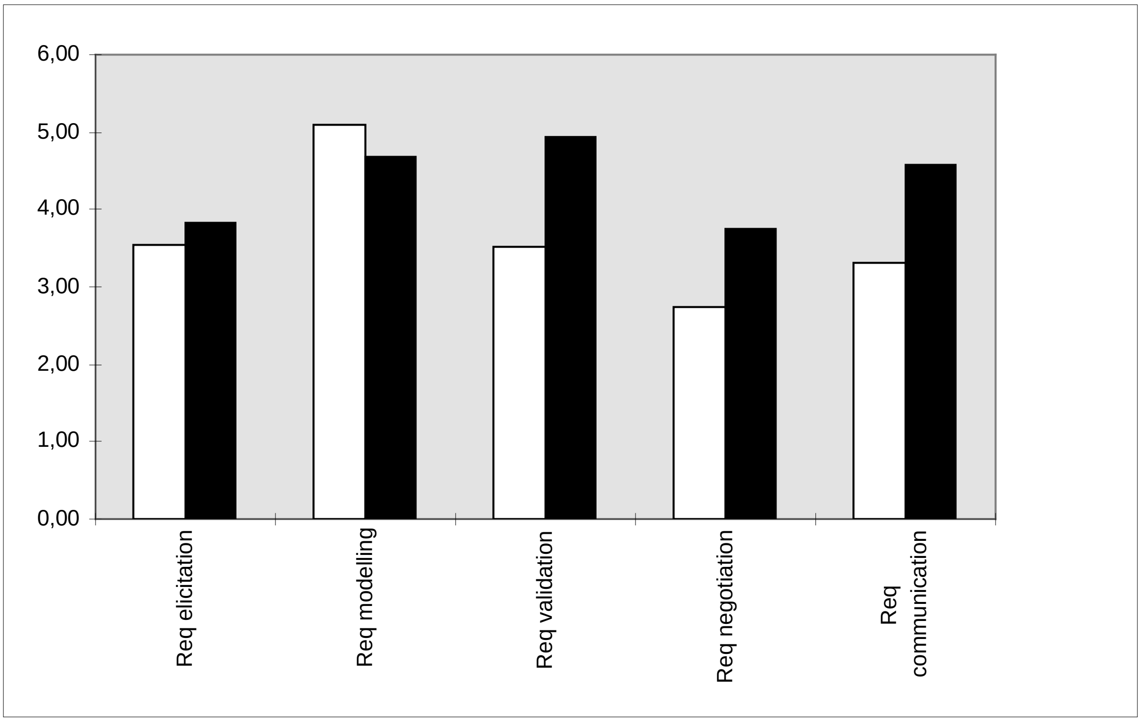

Figure 6: Frequency of Scenario Generation and Use in Specific RE Tasks.

and use of scenarios for these task indicates that scenarios originally generated for other purposes are reused.

4.5 Stakeholders

Figure 7 indicates the involvement of different stakeholder groups in the scenario generation and usage process. Of coarse this table reflects a significant correlation to the values of the different life-cycle phases, since typically each phase is related to a certain set of stakeholders. Besides the group of requirements engineering also the customer group plays an important role in scenario generation as well as use. Given the main uses of scenarios during requirements engineering (see section 4.4), this observation confirms the claim that scenario-based approaches enforce an interdisciplinary requirements engineering process.

