Survey on the Scenario Use in Twelve Selected Industrial Projects

M. Arnold[1], M. Erdmann[2], M. Glinz[3],
P. Haumer[4], R. Knoll[5], B. Paech[6], K. Pohl[4],
J. Ryser[3], R. Studer[2], K. Weidenhaupt[4]

GI FG 2.1.6 - - Working Group on
Scenario-based Requirements Engineering
Chair: Klaus Pohl
RWTH Aachen, InformatikV, 52056 Aachen
email: pohl@informatik.rwth-aachen.de

[1]FIDES Informatik, Zürich, Switzerland; [2]University of Karlsruhe, Germany; [3]University of Zürich, Switzerland; [4]RWTH Aachen, Germany; [5]RWG GmbH, Stuttgart, Germany; [6]Technical University of Munich, Germany

Table of Content

I. Introduction

Scenario-based requirements engineering approaches stress the use of examples, scenes, and narrative descriptions of context from a usage-oriented perspective, complementing the development of abstract conceptual models. In the past years, scenarios have been attracting increasing interest in both requirements engineering (RE) research and practice. However, due to the immaturity of the field, little consensus has been reached so far on how to treat scenarios as products and how to use scenarios throughout the requirements engineering process or even the whole software life cycle.

I.1 Overview

To highlight the current activities on scenario-based RE in Germany and Switzerland, in January 1997 the working group ”Scenario-Based Requirements Engineering” within the special interest group 2.1.6 “Requirements Engineering” of the German Informatics Society was founded. This working group comprises participants from both, universities and software companies.

The first objective of the working group was to investigate the current state of practice in scenario-based RE in German and Swiss software organisations as a complementary activity to the European-wide survey described in [Weidenhaupt et al., 1998].

In this paper, we report on the results of these industrial investigations. Altogether, twelve projects in nine German and three Swiss software organisations were examined during half- to one-day site visits. We did not select the projects randomly, but chose those projects from which we knew that scenario use played a significant role.

I.2 Classification Taxonomy

For making the results of the site-visits comparable, we developed a comprehensive classification taxonomy. This taxonomy was used to guide through the interviews and to evaluate the results of the site-visits according to a common scheme. To avoid bias, we let our interview partners talk freely about their overall development process, scenario usage, and experiences and used the taxonomy mainly as a checklist for asking additional questions as to not omit crucial information.

When developing the classification taxonomy we took a framework for scenario-based RE approaches described in [Roland et al., 1998] as a starting point. This classification framework, however, was developed to classify scenario approaches described in *literature* and neglects specific properties which are relevant for representing scenario usage in the context of *industrial projects*. In contrast, our taxonomy takes into account a set of project characteristics as well as experiences gained in the projects to enable a better understanding and assessment of the concrete scenario usage in the examined projects. In comparison to the evaluation framework underlying the European-wide survey described in [Weidenhaupt et al., 1998], our taxonomy has a richer and more formal structure. It is also noteworthy, that during the discussion about the site visit results the classification taxonomy was progressively refined to better reflect the results until it got its final form described in this report. The development of the classification taxonomy can hence be seen as an effort on its own, since it helped to make the role of scenarios within real RE processes (and software development processes in general) much clearer. The refinement of the classification taxonomy caused the need to re-interview some of our interview partners in order to elicit omitted information.

The taxonomy is structured into five major *sections*. Each section is further subdivided into *divisions*, which consist of several related *facets* covering an individual property. In some cases the facets are subdivided into *characteristics*. The five sections consider:

1. *Project Properties*. We examined several project properties which can significantly influence the way how scenarios are used and managed: project size and type, application domain, as well as the background and roles of the people involved in the project teams (Section 1);
2. *Scenario Contents and Representation*. We analysed the properties of scenarios as products, i.e. which kinds of scenarios are used, what content do they express, and which representations and structuring constructs are used (Section 2);
3. *Goals*. This section deals with the overall objective underlying the use of scenarios and the specific purposes with respect to other artefacts used in the process (Section 3);
4. *Process*. We took a detailed look at the projects’ process. We classified the establishment of relationships of scenarios to other software artefacts, the overall methodological context, the creation, usage, maintenance and quality control of scenarios as well as the tools used for manipulating scenarios (Section 4);
5. *Experiences and Expectations*. Finally, we summarised general project findings like benefits, problems, needs, and expectations from the interviewees’ viewpoint (Section 5).

The individual facets are typed according to the possible values they can carry, e.g. INTEGER for the number of persons involved in a scenario team or TEXT for describing the main purpose of scenario usage. For facets where a discrete qualitative assessment is adequate, we have introduced the type VALUE that comprises the following qualitative values:

- Plus sign (+): this property is true or is of major importance for the project;
- Circle (O): the property holds partially or is of medium importance.
- Minus sign (-): this property does not hold or is of no importance for the project;,

In addition, we added the following values which cover different cases where an assessment using the (+),(O) or (-) was not applicable:

- Blank (” ”): the interviewee did not mention this property.
- Slash (/): the property was not applicable in the examined project.
- Question mark (?): the information on the property was not available or unknown to the interviewee.

I.3 Structure of the Report

The results of the project classification are presented in section 1-5, which correspond to the respective sections of the classification taxonomy.

Each section starts with an introduction describing the scope of the section and giving an overview of the taxonomy covered in this section. Then the evaluation results of the projects according to the facets are presented division by division, mostly condensed in tabular form. The meaning of each facet is explained in detail and, if not self-explaining, the occurring facet values as well. Moreover, we describe how the data concerning a certain facet were elicited, e.g. if they were explicitly mentioned by the interviewee or if they were the result of our interpretations of the information obtained from the interviewee. Finally, we summarise the main findings of each section.

1 Project Properties

Besides collecting information about scenario characteristics, we are also interested in classifying the projects themselves to better understand scenario properties and usage in a broader context. To represent this information the classification taxonomy differentiates the project's domain and the application background (Div. 1.1), project size (Div. 1.2) and type (Div. 1.3). In addition, we inquired information about the people involved in the project, because a major application of scenarios is to support the communication of different stakeholders. Thus, we classified the stakeholders' level of education (Div. 1.4) as well as the team structures of the project (Div. 1.5).

Before describing the project characteristics in detail, we first give a short overview of the twelve examined projects in Tab. 1. Most interview partners asked us to anonymise the information obtained from them. We therefore cannot name the companies we interviewed nor can we give too detailed information about the projects, which might allow insiders to identify them. Rather, we introduce a unique identifier reflecting the projects domain (e.g., INS1 for insurance project No. 1, MED for medical system etc.) for each project which is used as reference instead.

Tab. 1: Short Description of the Examined Projects

Project	Description
NET1	The aim of the project was to develop a management system for heterogeneous (with respect to three dimensions: producer, OSI-level, technology) VLAN (virtual local area network) based on an existing network management platform (the customer was the supplier of the platform). This platform offers API for events, database, graphical user interface and management protocols like SNMP and OSI).
INS1	The project had to deliver a requirements specification for a particular department of a large insurance company. The department handles credits and securities. The system will substitute a very simple existing IS. The main difficulty is to find flexible computerised support for the many specific ways of handling the different customers and credits and securities. The system is now being implemented by the customer
SALES	At a large production company (cars) an existing sales support system is replaced. It deals with all special sales cases (like sales to employees). Therefore a lot of different departments are affected. The project is divided into 5 subprojects handling e.g. the ordering (including configuration of cars), plant specific sales and company cars.
MED	The aim the project was to develop a medical information system, which is 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.
NET2	The project takes place in the context of the privatisation of the telecommunication market in Germany. The project aims at building administration software for the management, monitoring, and billing of telecommunication network services. The background is that the customer, a consortium of large companies, intends to use its existing network infrastructure for public telecommunication services. This involves a lot of new functionality about which the customer has nearly no prior experience.
PUB1	The aim of the project is to build common software to be used by all authorities of a specific sector in the 16 German states. The vision is that the currently individual and only partial interoperable systems are finally replaced by an integrated system.
INS2	The project takes place at the life insurance department of a large insurance company. The goal is to build an integrated information system, which supports all services performed at the workplace of the insurance employees. The new system replaces an old, poorly integrated system.

Project	Description
DEV	The project is conducted at an international provider of railway systems and rail transportation and aims at the development of a software system with tools supporting project management and development of software for rail transportation systems and signal & traffic control systems.
PUB2	The project aims at building an emergency and security task force guidance system for public administration and services (e.g. police, fire department, ambulance / medical emergency services). The system is being built by a consultant firm, established as an IT consulting, training and solution provider.
BANK1	The goal of this project is the development of a system that should manage the transfer (especially the lending) of financial papers between different German banks. The system replaces the current manual practice and should be able to maintain the stocks of different papers, to whom or from whom they were lent etc. The system does not contain knowledge about actions; it is only used for bookkeeping.
BANK2	In this project at a large German bank a system that should help to manage, control and monitor all necessary activities for the introduction of the Euro, e.g. printing new forms, developing new or changing existing systems, offering new services, is being developed. The project affects all departments of the bank.
BANK3	The project at a large Swiss bank is the development of a new work platform supporting employees of the bank who are working in the corporate customer business.

1.1 Domain / Application

For this division, we differentiate the facets application domain (Facet. 1.1.1), system type (Facet 1.1.2) and the importance of static and dynamic properties, respectively (Facet 1.1.3).

We initially let the interviewees name suitable categories for the application domain and system type of their projects. To enable a better comparison of the projects, we then generalised the mentioned categories. For the application domain, the current values are *communication, financial, medical, public (emergency) service, sales,* and *software development*. This set can be extended if necessary, e.g. to classify a *control room* project. The system type facet is assigned to *information system, management system,* and *CASE tool*. Facet 1.1.3 (importance of static/dynamic properties) is classified using the VALUE type. The entries for this facet were directly asked from the interviewees.

Tab. 2 summarises the results for Div. 1.1. The majority of the examined projects (8 out of 12) are concerned with the development of information systems, mostly for administrative domains like financial, sales, and public service. Management systems, which are distinguished from information systems by their more active nature, are developed in 3 projects. Two of them deal with management of network communication facilities, one with the management of public emergency services.

The prevalence of information systems partly explains the major importance of static properties: in 10 projects static aspects were regarded as important as opposed to 6 projects which emphasise the relevance of dynamic aspects. For example, only one information system project (INS1) stresses the importance of dynamic aspects over static ones.

1.2 Project Size

Since the project size may significantly influence how scenarios are applied, we captured the project duration (measured in years) as well as the effort spent (measured in person years). Seven out of 12 projects were not yet finalised when we conducted our study so that we differentiate between the elapsed duration at the time when conducting the interview (1.2.1) and the (estimated) duration in total (1.2.2). In addition, the (estimated) total effort is captured in person years (1.2.3). We asked this information directly from the interviewees.

Tab. 2: Domain/Application

Project	1.1.1 Application domain	1.1.2 System type	1.1.3. Importance of	
			1.1.3.1 static properties	1.1.3.2 dynamic properties
NET1	communication	management system	+	+
INS1	financial	information system	O	+[1]
SALES	sales	information system	+	O
MED	medical	information system	+	O
NET2	communication	management system	+	O[2]
PUB1	public service	information system	+	+
INS2	financial	information system	+	O
DEV	software development	CASE Tool	–	+
PUB2	public emergency service	management system	+	+
BANK1	financial	information system	+	-
BANK2	financial	information system	+	O
BANK3	financial	information system	+	+

Tab. 3: Project Size

Project	1.2.1 Duration elapsed (years)	1.2.2 Duration total (years)	1.2.3 Effort total (person years)
NET1	1	1	3
INS1	0.66	0.66	4
SALES	2	5	200
MED	3	3	15
NET2	2	5	20
PUB1	2	10	1000
INS2	2	4	16
DEV	5	6	45
PUB2	2.5	3	18
BANK1	0.75	0.75	2.25
BANK2	0.5	0.5	2.5
BANK3	1	1.5	16

[1] Focus in dynamics, since mainly business processes were described.

[2] Focus on data was a specific choice of the project.

Tab. 3 summarises the duration and effort values of the examined projects. The majority falls into the small (less then 2 years duration, less than 5 person years) and medium (between 2 and 5 years duration, between 5 and 20 person years) categories. Exceptions are the projects PUB1 (1000 person years), SALES (200 person years) and DEV (45 person years).

1.3 Project Type

Div. 1.3 of the taxonomy accumulates more specific properties of the projects. In Facet 1.3.1 the experience of involved customers and suppliers in the application domain of the project is evaluated using the type VALUE. Facet 1.3.2 indicates whether the project was carried out by an *in-house* data processing department of the customer or an *external* supplier. In Facet 1.3.3 we classify the systems to be built as *individual* solutions, as *market*-driven products or as *prototype* developments. For Facet 1.3.4, we differentiate the type of system development in the projects, i.e. whether they develop a *new* system (with no predecessor), build upon an *existing platform*, or *replace* an existing system. Finally, we state in Facet 1.3.5 the absolute number of companies (or more general: profit centres) involved in the projects as customer, supplier or consultant.

Tab. 4: Project Type

Project	1.3.1 Experience in application domain		1.3.2 Type of customer	1.3.3. Type of resulting system	1.3.4 Type of development	1.3.5 Companies/ profit centres involved		
	1.3.1.1 Customer	1.3.1.2 Supplier				1.3.5.1 Customer	1.3.5.2 Supplier	1.3.5.3 Consultant
NET1	O	O	in-house	individual	new, existing platform	1	2	0
INS1	O	O	extern	individual	replacement	1	1	0
SAL	+	O	in-house	individual	replacement	>1	2	1
MED	+	–	extern	individual	new	1	1	0
NET2	O	–	extern	individual	new	1	1	0
PUB1	+	+	in-house	individual	replacement	1	1	1
INS2	+	+	in-house	individual	replacement	1	1[3]	1
DEV	+	+	in-house	individual[4]	replacement	4	6	1
PUB2	+	+	extern	market (individual[5])	replacement	O[6]	2	1
BANK1	O	O	in-house	prototype	new	1	1	1[7]
BANK2	+	O	extern	individual	new	1	1	1
BANK3	+	+	in-house	individual, prototype	replacement	4	1	1

[3] supplier = IT-department of customer company

[4] some components purchased

[5] in parallel to the market-oriented development, the system is individually customized for some pilot customers to minimise the risk for market introduction

[6] not directly involved because of market development

[7] OO-Coach for the IT-department

Tab. 4 summarises the project type aspects of the examined projects. In 8 projects the customer has significant experience in the application domain, in 4 projects he has a medium familiarity. In 8 cases, the experience of the supplier is about the same level as the experience of the customer. This is mostly explained by the fact that in 6 of these cases the development of the software is performed as an in-house development.

In 4 cases the supplier has a slightly lower experience. Most systems developed in the projects were individual in-house applications replacing existing systems. 4 of the 5 cases where customer and supplier experience differ are projects, which are conducted by an external supplier.

The huge majority of the projects develop individual or even prototypical solutions. Only in project PUB2 a market-driven system is developed. In 7 projects, an existing system is replaced, while 5 projects build a new system (one of them utilising an existing platform).

