universität wien

DIPLOMARBEIT

Titel der Diplomarbeit

“Implementing a Business Process into an ERP solution A Case Study with Use Case Requirements Model”

Verfasser

Thomas Sekulin

angestrebter akademischer Grad

Magister der Sozial- und Wirtschaftswissenschaften (Mag. rer. soc. oec.)

Wien, 2008

Studienkennzahl lt. Studienblatt: A 175
Studienrichtung lt. Studienblatt: Wirtschaftsinformatik

Betreuerin: ao. Univ. Prof. Dipl.-Ing. Dr. Renate Motschnig

ACKNOWLEDGMENTS

It is hard to write appropriate acknowledgments that reflect the assistance and inspiration I was given during the work on this thesis. Without this support, this project would never have been finished. Now is time to thank you all.

I start the expression of thanks with my supervisor Prof. Renate Motschnig, for guiding me through the work on my diploma thesis.

I appreciate my employer PPC Insulators for the opportunity to use this very interesting multinational enterprise as a source of evidence. I am especially grateful to all my colleagues at PPC Insulators for working with me on the implementation project, which was the object of my case study.

Let me thank my parents for supporting me during my studies and their tolerance in sharing this experience for so long.

Finally, and most importantly, my deepest thank to Sabine, for all her patience with me during the work on this thesis and for her energy to work through my text and polish my English.

TABLE OF CONTENTS

LIST OF FIGURES AND TABLES

LIST OF ABBREVIATIONS

BC	Business Case
CMM	Capability Maturity Model
CMMI	Capability Maturity Model Integration
ERP	Enterprise Resources Planning
ID	Identifier
IPD-CMM	Integrated Product Development Capability Maturity Model
IPPD	Integrated Product and Process Development
IT	Information Technology
NFR	Non-Functional Requirement
OMG	Object Management Group
OMT	Object-Modeling Technique
OOSE	Object-Oriented Software Engineering
POS	Point of Sales
PU	Production Unit
SECM	Systems Engineering Capability Model
SEI	Software Engineering Institute
SRS	Software Requirement Statement
SW-CMM	Software Capability Maturity Model
TBD	To Be Determined
UML	Unified Modeling Language
XML	Extensible Markup Language

Part I

REQUIREMENTS ENGINEERING

1. Introduction

Enterprise Resource Planning (ERP) solutions became the replacement for legacy systems for many companies starting in the 1990s. Multinational companies looked toward reengineering and cost cutting and often combined their ERP project with the breaking down of country barriers for manufacturing sites and centers of distribution.

Implementing an ERP solution is mostly done by using packaged software, but only in very few cases by building an own ERP solution. To go for the packed software removes some aspects from the implementation project, like choosing the systems technology and programming the fundamental features, which should reduce the complexity of the project. On the other hand, the complexity of the project will stay on a very high level, since many different organizational units and departments are involved. The difficulties are growing, if the implementation is extended over multination organizations to cover all the distributed sites.

This thesis investigates a project, which implemented a distributed business process for sales offer and order handling, where it was necessary to extend the packed ERP solution to cover the flow of data through different geographical distributed installations.

The thesis is organized in two main parts, where part one outlines some related work and part two explains the details of the case study and the experience with a use case requirements model.

Part I is divided into three Chapters, where Chapter 2 shows some principles of Requirements Engineering, Chapter 3 develops some guidelines for Improving Requirements Engineering based on Capability Maturity Model and Chapter 4 looks into the use of Requirements Modeling with UML and Use Cases.

Part II is divided into four Chapters, where Chapter 5 explains the project, which was the motivation for this thesis and is the objective of the case study. Chapter 6 shows

the research questions and the evaluation concepts, for the investigations during the case study. Chapter 7 shows the detail of the case study on the overall review of the existing requirements and in detail on an extension of the initial project. Finally, Chapter 8 will discuss the findings of the case study and the conclusions of the research questions and will present an outlook on further work on this project.

2. Requirements Engineering

The implementation of software products is a permanent struggle with misunderstandings and wrong translations between different languages of the project members. On one side are the developers, which know very well how to program the system, but they have less knowledge of the use of the system. They are thinking and talking in the terms of program codes and data models, which is their basic background for solving the task of implementation. On the other side are the users of the system, which have many ideas about the future use, but have no knowledge about the implementation or function in the system. The users of the system are thinking in real world tasks and artifacts but they have difficulties to transport there knowledge to the developers. Therefore the system analyst was introduced to translate between the different sets of know how about the existing environment and capture the requirements for the future system. During the process of requirement capture, the system analysts try to find the needs of the different users, so the software developers can start with the writing of the program code. Sometimes, if the capturing process is too difficult, the developers just start to write code, also without knowing the final goal, since this is the easy part of the work.

2.1. Reason for Requirements Engineering

The difficulties in the requirement capturing process are based on the different roles and interests of the involved parties. Since the developers are not using the software, they involved analysts to combine the knowledge of different stakeholders of the system. These stakeholders are the designated users and the owners of the system. Any system has usually many different possible users with many different roles. While each user may know in very detail what he or she does, they are not able to see the whole picture to find possibilities for efficiency improvements, or to describe their own requirements in a detailed and precise way.

Over the years, we have fooled ourselves into believing that users know what the requirements are and that all we need to do is to interview them. It is true that we want to build a system which supports the users, and that the interactions of the users

are an important input. However, it is even more important that the system will provide value to the business that uses it.[1]

Beside the conflict of objectives between the involved user roles, there is also an additional different view from the owner and sponsor of the system. In most cases, the owners will not work with the software, but they are defining why the product will be built, and which benefit is expected. During the traditional approach, the software analysts will interview all the different parties and combine all the whishes into documents with hundreds of pages, but the traceability between functions and requirements was lost in these documents.

The major challenge during requirements engineering is that the owners and users, who most frequently will not be a computer specialist, must be able to read and understand the results, but also the developers need to know which capabilities should be included in the future system and which should not be included. Therefore, it is very important to find a form of representation, which can be used for all parties that are involved in the process of capturing the requirements.

2.2. Goals of Requirements Engineering

Since every software development project is unique by the combination of project members and environment, it is important to resolve the problem with different views on the system and the use of the system. At least the structure of the planned outcome of the requirements engineering process should be the same for all the development projects.

2.2.1. Business Requirements

For most of the planned systems, except for very small and limited projects, it is very helpful to show the context of the system. The business requirements should articulate how the products stakeholders will benefit from this product. For this purpose, the vision including the scope of the project should be documented. A vision statement describes what the product could ultimately become and the included project scope will show the limitations with the business requirements. The scope description should set the border between major features included in the initial release and describe how the vision will be more fully realized through subsequent releases. The limitations identify

[1] Jacobson, Booch; Rumbaugh 1999, p.112

specific capabilities, which the product will not include. Documented business requirements will help to answer the first question to ask whenever someone proposes new functionality: "Is this in scope?"[2]

To improve the knowledge about the business context and the environment and to increase the change to build a useful system, it is recommended to build a Business Model or at least a Domain Model.

2.2.1.1. Domain Model

A domain model captures the most important types of objects and the links between them in the context of the system. Identifying and naming these objects, helps to develop a glossary of terms that will enable everyone who is working on the system to communicate with a common terminology. These domain objects come in three typical shapes: [3]

- Business objects that represent things that are manipulated in a business
- Real world objects and concepts that a system needs to keep track of
- Events that will or have transpired, such as deliveries

The domain model helps users, customers, developers and other stakeholders to illustrate the objects and how they are related one to another.

2.2.1.2. Business Model

Business modeling is a technique for understanding the business processes of an organization. The goal is to identify the processes and the relevant business entities to be supported by the software. This is an extension of the domain model, since it includes beside the business objects also the processes and the involved workers. Beside the specification of the involved tasks and objects, this model shows also the responsibilities and operations of the related workers. As analysts build such a business model, they learn a lot about the context of the system to build, and this knowledge is essential to identify the requirements of the users.[4]

[2] Wiegers 2000

[3] Jacobson, Booch; Rumbaugh 1999, p.119

[4] Jacobson, Booch; Rumbaugh 1999, p.122 ff

2.2.2. Functional and User Requirements

As starting point of collecting the requirements a features list can be build. This shows candidates for requirements and includes all upcoming ideas from users, customers, analysts or developers. Each proposed feature in the list should include a short name and description, but also the priority for implementation, the estimated costs, the risk of implementing and the actual status to track the progress of the work. Based on the attributes it is possible to order and decide about implementation or validation on the single feature.

With the features list it is possible to collect the user requirements, describing the tasks or business processes a user will be able to perform with the product, together with the functional requirements, describing the specific system behaviors that must be implemented. The traditional view of requirement specification is that of the functional requirements, which defines the functions and services of the developing product. A functional requirement specification shows the description of the fundamental functions of the software components. The functional requirements specify the functionality in term of process input and output like the controls and the behavior on exceptional situations.

2.2.3. Non-Functional Requirements

Finally, to complete the understanding of constrains that can be placed on the system, its environment or its development, it is important to collect the non-functional requirements. The non-functional requirements (NFR) are used to specify the properties and qualities of a system such as security, usability and performance. NFRs are not directly implemented in software, but they are properties of the design and architecture. During the specifications of the NFRs, it is important to find a way of expression, which is possible to test after the implementation. Otherwise, it will not be possible to take care on this NFR during implementation.