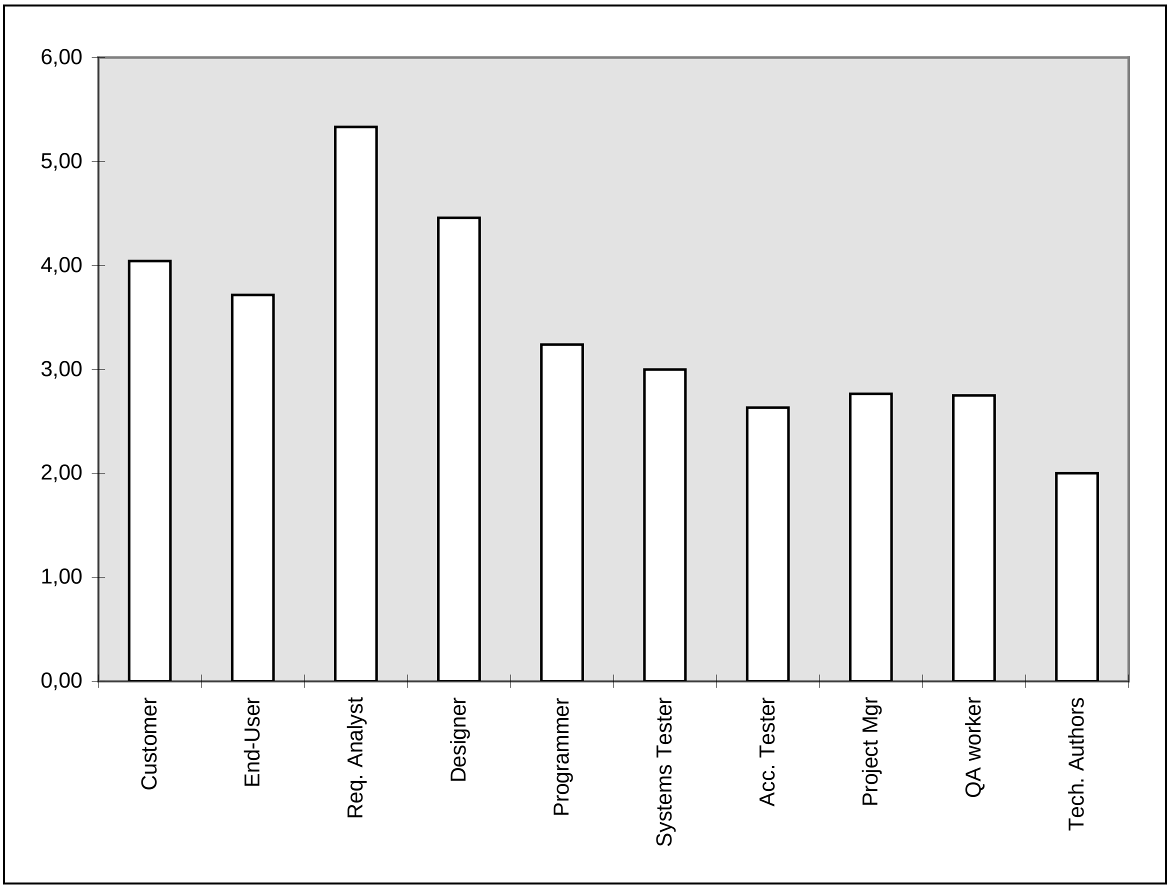

Figure 7: Stakeholder Involvement.

4.6 Perceived Benefits and Problems

4.6.1 Benefits of Scenario-based Approaches

Respondents pointed out the following main benefits of scenario-related techniques:

- early validation of requirements
- reduction of abstraction level
- help in verifying and checking completeness of requirements specification
- good communication base with customers and other non-technical people
- decision support for determining the optimal, i.e. most economic, solution
- resolution of ambiguities
- central reference in case of disagreement/negotiations
- illustration of dynamics

- ease of use
- complexity reduction through usage-oriented „slicing“ of overall system functionality
- illustrative representation of complex circumstances
- support for reverse engineering

4.6.2 Problems, Drawbacks and Negative Experiences

Most mentioned problems concern the lack of support by methods, tools, and management strategies for scenario-based approaches, more concretely:

- missing conventions and process guidelines
- weak understanding how to embed scenarios in existing notation
- misunderstandings because of informality
- checking completeness of scenarios not trivial
- missing traceability between scenarios and other software artifacts
- translation of operational user view to testable system view not trivial
- week visualization techniques
- management of scenario life-cycle
- weak tools and techniques
- hard to convince customer of the need of spending effort into scenario generation
- lack of trained practitioners

4.7 Future Plans

same level as now	slightly increased	significantly increased	extensive	no answers
9	2	7	4	4

Figure 8: Plans for Future Scenario Use.

Figure 8 indicates the plans of the respondents concerning future use of scenarios in their organization. Interestingly no respondent is going to decrease future scenario use. In average the organizations plan to slightly increase the application of scenario based techniques. The main topics for an enhancement of scenario use are:

- strengthen the coherence through all phases
- application of scenarios for testing and dialogue design
- development of tools for interactive scenario presentation to customer
- improvement of scenario formalisms

Acknowledgments

The authors like to thank their numerous interview partners and the questionnaire respondents from industry for giving insight into the current practice. This work was in part founded by the European Community under ESPRIT Reactive Long Term Research project 21.903 CREWS.

References

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

[McMenamin & Palmer, 1984] St.M. McMenamin, J.F. Palmer. *Essential Systems Analysis*. Prentice Hall, 1984.

[Rational, 1997] Rational Software Corporation. *Unified Modeling Language.* Available on the WWW: http://www.rational.com, 1997

[Rolland et al., 1997] C. Rolland, C. Ben Achour, C. Cauvet, J. Ralyté, A. Sutcliffe, N.A.M. Maiden, M. Jarke, P. Haumer, K.Pohl, E. Dubois, P. Heymans. *A Framework for a Scenario Classification Framework*, submitted to the Requirements Engineering Journal, 1997.

[Rumbaugh et al., 1991] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, W. Lorensen. *Object-oriented Model and Design.* Prentice Hall, 1991.

[Weidenhaupt et al. 1997] K. Weidenhaupt, K. Pohl, M. Jarke, P. Haumer. *Scenario Usage in System Development: A Report on Current Practice.* IEEE Software, March 1998