In 7 cases, the contractual structure of the projects only involves one customer, one supplier party and partially a consultant. 4 projects comprise more than one customer or more than one supplier party, respectively. Altogether, 8 projects are supported by a consultant.

Tab. 5: Stakeholder Experience in Software Development

Project	1.4.1.A End-user representative	1.4.1.B Project manager	1.4.1.C Software engineer	1.4.1.D Consultant	1.4.1.E Domain expert
NET1	/	O	+	/	+[8]
INS1	–	+	+	/	–[9]
SALES	/	+	+	+	O
MED	/	+	+	/	O
NET2	–	+	+	/	–
PUB1	–	+	+	+	–
INS2	–	+	+	+	–
DEV	+	+	+[9]	+	+[9]
PUB2	–	+	+	+	–
BANK1	–	+	+	+	–
BANK2	/	+	+	+	–
BANK3	–	+	+	–[10]	–

1.4 Stakeholder Experience

Using the VALUE type, we classify the experience of the different stakeholders involved concerning software development in general (Tab. 5) and scenario use in particular (Tab. 6). The considered roles are end-user representative, project manager, software engineer, consultant, and domain expert. Note that in some projects certain stakeholders do not occur (indicated by

[8] same as software engineer

[9] same as end-user

[10] one of the domain experts

a slash ”/”), i.e. here a classification was not applicable. Sometimes the same person(s) play(s) the role of different stakeholders (see footnotes). The information about stakeholder experience was directly provided by the interviewees (who were mostly the project managers) and thus, of course, represents a subjective assessment.

The usage of scenarios in the projects is justified by the low level of experience that end-users and domain experts have with software development as indicated in Tab. 5. Most of the end-users (if involved in the development at all) had no previous experience and were therefore not able to understand more formal representations. Nearly the same holds for the partially overlapping group of domain experts, who are only in two cases familiar with software development techniques. The project DEV is an exception to this trend, since it develops CASE tools, i.e. the end-users are software engineers themselves. As expected, project managers, software engineers and consultants have in nearly all cases significant experiences with software development (except project managers in project NET1).

Tab. 6: Stakeholder Experience in Scenario Use

Project	1.4.2.A End-user representative	1.4.2.B Project manager	1.4.2.C Software engineer	1.4.2.D Consultant	1.4.2.E Domain expert
NET1	/	–	–	/	–[11]
INS1	–	O	O	/	–[9]
SALES	/	–	–	O	–
MED	/	+	O	/	–
NET2	–	+	O	/	–
PUB1	–	O	–	+	–
INS2	–	O	O	+	–
DEV	O	O	O[9]	–	O[9]
PUB2	–	O	O	O	–
BANK1	–	–	+	–	–
BANK2	/	O	O	O	–
BANK3	–	O	O	–	–

In Tab. 6 it becomes apparent that scenario use is quite a new field with still a low level of experience in all stakeholder groups. Only in project BANK1 the software developers have an over-average experience in scenario use. In the projects NET2, MED, PUB1 and INS2 the project manager and the consultants, respectively, introduced scenario-based techniques into the projects.

1.5 Scenario Team Structure

In this division, we consider the formation of the teams working with scenarios in the projects.

[11] same as software engineer

Facet 1.5.1 counts the number of people, which are involved in the treatment of a single scenario. In Facet 1.5.2 we additionally differentiate the number of people from each stakeholder group. In Facet 5.3 we consider the total number of people which are involved in scenario creation or usage.

Although Tab. 7 shows that the total number of persons involved in the individual projects varies widely, an average of 3-5 people constructing and working with a single scenario at a time is quite stable among all projects. Exceptions are the projects MED, in which work is done in a larger team (up to 9 people), and BANK1 where scenarios are developed and used by only one or two persons. Typically 1-2 end-user representatives or domain experts and 1-3 software engineers are drawn into a scenario team.

Tab. 7: Scenario Team Structure

Project	1.5.1 Average size of team for treating a single scenario	1.5.2 Team structure for treating a single scenario					1.5.3 Total number of persons involved in scenario creation and usage
		1.5.2.1 End user repr.	1.5.2.2 Project manager	1.5.2.3 Software Engineer	1.5.2.4 Consultant	1.5.2.5 Domain expert	
NET1	4	0	0	3	0	2[12]	4
INS1	5-6	4	1	1	0	0	7
SALES	3-4	0	1	1-2	0	1	24
MED	5-9	0	0-1	2-3	0	2-5	10-12
NET2	3-5	1-2	0-1	1-2	0	1-2[13]	10-14
PUB1	3-5	1-2	0	1-2	0-1[14]	1-2[13]	~ 70
INS2	2-3	1-2	0-1	1-2	0-1[14]	1-2[15]	3-4
DEV	3	0	0	3	0	0	5 at visited site 15 in total
PUB2	4	0	0	2	1	1	6
BANK1	1-2	0	2	1[16]	1[16]	1[16]	2
BANK2	4	0	1	3	1[16]	1	5
BANK3	3	1	0	2	1	(1)[17]	5 at visited site 9 in total

1.6 Summary and Findings

Section 1 of the classification taxonomy captures a set of important project properties, which enable a better understanding of the scenario use in the concrete project context. The general findings from the project characterisation can be summarised as follows:

[12] One of them identical with one of the SW engineers.

[13] Partially identical with user representatives.

[14] Only for spot checks.

[15] Mostly identical with user representatives.

[16] Identical with project manager.

[17] Consultant is identical with domain expert.

- **Information systems projects prevalent:** The majority of the projects to which we had access are concerned with the development of information systems in the administrative area. It should be noted that we did not consider process control or embedded systems. This does not necessarily mean that scenarios are not suitable for these kinds of projects.
- **Emphasis on static properties**: The stress on static properties might, at first glance, appear surprising, since scenarios have traditionally been regarded as a means to elicit behavioural aspects. However, it will be shown in the following section that scenarios also play an important role for determining the dynamic aspects of the systems to be built.
- **Large variance/diversity in project size**: The size of projects utilising scenarios varies widely from very small projects (0.5 years with 2.5 person years) to huge projects spanning a time frame of 10 years with about 100 people involved. This can be seen as an indicator for the scalability of scenario-based techniques. The majority of the projects, however, belong to the small and medium category.
- **Differences in Experience (in external development projects)**: In the projects carried out by an external supplier, the supplier lacks familiarity with the application domain in comparison to the customer. In in-house projects, the customer and supplier parties have nearly the same experience in the application domain.
- **Small scenario team sizes**: The average team size for a single scenario treatment normally does not exceed 5 members. Besides the software engineers, usually one or two end-user representatives and/or domain experts participate in a scenario team. This underlines the interdisciplinary nature of scenario-based software development which was also observed in [Weidenhaupt et al., 1998].

2 Scenario Content and Presentation

In this section, we focus on scenarios as artefacts produced in the examined projects, i.e. we highlight various *product* properties of scenarios.

We first briefly describe the various kinds of scenarios used in our sample set of projects (Div. 2.1). In Div. 2.2, we consider the scenario content, i.e. what kind of information do scenarios convey, what viewpoints are reflected, what scope and granularity do the scenarios have and on what level of abstraction are they modelled?

Div 2.3 deals with the representation formats used, in particular: What primitives (building blocks) and structuring mechanisms are used to build scenarios? What kind of language and style is used to write scenarios? How formal is the representation?

Finally, we elicit two characteristic size metrics, namely the number of scenarios in each project and the typical size of a single scenario.

Tab. 8: Kinds of Scenarios

Project		2.1 Short description of kinds of scenarios used in the project
NET1	a) b)	Textual description of the services of the VLAN to be managed Textual description of the services of the management system
INS1		Description of business processes
SALES	a) b) c)	Description of business processes Uses cases (interaction with the system) Event traces (interaction between classes)
MED	a) b) c)	Informal scripts of typical system usage CRC cards UI centred use cases describing a certain task/working step performed through an individual UI dialogue
NET2		Use case descriptions closely related to UI prototype forms generated from an EER data model
PUB1	a) b)	Use cases describing system functionality for supporting a business process from a user’s perspective Scenarios representing single threads through a use case
INS2	a) b)	Use cases describing system functionality for supporting a business process from a user’s perspective Scenarios representing single threads through a use case
DEV		Design scenarios: Textual descriptions of interaction and (complicated) dialogue sequences
PUB2		Textual descriptions of interaction sequences from a user’s perspective, augmented by use case overview diagrams
BANK1		Textual descriptions of interactions between users and system, and system responses
BANK2		Textual descriptions of envisaged system functionality; closely related to UI
BANK3	a) b)	Textual scenarios augmented with co-operation diagrams describing the application domain as is. Textual scenarios (called visions) with tables to depict relationships and pictures (screenshots of interface elements and masks) describing the desired state

2.1 Kinds of Scenarios

Div. 2.1 of the taxonomy provides a short description of the different kinds of scenarios used in the projects under consideration. In inquiring about different kinds of scenarios we did not directly ask “What kind(s) of scenarios do/did you use in your project?” Most interviewed persons would not have been able to answer this question. Instead, we formulated the entries when evaluating the interview data and by examining sample scenario documents. The data for Div. 2.1 are thus a condensate of information collected in the interviews.

Tab. 8 summarises the corresponding answers, as extracted from the interview data. In several projects, scenarios of more than one kind are used. This is indicated by the sub-rows a), b), c) in column one of Tab. 8 and in all subsequent tables.

In summary, scenarios describe services and functionality (NET1, MED b, PUB1 a, INS2, BANK2), processes (INS1, SALES a, partially PUB1 a, INS2 a), user interfaces and dialogue sequences (MED c, partially NET2, DEV), and system usage (SALE b, MED a, NET2, PUB1 a+b, INS2 a+b, PUB2, BANK1, BANK3 a+b, DEV in parts). They model the system as is, as desired/required, and as designed. Scenario types include (and are as different as) event traces, scripts, textual descriptions and CRC cards. The variety of the observed scenario kinds will be presented in more detail in the subsequent divisions of this section.

2.2 Content of Scenarios

This division considers the information conveyed by the scenarios used in the projects. The content of scenarios on an abstract level is described by the:

1. *Main modelling focus* (Facet 2.2.1); i.e. is the
   - *current* situation modelled primarily (analysis of the concrete or conceptual existing system),
   - *desired* new system modelled (focus on the problem to be solved, requirements of the desired system are modelled) or
   - *solution* modelled (the design of the desired system is specified and defined)?
2. *Scope* (Facet 2.2.2); are scenarios used to describe
   - system internal behaviour and sequences of actions (*internal* scope),
   - the interaction of the system with users and other systems or the environment respectively (*interaction* scope),
   - the associations, relations and connections between any parts of the environment as well as between the system and its environment (*contextual* scope).
3. *Abstraction level* (Facet 2.2.3); i.e. are scenarios described at the
   - *instance* level (John Doe does ..., then he ...);
   - *type* level (the operator does ..., then she ...)?
4. *Modelling viewpoints* (Facet 2.2.4); i.e. do the scenarios centre around
   - *static* aspects (data);
   - *dynamic* aspects (behaviour) or
   - *non-functional* requirements (e.g., performance, reliability, security, safety etc.)?
5. *Granularity* (Facet 2.2.5); i.e. are scenarios described at the level of
   - *business processes* (business processes describe tasks and relationships among them);
   - *tasks* (tasks describe working steps and relationships among them);
   - *working steps* (working steps consist of elementary interactions)?
6. *Cases being modelled* (Facet 2.2.6); i.e. are scenarios used to model the
   - *normal* flow of actions;
   - *exceptional* flows?

The criteria of Div. 2.2 are not entirely unrelated or orthogonal. Scope and granularity in particular are not independent facets. Characteristics are not mutually exclusive; scenarios can cover more than one characteristic in all facets. For example, it is perfectly feasible that the same kind of scenarios is used to model the current situation as well as to capture and document the requirements.

The entries were elicited partly through direct questions, partly the entries were condensed and formulated after the interviews. In any case the answers were enhanced and reconciled with our observations and findings in sample scenarios. The information found in Facet 2.2.4 relates to the question of how important the different aspects of data (static view), behaviour (dynamic view) and non-functional requirements are in the projects described.

Tab. 9: Scenario Content I

Project		2.2.1 Scenarios used to model			2.2.2 Scope			2.2.3 Abstraction	
		2.2.1.1 Current situation	2.2.1.2 Requirem. (problem)	2.2.1.3 Design (solution)	2.2.2.1 Internal	2.2.2.2 Inter-action	2.2.2.3 Context	2.2.3.1 Instance	2.2.3.2 Type
NET1	a)	–	+[18]	–	O	+	O	–	+
	b)	–	+	–	+	O	O	O	O
INS1		–	+	–	O	–	O	O	+
SALES	a)	+	+	–	–	–	+	O	+
	b)	–	+	–	–	+	–	O	+
	c)	–	–	+	+	–	–	–	+
MED	a)	–	+	O	–	+	O	+	O
	b)	–	+	O	–	+	–	–	+
	c)	–	O	+	–	+	–	–	+
NET2		–	+	O	–	+	–	–	+
PUB1	a)	O	+	O	–	+	O	–	+
	b)	+	O	–	–	+	O	+	–
INS2	a)	O	+	–	–	+	–	O	+
	b)	O	+	–	–	+	–	+	O
DEV		–	O	+	+	+	–	O	+
PUB2		–	+	O	O	+	–	O	+
BANK1		–	+	O	O	+	–	–	+
BANK2		–	+	O	O	+	–	–	+
BANK3	a)	+	– (O)	–	O	+	+	O[19]	+
	b)	–	+	–	O	+	+	–	+

Tab. 9 summarises the data for the facets *main modelling focus* (Facet 2.2.1), *scope* (Facet 2.2.2), and *abstraction level* (Facet 2.2.3).

- As indicated by the data for Facet 2.2.1, the modelling of requirements[20] prevails. All but one project (DEV) use scenarios heavily to capture and model requirements. Only in few

[18] Requirements of the LAN to be managed.

[19] Instance scenarios were partially used to elicit type scenarios, they were not documented though.

projects (3 out of 12), they make a significant contribution to design solutions and/or are used to model the current situation. In projects that utilise more than one kind of scenario, the different types of scenario have different characteristics, too. For example, in project PUB1 one type is mainly used to model the current situation, the other to specify the requirements.

- The scope of scenarios (Facet 2.2.2) lies mainly on the boundaries between user and system (facet 2.2.2). All projects but one use at least one kind of scenarios to model interaction between user and system. Yet, almost half of the projects use scenarios complementary also on a contextual level or to model system internal interactions, respectively.
- Scenarios capture requirements on both, the instance and the type level (Facet 2.2.3). Even though all projects use scenarios at the type level, yet instance level scenarios are also used at least partially. In many projects they are used to complement type level scenarios (e.g. in project PUB1 and INS2): They help the user to relate to specific situations. Some projects use instance scenarios at the beginning of requirements elicitation and scenario creation. Later on they abstract from the instance level and use type level scenarios.