2.3. Difficulties during Requirement Engineering

Most of the difficulties during requirements elicitation are based on cognitive psychology. Each individual interprets the world by using existing memories. This is one of the main causes of misunderstanding in requirements engineering. Two different people may see and hear the same, but because they have different experiences, they will end up with different interpretation of the input. The same is

true for the different approaches for solving a problem. The way individuals think and process information involves mental models of problem and possible solutions. Everyone is influenced from the own knowledge about the problems environment and tends to reuse previous solutions for similar tasks. Mismatch between memory of previous application and the current requirements problem can lead to inappropriate reuse.[5]

Human behavior is also influenced by social relationships, which are complex and depending from many different circumstances such as culture and group identity. However, the most important are power and trust between the involved parties. Power reflects the authority and discipline aspects of an organization, while trust is related to commitment and experiences between people. During requirement engineering, it is important to avoid excessive use of power and dissolve the existence of mistrust.[6]

2.4. Process of Requirements Engineering

A requirement engineering process is a structured set of activities, which is followed to deliver, validate and maintain a systems requirements document. A good process description will provide good guidance to the people involved and will reduce the risk of missing activities or working in an unguarded way. The description should include which activities take place, the structure and schedule of these activities, the responsibilities for each activity, the inputs and outputs of the activity and the tools used during the process of requirement engineering.

If this roadmap for the process is missing, it is very likely, that the software development projects will miss budget and, or schedule. Furthermore, the number of change requests after delivery of the system will be very high and it will take very long time to implement changes based on new requirements.

[5] Sutcliffe 2002 p 42f

[6] Sutcliffe 2002 p 43

2.4.1. Framework of Requirements Engineering Process

A framework for describing requirements engineering processes can be constructed by considering three fundamental concerns of requirements engineering.[7]

- Elicitation: the concern of understanding a problem
- Specification: the concern of formally describing a problem
- Validation: the concern of attaining an agreement on the nature of the problem

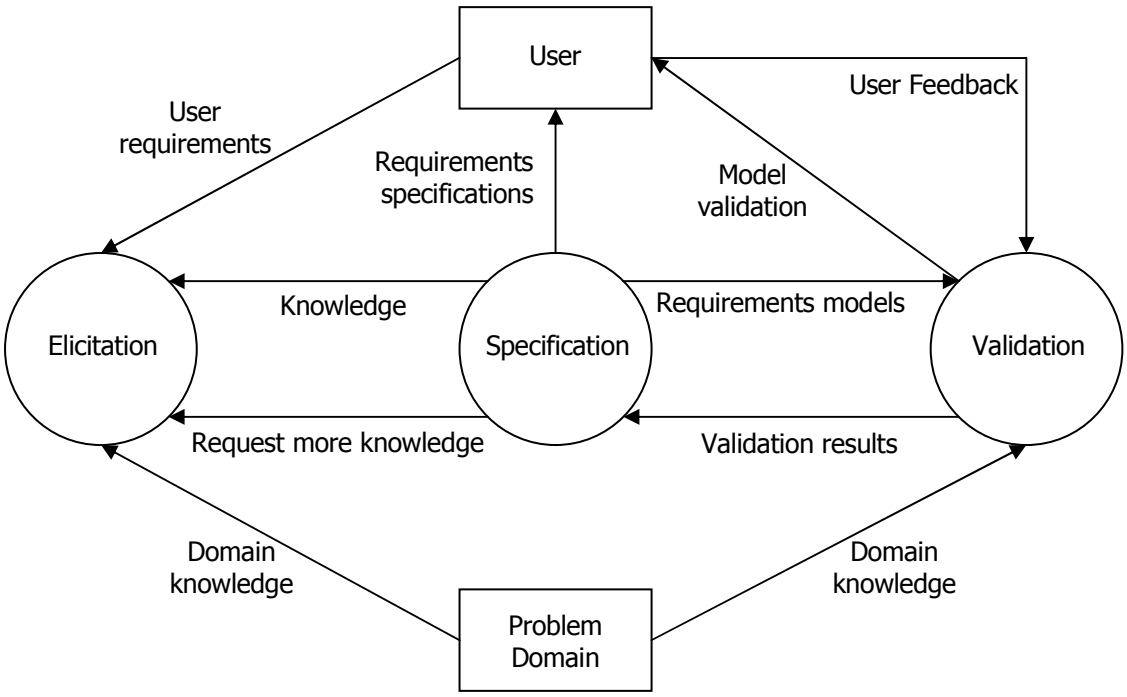

Figure 2.4.1-1 Framework of requirements engineering process[8]

Each of the above concerns implies that some activities must take place in order to provide answers, and in doing this, some resources must be used. For example, in order to understand the problem, relevant information about it must be available to the problem solver. If that information is not already available to the problem solver, then it must be obtained. In the same way, the relevant information must be validated in order to ensure its accuracy, consistency and relevance. The framework of the requirements engineering process is shown in Figure 2.4.1-1.[9]

[7] Loucopoulos, Karakostas 1995 p 20 ff

[8] Loucopoulos, Karakostas 1995 p 21

[9] Loucopoulos, Karakostas 1995 p 20

In requirements elicitation, the analyst is identifying the sources of problem knowledge, acquiring it and understanding the significance of the elicited knowledge and its impact. The most typically used techniques elicit the requirements from users through interviews and the creation of prototypes. Elicitation can be considered as an ongoing process, since it delivers the raw material to other processes. Therefore, elicitation occurs in parallel with specification and validation process.[10]

During the specification process, the collected knowledge must be analyzed to find cross relations and the heterogeneous parts of information must be combined to produce a logical and coherent whole. Since this is the central process of requirements engineering, it controls the elicitation and validation processes. Because of this central role, any interaction between the specification process and one of the other processes triggers always an additional task in the specification and the third process.[11]

Validation is a process, which requires interactions between analysts, customers of the intended system and users. In some occasions, the analyst can test the validity of the requirements model by common sense, but in others, it is necessary to initiate a test and analyze the results. This test can take place with a prototype, or with discussions involving the users of the system. The validation process starts for each additional requirement to check correctness, logical consistency and coherence.[12]

For most of the users, it is very difficult to describe their work in an abstract way. They are well aware about their daily tasks and know the process in the very last details but mostly they are not able to describe the task with general terms or find their role in the big picture of the overall business process. To assist the analyst to create from the concrete steps an abstract view of the requirements a possible technique is the work with user related scenarios.

2.4.2. Scenario-Based Requirements Engineering

This method involves a prototyping style of approach to requirements engineering, motivated by the need to get users actively engaged in argumentation about how a design will help to archive their goals. Scenario-Based requirements engineering does

[10] Loucopoulos, Karakostas 1995 p 21 ff

[11] Loucopoulos, Karakostas 1995 p 23 ff

[12] Loucopoulos, Karakostas 1995 p 25 ff

not exclude formal specification and modeling of requirements, but it sees such activity in parallel with prototyping.[13] The approach is based on the thesis, that technique integration provides the best alternative for improving requirements engineering and that active engagement of users in trying out designs is the best way to get effective feedback for requirements validation. Another motivation is to use scenarios as a means of situating discussion about the design, so that new requirements can be elicited by argumentation about problems confronted with scenarios describing a context of use. Three techniques are used[14]

- Storyboards: concept demonstrators and prototypes: to provide a designed, interactive artifact to which users can react to.
- Scenarios: the design artifact is situated in a context of use; thereby helping users relate the design to their work and task context.
- Design rationale: The designers reasoning are exposed to the user to encourage user participation in the decision process.

The techniques are combined with the Scenario-Based Requirements Engineering method to guide the requirements engineer. The method is composed of advice on setting up sessions, use of the above techniques, and more detailed guidance on fact acquisition and requirements validation strategies. Figure 2.4.2-1 shows the overview of the Scenario-Based Requirements Engineering, which consists of the following four phases[15]

- Initial requirements capture and domain familiarization: This is conducted by conventional interviewing and fact-finding techniques to gain sufficient information to develop a first concept demonstrator.
- Storyboarding and design visioning: This phase creates early visions of the required system that are explained to users in storyboard walkthroughs to get feedback on feasibility.
- Requirements exploration: This uses concept demonstrators and early prototypes to present more detailed designs to users in scripted, semi-interactive demonstrations so the design can be critiqued and requirements validated.

[13] Sutcliffe 2002 p 127

[14] Sutcliffe 2002 p 127

[15] Sutcliffe 2002 p 127f

- Prototyping and requirements validation: This phase develops more fully functional prototypes and continues defining requirements until a prototype is agreed to be acceptable by all the users.