Tab. 10: Scenario Content II

Project		2.2.4 Viewpoints			2.2.5 Granularity			2.2.6 Cases being modelled	
		2.2.4.1 Static	2.2.4.2 Dynamic	2.2.4.3 NFR	2.2.5.1 Business process	2.2.5.2 Task	2.2.5.3 Working step	2.2.6.1 Normal Cases	2.2.6.2 Exceptions
NET1	a)	O	+	+	+	O	–	+	–
	b)	O	+	O	–	O	+	+	–
INS1		–	+	+	+	O	–	+	+
SALES	a)	+	O	–	O	+	–	+	–
	b)	+	+	–	–	–	+	+	–
	c)	+	+	–	–	–	+	+	–
MED	a)	+	O	–	–	+	O	+	O
	b)	+	+	–	–	+	O	+	O
	c)	+	O	O	–	+	O	+	O
NET2		+	O	–	–	O	+	O	+
PUB1	a)	O	+	–	+	O	–	+	O
	b)	O	+	–	+	O	–	O	+
INS2	a)	+	+	–	+	+	O	+	O
	b)	+	+	–	+	+	O	+	+
DEV		–	+	O	–	+	+	+	O
PUB2		O	+	O	O	+	O	+	O
BANK1		+	O	–	–	+	–	+	O
BANK2		+	O	O	–	+	O	+	–
BANK3	a)	– (O)	+	–	+	+	–	+	–
	b)	(O) +	+	–	+	+	+	+	O

[20] The term *requirements* in this facet (2.2.1) denotes the artefact(s) capturing required functionality and qualities of a future envisioned system, the target aimed for, as opposed to facet 3.1.3 where the term is used to denote the system specification.

Tab. 10 summarises the data for the facets *viewpoint* (Facet 2.2.4), *granularity* (Facet 2.2.5), and *cases being modelled* (Facet 2.2.6).

- The data for Facet 2.2.4 (viewpoints) shows that in all the projects, the dynamic viewpoint is important and is captured through scenarios. But scenarios are also heavily related to the static viewpoint: in 17 out of 20 scenario types data is included or plays a vital role. Non-functional requirements (NFR)[21] on the other hand play only a minor role within the examined scenarios: They are heavily considered only in the projects NET1 and SALES and to lesser extent in five further projects.
- Scenarios are used on all levels of granularity (Facet 2.2.5). We find scenarios to be mostly describing tasks (8 out of 12 projects, 12 out of 20 scenario types), but significant fractions model business processes and/or single working steps, too.
- Scenarios are partial models (Facet 2.2.6) inasmuch as they capture predominantly the normal flow of actions; alternative and exceptional flows are modelled only partially: All scenario types model the normal flow of actions, but only four out of 20 scenario types fully model exceptional flows.

2.3 Representation of Scenarios

Div. 2.3 of the taxonomy deals with the representational aspects of the scenarios used in the projects under consideration.

In Facet 2.3.1, we examine the underlying ontology by which the knowledge captured in the scenarios is expressed. This facet is subdivided into the following characteristics:

- *Key primitives*: what are the building blocks of scenarios (e.g. actors, use cases, objects, events or messages)?
- *Structuring constructs:* to which degree are certain structuring mechanisms (sequence, iteration, alternative, composition) and abstraction constructs (type abstraction, hierarchical decomposition) employed to organise the knowledge captured in the scenarios? Is there some kind of graphical overview representation?
- *Formality*: are the scenarios represented in a formal, semiformal or informal language[22].

Facet 2.3.2 deals with the notation used for the scenarios. We differentiate the following categories:

- *Free text*: We consider flow-text having only little structure like chapters, subchapters, sections, pages and paragraphs to be free text.
- *Structured text*: If a structure, a layout and/or fixed definite parts are given, that need to be filled and specified, we speak of structured text. Usually a template is given; often selections, values to choose from and ranges are specified and the process of filling the template might even be machine-supported. Certain constructs might be marked by keywords; for example, "if ... then ... else ..." or "while ... do ...".

[21] excepting the user interface and related requirements

[22] We define a scenario that uses primarily free text and only some tables, pictures and/or diagrams to be informal. A scenario that is presented in a concise and syntactically as well as semantically defined graphical representation, but includes natural language text as in annotations and the like, is considered to be semiformal. A scenario with a syntactically and semantically completely defined notation is formal.

- *Restricted text*: If keywords are specified and the language to describe scenarios is restricted we speak of restricted text. For example, restrictions may concern precise meanings of words such as ‘and’ (logical conjunction) or ‘then’ (temporal sequence). The language could also be constrained to use program-like structures only (for example, sequence, alternative and iteration). In its extreme, restricted text is a programming language.
- *Table*: Scenarios are represented in a tabular form or tables are added to the text of the scenarios[23]. Tables added could be decision tables, flow of action tables and the like.
- *Diagram*: Graphical representations help in conveying the overall picture and the connections that exist among scenarios, objects and other modelling artefacts. Examples are message sequence charts, interaction diagrams, state-transition diagrams and so on.
- *Image*: Images might be pictures of the desired UI, screenshots and the like.

It should be noted that these categories are not exclusive, e.g. textual scenarios may be enriched by tables, diagrams and images.

In Facet 2.3.3, we consider the presentation style of the scenarios, which we differentiate into

- *static*;
- *interactive* (UI-models);
- *animated.*

The data for Div 2.3 were extracted primarily from samples of scenarios shown to us as used in the different projects. They also represent a condensate of the answers to questions concerning notation, structuring mechanisms and abstractions used for scenarios. Thus they blend answers to direct questions with our own observations in sample scenarios.

Tab. 11: Representation I

Project		2.3.1 Ontology								
		2.3.1.1[24] Primitives	2.3.1.2[25] Structuring/abstraction constructs							2.3.1.3 Formality[26]
			sequ.	iteration	alterna-tive	compo-sition	overview diagram	type abstr.	hier. decomp.	
NET1	a)	keywords for views	–	–	–	–	–	–	–	IF
	b)	none	–	–	–	–	–	–	–	IF
INS1		events, activities, actors, control flow	+	–	+	–	+	+	+	SF

[23] Often structured text can be presented as a table, yet to keep tables easily comprehensible, expressive and of manageable size the values in table fields are often restricted.

[24] Primitives are listed in arbitrary sequence. Their place in the list neither implies a valuation of their importance in scenarios nor the frequency of their appearance and use in scenario descriptions.

[25] Most scenarios are written in natural language and natural language per se provides for most structures like sequence, alternative and iteration. So the values presented for the characteristic “*Structuring/ abstraction constructs*” are somewhat arbitrary.

[26] F = Formal, SF = Semiformal, IF = Informal

Project		2.3.1 Ontology								
		2.3.1.1[24] Primitives	2.3.1.2[25] Structuring/abstraction constructs							2.3.1.3 Formality[26]
			sequ.	iteration	alterna-tive	compo-sition	overview diagram	type abstr.	hier. decomp.	
SALES	a) b) c)	activities, dataflow actors, use case actors, messages	+ – +	– – +	+ – +	– + –	+ + –	+ + –	+ – –	SF

MED	a)
	b)
	c)
		actors, actions, objects, use cases
		classes, responsibilities, collaborations
		data fields, constraints, events, (button) effects
			–
			+
			+
				–
				–
				–
					–
					–
					O
						–
						–
						–
							–
							–
							–
								O
								+
								+
									–
									–
									–
										IF
										SF
										IF - SF

NET2		data fields, constraints, events, button effects, deviations from standard behaviour	+	–	O	–	–	+	–	IF
PUB1	a) b)	both for a) and b): use case/scenario id, owner, version, overview, event, precondition, action, result, postcondition, exception, comments, sources	+ +	+ O	+ –	O O	+ –	+[27] +	O –	IF IF
INS2	a) b)	both for a) and b): use case id, goal, actor, event, objects, precondition, action, postcondition, exception	+ +	O O	O –	O O	+ –	+ +	O –	IF IF
DEV		use cases, objects, events, actions	–	–	+	–-	O	+	O	IF
PUB2		use case diagram, actors, use cases, partially pre- and post-conditions, events, GUI-elements (windows), data fields	O	–	+	–	+	+	O	IF
BANK1		user role, pre/ postcondition, predecessor use case, successor use case	–	–	–	–	–	–	–	IF
BANK2		actor, pre/ post condition, triggered by, triggers	–	–	–	–	–	–	–	IF - SF
BANK3	a) b)	for a) and b): no explicit, specific primitives	– –	– –	– –	– –	O –	– –	O O	IF IF

[27] Scenarios of type b) are regarded as instances of a).

Tab. 11 summarises the data for the facet *ontology* (Facet 2.3.1).

- The primitives used for expressing scenarios (Characteristic 2.3.1.1) vary widely among the various scenario types. However, certain primitives are rather common, such as the concept of actors or objects which occur in 8 out of 20 scenario types or event which is used in 5 scenario types. Pre/post-conditions are also frequently represented (6 scenario types). It should be noted that in most cases there is no explicit metamodel of the primitives to be used in the scenario such as in, e.g., UML message sequence charts. Rather the primitives are merely headings of the sections of a textual scenario template.
- Most scenarios make only little use of structuring mechanisms (Characteristic 2.3.1.2). In our sample only few scenario types utilised structuring mechanisms like sequence (used in 10 out of 20 scenario types), iteration (2 out of 20) and alternative (6 out of 20). Only one scenario type utilises an explicit composition mechanism.
- Abstraction mechanisms are not commonly used either. Even though an overview diagram is part of use case modelling in many methodologies, in the sample projects reflected in this paper only five out of 12 projects used overview diagrams. Even less provided an explicit mechanism for hierarchical decomposition (two scenario types out of 20).
- Scenarios are mainly informal or at most semiformal models; no formal models are used (characteristic 2.3.1.3). More precisely, all scenarios are informal (13 out of 20 scenario types), informal including some semiformal aspects (2 out of 20) or semiformal (5 out of 20).
- Tab. 12 condenses the data concerning the notation (Facet 2.3.2) and presentation (Facet 2.3.3) of the scenarios found in the projects.
- Natural language is the dominant means of notation. (Facet 2.3.2) All but one project in the sample use free or structured text to describe scenarios. The one exception uses a diagrammatic notation (in project SALES). Restricted text is used in only four projects. Diagrams and images are used to enhance textual descriptions. Images (for example screenshots, mock-ups or sketches of the user interface) are used in seven projects as a supporting means to the other notations applied. Only in three projects images are more than just an extra.
- The presentation of the scenarios is static throughout all projects (Facet 2.3.3). No project uses any form of animation and only one scenario kind engages some form of interactivity (SALES b).

Tab. 12: Representation II

Project		2.3.2 Notation						2.3.3 Presentation		
		2.3.2.1 Free Text	2.3.2.2 Struct. Text	2.3.2.3 Restr. Text	2.3.2.4 Table	2.3.2.5 Diagram	2.3.2.6 Image	2.3.3.1 Static	2.3.3.2 Inter- active	2.3.3.3 Anima- ted
NET1	a)	+	+	–	–	O	–	+	–	–
	b)	+	–	–	–	O	O	+	–	–
INS1		+	+	–	–	+	–	+	–	–
SALES	a)	–	–	O[28]	–	+	–	+	–	–
	b)	–	–	O	–	+	+	+	+	–
	c)	–	–	O	–	+	–	+	–	–
MED	a)	+	–	–	–	–	–	+	–	–
	b)	–	–	–	+	–	–	+	–	–
	c)	–	+	–	–	–	+	+	–	–
NET2		O	+	–	–	–	+	+	–	–
PUB1	a)	O	+	–	–	O	–	+	–	–
	b)	O	+	–	–	–	O	+	–	–
INS2	a)	O	+	–	–	O	O	+	–	–
	b)	O	+	–	–	O	O	+	–	–
DEV		+	O	O	–	–	O	+	–	–
PUB2		+	O	–	O	O	–	+	–	–
BANK1		+	+	–	–	–	–	+	–	–
BANK2		+	+	–	–	–	–	+	–	–
BANK3	a)	+	–	–	O	O	O	+	–	–
	b)	+	O	–	O	–	O	+	–	–

2.4 Scenario Figures

Div. 2.4 captures two size metrics for each scenario type: the total number of scenarios and the typical size of a single scenario. Size is measured in pages A4 of text, as there is no suitable measure available to evaluate the size of a scenario. Both numbers were asked for directly. In some cases, they are estimates by the project managers.

Tab. 13 shows the data concerning the number and size of the scenarios encountered in the projects. The main observation is that the number and size of scenarios is relative small. Although the number ranges from 10 to 2000 (Facet 2.4.1), only in two of the 20 projects, more than 100 scenarios of one type were created (PUB1, PUB2[29]). The typical size of a scenario is 1-3 pages, but again size varies significantly across projects ranging from a few sentences up to 30 pages (Facet 2.4.2).

[28] Text associated with the diagram elements (by the diagram tool).

[29] Project PUB1 is huge: Total effort when finished will be 1000 person years and an estimated 2000 scenarios will have been written by the time.

Tab. 13: Scenario Figures

Project		2.4.1 Total number of scenarios	2.4.2 Typical size of a single scenario
NET1	a)	10	1-2 pages
	b)	24	1 sentence - 1 page
INS1		40[30]	2-5 pages
SALES	a)	50 - 70	?
	b)	?	?
	c)	?	?
MED	a)	20	1-3 pages
	b)	?	1 page
	c)	40	1-3 pages
NET2		60[31]	1-2 pages
PUB1	a)	350[32]	3-8 pages
	b)	1400 - 2000[33]	5-10 pages
INS2	a)	5[34]	3 pages
	b)	25	3-6 pages
DEV		50	1 page
PUB2		170	1-2 pages
BANK1		20	max. 1.5 pages
BANK2		20	1-2 pages
BANK3	a)	10	2-3 pages
	b)	23	5-30 pages

2.5 Summary and Findings

In Section 2, we have highlighted the product properties of the scenarios used in the examined projects concerning their content, their representation and their size and number. The main findings about the content expressed in the scenarios can be summarised as follows:

- **Modelling of requirements prevails** (Facet 2.2.1): Scenarios are mainly used to capture and model requirements. Only in few projects they are applied to design solutions and even less to model the current situation.
- Scope of scenarios lies on interaction (Facet 2.2.2).
- **Type level scenarios are predominant** (Facet 2.2.3) but frequently complemented by instance level scenarios.
- **Scenarios capture not only behaviour, but also data aspects** (Facet 2.2.4): In all the projects, behaviour – be it on an internal or an external level – is important and is captured through scenarios. But scenarios are also heavily related to the data viewpoint.
- **Scenarios are medium grained** (Facet 2.2.5): we find scenarios to be mostly describing tasks, but significant fractions model business processes and/or single working steps, too.
- **Scenarios are partial models** (Facet 2.2.6): the by far predominant fraction of the projects only capture normal cases in scenarios which is somewhat contradicting to what

[30] 20-25 of them self-contained.

[31] In general 2-4 per domain object.

[32] So far, ~ 1000 expected.

[33] Each business process is documented by a use case (a). For each use case 4-6 scenarios (b) are developed.

[34] So far, 27 expected.

is often claimed in research literature, namely that scenario are ideal means to explore exceptional situations (e.g. [Maiden et al., 1998]).

From the representational point of view, the main findings of Section 2 are:

- **Little use of structuring mechanisms** (Characteristic 2.3.1.2): only few projects employed structuring mechanisms like sequence, iteration and alternative or even an explicit composition mechanism.
- **Abstraction mechanisms are restricted to overview diagrams** (Characteristic 2.3.1.2): Few projects utilised overview diagrams known from well-known object-oriented methodologies, even less provided an explicit mechanism for hierarchical decomposition.
- **Scenarios are mainly informal or at most semiformal models** (Characteristic 2.3.1.3); no formal models are used. One reason is that scenarios serve as a means to elicit, document and validate requirements. They facilitate and enhance communication between users and requirements engineers (see section 3 in this paper). As they are used to elicit and validate requirements, they have to be understandable to users who might be unacquainted with or unwilling to learn (graphical) semiformal or formal languages.
- **Text dominates as notation** (Facet 2.3.2). This is also an indicator that scenarios have to be communicable. However, most projects feel the need for at least some structure and hence often impose a template format upon the scenario documents. Diagrams and images are sometimes used to enhance textual descriptions.
- **Presentation is static** (Facet 2.3.3); advanced animated or even interactive presentation styles are not state of current practice.
- From the size metrics that have been raised, the following conclusions can be drawn: **Small scenario size** (Facet 2.4.2): The typical size of a scenario is 1-3 pages, but size varies significantly across projects ranging from a few sentences up to 30 pages.
- **Small number of scenarios** (Facet 2.4.1): The same is true for the total number of scenarios: It ranges from 10 to 2000. It has to be noted though, that in only two of the 20 projects, more than 100 scenarios of one type were created (PUB1, PUB2[35]).

As most of the observed projects are in the range of 10 to 60 scenarios (14 out of 17), the conclusion may be drawn that it is reasonable to divide bigger projects into manageable subprojects of about this size (up to 100 scenarios). Projects bigger than that are increasingly hard to manage: There is a lack of C&V management tools suited for use with scenarios. As a consequence two third of all the projects report that they had problems in managing scenario documents (see section 4 and 5).

A general finding from this section is that many projects claim to use scenarios according to the Jacobson OOSE approach. These scenarios are mostly informal and use very little structuring constructs and abstractions. However, the scenarios observed vary very widely in terms of their content, representation and size, even within a single project. This can be seen as an indicator that current methodologies leave (too) much room for interpretation and call for specific adaptations of the scenario approach used in the projects.

[35] Project PUB1 is huge: Total effort when finished will be 1000 person years and an estimated 2000 scenarios will have been written by the time

3 Goals of Scenario Use

This section deals with the main goals underlying the use of scenarios in software development projects. We are particularly interested in those reasons that can be inferred from real life development experiences and go beyond those described in the scenario-oriented research papers and textbooks, e.g. the high comprehensiveness for developers *and* clients/users, the role as a communication medium for system requirements [Jacobson, 1995; Potts et. al., 1994], or as a means to model and possibly reengineer business processes [Jacobson et al., 1995]. Nevertheless, these *standard purposes* have been mentioned as relevant for some of the inspected projects as well. We are also interested in whether scenarios are used to describe the current situation or to create a vision of a future system. The data to answer questions like these are presented in this section.

3.1 Purpose

Div. 3.1 highlights the different purposes underlying the use of scenarios. After reviewing the interview material gathered from twelve projects, it became obvious that the objectives for using scenarios in software development projects can be characterised by an activity associated with a subject. To structure the objectives for using scenarios we classified the insights gained in a two dimensional matrix (cf. Figure 3.1). The matrix pairs certain activities (forming the columns) with certain subjects (the rows), e.g. *understanding* the *application domain*, *eliciting* the *requirements*, or *negotiating* the *contract*. We discovered four important subject groups relevant for understanding the objectives for using scenarios:

- the *application domain* describing the environment the system will be installed in;
- the *vision of the projected system* describing the product to be developed;
- the *software artefacts* created during the development process and
- the *contract* between software developers and software procurers.

Subjects / Activity : *VALUE*	Understanding	Elicitation	Validation	Documentation	Mediation/ Negotiation
Application Domain					
Business Process					
Terminology					
Domain Knowledge					
...					
System Vision					
Software Artefacts					
Requirements					
Architecture					
Code					
Test Cases					
Interface					
...					
Legal Issues					
Contract					
...					
...					

Figure 3.1: Activity/Subject Matrix for the Division “Purpose”

Besides the subjects, we identified five important activities on these subjects:

- understanding, i.e. the creation of a mental model in the minds of (typically) the developers;
- elicitation, i.e. the process of making knowledge about the subject explicit;
- validation, i.e. the capability to judge about the subject’s correctness or adequacy;
- documentation, i.e. the storage of knowledge on a persistant medium, and
- mediation/negotiation, i.e. the process of communication between client and developer to convey the subject.

The pairs of the matrix are instantiated with the VALUE type to indicate strong, weaker, or no relevance of the appropriate subject/activity pair for the current project. The data presented below has been derived from the interviews, without explicitly asking about the relation of scenarios to certain subjects and certain activities (through interpretation of the data). Therefore, an empty cell indicates that the interviewee did not explicitly mention the corresponding purpose.

The following tables represent all 12 projects and contain the facets of the above presented taxonomy. To allow for an easier comparison of projects the tables have been reorganised. Each row contains data of one project and each column represents one subject/activity pair.

3.1.1 Scenario Purposes with respect to Application Domain

First, we present the gathered data concerning the relationship between scenarios and the *application domain*. This facet has been divided into *business process*, *domain terminology*, and *domain knowledge* to increase the level of detail.

Tab. 14 (covering the characteristics 3.1.1. A – C) describes the relationship between scenarios and aspects of the application domain. It is obvious that all 12 projects use scenarios to understand the application domain knowledge; 9 projects use scenarios to understand the domain terminology, and 7 projects use scenarios to understand business processes. Thus, understanding aspects of the application domain is one main purpose for using scenarios. The activities *elicitation* and *documentation* are similarly important for all subtopics of the application domain (6 to 10 projects out of 12). In the projects PUB1 and INS2 scenarios are used for nearly all mentioned activities with a stress on understanding and documenting business processes and the domain knowledge.

3.1.2 Scenario Purposes with respect to System Vision

The next table deals with the system vision, i.e. descriptions of the functionality of the system that should be built in the project, and in which way scenarios are used to understand, elicit, or document this vision.

Tab. 14: Scenarios Purposes with respect to Application Domain

Project		3.1.1.A Application domain: business process					3.1.1.B Application domain: terminology					3.1.1.C Application domain: domain knowledge				
		Understanding	Elicitation	Validation	Documentation	Mediation/ Negotiation	Understanding	Elicitation	Validation	Documentation	Mediation/ Negotiation	Understanding	Elicitation	Validation	Documentation	Mediation/ Negotiation
NET1	a)	+			O	+	+			O	+	+			O	+
	b)															
INS1		+	+		+	+	+	+		+	+	+	+		+	+
SALES	a)	+			+	O	+					+			+	
	b)	+			+	O	+					+			+	
	c)	+			+	O	+					+			+	
MED	a)	O	O	O			+	+				+	+			
	b)			O	O		O	O				O	O			
	c)									O					O	
NET2							O	O		O	O	O			O	O
PUB1	a)	+	+	O	+	+	O[36]	O	O	O	O	+	+	O	+	+
	b)	+	+	O	+	+	O	O	O	O	O	+	+	O	+	+
INS2	a)	+	O	O	+	O	+	O	O	O	O	+	+	O	+	
	b)	+	O	O	+	O	+	O	O	O	O	+	+	O	+	
DEV										O		O			O	
PUB2		+	O		O		O					+	+	O	O	
BANK1[37]												O	O			
BANK2[38]							O	O				O	O			
BANK3	a)	+	+	+	O		+	+	+	O		+	+	+	O	
	b)	O	O	O	O		O	O	O	O		O	O	O	-	

As can be seen in Tab. 15 most projects (9 of 12) use scenarios to understand the system vision or to elicit, document, and mediate/ negotiate it (each 8 of 12). Exactly those projects use scenarios to document the system vision that also use them to mediate and negotiate it, and vice versa. A similar strong correlation can be found between understanding and eliciting the vision (except for project NET1). 10 out of 12 projects mentioned one or more activities related to the system vision as an objective for using scenarios, thus this subject is one main reason for using scenarios in the visited projects. The only exceptions are projects SALE and NET2.

[36] in conjunction with a glossary.

[37] The interviewees explicitly mentioned that scenarios were not used in relation to business processes at all. The application domain terminology has been acquired in an earlier project without the use of scenarios.

[38] The interviewees explicitly mentioned, scenarios were not used in relation to business processes at all.

Tab. 15: Scenario Purpose with respect to System Vision

Project		3.1.2 System Vision				
		Understanding	Elicitation	Validation	Documentation	Mediation/ Negotiation
NET1	a)	+			O	+
	b)					
INS1		O	O		O	O
SALES	a)					
	b)					
	c)					
MED	a)	O	+			
	b)					
	c)	O				
NET2						
PUB1	a)	+	+	O	+	+
	b)	O	O	+	O	O
INS2	a)	O	+	+	+	O
	b)	O	+	+	+	O
DEV					O	O
PUB2		O	+	-	+	O
BANK1		+	+	+	+	+
BANK2		+	+		+	+
BANK3	a)	-	-	-		
	b)	+	O	+		

3.1.3 Scenario Purposes with respect to Software Artifacts

Because several rather different *software artefacts* were mentioned as reasons for using scenarios they are discussed individually in the following table (3.1.3.A-E).

Tab. 16 (covering the characteristics 3.1.3. A - E) shows how software artefacts represent reasons for using scenarios. The gathered data reveal significant differences concerning the different artefacts. Scenarios were used to a large extent to understand, to elicit, to validate, to document, and to mediate/ negotiate requirements and the software interface. The projects NET1 and INS1 are exceptions to this trend, since hardly any relationship between scenarios and software artefacts are documented. The reason is that these projects stopped before developing these artefacts. In addition, the project PUB1 uses scenarios for the acquisition of requirements but does not use them in relation to the interface.

The relation between scenarios and requirements is very close. 11 out of 12 projects use scenarios to document requirements, 8 projects use scenarios to understand, to mediate/negotiate and to validate requirements, and 7 projects use scenarios for the elicitation of requirements. A similar but weaker relationship holds between scenario use and interfaces. Here the main purpose of scenarios lies in understanding and validating user interfaces (7 of 12). The elicitation and the documentation facet is relevant in 6 of 12 projects.

The tables show very weak relationships between scenario use and the code or the software architecture. Only project SALE mentioned documentation of code as an objective for the use of scenarios. In the project DEV the main objective for scenario use is their relationship to the software architecture, since scenarios were used to elicit, to understand, to negotiate, to document, and to validate the software architecture.

The relationship between scenarios and test cases is of medium importance as shown in table 3.1.3.D. Only 4 projects use scenarios to elicit or to document test cases. Only the project BANK1 shows a tight relationship between scenarios and test cases, since scenarios were used to elicit, to document, to negotiate, and to validate test cases.

Tab. 16: Scenario Purposes with respect to Software Artefacts

Project		3.1.3.A software artefacts: requirements					3.1.3.B software artefacts: architecture					3.1.3.C software artefacts: code					3.1.3.D software artefacts: test cases					3.1.3.E software artefacts: interface				
		Understanding	Elicitation	Validation	Documentation	Mediation/ Negotiation	Understanding	Elicitation	Validation	Documentation	Mediation/ Negotiation	Understanding	Elicitation	Validation	Documentation	Mediation/ Negotiation	Understanding	Elicitation	Validation	Documentation	Mediation/ Negotiation	Understanding	Elicitation	Validation	Documentation	Mediation/ Negotiation
NET1	a)																									
	b)				+																					
INS1																										
SALES	a)	+			+	O				+					+					+		+		+	+	
	b)	+			+	O				+					+					+		+		+	+	
	c)	+			+	O									+					+		+		+	+	
MED	a)	O	+	+		O												O				O	+	+		
	b)	O	+	+		O												O				O	+	+		
	c)	O			+	O																			+	
NET2				+	O																			+	+	
PUB1	a)	+	+	+	+	O																				
	b)	+	+	+	+	O																				
INS2	a)	+	+	+	+	O												O		O			O	O		
	b)	+	+	+	+	O												O		O			O	O		
DEV				O	O		O	O	+	+	O						O					O		+	O	O
PUB2		+	+	+	+	+	O															O	O		O	O
BANK1		+	+	+	+	+												+	+	+	+	+	+		+	+
BANK2		+	+		+	+											O	O		O		+	+	O		O
BANK3	a)	O	O	O	O																	-	-	-		
	b)	+	+	+	+																	O	O	O		

3.1.4 Scenario Purposes with respect to Legal Issues

Tab. 17 highlights the role scenarios play wrt. to the contract between supplier and customer.

In 2 of the analysed projects scenarios are heavily used to document the contract between the system procurers and the system developers, i.e. the scenario descriptions become a part of the contract. In five other projects, scenarios serve to a lesser extent for documenting the contract. The project BANK2 also uses scenarios as a mediation and negotiation means when it comes to fixing a contract. Other purposes with respect to legal issues play no or only a minor role.

Tab. 17 : Scenario Purposes with respect to Legal Issues

Project		3.1.4 Legal issues: contract				
		Understanding	Elicitation	Validation	Documentation	Mediation/ Negotiation
NET1	a)				+	
	b)					
INS1						
SALES	a)					
	b)					
	c)					
MED	a)					
	b)					
	c)	o			o	o
NET2					o	
PUB1	a)					
	b)					
INS2	a)				o	
	b)				o	
DEV						
PUB2		o			o	
BANK1						
BANK2					+	+
BANK3	a)	-		-		
	b)	o		o		

3.2 Main Objectives

Because not all relevant rationales for using scenarios in software development projects can be classified according to a rigid scheme as that of Figure 3.1, another division of the taxonomy summarises the main objectives for using scenarios. The data gathered in interviews concerning the main objectives represent explicit statements about the objectives for using scenarios in the concrete projects. Thus, the data may partly overlap with that classified with the classification scheme of Figure 3.1.

Tab. 18 contains some explicit statements about the objectives for using scenarios in the concrete projects. These statements are partly reflected in the discreticized tables 14-17. But they also contain some additional viewpoints, e.g. in project MED scenarios have been used to *structure* the understanding of the future system. In PUB1 scenarios were used to *manage the complexity* of the application domain and project DEV employs scenarios to document and communicate *design decisions*. This use of scenarios in project DEV is an explanation for the special role this project plays in some of the presented tables of this section, e.g. in DEV scenarios are not mentioned at all in relation to the application domain.