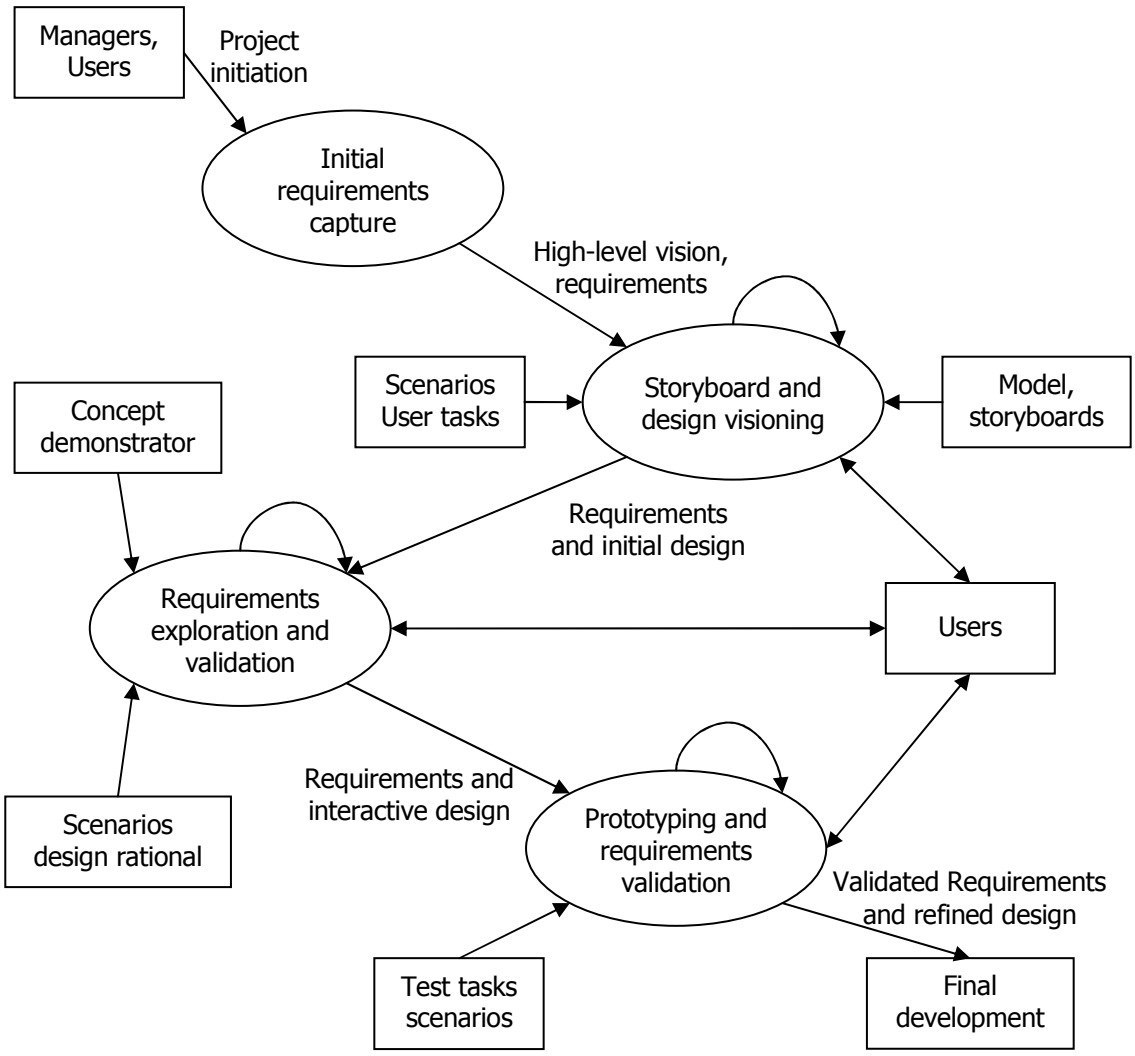

Figure 2.4.2-1 Overview of the Scenario-Based Requirements Engineering[16]

The phase requirements capture gathers facts about the domain and captures users' high-level goals for the new system. Scenarios are elicited as examples of everyday use of the current system, with stories of problem encountered and how they are dealt with. Scenarios complement goal modeling since goals focus on abstractions that describe users' intentions, whereas scenarios make abstract intentions clearer by giving

[16] Sutcliffe 2002 p 128

examples of how a new system might work to fulfill users' goals. To assist making the abstract, concrete two types of scenario can be used.[17]

- System visions express policies and high-level aims are frequently related to mission statements. While these are not scenarios in the sense of specific examples, they do provide a framework within users' intentions can be analyzed.
- Impact scenarios are visions of the future systems usage, once the goal has been implemented in a design. These scenarios cannot be created until analysis has decomposed the system to a level where some detail of sub-goals is apparent. Impact scenarios describe the effects that the designed system will produce and hence test whether the goal is suitable.

The output from the phase requirements capture is a set of user goals, organized in a goal hierarchy if necessary, domain facts and a set of scenarios of use including normal and exceptional episodes. These outputs are used to create early visions of the system in the next phases.[18]

The phase storyboarding and design visioning creates early visions of the system. Storyboards are sketches that show key steps in user system interaction. A combination of storyboards is advisable, where high-level overviews of the system functionality can be presented in an analyst-led walkthrough while areas of system detail are explored in user-led walkthroughs to test how well the early design ideas support the tasks. During the walkthroughs, it is important to record problems reported by the users as well as suggestions for improvements to reuse them for redesign. One of the advantages of the use of storyboards and scenarios is that they help involve users in designing the system.[19]

The phase requirements exploration starts when the design vision has stabilized and sufficient analysis has been undertaken to illustrate how the system will work with a limited set of scenarios. The additional task in this phase is to specify the system functions. At this stage, some modeling and specification will have been undertaken so

[17] Sutcliffe 2002 p 129 f

[18] Sutcliffe 2002 p 130

[19] Sutcliffe 2002 p 130 f

that system functionality can be simulated, although details of algorithms and data structures can be left until later in the process.[20]

The objective of the phase prototyping and requirements validation is to obtain more detailed user requirements and validation feedback on the design. User reaction is stimulated most effectively by hands-on testing. At this stage, requirements validation converges with usability testing to gain feedback on the user interface as well as on functionality.[21]

2.5. General Requirements Characteristics

There are several characteristics available, that each individual software requirement statements should exhibit to define the quality of requirements. It is not sufficient to have excellent individual requirement statements. Additional characteristics are used to describe the total set of software requirements statements as a whole.[22] It will never happen that all requirements demonstrate all these ideal attributes. However, to keep these characteristics in mind while writing and reviewing the requirements, will produce better requirements documents and better products.

2.5.1. Requirement Statement Characteristics

Requirement Statement Characteristics describe the qualities of every individual requirement as following:[23]

- Complete: Each requirement describes fully the functionality to be delivered. It contains all the information necessary for the developer to design and implement that functionality. If there is a lack of certain information, the requirement should show a TBD (to be determined) as a standard flag to highlight these gaps. Before the design and implementation is proceed, all TBDs have to be resolved.
- Correct: Each requirement describes accurate the functionality to be built. The reference for correctness is the source of the requirement, such as stakeholder or a high-level system requirement. A software requirement that conflicts with its system requirement is not correct.

[20] Sutcliffe 2002 p 132

[21] Sutcliffe 2002 p 141

[22] IEEE 1984 p 11 ff

[23] Wiegers 2003 p 22 f

- Feasible: It must be possible to implement each requirement within the known capabilities and limitations of the system and its operating environment.
- Necessary: Each requirement should document a feature, which one of the stakeholders really need or is required for conformance to an external system requirement or a standard. Every requirement should originate from a source that has the authority to specify requirements.
- Prioritized: For each requirement, feature, or use case an implementation priority should be assigned to indicate how essential it is to a particular release.
- Unambiguous: All readers of a requirement statement should arrive at a single, consistent interpretation of it. Since natural language is highly prone to ambiguity, each requirement should be written in simple, concise, straightforward language appropriate to the user domain. Readers must be able to understand what each requirement is saying. All specialized terms and terms that might confuse readers need a definition in a glossary. In addition, the terms that are used for the same object needs to be consistent.
- Verifiable: Each requirement should be examined, whether tests or other verification approaches, such as inspection or demonstration, are able to determine whether the product properly implements the requirement. If a requirement is not verifiable, determining whether it was correctly implemented becomes a matter of opinion, not objective analysis.

2.5.2. Requirements Specification Characteristics

Requirements Specification Characteristics relate to the set of requirements that are collected into a specification. The complete documentation should fulfill the following characteristics[24]:

- Complete: The set of requirement specification is complete, if it includes all the significant requirements, whether relating to functionality, performance, design constraints, attributes or interfaces. No requirements or necessary information should be absent. Missing requirements are hard to spot because they are not there. Focusing on user tasks, rather than on system functions, can help you to prevent incompleteness.

[24] Wiegers 2003 p 24 f

- Consistent: Consistent requirements do not conflict with other requirements or with high-level business, system, or user requirements. Conflicts between requirements must be resolved before development can proceed.
- Modifiable: It must be possible to revise the requirements specifications when necessary and to maintain a history of changes made to each requirement. This dictates that each requirement be uniquely labeled and expressed separately from other requirements so that references are unambiguously. Each requirement should appear only once. The requirement specification should have a coherent and easy to use organization with a table of contents, an index and explicit cross-referencing.
- Traceable: A traceable requirement can be linked backward to its origin and forward to the design elements and source code that implement it and to the test cases that verify the implementation as correct. Traceable requirements are uniquely labeled with persistent identifiers. A single statement must not contain multiple requirements since the different requirements might trace to different design and code elements. Trace each requirement back to specific input, such as a use case, a business rule, or some other origin.

3. Improving Requirements Engineering

To improve the results of requirements engineering, the initial task is the work on process improvement as a series of actions to identify, analyze, and improve existing processes within the organization to meet new goals and objectives. These actions have to follow a specific methodology or strategy to create successful results.