Tab. 18: Main Objectives of Scenario Use

Project		3.2 Main Objectives of Scenario Use
NET1	a)	▪ understand domain
	b)	▪ fix requirements
INS1		▪ understand and document future handling ▪ improve communication with customer
SALES	a) b) c)	in all cases: understand processes and document them
MED	a)	▪ document and structure software engineer's understanding of future system ▪ validation through user
	b)	▪ intermediary step between scenario of a) and class model
	c)	▪ documenting system functionality from user perspective, communicating system functions to user
NET2		▪ communicate underlying data-model to user/domain expert, thereby, enabling a validation ▪ isolate and document non-standard features apart from generated standard behaviour of prototype
PUB1	a)	▪ identify, understand, and document business processes, transferring domain knowledge from user to sw eng., manage complexity of application domain
	b)	▪ - dito -
INS2	a)	▪ identify, understand, and document business processes, transferring domain knowledge from user to software engineer
	b)	▪ - dito -
DEV		▪ document and communicate design decisions
PUB2		▪ communication means between developer and domain experts, specify reqs from the user's perspective
BANK1		▪ acquire initial understanding of domain, documentation of user requirements
BANK2		▪ enable sw engs. to understand desires of customer, medium for domain experts to express their vision of the future system without the need for learning a new notation
BANK 3	a)	▪ understand the appl. Domain/the business and its context/environment, enable and improve communication between developers and customer
	b)	▪ develop a vision of the desired system, communicate this vision to customer, capture requirements of system to be (Requirements specification) down to system components

3.3 Summary and Findings

The general objectives for using scenarios that were derived from the given tables can be summarised as follows:

- **Scenarios as communication means**: Scenarios are mainly used to create a shared understanding between the different stakeholders by activities such as *understanding*, *elicitation*, and *mediation/negotiation*. These activities have been mentioned as main objectives for using scenarios.
- **Varying importance of different purposes in different projects**: The most important subjects according to the gathered data are the *domain knowledge* and *terminology*, the *system vision*, the *requirements*, the *interface*, and partly the *business processes*.
- **Scenarios and test cases**: The relationship between scenarios and *test cases* is less important which is due to maintenance problems of the initial scenarios that do not reflect

the current requirements at the time when the system is tested (cf. [Weidenhaupt et al., 1998]).

- **Scenarios and architecture**: Scenarios play a minor role in relation to the subject *software architecture*, except for the project DEV, where the main objective for using scenarios lies in their ability to document and validate the software architecture and design decisions.
- **Scenarios and code**: The software artefact *code* was only seldom mentioned in the interviews.
- **Validation only for requirements and interfaces**: The activity *validation* was mentioned much more seldom than the other activities. If validation was mentioned, the projects use scenarios to validate *requirements* and *interfaces* and partly *business processes* and the *system vision.*

4 Process

Scenarios can be used in a variety of ways during the software development process. These are determined on the one hand by the purpose and on the other hand by the software development method used. The software development method determines the artefacts capturing the main decisions regarding requirements, interface, architecture, implementation and test. Scenarios (and all other kinds of modelling techniques) are used as a medium to understand and elicit information on which decisions are based and to document, validate and negotiate these decisions.

The main purpose of scenario usage has been discussed in section 3. Here we concentrate on the embedding of scenario usage in the software development process. Since it is impossible to capture all the details of a software development process during a project, this embedding can only be described very roughly. We therefore split this section into two parts: in subsections 4.1-4.3 the major facets of scenario based software development processes are described, in subsection 4.4 we go into detail for particularly interesting elements of such a process. The latter describe the highlights of the projects.

Regarding the overall software development process we have captured the artefacts and the modelling techniques used. For requirements engineering we have also been interested in the following questions: How was the information captured in the scenarios obtained? What kind of management was applied to the scenario documents? Which tools had been used?

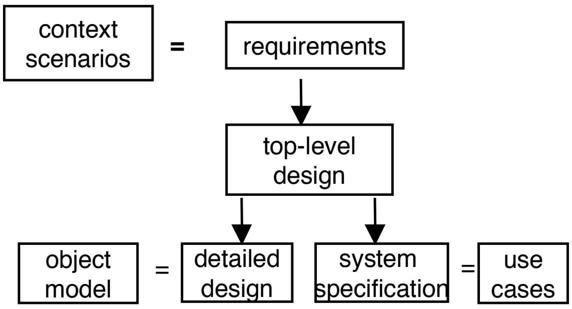
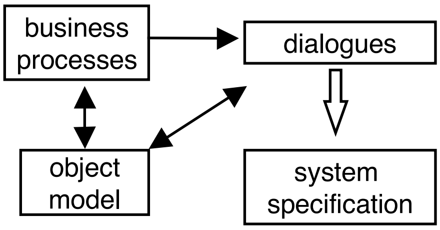
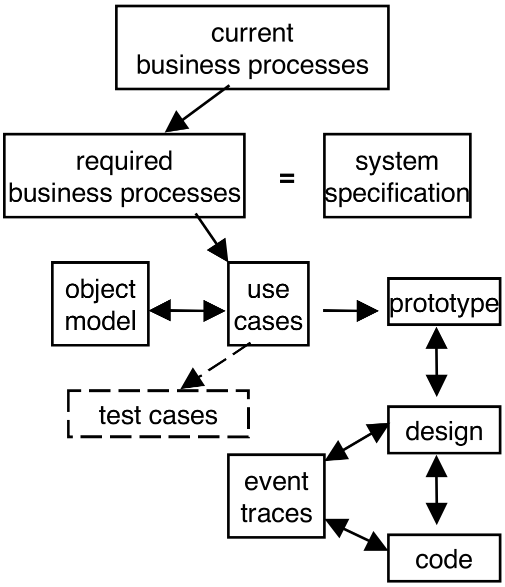
The following list gives an overview of the taxonomy divisions:

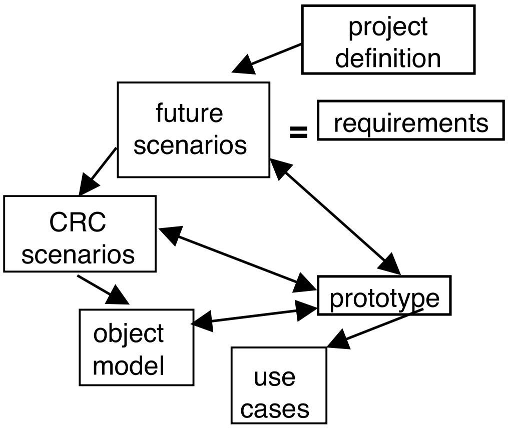
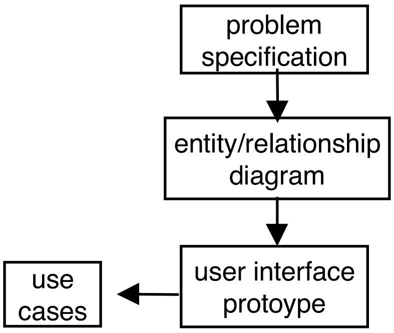
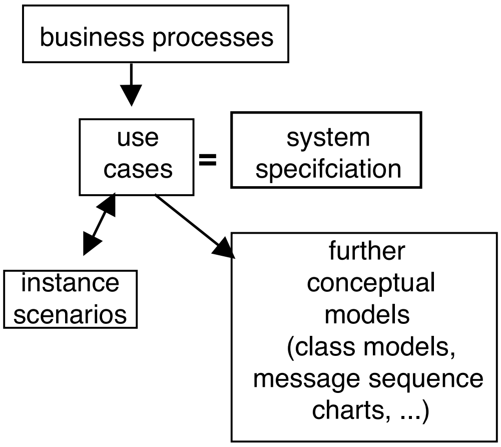
- a short description of the overall software development process (Div. 4.1),
- the documented or maintained relationships of scenarios to requirements, architecture, code, test cases, interface and glossary/object model (Div. 0),
- a characterisation of the requirements engineering method used (Div. 4.3),
- a description of the creation, usage, maintenance , change and quality control activities in the scenario lifecycle (Div. 4.4), and
- the tools that were used (Div. 4.5).

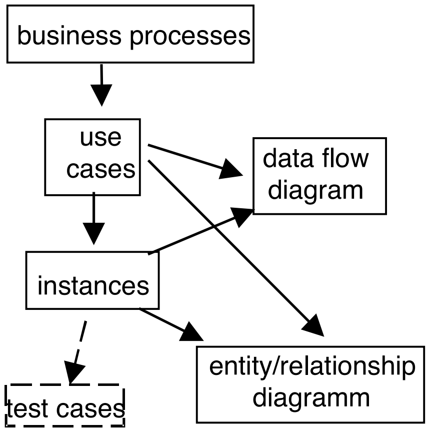
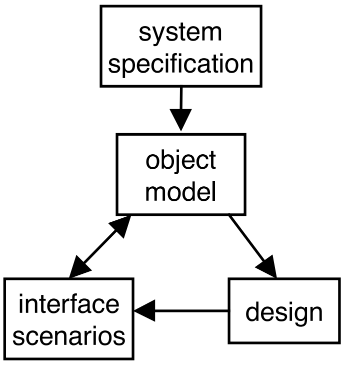
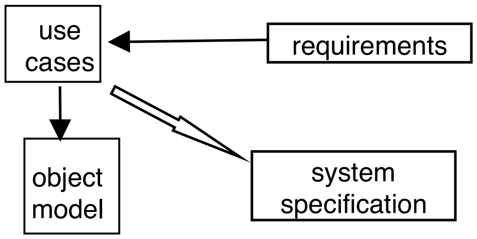
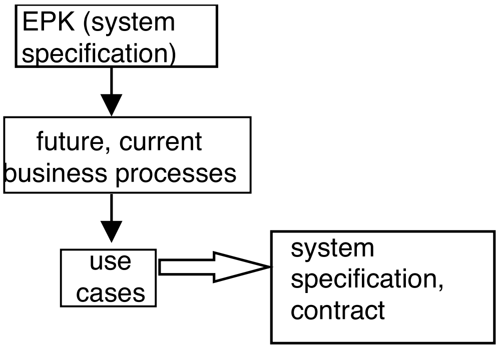
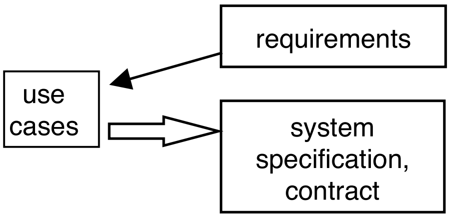
4.1 Process Context

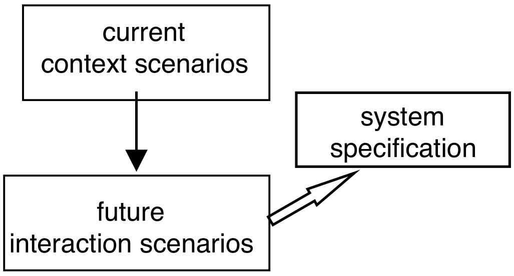
Div. 4.1 provides a short overview of how scenario generation and usage is embedded in the overall requirements engineering process. In Tab. 19, we show for each project the artefacts (depicted as boxes) developed, as well as their relationships: one-way arrows denote derivation, two-way arrows denote validation against each other. The equality sign between artefacts denotes the fact that one is a major part of the other, while a thick empty arrow stands for inclusion. Dashed arrows denote planned relationships.

Tab. 19 : Overview of the Scenario Generation and Usage Process

Project	4.1 Process Context	Overview Picture
NET1	The first kind of scenarios constitute a major part of the requirements specification. These were visions for services of a VLAN and their management. From that a top-level design for the management system was developed. This is detailed into a system specification consisting of use cases and a detailed design given by an object model.	 context scenarios = requirements top-level design object model = detailed design system specification = use cases
INS1	The project had to deliver a system specification. First, business process were defined. After one quarter also the object model was started and the two models were validated against each other. The business processes were refined into dialogues, which are part of the system specification. Validating the object model and the dialogues against each other also lead to a refined object model.	 business processes dialogues object model system specification
SALES	Business process describing the current state were derived from the existing system. From them the required business processes - constituting the system specification - were developed together with the domain experts. Common activities in the business processes were extracted and detailed into use cases. In parallel the object model was developed, and use cases and the object model were validated against each other. Then use cases were implemented in a prototype which was evaluated by end-users. Prototyping required to fix some design and implementation details. These were documented with event traces.	 current business processes required business processes = system specification object model use cases prototype test cases design event traces code

Project	4.1 Process Context	Overview Picture
MED	Based on a project definition provided by the customer, informal scenario scripts were developed documenting the requirements. These were structured into CRC cards from which an initial object model was developed. To validate the object model a prototype was developed. CRC cards and the scenarios served as test cases for the prototype. This lead to changes of both, the object model and the prototype. User interface centred use cases were used to document the system functionality from the point of the user.	 project definition future scenarios = requirements CRC scenarios prototype object model use cases
NET2	Starting from a coarse project specification, a problem specification was developed in workshops between the customer and the developers. The terminology defined in the workshop was the basis for an initial EER model. To communicate the EER model to the customer an user interface prototype was generated. Each dialogue was documented as a use case, especially deviations from generated functionality.	 problem specification entity/relationship diagram user interface protoype use cases
PUB1	First relevant topics were identified and pre-structured into business processes. Based on interviews with domain experts use case descriptions envisaging the future system usage are created as well as instance scenarios. The overall set of use cases constitutes the system specification from which further, more formal conceptual models such as class diagrams, message trace diagrams and user interface models are derived in later development stages.	 business processes use cases = system specifciation instance scenarios further conceptual models (class models, message sequence charts, ...)

Project	4.1 Process Context	Overview Picture
INS2	The scenario generation was based on more than 100 business processes identified earlier using a petrinet-based method. These were restructured into 27 business processes. Each business process was elaborated by conducting interviews with the domain experts/users and detailed as use case. The latter were enhanced by 4-6 instance level scenarios. Use case and scenarios built the basis for deriving data flow diagrams and EER models.	 business processes use cases data flow diagram instances test cases entity/relationship diagramm
DEV	From the system specification an object model was derived and detailed into the design. Scenarios are used in the design stage. There they help to structure, communicate and validate complicated dialog sequences. Also, they are used to validate the object model.	 system specification object model interface scenarios design
PUB2	First requirements are elicited and use cases written by the developers. They are iteratively reviewed and refined. The final version is part of the system specification and serves as basis for the object model.	 use cases requirements object model system specification
BANK1	Starting from an existing system specification, current and future business processes were identified to gain an understanding of the application domain. Then use cases were acquired with the domain experts that result in a system specification that serves as a major part of the contract between developers and customers.	 EPK (system specification) future, current business processes use cases system specification, contract
BANK2	Through interviews with domain experts requirements are identified. From them the developers create a first version of use cases. These were reviewed and refined and serve as a contract with the customer.	

Project	4.1 Process Context	Overview Picture
BANK3	Developers observed the work of the users and interviewed them. On that basis, scenarios describing the current application are written by the developers. Visionary scenarios are developed which are part of the system specification.	 current context scenarios system specification future interaction scenarios
	Legend:	= major part ↔ validation → derivation ⇨ inclusion ⇢ planned relationships

All projects use scenarios to describe interaction between the user and the software system. One half of the projects uses additionally either instances or visionary business processes.

In the following, we examine the relationships of scenarios to other artefacts in detail.

- **Relationship to requirements and system specification:** The relationships of scenarios to requirements are as diverse as the use of the term requirement itself. In one interpretation, requirements describe the envisioned effects of the software system on the processes of its environment. Seven of the projects used scenarios for that (NET1, SALES, INS1, MED, PUB1, INS2, BANK3), another four just used informal text (NET2, PUB2, BANK1, BANK2). Requirements often also refer to the system specification, namely the required system functionality. We found that the majority of projects uses Jacobson's use cases or other interaction scenarios for that (NET1, SALES, INS1, PUB1, PUB2, BANK2, BANK3). Given the exemplary nature of scenarios this is somewhat surprising. We only found three projects which used other means of system specification (DEV, NET2, BANK1), while two projects (MED, INS2) had no explicit system specification.

- **Relationship to user interface prototype:** In the literature, use cases are often proposed as a means to describe user interaction as input to or validation of user interface prototype development. This is also reported in [Weidenhaupt et al., 1998] and was the case for two of our projects (SALES, MED) . Interestingly, in three projects use cases were also generated from the prototype to document special features (DEV, NET2, MED), e.g. non-standard functionality.

- **Relationship to object model/ data model:** 8 projects produce an object model. In two projects (PUB2, PUB1) scenarios are input to object model development, in two projects (INS1, SALES, DEV) they are a means of validation. In project MED, CRC-cards are used for object model development and the prototype for validation. In project NET2 an EER model was developed from the specification and from that an interface prototype. In project INS2 scenarios have been used as a front-end to a structured method.

- **Relationship to existing system:** Requirements are often heavily influenced by existing systems. In three projects (SALES, BANK1, BANK3) scenarios have been used to model and understand existing business processes or use cases. In project PUB1 the current system has not been documented by scenarios, but the future use cases have been validated against the existing system.

For the projects, which used several kinds of scenarios (NET1, SALES, PUB1, MED, INS2), one kind of scenario was input for the other one: Instances are derived from scenarios on the type level, or the focus is shifted from current to future processes, or from the software context to the interaction.

Tab. 20: Documented and Maintained Relationships of Scenarios to Artefacts

Project		4.2.1 Require-ments		4.2.2 Architecture		4.2.3 Code		4.2.4 Test cases		4.2.5 Interface		4.2.6 Glossary / Obj. model	
		D[39]	M[40]	D	M	D	M	D	M	D	M	D	M
NET1	a)	O	–	O	–	–	–	–	–	–	–	–	–
	b)	+	+	O	–	–	–	–	–	–	–	O[41]	O[41]
INS1		+	+	–	–	–	–	–	–	O	O	O[41]	O[41]
SALES	a)	+	+	–	–	–	–	–	–	–	–	–	–
	b)	O	O	–	–	–	–	–	–	+	+	+	+
	c)	–	–	+	+	–	–	–	–	–	–	–	–
MED	a)	+		–		–		O	–			O	–
	b)	+		–		–		O	–			O	–
	c)	+		–		–		–		+	O	–	
NET2		+	–					–		+	O	O	–
PUB1	a)	+	O					–		–		O[42]	O
	b)	+	O					–		–			
INS2	a)	+	–						–		–	O[42]	–
	b)	+	–					O	–	O	–	–	
DEV		–	–	O	–	–	–	–	–	O	–	O[41]	–
PUB2		+	–	–	–	–	–	–	–	O	–	O[41]	–
BANK1		–		–		–		–		–		–	
BANK2		–		–		–		–		–		–	
BANK3	a)	–	–	–	–	–	–	–	–	–	–	–	–
	b)	O	–	–	–	–	–	–	–	O	–	O[41]	–

4.2 Relationship of Scenarios to Artefacts

In this division, we examine the documented and maintained relationships of scenarios to the major artefacts in each project, respectively. We call a relationship (e.g. a derivation or validation relationship) *documented* if this relationship is explicitly mentioned in the respective documents. An example is an explicit reference stated in a class diagram to those scenarios from which a certain class was elicited. Such relationships are regarded as *maintained* if a

[39] D = documented

[40] M = maintained

[41] Object model

[42] Glossary

change in either the scenarios or the related artefacts is reflected in the respective counterpart, e.g. that if a scenario is changed the corresponding class model is adapted accordingly, if necessary. The fact that scenarios are not related to an artefact not necessarily means that there has been no influence between the two. A reason might be that the project has not yet produced the artefact.

The data in Tab. 20 were extracted from the process descriptions given by the interviewees and the sample documents to which we had access.

Not surprisingly, all projects document relationships of scenarios either to the requirements (9 out of 12) or the user interface prototype (8 out of 12). 10 projects also document the relationship to the object model, however only in one project this was a major issue. Documented relationships to architecture or test cases are much less frequent, and no relationships to code are documented. The maintenance of these relationships is neglected more often than not.

4.3 Used Requirements Engineering Method

For the used requirements engineering method we distinguished standard methods from non-standard ones, and also identified the type of the method. Tab. 21 lists the interview data.

Tab. 21: Requirements Engineering Method Used

Project		4.3.1 Standard method	4.3.2 Which standard method OR description of in-house method	4.3.3 Type of method
NET1	a) b)	No Yes	(but experience with similar studies) Jacobson OOSE	OO
INS1		No	adaptation of method used in different project in the same company	OO
SALES	a) b) c)	Yes Yes Yes	OMT, StP as supporting tool	OO
MED	a) b) c)	No No No	Isotec (in-house method)	OO
NET2		Yes	EER + incremental prototyping	Data
PUB1	a) b)	No No	Based on Jacobson Use Case modelling (OOSE). Notation based on OMT, now UML	OO
INS2	a) b)	No	Based on Jacobson Use Case modelling and SERM (derivative of SA)	Functional/ Data
DEV		No		OO
PUB2		Yes	Jacobson OOSE	OO
BANK1		Yes	Jacobson Use Cases and OMT	OO
BANK2		No		OO
BANK3	a) b)	No No		OO

Only one project does not use an object-oriented method. Five of the projects apply a variant of Jacobson’s use case approach. As reported in [Weidenhaupt et al., 1998] this approach has been adapted to the project specifics.

4.4 Scenario Lifecycle

Tab. 22 and Tab. 23 summarise the interview data about scenario creation, usage, maintenance, change and quality control and the supporting tools.

As expected, scenarios are often created based on interviews (INS1, PUB1, INS2, PUB2, BANK3), on-site observation (PUB2, PUB1, BANK3) or on workshops (INS1, SALES) together with users and domain experts. For project DEV the scenarios were developed solely by the software engineers, since they coincide with the domain experts. In four projects (MED, NET2, BANK1, BANK2) the users were not involved with the first draft of the scenarios, but in three of them scenarios were reviewed by the users. In that case the source of information were informal requirements specification, a user interface prototype or an existing system specification which had to be reworked. Project MED used the prototype as discussion focus with the user and scenarios only as documentation. The usage data listed in Tab. 22 is quite diverse, sometimes emphasising the relationships to other artefacts as described in the process pictures of Div. 4.1 and sometimes putting emphasis on the purpose as described in Div. 3.1.

Tab. 22: Creation and Usage of Scenarios

Project		4.4.1 Creation	4.4.2 Usage
NET1	a) b)	▪ Brainstorming in team ▪ Documentation according to guidelines ▪ Reviews with additional domain expert	Used for top level design of the MS. Used for system function specification and analysis object model.
INS1		Business processes developed based on ▪ interviews ▪ workshops (for critical issues only)	Used for documenting the new business processes, for definition of object model and definition of dialogs.
SALES	a)	Business process derivation from old system and workshops with domain experts for new processes.	Used for documenting the new business processes and for derivation of use cases.
	b)	Extracted and detailed from business processes.	Used for describing system functions.
	c)	Created while designing and coding the prototype.	Used for design decisions of the prototype.
MED		▪ Creation of informal scripts from project definition. ▪ Reviews and refinement of scripts. ▪ Creation of class model using CRC cards. ▪ Development of UI prototype. ▪ Development of use case descriptions describing (prototype) UI dialogues.	▪ Informal scripts used to derive CRC cards. ▪ CRC cards are used to derive class model. ▪ Informal scripts and CRC used to validate prototype (and indirectly the class model). ▪ Use case descriptions used to document system from user view by describing individual UI dialogues
NET2		Use cases created by developers after discussing parts of the prototype generated from the EER data model with the customers.	▪ Use cases mainly used to document deviations from generated functionality. ▪ Means of communication. ▪ Basis for the implementation of non standard functionality.

Project	4.4.1 Creation	4.4.2 Usage
PUB1	▪ Potential use cases identified by developers from personal domain knowledge. ▪ Interviews with the subject matter experts ▪ Information collection at the workplace. ▪ Discussion of most likely sample threads through a use case. ▪ Developers elaborate the information retrieved into a use case document.	Use case model lays the foundation for all later phases. Project guideline explicitly states that requirements not covered by the use cases modelled will not be considered in the later phases. The guideline document describes a method how to derive a "resource/ responsibility" model from the use case model Scenarios are mainly used for validating the use case descriptions.
INS2	Development based on existing business process model: ▪ Restructuring ~100 workflows of existing model to 27 potential use cases. ▪ Interviews with domain experts. ▪ Development of use case descriptions and 4-6 scenarios of most likely threads. ▪ One to two review cycles.	Use cases and scenarios used as starting point for developing functional and data models (i.e. in a McMenamin/Palmer inspired fashion the scenarios were transformed into physical DFD and ER models and then consolidated to essential models).
DEV	Developers are domain experts → elicitation phase is short and informal ▪ Development of requi. catalogues. ▪ Creation of scenarios by developer during design. ▪ Validation of scenarios in team reviews.	Scenarios used to communicate between developers, argue about design decisions and validate dialog sequences.
PUB2	▪ Interviews, on-site observation, present know-how/knowledge and analysis of existing system used to determine requirements. ▪ Creation of scenarios/use cases. ▪ Validation & refinement of scenarios during several review cycles. ▪ Stakeholders agree to a definitive final version (→ requ. spec.).	Scenarios used to create the OO model.
BANK1	▪ Developers formulate first version of use cases. ▪ Use cases discussed with the domain experts.	Use cases used to communicate requirements between domain experts and developers, served as a basis for the object model and for UI development.
BANK2	▪ SW developers create use cases ▪ Use cases are reviewed by domain experts	Use cases serve as contract between bank and SW company.
BANK3	No explicit methods or guidelines ▪ Observations, interviews (open questions) ▪ Scenarios written by developers, supported by a domain expert in the team ▪ Validation by author-critic-cycles	▪ Based on scenarios and first visions, the developers create an evolutionary prototype. Thereby gained knowledge leads to enhanced visions, a new prototype is developed and this process is carried out iteratively ▪ Validation of prototype with users (usability lab)

As indicated by Tab. 23, scenarios suffer from the same maintenance problems as other requirements documents: typically, only one version is kept and it is not updated with changes due to design or implementation considerations. Only, the very large project PUB1, has established a library of use cases for reuse (distinguishing public and preliminary use cases) as well as a formal policy for change. The most prevalent quality assurance technique are reviews. These are mainly informal. Again, only the big project PUB1 has established a special quality assurance group. The use of templates for scenario description is another way of establishing some consistency throughout the documentation.

Tab. 23: Maintenance, Change Control and Quality Control of Scenarios

Project	4.4.3 Maintenance and change control	4.4.4 Quality control
NET1	Different versions kept	Reviews
INS1	None.	Reviews by customers
SALES	No versions; managed as part of the overall project documents; tool support (StP)	Only as part of the overall documentation
MED	No systematic change management (not desired); Basic document versioning as provided by Word; Use of Word templates.	Reviews by domain experts in workshops
NET2	None.	▪ Reviews ▪ Use case descriptions have to follow a pre-defined template
PUB1	Reuse encouraged, special team responsible for maintaining a library of all published and preliminary use cases. Formal policy for publishing use cases and negotiating changes in use cases defined	Project guideline defines the formal structure of use case descriptions. Word templates ensure a uniform appearance of all documents produced. Use cases reviewed by the subject matter experts (interviewee) and by communication partners in the 16 states (Use cases consistent to different legal conditions or work procedure). Quality assurance group ensures formal adherence of use cases to guidelines (spot checks).
INS2	None.	Reviews by domain experts; No further quality mechanisms or policy.
DEV	Maintenance of scenarios is up to developer. Use cases are not kept up to date to reflect the last changes in design or implementation.	Informal reviews in the development team only.
PUB2	Document version control; use cases are not versioned. No tracing Scenarios are not maintained, changes to design and code are not incorporated in scenarios.	Reviews, inspections. Quality handbook defining formal procedures. Use of templates, check list.
BANK1	None.	Direct feedback by domain experts through reviews. Use of templates.
BANK2	None.	Reviews by domain experts, close feedback loops. Use of templates.
BANK3	No formal maintenance, no change control. Document versioning by last two digits in file names. Release management is being introduced.	Quality management system does not exist. Validation by author-critic-cycles, prototypes and tests in usability lab.

4.5 Tools used

We distinguished six different kinds of tools: word processor, graphics processor, CASE tool, hypertext tool, configuration and management tool and GUI builder. Tab. 24 – Tab. 26 show the corresponding data.