3.1. Capability Maturity Model (CMM)

The CMM was originally developed as a method of assessing the capabilities of companies bidding for defense contracts. Though it comes from the area of software development, it can be applied as a generally applicable model to assist in understanding the process capability maturity of organizations in diverse areas. The active development of this model by the Software Engineering Institute (SEI) began in 1986. The SEI is a federally funded research and development center as part of the Carnegie Mellon University in Pittsburgh, Pennsylvania.

The basic idea underlying the CMM approach is that organizations should assess their maturity and then introduce process changes, which will enable them to progress up the maturity ladder. [25] These models use two dimensions to address discipline practices and the level of process practices.

- Process Areas or Focus Areas

The first dimension is a collection of best practices in the discipline being addressed. In this regard, they overlap typical standards that also provide this type of information. The models are structured around groupings of practices such as requirements, design, verification, configuration management, planning, or risk management. These groupings may be called Key Process Areas, Process Areas, or Focus Areas. [26]

- Levels of Capability/Maturity

The second dimension provides levels of practice from chaos to quantitative improvement and tailoring that are associated with defining and implementing the processes effectively. Most importantly, they provide a sequence for addressing these practices that has proven effective in progressing from random practice to effective

[25] Sommerville, Sawyer 1997, p. 19

[26] INCOSE 2003, 2.4.1.1.

application of the discipline.[27] Maturity levels are used to characterize organizational improvement relative to a set of process areas, and capability levels characterize organizational improvement relative to an individual process area.[28]

To avoid misunderstandings it is important to be aware, that the models do not describe how the processes should be performed or the quality of the processes, only that some processes are defined, used, and cover the minimum practices in the model. Bad process well implemented is a definite possibility. However, specifics of process improvement create an environment that, by its nature, drive organizations to improve how they do business.[29]

Since 1991, CMMs have been developed for many disciplines. Some of the most notable include models for systems engineering, software engineering, software acquisition, workforce management and development, and integrated product and process development (IPPD). Although these models have proven useful to many organizations in different industries, the use of multiple models has been problematic.[30]

The CMM Integration project was formed to combine three source models, the Capability Maturity Model for Software (SW-CMM), the Systems Engineering Capability Model (SECM) and the Integrated Product Development Capability Maturity Model (IPD-CMM). The combination of these models into a single improvement framework was intended for usage by organizations in their pursuit of enterprise-wide process improvement.[31]

3.2. Capability Maturity Model Integration

CMMI® (Capability Maturity Model® Integration) is a process improvement maturity model for the development of products and services.[32]

Levels are used in CMMI to describe an evolutionary path, which is recommended for an organization that wants to improve the processes that are used to develop and

[27] INCOSE 2003, 2.4.1.1.

[28] CMMI 2006, p. 36

[29] INCOSE 2003, 2.4.1.2

[30] CMMI 2006, p. 6

[31] CMMI 2006, p. 6

[32] CMMI 2006, p. 29

maintain its products and services. CMMI supports two improvement paths. One path enables organizations to improve processes corresponding to selected process areas. For the continuous representation, the term "capability level" is used. The other path enables organizations to improve a set of related processes by addressing successive sets of process areas. For the staged representation, the term "maturity level" is used.[33] Figure 2.5.2-1 shows the improvement path trough the different CMMI maturity levels.

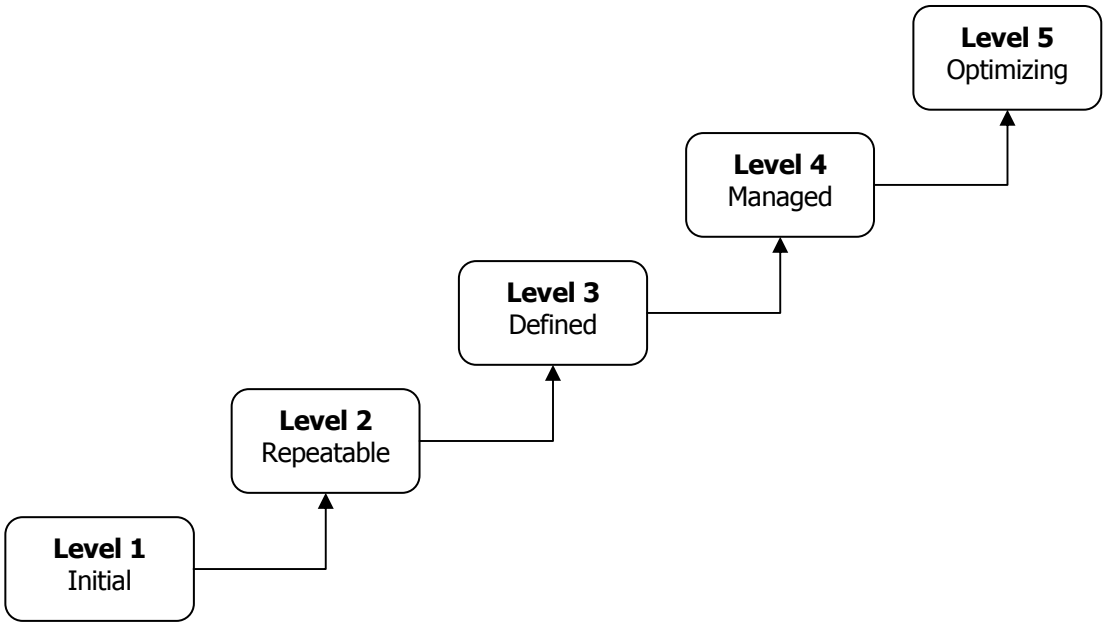

Figure 2.5.2-1 CMMI maturity levels[34]

To reach a particular level, an organization must satisfy all of the appropriate goals of the process area or set of process areas that are targeted for improvement. A maturity level consists of related specific and generic practices for a predefined set of process areas that improve the organization's performance. The maturity level of an organization provides a way to predict an organization's performance in a given discipline or set of disciplines. Experience has shown that organizations do their best when they focus their process improvement efforts on a manageable number of process areas at a time and that those areas require increasing sophistication as the organization improves. A maturity level is a defined evolutionary plateau for organizational process improvement. Each maturity level matures an important subset

[33] CMMI 2006, p. 29

[34] Sommerville, Sawyer 1997, p. 20

of the organization's processes, preparing it to move to the next maturity level. The maturity levels are measured by the achievement of the specific and generic goals associated with each predefined set of process areas.[35]

There are five maturity levels, each a layer in the foundation for ongoing process improvement:

- Maturity Level 1: Initial

At maturity level 1, processes are usually ad hoc and chaotic. The organization usually does not provide a stable environment to support the processes. Success in these organizations depends on the competence and heroics of the people in the organization and not on the use of proven processes. Nevertheless, maturity level 1 organizations often produce products and services that work; however, they frequently exceed their budgets and do not meet their schedules. Maturity level 1 organizations are characterized by a tendency to over commit, breakup of processes in a time of crisis, and an inability to repeat their successes.[36]

- Maturity Level 2: Managed

At maturity level 2, the organization has ensured that processes are planned and executed in accordance with policy; that projects employ skilled people who have adequate resources to produce controlled outputs, involve relevant stakeholders, are monitored, controlled, and reviewed; and are evaluated for compliance with their process descriptions. The process discipline reflected by maturity level 2 helps to ensure that existing practices are retained during times of stress. When these practices are in place, projects are performed and managed according to their documented plans. At maturity level 2, the status of the work products and the delivery of services are visible to management at defined points (e.g., at major milestones and at the completion of major tasks). Commitments are established among relevant stakeholders and are revised as needed. Work products are appropriately controlled. The work products and services satisfy their specified process descriptions, standards, and procedures.[37]

[35] CMMI 2006, p. 35

[36] CMMI 2006, p. 36

[37] CMMI 2006, p. 36 f

- Maturity Level 3: Defined

At maturity level 3, processes are well characterized and understood, and are described in standards, procedures, tools, and methods. The organizations set of standard processes, which is the basis for maturity level 3, is established and improved over time. At maturity level 3, processes are managed more proactively using an understanding of the relationships between the process activities and detailed measures of the process, its work products, and its services. At maturity level 3, processes are typically only qualitatively predictable.[38]

- Maturity Level 4: Quantitatively Managed

At maturity level 4, the organization and projects establish quantitative objectives for quality and process performance and use them as criteria in managing processes. Quality and process performance is understood in statistical terms and is managed throughout the life of the processes. At maturity level 4, the performance of processes is controlled using statistical and other quantitative techniques, and is quantitatively predictable.[39]

- Maturity Level 5: Optimizing

At maturity level 5, an organization continually improves process performance through innovative process and technological improvements. Process improvement objectives for the organization are established, continually revised to reflect changing business objectives, and used as criteria in managing process improvement. At maturity level 5, the organization is concerned with addressing common causes of process variation and changing the process to improve process performance and to achieve the established quantitative process improvement objectives.[40]

Organizations can achieve progressive improvements in their organizational maturity by achieving control first at the project level and continuing to the most advanced level, organization-wide continuous process improvement, using both quantitative and qualitative data to make decisions.[41]

[38] CMMI 2006, p. 37

[39] CMMI 2006, p. 37 f

[40] CMMI 2006, p. 38

[41] CMMI 2006, p. 39

Process Area	Category	Maturity Level
Requirements Management	Engineering	2
Project Monitoring and Control	Project Management	2
Project Planning	Project Management	2
Supplier Agreement Management	Project Management	2
Configuration Management	Support	2
Measurement and Analysis	Support	2
Process and Product Quality Assurance	Support	2
Product Integration	Engineering	3
Requirements Development	Engineering	3
Technical Solution	Engineering	3
Validation	Engineering	3
Verification	Engineering	3
Organizational Process Definition	Process Management	3
Organizational Process Focus	Process Management	3
Organizational Training	Process Management	3
Integrated Project Management	Project Management	3
Risk Management	Project Management	3
Decision Analysis and Resolution	Support	3
Organizational Process Performance	Process Management	4
Quantitative Project Management	Project Management	4
Organizational Innovation and Deployment	Process Management	5
Causal Analysis and Resolution	Support	5

Table 3.2-1 Process Areas and Associated Categories and Maturity Levels[42]

The process areas are organized by maturity levels to reinforce the concept to focus on process areas in the context of the maturity level to which they belong. The staged representation provides a predetermined path of improvement from maturity level 1 to maturity level 5 that involves achieving the goals of the process areas at each maturity level. Process areas are grouped by maturity level, indicating which process areas have to be implemented to achieve each maturity level. The generic goals that apply to each

[42] CMMI 2006, p. 44

process area are also predetermined.[43] Table 3.2-1 shows the 22 defined process areas and the associated categories and maturity levels.