Tab. 24: Graphics and Word Processor Used

Project		4.5.1. Word processor				4.5.2. Graphics processor			
		editing	manage-ment	checking	trans-formation	editing	manage-ment	checking	trans-formation
NET1	a)	+ MS Word	–	–	–	+ MSPower-point	–	–	–
	b)	+ MS Word	–	–	–	+ MSPower point	–	–	–
INS1		+ MS Word	–	–	–	+ Visio	–	–	–
SALES	a)	+	–	–	–	+ Visio	–	–	–
	b)	+	–	–	–	–	–	–	–
	c)	+	–	–	–	–	–	–	–
MED	a)	+ MS Word	O MS Word	–	–	+ MS Word	–	–	–
	b)	+ MS Word	O MS Word	–	–	+ MS Word	–	–	–
	c)	+ MS Word	O MS Word	–	–	+ MS Word	–	–	–
NET2		+ MS Word	O MS Word	–	–	–	–	–	–
PUB1	a)	+ MS Word	+ MS Word	–	–	–	–	–	–
	b)	+ MS Word	+ MS Word	–	–	–	–	–	–
INS2	a)	+ MS Word	O MS Word	–	–	–	–	–	–
	b)	+ MS Word	O MS Word	–	–	–	–	–	–
DEV		+ MS Word	O	O	–	–	–	–	–
PUB2		+ MS Word	+ MS Word	O MS Word	–	O Visio	–	–	–
BANK1		+ AmiPro	–	–	–	O Visio	–	–	–
BANK2		+ MS Word	O MS Word	–	–	–	–	–	–
BANK3	a)	+ MS Word	+ MS Word	–	–	+	–	–	–
	b)	+ MS Word	+ MS Word	–	–	+ CorelDraw Rat. ROSE	–	–	–

Tab. 25: CASE Tools and Hypertext Tools Used

Project		4.5.3 CASE tool				4.5.4 Hypertext tool			
		editing	management	checking	transformation	editing	management	checking	transformation
NET1	a)	–	–	–	–	–	–	–	–
	b)	–	–	–	–	–	–	–	–
INS1		+ Rat. Rose[43]	– –	– –	– –	– –	– –	– –	– –
SALES	a)	–	–	–	–	–	–	–	–
	b)	+ StP	+ StP	+ StP	–	–	–	–	–
	c)	+ StP	+ StP	+ StP	–	–	–	–	–
MED	a)	–	–	–	–	–	–	–	–
	b)	–	–	–	–	–	–	–	–
	c)	–	–	–	–	–	–	–	–
NET2		+ EER modelling tool (propriet.)	–	–	–	–	–	–	–
PUB1	a)	+ Paradigm Plus[44]	+ Paradigm Plus	–	O Paradigm Plus	–	–	–	–
	b)	+ Paradigm Plus[45]	+ Paradigm Plus	–	O Paradigm Plus	–	–	–	–
INS2	a)	+	O	O	O	–	–	–	–
	b)	+	O	O	O	–	–	–	–
DEV		–	–	–	–	–	–	–	–
PUB2		–	–	–	–	–	–	–	–
BANK1		+ Visual Age C++	–	–	–	–	–	–	–
BANK2		–	–	–	–	–	–	–	–
BANK3	a)	–	–	–	–	–	–	–	–
	b)	–	–	–	–	–	–	–	–

[43] for editing object models

[44] Rational Rose currently evaluated

[45] Rational Rose currently evaluated (Fussnoten zusammenfassen)

Tab. 26: Configuration and Management Tool or GUI Builder Used

Project		4.5.5 C&V management tool				4.5.6 (G)UI builder			
		editing	management	checking	transformation	editing	management	checking	transformation
NET1	a)	O file system	–	–	–	–	–	–	–
	b)	O file system	–	–	–	–	–	–	–
INS1		O RCS	–	–	–	–	–	–	–
SALES	a)	–	–	–	–	–	–	–	–
	b)	–	–	–	–	–	–	–	–
	c)	–	–	–	–	–	–	–	–
MED	a)	–	–	–	–	–	–	–	–
	b)	–	–	–	–	–	–	–	–
	c)	–	–	–	–	O Visual Basic	–	–	–
NET2		–	+	–	–	+ UI prototype generator (propriet.)	–	O UI prototype generator (propriet.)	+ UI prototype generator (propriet.)
PUB1	a)	?	?	–	–	–	–	–	–
	b)	?	?	–	–	–	–	–	–
INS2	a)	–	–	–	–	–	–	–	–
	b)	–	–	–	–	–	–	–	–
DEV		–	–	–	–	–	–	–	–
PUB2		–	O	–	–	–	–	–	–
BANK1		–	–	–	–	–	–	–	–
BANK2		–	–	–	–	O Visual Builder	–	–	–
BANK3	a)	–	–	–	–	–	–	–	–
	b)	–	–	–	–	–	–	–	–

From the tables it is obvious, that little support for graphics is used, that CASE-tools mostly only support editing and that almost no configuration management of scenarios was done.

4.6 Summary and Findings

Facets of the scenarios process can be characterised as follows:

- **Scenarios are almost exclusively applied in the early phases of software development**: They are used to develop requirements, system specification, prototypes and object models. Sometimes they refer to the existing system, but future scenarios prevail.
- **Scenario usage is as diverse as the process of requirements capture in general**: Sometimes they are part of the very first stage of context understanding and description (including the existing system), often they constitute the system specification. There is no standard sequence of the development of prototypes, system specification and scenarios. Each can be input to or derivation of the development of the others. It is an important issue for further research whether the often exemplary nature of scenarios is adequate to their use as major constituent of the system specification
- **For scenario creation, a whole variety of inputs exists:** This depends on the artefacts developed before scenario creation. With one exception the scenarios were the main focus of negotiation with the users and domain experts.

- **Scenario maintenance and quality control shows no difference to requirements documents in general**: Mostly only one version is kept, which is not updated systematically. Reviews are informal.
- **No specific tools are used**.

4.7 Scenario Usage Highlights

In this subsection we describe for each project process aspects which are particularly interesting.

4.7.1 Project NET1

Project NET1 used visionary scenarios to understand the problem domain as well as use cases for definition of the system functionality. The latter caused no problem, since the use cases mostly dealt with one input and one output. Use cases were also related to the object model. A major difficulty encountered was that use cases do not allow to describe invariants.

4.7.2 Project INS1

Project INS1 mainly aimed at delivering a requirement specification consisting of an object model and dialogue definitions. Scenarios were used to understand and fix the business process to be supported by the system. The detailed interaction between system and the user was not studied. The project reports that the customers at first did not like to spend time on defining the business processes because they seemed too little related to the requirement specification. However, it forced the customers to agree on the business processes without reference to the software functionality. That way the often reported blend of political and business specific issues with technical issues was avoided. Since the system is now implemented in-house, no data is available whether the defined functionality really fulfils the customers' needs.

4.7.3 Project SALES

Project SALES is the only one using scenarios for describing communication between design objects. In particular, event traces as supported by the tool SoftwarethroughPictures are developed. These are helpful to document design options, but the tool does not support relationships to other artefacts. This project also used scenarios for context and interaction description. The main purpose was to understand and document the context and interaction. Due to the organisation of the company, interaction between users and developers is not emphasised. By using scenarios for three different purposes a whole lot of documentation was created, which is difficult to keep consistent. In particular, it is an open issue how to check consistency between different kinds of scenarios, between scenarios and prototypes or object models. This project used scenarios as the only means to define system functionality. The basic reason was that the functional model supported by the tool did not seem adequate. It is an interesting issue for further research, how scenarios relate to more traditional function models like data flow diagrams.

4.7.4 Project MED

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 artefacts changed in that they acted now as a "guideline" how to use the prototype for validation.

4.7.5 Project NET2

After having made bad experiences in attempting to talk directly with the customer about the abstract EER data model, scenarios in the form of prototype usage situations were introduced as a successful means for validating the data model. Nevertheless, it turned out that it is hard to elicit scenarios (sample system usage, test cases) from customers if they have 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. First the prototype provided a view on the data model for validation which was understandable by the customer, and secondly, 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.

4.7.6 Project PUB1

In PUB1 the terms use case and business process were used interchangeably (but considered at a fine-granular level). Use case modelling is seen as a problem understanding activity to make the developers familiar with the business processes to be supported by the future system. The process definition of PUB1 was well defined and due to its scale adapted for specific project purposes (e.g., distribution of customers, differences in current practices depending on location etc.). Special features of the process were its interview based generation technique, its reviewing and quality assurance procedures, and its systematic reuse of use cases.

4.7.7 Project INS2

A peculiarity of this project was the generation of use cases based on the reuse of workflows identified in a failed petrinet-based approach. Illustrated by the small number of resulting use cases (27), it turned out that the use cases were far more abstract than the earlier defined petrinet-based workflows (more than 100). For each use case five instance-level scenarios representing typical threads through the use case are described. After checking the scenarios against the use cases with domain experts Data Flow Diagrams using the classical techniques from McMenamin and Palmer were derived.

4.7.8 DEV

Most surprising and unique to this project is its use of scenarios as a means not to capture requirements, but to document design decisions and to serve as a help to better communicate these decisions among developers. Scenarios are used because of their suitability to structure complicated dialog sequences. It is the only CASE tool development project in this survey and as such developers are domain experts and may be future users of the tool themselves.

4.7.9 PUB2

As this is the only project in this survey developing a product for market, it is interesting that - even though the project does not have an explicit customer - nevertheless use cases were used to capture functionality. Scenarios were created from interviews with potential future procurers, on-site observations and know-how from an existing system and were validated with potential future users.

4.7.10 BANK1

In this project a model of current and future business processes, i.e. an analysis of the application domain, already existed. This model is represented in Event-driven Process Chains (EPK). These EPKs define the context of the development process and support the development team in understanding system requirements. Use cases then, have been acquired to model the user interactions with the (concrete) system.

4.7.11 BANK2

The system requirements are stated in a set of use cases, that were acquired in several workshops with domain experts. The use cases have been altered and refined and new use cases have been added until the system functionality has been sufficiently specified/documented. Then, the developed use cases became the core of the contract between the clients and the software company.

4.7.12 BANK3

Interesting in this project are the two kinds of scenarios used: First, scenarios were utilised to help developers understand the application domain, capture business processes and to describe the current state. Based on these scenarios enhanced visions were developed and in an iterative process by use of prototypes these visions were refined. Thus scenarios of the second kind (visions) were used to capture the desired state serving as the specification of the system.

5 Benefits, Problems, Needs, and Future Plans

In this section we describe the benefits, problems, needs and future plans related to scenario usage that the interviewees encountered during the course of their projects. All this information is based on explicit statements by the interviewees and therefore represent their subjective perception. In evaluating their answers we recognised and extracted several common factors mentioned in many interviews and arranged them in a table. The listed benefits, problems, and needs were supplemented (esp. the benefits-table) with information gained about the objectives for using scenarios (cf. section 3) in the visited projects.

Tab. 27: Benefits

Project	for communication and discussion with customer	reflect users point of view, user is involved	transfer, understand domain knowledge	understand and document business process	document understanding of future system	document/ structure requirements	further benefits
NET1	+		+	+	+	+	to write down vague ideas to describe changes (functionality)
INS1	+[46]			+			
SALES				+			helpful for maintenance and reengineering
MED		+	+		+	+[47]	complementary to prototypes
NET2	+	+					basis for validating data model less effort in creating specs and more comprehensible specs
PUB1			+	+			
INS2		+	+	+			basis for DFD & EER data model
DEV	-						document and communicate design decisions; structure, validate and harmonise complicated dialogue sequences; focus on essential behaviour; alternatives can be appended to normal flows
PUB2	+	+	o			o	
BANK1		+	+			+	
BANK2		+	+		+		
BANK3	+		+	+	+	+	scenarios allow rapidly and successfully designing prototypes

[46] basis for discussion

[47] requirements were early validated

5.1 Benefits

In this division, we elicited information about the benefits gained from using scenarios. Because several similar benefits were mentioned in various interviews we categorised them into a set of six classes. The following table displays the benefits from scenario use, representing each class as one column. Further benefits that are special to single projects are listed in the rightmost column.

8 of the 12 projects benefited from the ability of scenarios to transfer domain knowledge from domain experts to developers and to help the developers understand it. The involvement of the user/ the client achieved by scenario usage has been mentioned to be an advantage in 6 projects, as was the ability to document and help understand business processes. The other three benefits apply to 4 or 5 projects, i.e. the support of scenarios in communication and discussion between developers and customers, their ability to document and structure system requirements, and their role in documenting the developers’ understanding of the future system.

None of the listed advantages was mentioned for the project DEV. This project uses scenarios during the design phase and thus gains from scenarios through their ability to document and communicate design decisions and to represent internal and external behaviour.

Besides the described benefits several projects facilitate scenarios for individual purposes, e.g. to easily derive data models. Two projects mentioned the relationship between scenarios and prototypes, esp. their complementary character as an advantage --- a point that is also documented in [Weidenhaupt et al., 1998].

5.2 Problems

Tab. 28 describes the problems that arose during the development and use of scenarios. Each problem is connected with the wish (or the need) to overcome this problem by some means. Again, similar problems and needs were mentioned for several projects, so that a tabular representation seems appropriate to display which projects have which problems in common. Further problems and needs, unique for single projects are listed in the two rightmost columns.

The common problems listed in the table are:

- At what level granularity should scenarios be formulated?
- How should/could scenarios be structured?
- How can the completeness of the set of scenarios be determined?
- How can consistency between scenarios and other models be achieved/guaranteed?
- How can the (large) set of (scenario) documents be managed?
- How can traceability be ensured/ handled? (The notion of traceability in this table is rather broad, it embraces the traceability of models back to the acquired scenarios as well as the traceability of changes (and their rational) during the development process.)
- How to cope with overlapping of scenarios?

Tab. 28: Problems and Need

Project	granularity	how to structure scenarios	completeness of scenarios	consistency with other models	management of scenario documents	traceability	overlapping scenarios	further problems	further needs
NET1	+[48]	+	+	+				invariants cannot be formulated	
INS1	+							use case did not refer to software system requirements were not clear → many reviews	tool for editing structured text and images
SALES				+	+[49]			implementation was not performed in oo-fashion	
MED				+[50]	+[51]	+			
NET2		+	+		+			customer had to be convinced of scenario use mapping of a scenario to global architecture	systematic generation of test cases
PUB1					+[52]	+	+	(geographically) distributed teams	
INS2					+	+	+	rework because of usage oriented decomposition of functionality how to determine effects of changes	
DEV		o		-					use cases should be used in a broader field of RE
PUB2	-		-	o	o	+		GUI and user manual were not developed in parallel with scenarios, thus no synergies	support in parallel development of GUI
BANK1								no process model, only a huge set of documents	need of a process model and a tool for it
BANK2					+				
BANK3					+	o		developers partly ignored scenarios	glossary of terms

[48] guidelines needed for how to write scenarios

[49] flexible CASE tool needed

[50] esp. when using scenarios as test cases

[51] tool for scenario management and traceability needed

[52] huge number of scenarios

The main problem mentioned in 8 of 12 projects is the management of scenarios as documents. Developers need flexible tools that help manage the set of scenarios, esp. in large projects. Another problem, mentioned as relevant for 5 of 12 projects, is the traceability of scenarios.

Closely related to the management of scenario documents and the traceability is a problem mentioned four times: the consistency between the set of scenarios and other models. To support the process of checking two models for consistency (or guaranteeing consistency) both have to be represented explicitly, thus a prerequisite for consistency is the management of scenario documents.

The needs that follow from the mentioned problems ask for methods and tools to help manage scenarios and their relationships to other models of the development process. Those methods should include guidelines for writing scenarios and should enforce or at least support the representation of traceability information, e.g. by explicitly relating scenarios to GUI prototypes or test cases. Tools should allow the creation of textual and graphical documents. They should be able to manage these documents and relate them to each other or to other models to achieve traceability.

5.3 Future Plans

The final section of the survey’s taxonomy deals with the question if scenarios will be used in future projects again.

Tab. 29: Future Plans

Project	5.3 Will scenarios be used again?
NET1	+
INS1	+ (for the definition of business processes by the customer)
SALES	
MED	+ (depending on customer and project
NET2	+ (depending on customer and project)
PUB1	? (due to the early stage of the project, no future plans have been made)
INS2	? (due to the early stage of the project, no future plans have been made)
DEV	+ (for the design phase, maybe also in RE activities)
PUB2	+
BANK1	+
BANK2	O (probably, to elicit system requirements)
BANK3	+ (a glossary would be included)

Tab. 29 shows the explicit answers of our interviewees to this question. Note that three interview partners were not able to answer this question as their projects were in too early a stage. None of the projects under consideration is going to definitely retract from using scenarios in future projects. 8 of 12 project managers mentioned that they would continue using scenarios in further projects, thus in these projects scenarios were found to be satisfactory for the needs in software development. One interviewee mentioned that they were planning on extending scenario use in future projects (DEV). Other projects concretised the way scenarios will be facilitated in future projects, e.g. to define business processes by the customer (INS1) or to elicit system requirements (BANK2). Some decisions for scenario usage in future projects included improvements in relation to the current project, e.g. the use of a glossary (BANK3).