For each of these process areas, the generic practices describe how to institutionalize a managed process to reach maturity level 2 and improve the process to a defined status, going further to a quantitatively managed process to reach finally an optimizing process for maturity level 5. Therefore, the further investigations are focused on the specific practices, which are already needed to fulfill maturity level 2. For the first improvement to maturity level 2, only one process area is connected to the engineering category, this is the process area Requirements Management.

3.2.1. Requirements Management

The purpose of Requirements Management is to manage the requirements of the project's products and product components and to identify inconsistencies between those requirements and the project's plans and work products.[44]

The specific goal is that requirements are managed and inconsistencies with project plans and work products are identified. The project maintains a current and approved set of requirements over the life of the project by managing all changes to the requirements, maintaining the relationships among the requirements, the project plans, and the work products, identifying inconsistencies among the requirements, the project plans, and the work products and taking corrective action. To achieve this, the following specific practices are defined.[45]

- Develop an understanding with the requirements providers on the meaning of the requirements. [46]

As the project is developing, the number of collected requirements is growing. To avoid requirements creep in, it is important to define and document the sources, from which the requirements are received. Furthermore, it is important to analyze the requirements with the provider to ensure that a compatible, shared understanding is

[43] CMMI 2006, p. 43

[44] CMMI 2006, p. 408

[45] CMMI 2006, p. 410

[46] CMMI 2006, p. 410 f

reached on the meaning of the requirements. The result of this analysis and dialog is an agreed-to set of requirements.

Lack of evaluation and acceptance criteria often results in inadequate verification, costly rework, or users' rejection.

- Obtain commitment to the requirements from the project participants. [47]

Whereas the previous specific practice dealt with reaching an understanding with the requirements providers, this specific practice deals with agreements and commitments among those who have to carry out the activities necessary to implement the requirements. As the requirements evolve, this specific practice ensures that project participants commit to the current, approved requirements and the resulting changes in project plans, activities, and work products. The impact on the project participants should be evaluated when the requirements change or at the start of a new requirement. Changes to existing commitments should be negotiated before project participants commit to the requirement or requirement change.

- Manage changes to the requirements as they evolve during the project. [48]

During the project, requirements change for a variety of reasons. As needs change and as work proceeds, additional requirements are derived and changes may have to be made to the existing requirements. It is essential to manage these additions and changes efficiently and effectively. To analyze the impact of the changes, it is necessary that the source of each requirement is known and the rationale for any change is documented. The project manager wants to track appropriate measures of requirements volatility to judge whether new or revised controls are necessary. Maintaining the change history helps to track requirements volatility.

- Maintain bidirectional traceability of requirements. [49]

The intent of this specific practice is to maintain the bidirectional traceability of requirements for each level. When the requirements are managed well, traceability can be established from the source requirement to its lower level requirements and from the lower level, requirements back to their source. Such bidirectional traceability helps

[47] CMMI 2006, p. 411 f

[48] CMMI 2006, p. 412 f

[49] CMMI 2006, p. 413

to determine that all source requirements have been completely addressed and that all lower level requirements can be traced to a valid source.

Requirements traceability can also cover the relationships to other entities such as intermediate and final work products, changes in design documentation, and test plans. The traceability can cover horizontal relationships, such as across interfaces, as well as vertical relationships. Traceability is particularly needed in conducting the impact assessment of requirements changes on the project's activities and work products.

- Identify inconsistencies between project work and requirements. [50]

This specific practice finds the inconsistencies between the requirements and the project plans and work products and initiates the corrective action to fix them.

All these specific practices together have the common goal to trace requirements, starting from who delivers the requirements, what happens with the requirement during the project and how they are connected to final implementation. Therefore, let us take a closer look into requirements tracing.

3.3. Requirements Tracing

Requirements traceability is an important feature of requirements processes and software systems. It is used to keep track of the relationship between individual requirements, of the change history and of the relationships between software process artifacts. A Software Requirements Specifications is traceable if the origin of each of its requirements is clear and if it facilitates the referencing of each requirement in future development or enhancement documentation.[51]

3.3.1. Purpose of Traceability

The purpose or rather the need for traceability depends on the stakeholder and the task that should be supported by the traceability. Therefore, different views of traceability exist.[52]

[50] CMMI 2006, p. 414

[51] IEEE 1984 p 13

[52] Lindvall 1994 p 11 ff

- Customer, User: Traceability ensures customer satisfaction by ensuring the customer that all stated requirements are met and that the task is completed. In addition, the effect of a requirement change can be demonstrated.
- Project planner: Project planners apply the tracing approach to perform impact analysis. Requirements can be tracked to determine the impact of a required change.
- Project manager: Project managers use the traceability information to control project progress.
- Requirement engineer: Requirement engineers use the traceability information to check correctness and consistency of the requirements.
- Designer: Designers use the traceability information to understand dependencies between the requirements and to check whether all requirements are considered by the design.
- Testers: Testers require traceability relationships between requirements and test plans to prove that the system meets the needs of the customer. In addition, test procedures that should be rerun to validate an implemented change can be identified. This saves test resources and allows the schedule to be streamlined.
- Maintainer: Maintainers use the traceability information to decide how a required and accepted change will affect the system and to clarify in detail which module is directly affected and which other modules will experience residual effects. The proper documentation of the design rational helps to understand the system to avoid degrading the system during the implementation of the change request.

3.3.2. Traceability Process

Similar to the requirement engineering process, the traceability process is a structured set of activities, to enable the traceability of requirements.

3.3.2.1. Define Requirements

The first step in implementing traceability is the identification of traceability information that the organization intends to capture and use.[53] This means the identification and description of the requirements, which should be realized in the software product. This is generally done during the analysis phase of the software engineering process and documented in various documents as requirement specifications.

[53] Ramesh 1995

3.3.2.2. Capture Traces

In the second step, it is necessary to define when the identified requirements should be captured and by whom. This step of trace production can be done in two different ways.[54]

Off-line, if capturing of relationships between documentation entities is performed as a post implementation activity. The off-line production of relationships may be manual or automatic, where in the case of the automatic approach analysis software will try to extract the dependencies. To build up the traces manual can be very cost intensive, and the downside in the automatic tracing is the large number of possible relations, which needs manual analyzes anyway.

On-line, if capturing of relationships is integrated during performing the development activities. The productions of traces as side effect of the development process supports also the verification of the realization of these trace relationships. Therefore, this is the more effective and useful way.

3.3.2.3. Extract and Represent Traces

The third step is to find a suitable representation of the traces depending on the purpose of traceability. For the representation are three different solutions in common use.[55]

- Graphical models, where documentation entities are represented by entities and relationships between them.
- Cross references, where links between documentation entities are embedded as hyperlinks in the text, which is either an informal language text or a formal specification. The use of cross references enables the navigation through the related documents
- Traceability matrices, where links between documentation entities are represented in a matrix. The horizontal and vertical dimensions of the matrix list the documentation entities that are able to link. The entries inside the matrix are representing the existing link between the entities. The entities, which are listed in the different dimension, can be the same or also different.

[54] Pinheiro 1996

[55] Wieringa 1995

3.3.2.4. Maintain Traces

The final and most important step is to ensure the maintenance of the requirements traceability. This is very important, since any future change request represents a change in the requirements, and this can affect the existing traces between requirements. To ensure the good maintainability of requirements and traces the effective communication between all involved parties is very important. To support the understanding between the project members the used model of the documentation is very important.

4. Requirements Modeling

All engineering disciplines use models to develop the products they intend to build. A model serves as an abstraction, an approximate representation of the real item that is being built. The same basic reasons why other complex systems are modeled apply to software, too. The goal is to manage the complexity and to understand the design and associated risks. More specifically, by modeling software, developers can create and communicate software designs before committing additional resources, trace the design back to the requirements, helping to ensure that they are building the right system and practice iterative development, in which models and other higher levels of abstraction facilitate quick and frequent changes.

Requirements models are used to discover and clarify the functional and data requirements for software and business systems. Additionally, the requirements models are used as specifications for the designers and builders of the system. Requirements models are used when gathering requirements, during systems analysis and will be reused when the requirements change. Building accurate models increase the change to eliciting correct requirements. To specify, visualize, construct and document the artifacts of a software systems the graphical language UML can be used.

4.1. Unified Modeling Language

The Unified Modeling Language (UML) was developed in the mid-nineties. At that time, different modeling methods were available which addresses different problems modeling object-oriented systems. The basic idea behind UML was to unite the main three object oriented modeling methods Booch, developed by Grady Booch, OMT, introduced by James Rumbaugh and OOSE, created by Ivar Jacobson to one non-proprietary language. Concepts from many other object-oriented methods were also integrated with UML with the intent that UML would support all object-oriented methods. It also incorporated a number of best practices from modeling language design, object-oriented programming and architectural description languages. The first version UML 0.9 was released in 1996 and after feedback from the community contribution from several organizations, UML 1.0 was submitted to the OMG in 1997. The current version is 2.1.2.

UML is a language with a very broad scope that covers a large and diverse set of application domains. Not all of its modeling capabilities are necessarily useful in all

domains or applications. This suggests that the language should be structured modularly, with the ability to select only those parts of the language those are of direct interest.[56]

A UML model consists of elements such as packages, classes, and associations. The corresponding UML diagrams are graphical representations of parts of the UML model. UML diagrams contain graphical elements (nodes connected by paths) that represent elements in the UML model.[57] UML 2.1.2 defines thirteen types of diagrams, divided into three categories: Six diagram types represent static application structure; three represent general types of behavior; and four represent different aspects of interactions:[58]

Structure Diagrams represent the static application structure of the objects in a system and include the Class Diagram, Object Diagram, Component Diagram, Composite Structure Diagram, Package Diagram, and Deployment Diagram. They describe those elements in a specification that are irrespective of time. The elements in a structure diagram represent the concepts of an application, and may include abstract, real-world and implementation concepts. Structure diagrams do not show the details of dynamic behavior, which are illustrated by behavioral diagrams. The different structure diagrams are used to show different views on the system.[59]

- Class Diagram: Shows the static logical design of a system by displaying the objects as classes with their attributes, and the relationships between them.
- Object Diagram: Shows static snapshots of instances of objects from the class diagrams in the perspective of real cases.
- Composite Structure Diagram: Shows the internal structure of a class, component, or use case, including the interaction points to other parts of the system.
- Component Diagram: Shows the static implementation view of a system with a set of components and their relationships. A component represents a physical implementation like a part of the software, and sometimes hardware components, that make up the system.

[56] OMG 2007, p 1 f

[57] OMG 2007, p 681

[58] OMG 2007, p 684

[59] OMG 2007, p 684

- Deployment Diagram: Show the connectivity of physical nodes in an architectural view of the system. A node is a resource that provides a physical operating environment for executing one or more components.
- Package Diagram: Shows how model elements are organized into packages as well as the dependencies between packages.

Behavior Diagrams represent general types of behavior and include the Use Case Diagram, used by some methodologies during requirements gathering, Activity Diagram, and State Machine Diagram. Behavior Diagrams show the dynamic behavior of the objects in a system, including their methods, collaborations, activities, and state histories. The dynamic behavior of a system can be described as a series of changes to the system over time.[60]

- Use Case Diagram: Shows the interaction and the boundary between the system and users.
- Activity Diagram: Shows different workflows of activity within the system. Including the sequential flow between activities, parallel processing, and the possible decision paths.
- State Machine Diagram: Shows the states transitions or changes of state how an object moves from one state to another and the rules that govern that changes over time.

Interaction Diagrams, all derived from the more general Behavior Diagram, represent different aspects of interactions, and include the Sequence Diagram, Communication Diagram, Timing Diagram, and Interaction Overview Diagram.

- Sequence Diagram: Shows the interaction between users, screens, objects and entities within the system. It provides a sequential map of message passing between objects over time.
- Communication Diagram: Shows instances of classes, their interrelationships, and the message flow between them. Communication diagrams typically focus on the structural organization of objects that send and receive messages.
- Timing Diagram: Shows the change in state or condition of one ore more objects, instances or roles over time. It is typically used to show the change in state of an object over time in response to external events.

[60] OMG 2007, p 684

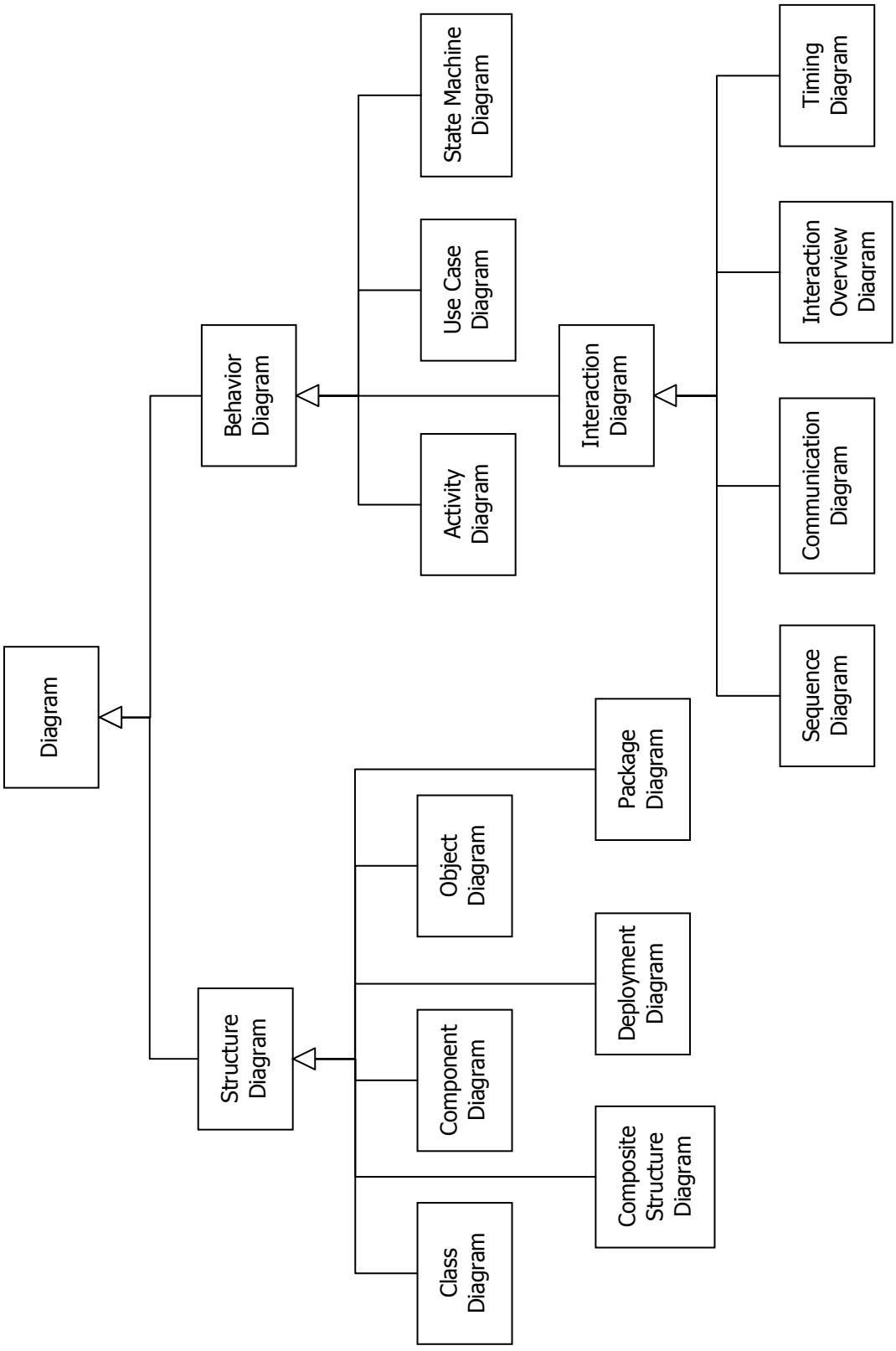

Figure 3.3.2-1 The taxonomy of structure and behavior diagram[61]

[61] OMG 2007, p 684

- Interaction Overview Diagram: Are variants on UML activity diagrams which overview control flow. The nodes within the diagram are frames instead of the normal activities on an activity diagram. These frames are either interaction frames that show any type of UML interaction diagram like sequence diagram, communication diagram, timing diagram or interaction overview diagram, or these frames are interaction occurrence frames that indicate an activity or operation to invoke.

Each of the different diagrams focuses only on particular fractions or on special views of the system. However, combining them together delivers a model to describe the system. Figure 3.3.2-1 shows the taxonomy of UML structure and behavior diagrams.

4.1.1. UML Use Case Diagram

Use cases are specifying required usages of a system. Typically, they are used to capture the requirements of a system, that is, what a system is supposed to do. The key concepts associated with use case diagrams are actors, use cases, and the system under consideration to which the use cases apply. The users and any other systems that may interact with the subject are represented as actors. Actors always model entities that are outside the system. The required behavior of the system is specified by one or more use cases, which are defined according to the needs of actors. Use cases, actors, and systems are described using use case diagrams.[62] UML use case diagrams consist of the following elements.