5.4 Summary and Findings

The benefits mainly mentioned refer to the ease understanding of scenarios for developers and clients, and thus support the process of transferring knowledge to the developers. The role of scenarios as a communication medium between the two parties is stressed in the list of benefits. It is noticeable that all mentioned benefits apply to scenarios independently of their representation medium or formalism. Even if scenarios were written down on paper or on a white board, these benefits still hold, e.g. the role of scenarios in transferring knowledge, their ability of documenting several kinds of information, or their degree of user involvement.

The discovered problems mainly call for methods and tools to support and structure the process of scenario development and usage, esp. the management of created documents. The interviewees request an environment that is capable of managing scenarios as if they were paper based and additionally offers a functionality that is only possible in electronic media, e.g. represent explicit relationships between scenarios and GUI prototypes.

Nevertheless, as the last table shows, even without flexible electronic support the utility “scenario” can be successfully applied to real software development projects.

6 Summary

To conclude this report about the use of scenarios in industrial software development projects we will summarise the findings of the previous chapters and present some observations about the survey’s results and our interpretation of these findings.

We visited twelve projects in Germany and Switzerland that used scenarios in their software engineering process in one way or another. Due to the small number and the contingent selection of projects our survey is not representative in a statistical sense (doing a representative study would have been beyond the resources and capabilities of our working group), but as the results presented in this report neither radically deviate from another recently published survey [Weidenhaupt et al., 1998] nor from our general experience in software development, we think that we have taken a valid and fairly typical sample of projects, giving a “snapshot” picture of scenario use in industry.

The visited projects differ in size, in their goals and in many other attributes, yet all of them use scenarios, but in many different ways. This diversity, the breadth in scenario usage, is probably the reason why we cannot derive many concise, general findings[53]. There is no *typical* scenario project. Neither is there the one true process to create, maintain and use scenarios. Thus, our findings are not spectacular. Nevertheless, we find them valuable: Some of them surprised us, others confirmed “folklore” knowledge as well as our previous experience with scenario use.

It is quite surprising to see the breadth, both in scope and purpose, of using scenarios, be it in requirements engineering, in design or in validation and verification activities. The following two sections present the commonalties we found in the different projects and some general observations, respectively.

6.1 Commonalties of the visited projects

Even though the projects are quite different[54], some interesting commonalties can be identified.

First of all, the survey confirms the following statements about scenarios:

There is no single purpose for using scenarios. Nevertheless three different main objectives for using scenarios can be derived from the data collected:

- Improving the communication between software engineers and customers,
- Understanding the application domain, a system vision and/or the system/software requirements, and
- Writing requirements in a new, convenient way and thus documenting them.

These advantages of scenario-like descriptions are well known from literature and in this survey they prove to be valid in practice. Although we could identify these three main objectives, it is not possible to classify projects according to this classification scheme, since in most projects scenarios fulfil several tasks and cover different objectives.

[53] This diversity (of scenario use and in applying the scenario approach) is in itself an interesting finding

[54] Only few facets hold for all projects in the survey and probably even less may be applied more generally to any project, i.e. hold for most other industrial projects and for their use of scenarios as well.

A second point common to most projects is the use of natural language as the dominant means for representing scenarios, i.e. scenarios are episodic prose texts. Only few and weak structuring mechanisms and hardly any abstractions are employed and formal approaches are not used at all. From our data, it is not clear whether semiformal or formal methods and appropriate abstraction techniques are not really needed in practice or whether the existing approaches are insufficient or too complicated for practical use. We think the latter is true which means that there is substantial need for research in this area.

A hint to why neither formal methods nor abstractions and structuring mechanisms are used is given by the following observation that is related to the last (dominance of natural language for representing scenarios) — tool support is lacking almost completely. Among the most striking findings of this survey is the fact that hardly any special tools for scenario creation, usage, and maintenance are used in the projects. Most projects use standard word processors to create, update, and manage scenario documents, possibly supplemented by graphical editors/drawing tools to create diagrams. Thus, a reason for not using more formal approaches might be the inappropriateness or lack of utilities to support them.

Most of the projects under consideration use object-oriented methods together with scenarios. This is not surprising as popular textbooks such as Jacobson et al. (1992) introduce object-oriented development and scenarios together. Nevertheless, it is remarkable, because scenarios themselves employ a functional approach and there is no straightforward way of interconnecting scenarios and object/class models.

Further, we observed (and were surprised by the fact) that scenario management is not – yet? – an issue in industry: In the projects of our survey, scenario management does not really happen and it is not even perceived everywhere as a need.

6.2 Further general observations

Scenario-related literature suggests scenarios as an elicitation medium that leads to system specifications or other more abstract models, i.e. they complement other requirements specifications. Our experience with the visited projects is quite different. Scenarios here are often used *instead of*, and not complementary to, a system specification. They replace more traditional specifications. Moreover, because scenarios (as used in the projects under consideration) model to a large extent only normal cases and no exceptional flows, only a partial specification of the system is being created.

The sample projects differ in many respects. We could not identify a typical process of creating, using, and maintaining scenarios. As stated above, in some projects scenarios replace traditional requirements specification and other projects use scenarios to validate already implemented system prototypes, etc.; thus the lack of a common process is not surprising. Again the broadness of applicability of scenarios (and their actual usage) hinders general conclusions.

Our data show that scenarios are used in projects of very different size. This gives evidence for the usability of scenarios regardless of project size: The scenario approach is scaleable. However, it should be noted that we have no data about the *degree* of usefulness depending on project size.

Scenarios are seen in relation to many artefacts within the projects, but many relationships only exist implicitly or are not maintained once established. The interviewees did realise the potential of interconnecting scenarios with other artefacts in the software development process, but they established neither means to support nor a process to maintain those relationships. This might have been the case because supporting tools to accomplish this task are lacking and/or because the overhead of maintaining the relationships and keeping all related documents up to date was too large to justify the additional work.

6.3 Conclusion

The survey revealed that scenarios are a flexible and broadly applicable approach. They are used for many purposes and in different phases of the projects, e.g. during the first phases of requirements elicitation and specification, as well as during the design phase and for validation and verification purposes.

The collected data proves scenarios to be applicable in projects of varying size and located in different domains, e.g. financial or communication.[55]

These implications are underpinned by the answers of the project managers to the question if scenarios would be used again in future projects. Nine out of twelve would use them again in new projects[56] and none reported a failure with scenario usage. Thus, one can say that scenarios are successfully employed in practice.

[55] Since the majority of the visited projects develop information systems, no statements can be made about the applicability of scenarios to other system types.

[56] For the other three interviewees it was too early to make a statement about future use of scenarios.

7 References

[Jacobson, 1995] I. Jacobson: The Use Case Construct in Object-Oriented Software Engineering. In: J. Carroll (ed.): *Scenario-Based Design: Envisioning Work and Technology in System Development*. John Wiley and Sons, New York, 1995. pp. 309-336

[Jacobson et al., 1995] I. Jacobson, M. Ericsson, and A. Jacobson. *The Object Advantage: Business Process Reengineering with Object Technology*. ACM Press, Addison-Wesley, 1995.

[Maiden et al., 1998] N. Maiden, S. Minocha, K. Manning, and M. Ryan: CREWS-SAVRE: Systematic Scenario Generation and Use. In: *Proceedings 3rd International Conference on Requirements Engineering*. Colorado Springs, CO, USA, 1998. pp. 148-155

[Potts et al., 1994] C. Potts, K. Takahashi, and A. Anton: Inquiry-Based Requirements Analysis. In: *IEEE Software 11(2)*, March 1994. pp. 148-155

[Rolland et al., 1998] C. Rolland, C. Ben Achour, C. Cauvet, J. Ralyté, A. Sutcliffe, N. Maiden, M. Jarke, P. Haumer, K. Pohl, E. Dubois, and P. Heymans: A Proposal for a Scenario Classification Framework. To appear in: *Requirements Engineering Journal*, 1998.

[Weidenhaupt et al., 1998] K. Weidenhaupt, K. Pohl, M. Jarke, and P. Haumer: Scenario Usage in System Development: Current Practice. In: *IEEE Software 15(2)*, 1998. pp. 34-45

Appendix

Classification Taxonomy

(Version 2.4)

Survey Data

Project:	<Real name or a letter code if confidential>	*TEXT*
Company:	<Real name or a letter code if confidential>	*TEXT*
Studied by:	<Names of AK members>	*TEXT*
Classification taxonomy:	<Version of class. tax. (2.4 in this case)>	*INTEGER*
Classification version:	<Version of the class. of the specific project>	*INTEGER*
Date(s) of interview(s):	*DATE*	
Date of classification:	*DATE*	

1 Project properties

<short description of organization and project> : TEXT

1.1 Domain / Application

1.1.1 Application domain:
{communication, financial, sales, medical, public service, software development, public emergency service, ...}

1.1.2 System type:
{management system, information system, CASE tool, ... }

1.1.3 Importance of
.1 Static properties (data): *VALUE*
.2 Dynamic properties (behavior, function): *VALUE*

1.2 Overall project size

1.2.1 Duration elapsed: *YEARS*

1.2.2 Duration total: *YEARS*

1.2.3 Effort total: *PERSON YEARS*

1.3 Project type

1.3.1 Experience in application domain of
.1 Customer: *VALUE*
.2 Supplier: *VALUE*

1.3.2 Type of customer:
{ in-house, extern }

1.3.3 Type of resulting system:
{ individual, market, prototype, ... }

1.3.4 Type of development:
{ existing platform, replacement, new... }

1.3.5 Companies/Profit Centers involved
.1 Customer: *INTEGER*
.2 Supplier: *INTEGER*
.3 Consultant: *INTEGER*

1.4 Stakeholder Experience

Stakeholder experience in domain: VALUE	*A End-user repr.*	*B Project manager*	*C Software engineer*	*D Consultant*	*E Domain expert*	...
1.4.1 Software development						
1.4.2 Scenario use						
...						

1.5 Scenario Team Structure

1.5.1 Average size of team for treating a single scenario: *INTEGER*

1.5.2 Team structure for treating a single scenario

.1	Number of user representatives involved:	*INTEGER*
.2	Number of project managers involved:	*INTEGER*
.3	Number of software engineers involved:	*INTEGER*
.4	Number of consultants involved:	*INTEGER*
.5	Number of domain experts involved:	*INTEGER*

< Remark: potentially same persons in different roles (e.g. user representative and domain expert)>

1.5.3 Total number of persons involved in scenario creation and use: *INTEGER*

2. Scenario Contents and Presentation

2.1 Kinds of scenarios

<short description of the kinds of scenarios used in the project>: TEXT

2.2 Content

2.2.1 Main modeling focus

.1 Current situation (as is): *VALUE*
.2 Requirements of the software system (problem): *VALUE*
.3 Design of the software system (solution): *VALUE*

2.2.2 Scope

.1 Internal : *VALUE*
.2 Interaction: *VALUE*
.3 Context: *VALUE*

2.2.3 Abstraction

.1 Instance scenarios: *VALUE*
.2 Type scenarios: *VALUE*

2.2.4 Viewpoints

.1 Static: *VALUE*
.2 Dynamic: *VALUE*
.3 NFR: *VALUE*

2.2.5 Granularity

.1 Business process (describes relationships among tasks): *VALUE*
.2 Task (describes relationships among working steps): *VALUE*
.3 Working step (consists of elementary interactions): *VALUE*

2.2.6 Cases being modeled

.1 Normal cases: *VALUE*
.2 Exceptions: *VALUE*

2.3 Representation

2.3.1 Ontology (modeling primitives and constructs)

.1 Primitives: *TEXT*
.2 Structuring/abstraction constructs:
  .1 Sequence: *VALUE*
  .2 Iteration: *VALUE*
  .3 Alternative: *VALUE*
  .4 Composition: *VALUE*
  .5 Overview diagram: *VALUE*
  .6 Type abstraction: *VALUE*
  .7 Hierarchical decomposition: *VALUE*
.3 Formality: *{ informal, semi-formal, formal }*

2.3.2 Notation

.1 Free text: *VALUE*
.2 Structured text/ templates: *VALUE*
.3 Restricted text (e.g. programming languages): *VALUE*
.4 Table: *VALUE*
.5 Diagram: *VALUE*
.6 Image (screenshot, picture, video, ..): *VALUE*

2.3.3 Presentation

.1 Static: *VALUE*
.2 Interactive: *VALUE*
.3 Animated/simulated: *VALUE*

2.4 Scenario figures

2.4.1 Total number of scenarios: *INTEGER*

2.4.2 Typical size of a single scenarios: *PAGES A4*

3. Goals of Scenario Use

3.1 Purpose

Subjects/activity: *VALUE*	Understanding	Elicitation	Validation	Documentation	Mediation/ Negotiation
3.1.1 Appl. domain					
.A Business process					
.B Terminology					
.C Dom. Knowledge					
3.1.2 System vision					
3.1.3 Software artifacts					
.A Requirements					
.B Architecture					
.C Code					
.D Test cases					
.E Interface					
3.1.4 Legal issues					
.A Contract					

3.2 Main objective

<one short paragraph on ”why” using scenarios in the project>: TEXT

4. Process

4.1 Process Context

*<short description of **scenario** process >: TEXT*

4.2 Relationships of scenarios to artifacts

<list of artifacts can be extended by those mentioned by the interviewees>

Artifacts	A Documented relationship: *VALUE*	B Maintained relationship: *VALUE*
4.2.1 Requirements		
4.2.2 Architecture		
4.2.3 Code		
4.2.4 Test cases		
4.2.5 Interface		
4.2.6 Glossary/object model		

4.3 Used RE method

4.2.1 Standard method: *BOOLEAN*

4.2.2 Which standard method OR description of in-house method: *TEXT*

4.2.3 Type of method: *{ oo, functional, data, behavior, ... }*

4.4 Scenario life cycle

4.3.1 Creation: *TEXT*
<activities, methods & guidelines used, roles of people involved>

4.3.2 Usage: *TEXT*
<activities, methods & guidelines used, roles of people involved>

4.3.3 Maintenance and change control*:* *TEXT*
<activities, methods & guidelines used, roles of people involved; in particular change control procedures, change propagation from and to scenarios>

4.3.4 Quality control: *TEXT*
<activities, methods & guidelines used, roles of people involved; in particular walkthroughs, inspections etc.>

4.4 Tools

<For each kind of tool the VALUE and possibly the name of the tool is given.>

Tool / purpose	Editing	Management	Checking	Transformation	...
Word processor	*VALUE*				
	TEXT				
Graphics processor					
CASE tool					
Hypertext tool					
C&V Mgt tool					
(G)UI Builder					

5. Experiences and Expectations

5.1 Benefits: ***TEXT***

5.2 Problems and Needs: ***TEXT***

5.4 Future Plans: ***TEXT***