4.1.1.1. Use Case

Each use case specifies a set of actions that the system can perform in collaboration with one or more actors and delivers an observable result that is of value for one or more actors or other stakeholders of the system. Use cases define the offered behavior of the system without reference to its internal structure. These behaviors, involving interactions between the actor and the system, may result in changes to the state of the system and communications with its environment. A use case can include possible variations of its basic behavior, including exceptional behavior and error handling. The behavior of a use case can be described by a specification that is some kind of behavior, such as interactions, activities, and state machines, or by pre-conditions and post-conditions as well as by natural language text. Which of these techniques, or

[62] OMG 2007, p 585 ff

combination of it, will be use depends on the nature of the use case behavior as well as on the intended reader.[63]

Use cases may have associated actors, which describe how an instance of the classifier realizing the use case and a user playing one of the roles of the actor interact. Two use cases specifying the same system cannot be associated since each of them individually describes a complete usage of the system.[64]

A use case is shown as an ellipse, either containing the name of the use case or with the name of the use case placed below the ellipse[65], shown in Figure 4.1.1-1.

Figure 4.1.1-1 Use case element

4.1.1.2. Actor

An Actor specifies a type of role played by a user or any other system that interacts with the system (e.g., by exchanging signals and data), but which is external to the system. Actors may represent roles played by human users, external hardware, or other system. An actor does not necessarily represent a specific physical entity but particular facet of some entity that is relevant to the specification of its associated use cases. Thus, a single physical instance may play the role of several different actors and, a given actor may be played by multiple different instances.[66]

An actor is represented by "stick man" icon with the name of the actor, shown in Figure 4.1.1-2.[67]

[63] OMG 2007, p 594

[64] OMG 2007, p 595

[65] OMG 2007, p 596

[66] OMG 2007, p 586

[67] OMG 2007, p 587

Figure 4.1.1-2 Actor element[68]

4.1.1.3. Include Relationship

An include relationship between two use cases means that the behavior defined in the including use case is included in the behavior of the base use case. The included use case is not optional, and is always required for the including use case to execute correctly. The include relationship is intended to be used when there are common parts of the behavior of two or more use cases. This common part is then extracted to a separate use case, to be included by all the base use cases having this part in common. Since the primary use of the include relationship is for reuse of common parts, what is left in a base use case is usually not complete in itself but dependent on the included parts to be meaningful. This is reflected in the direction of the relationship, indicating that the base use case depends on the addition but not vice versa.[69]

Execution of the included use case is analogous to a subroutine call. All of the behavior of the included use case is executed at a single location in the included use case before execution of the including use case is resumed.[70]

An include relationship between use cases is shown by a dashed arrow with an open arrowhead from the base use case to the included use case. The arrow is labeled with the keyword «include», shown in Figure 4.1.1-3.[71]

[68] OMG 2007, p 587

[69] OMG 2007, p 593

[70] OMG 2007, p 593

[71] OMG 2007, p 593

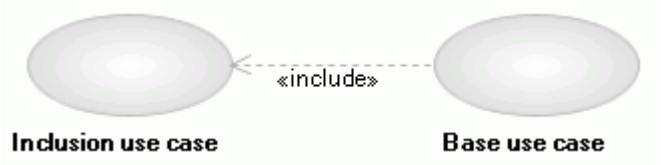

Figure 4.1.1-3 Include Relationship element

4.1.1.4. Extend Relationship

An extend relationship specifies that the behavior of a use case may be extended by the behavior of another, usually supplementary, use case. The extension takes place under specific conditions at one or more specific extension points defined in the extended base use case. The base use case is defined independently of the extending use case and is meaningful independently of the extending use case. On the other hand, the extending use case typically defines behavior that may not necessarily be meaningful by itself. The same extending use case can extend more than one base use case.[72]

An extend relationship between use cases is shown by a dashed arrow with an open arrowhead from the use case providing the extension to the base use case. The arrow is labeled with the keyword «extend». The conditions of the relationship as well as the references to the extension points are optionally shown in a note attached to the corresponding extend relationship, shown in Figure 4.1.1-4.[73]

Figure 4.1.1-4 Extend Relationship element[74]

4.1.1.5. Extension Point

An Extension Point is a feature of a use case that identifies a point where the behavior of a base use case can be extended with elements of another extending use case. An extension point is a reference to a location within a use case at which parts of the

[72] OMG 2007, p 589

[73] OMG 2007, p 591

[74] OMG 2007, p 591

behavior of other use cases may be inserted.[75] Each extension point has a unique name within a use case, shown in Figure 4.1.1-5 .

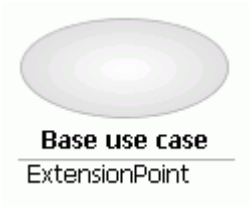

Figure 4.1.1-5 Use case element with Extension Point

4.2. Requirements gathering with Use Cases

As described in Chapter 2, Scenario-Based Requirements Engineering was frequently used from system analysts to describe ways a user can interact with a software system to help elicit requirements. The object-oriented development methodologies formalized the use case approach to requirements elicitation and modeling. Although the origin is based on the object-oriented development, this approach can be applied to projects that follow any other development procedure. The shift in perspective and thought processes that use cases bring to requirements development is more important than drawing formal use case diagrams. The focus on what the users need to do with the system is much more powerful than other traditional elicitation approaches of asking users what they want the system to do.

4.2.1. Capturing User Requirements with Use Cases

A use case describes a sequence of interactions between the system and external actors. An actor is a person, another software application, or some other entity that interacts with the system to achieve some goal. Actors show the roles that members of one or more user groups can perform.

A single use case might include a number of logically related tasks and several interaction sequences that lead to completing the task of a system user. The use case is a collection of related scenarios, where one scenario is identified as the primary scenario. The primary scenario is described by listing a sequence of events or interactions between the actors and the system. After all these steps are completed, the actor has accomplished the intended goal for the use case.

[75] OMG 2007, p 591 f

Other scenarios within the use case are described as alternative scenarios. These alternate scenarios result also in successful task completion, but they represent variations in the specifics of the sequence of events or interactions. The primary scenario can branch off into an alternative flow at some point in the sequence, and rejoin the primary scenario later. Several use cases might share some common functionality.

To avoid duplication, it is possible to define separate use cases that contain the common functionality and indicate that other use cases should be include that common use case. Furthermore, use cases can also extend the behavior described in another use case.

Conditions that result in the task not being successfully accomplished are documented as exceptions. It is important to include the exceptions scenario in the description of the use case, because this represents the user view of how the system should behave in the case an unsuccessful outcome may result. [76]

4.2.2. Capturing System Requirements with Use Cases

The use case description often does not provide the system developers every detail about the functionality they must build. Gathering all requirements in use cases descriptions will require writing highly detailed use cases. On the other hand, stopping requirements development at the user requirements stage keeps developers asking questions to get the additional requirements that were not included in the use cases.

To deliver any possible functional and non-functional requirement in use cases only forces to invent pure system use cases to hold all the functional requirements because a use case is the only container available to describe system functionality. However, some system functionality does not fit appropriately into a use case. If only use cases are used to capture functional requirements, this attempt wind up inventing artificial use cases, those that do not provide user value, just to have a place to store certain functionality. This artificiality does not add value to the requirements development process.[77]

[76] Armour, Miller 2001, p 24

[77] Wiegers 2006 p 94

It gets even more confusing if use cases describe the bulk of the functionality but place additional functional requirements that do not relate to specific use cases into a supplemental specification. This approach forces the developer to get some information from the use case documentation and then to the supplemental specification for other relevant inputs.

To avoid this, the preferable strategy is for the analyst to create a Software Requirement Statement (SRS) as the ultimate deliverable for the developers and testers. This SRS should contain all the known functional and nonfunctional requirements, regardless of whether they came from use cases or other sources. Functional requirements that originated in use cases should be traced back to those use cases so that readers and analysts know where they came from.[78]

4.3. Iterative and Incremental Use Case Model

In addition to the use cases themselves, the use case model includes the complete set of diagrams and descriptions like supporting text, glossaries, and other documentation used in specifying the use cases. When creating use cases for large systems, the use case diagrams, descriptions, and other model elements will be evolving and will be refined as the use case modeling effort progresses. [79]

With an iterative and incremental approach, use cases are first used to model the system at a conceptual level, focusing on the primary behaviors of the system, and they are redefined to describe the increasing levels of detailed requirements information.[80] As the requirements analysis process proceeds, the use case descriptions go through a series of abstract levels, with more details added to the descriptions as the knowledge about the problems increases.[81]

For this reasons, a single, flat level of detail is normally insufficient to support an ongoing and evolving analysis process and to provide the ability to robustly and effectively capture, partition and represent the vast functionality of a large system. It is necessary, as the use case descriptions are progressively defined, to represent them at

[78] Wiegers 2006 p 95

[79] Armour, Miller 2001, p 63

[80] Armour, Miller 2001, p 63

[81] Armour, Miller 2001, p 64

higher levels of detail. As more about the requirements is learned, more details are added to the use case descriptions. Details can be added by[82]

- Expanding information within a specific use case, that is, adding to its description
- Finding additional use cases as the requirements become clearer
- Extending the details of the use case model, through extend, include and generalization relationships
- Integrating multiple use cases to define larger system processes and functions and the bigger picture of the system

A use case model can include the following[83]

- Initial use case description: Use case descriptions developed during the beginning of the requirements analysis identify and broadly describe system behaviors initiated by actors. They provide a conceptual representation of the system.
- Base use case description: Base use case descriptions expand on the initial use case descriptions by documenting use case behavior in detail. Base use cases focus on the ideal behaviors and paths of use cases and avoid documenting exceptions or alternatives.
- Elaborated system use case descriptions: In elaborated descriptions, details of behavior such as condition logic and alternative flows are added to the base use case descriptions.

Figure 4.2.2-1 shows the growing details of the use case description during requirements capturing.

[82] Armour, Miller 2001, p 64

[83] Armour, Miller 2001, p 64 f

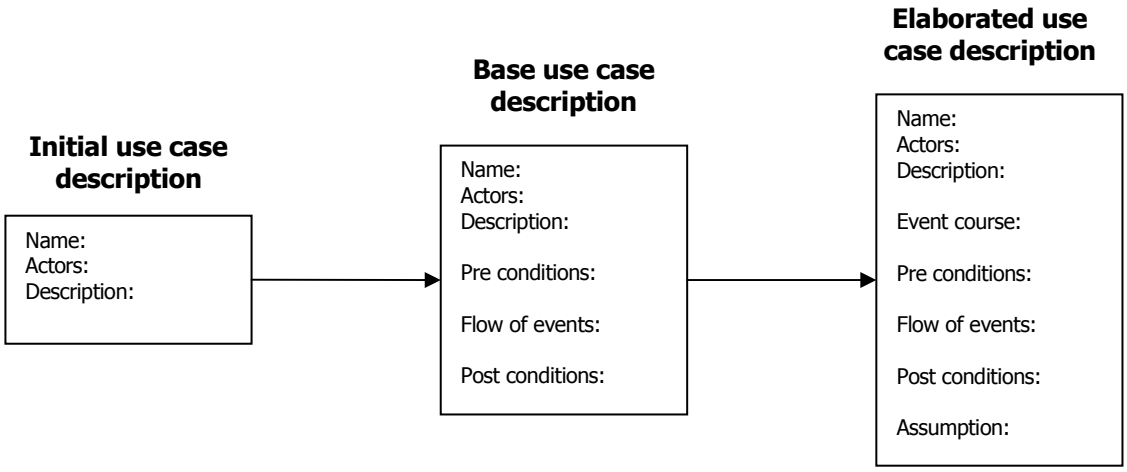

Figure 4.2.2-1 More details added to the use case descriptions[84]

There are a number of reasons to describe use cases at various forms and levels of abstraction. First, as mentioned earlier, a complete use case model is not built in just one step. The model is built progressively and iteratively, because of the process of analysis, discovery, and modeling of system functionality. As more knowledge is acquired, the model is elaborated and the representations capture this progress as it occurs.[85]

Second, the use case model has several audiences like users, analysts, designers to name a few. The different audiences view the use cases in different ways. Different stakeholders of the system have different viewpoints from which they examine and validate the use case model. Multiple representations help facilitate their understanding and their validation of the model.[86]

Third, multiple representations allow use cases to be more easily partitioned for further development. A model containing only initial use case descriptions can be created for a system concept or concept of operation document. Elaborated use case descriptions and a comprehensive set of extend and include use cases can be developed during analysis activity. Incremental delivery is also supported, as each increment is

[84] Armour, Miller 2001 p65

[85] Armour, Miller 2001 p66

[86] Armour, Miller 2001 p66

developed, the use cases selected for development in that increment are further refined.[87]

Forth, multiple use case representations are an organizational technique for adding order and understandability to a large use case model.[88]

In addition to the use case descriptions, the system use case model includes other elements[89]

- Use case diagrams: Diagrams provide a high-level visual representation of the actors, the use cases and the relationships between them.
- Extend, include and generalization relationships: These relationships document extensions and common parts in use cases and the major exception conditions and alternative flows of events that can occur in a use case. In an include relationship, a use case includes the behavior that are included in multiple use cases. Extend relationships document extensions in the use case flow of events. Generalization relationships are similar to those in object modeling and document relationship between an abstract or more general use case and more specific sub use case.
- Instance scenarios: These scenarios describe instances or examples of how use cases are executed. They are useful as a validation mechanism for users and are able to use as source to generate test cases.
- Analysis object model: These models map the use cases to the objects needed to realize the behaviors in the use case. They help to identify requirements that are more detailed and to map the use cases to the design. The use case to object model mapping activity can normally be performed by using UML sequence or collaboration diagrams.
- Business function packages: When several use cases are part of a large business process, these packages group the use cases by common business functions or areas.

Beside the use of the use case model during development of the system, the created documentation will help to develop various documents for further use like User Guides and Training documentation. The description of use cases and their scenarios are a

[87] Armour, Miller 2001 p67

[88] Armour, Miller 2001 p67

[89] Armour, Miller 2001 p65

well-defined starting point for user guides. Each use case describes to the user how the system behaves and how they interact with it. Furthermore, the use cases can be put together with screen shots of the user interface and combined with exercises to build the starting point for training materials.[90] The use cases are also repository of knowledge to store expertise about the business, which was collected during the implementation project.[91]

[90] Schneider, Winters 1999, p 166

[91] Schneider, Winters 1999, p 167

Part II

CASE STUDY ON AN IMPLEMENTATION PROJECT

5. Project Description

Companies collect, generate and store big quantities of data. In the past that data was spread across separate computer systems, each housed in an individual function. These legacy systems had a huge impact on business productivity due to the additional workload of re-keying, reformatting, updating and debugging the same data in different places. Enterprise Resource Planning (ERP) solutions became the replacement for obsolescent legacy systems for many companies. The ERP was promoting common data, standard business processes, and were promising efficiencies such as shorter intervals between orders and payments, lower back-office staff requirements, reduced inventory and improved customer service.

Large organizations, in particular multinational enterprises, have been at the forefront of the ERP movement since its origins and replaced the legacy systems with ERP solution. Nevertheless, a multinational enterprise has several units across separate geographical locations. Many of these enterprises do not use a central single instance of the ERP, instead they are utilizing local instance of the same ERP software. The number of systems was reduced to a single ERP environment inside the units, but the enterprise itself stays still with several separate instances of ERP. As the information used in managing a multinational enterprise originates in different locations, it is fragmented between these instances. With such an environment in place, it can be difficult to get a global distributed process mapped into the ERP systems.

The objective implementation project started in such an environment, where the existing support of the process by IT systems has been found incomplete, especially concerning sales cases that transcend organizational units and therefore transcend ERP systems, which results in separated representations in different ERP installations. To improve this situation, an envisaged process, which differs from the given situation regarding its IT support and in detailed process definition, has been designed. Additional, basic requirements for the ERP systems covering both the support of the envisaged process and respective reporting have been developed and described.

Each of the companies involved uses the standard ERP software Microsoft Business Solutions Dynamics NAV, Navision. During this project an additional granule was developed, which supports the complete sales process, starting from the first inquiry from the customer until the final delivery and invoicing. Because of the central development of this granule, the implementation of this business process is based on the same rule set and the collection of the reporting data will be consistent in all systems.

The main objectives of the implementation project is the extension of existing standard ERP software Navision (Microsoft Business Solution Dynamics NAV) to support the sales process on the operational side to increase efficiency by reducing the workload trough elimination of multiple data entry, and to ensure consistent and comparable data for the sales reporting on the analytic side. Beside that, the collection of additional data during the work on the different steps of the process should enable the monitoring of the process flow.

The collected reporting data is utilized in a central reporting system. The creation of a central reporting solution results in a massive reduction of workload by creating a high degree of automation in reporting. This automation ensures higher information quality by covering detailed data all over the distributed ERP installations and supports the operative processes by presenting the big picture across all organizational units involved.

5.1. Organizational Environment

The PPC Insulators group is a manufacturer of electro-porcelain supplying products and services to customers worldwide. The different subsidiary companies of the group are internationally distributed and located in Austria, Germany, Sweden, Slovakia, USA, Thailand and China.

The responsibilities during the sales process are divided into two different roles. One role is the Point of Sales (POS), where all communication with the customer is going through. The second role is the production unit, where the production of the articles takes place. Each sales department of the company, which acts as POS, sells articles from the production unit located in the same company, and products from any other production unit located in one of the other companies inside the group. Therefore, a

sales process for one single customer request can involve one location, which covers the role of POS and production unit. For this case, the process is covered by one ERP system and the complete data is available in this single instance. Alternatively, many different locations are involved, where one site is the POS and any other site may fulfill the role of the production unit. For this case, the process is covered by more than one ERP system and the data inside of one ERP system shows only a fragment of the complete process.

5.2. The given Situation of the Sales Process

To explain the scope of the project, it is necessary to take a closer look at the workflow of the sales process itself.

5.2.1. Overview of the Sales Process

Most of the articles, which are requested from the customers, are not sold from stock, but make-to-order production on customer request only. For such cases, the delivery of the products will go directly from the production unit to the customers requested location.

5.2.1.1. Make-to-order Production

The following describes the main sub-processes of the order intake process and their timely orders for the case of make-to-order production and direct shipment from the production plant to the customer. This process flow is shown as activity diagram in Figure 5.2.1-1.

- The POS receives the inquiry from the customer.
- The POS sends the inquiry to the plant, which is either the local production unit or a remote production plant. To ask different production units for quotes for one article requested from the customer, more than one inquiry to the plant is possible. In the case, that different articles are included in a single inquiry from the customer, multiple inquiry to plants can be initiated for each of these articles.
- The production unit, either the local production unit or a remote production plant, receives the internal inquiry.
- The production, either the local production unit or a remote production plant, sends the calculated internal offer.

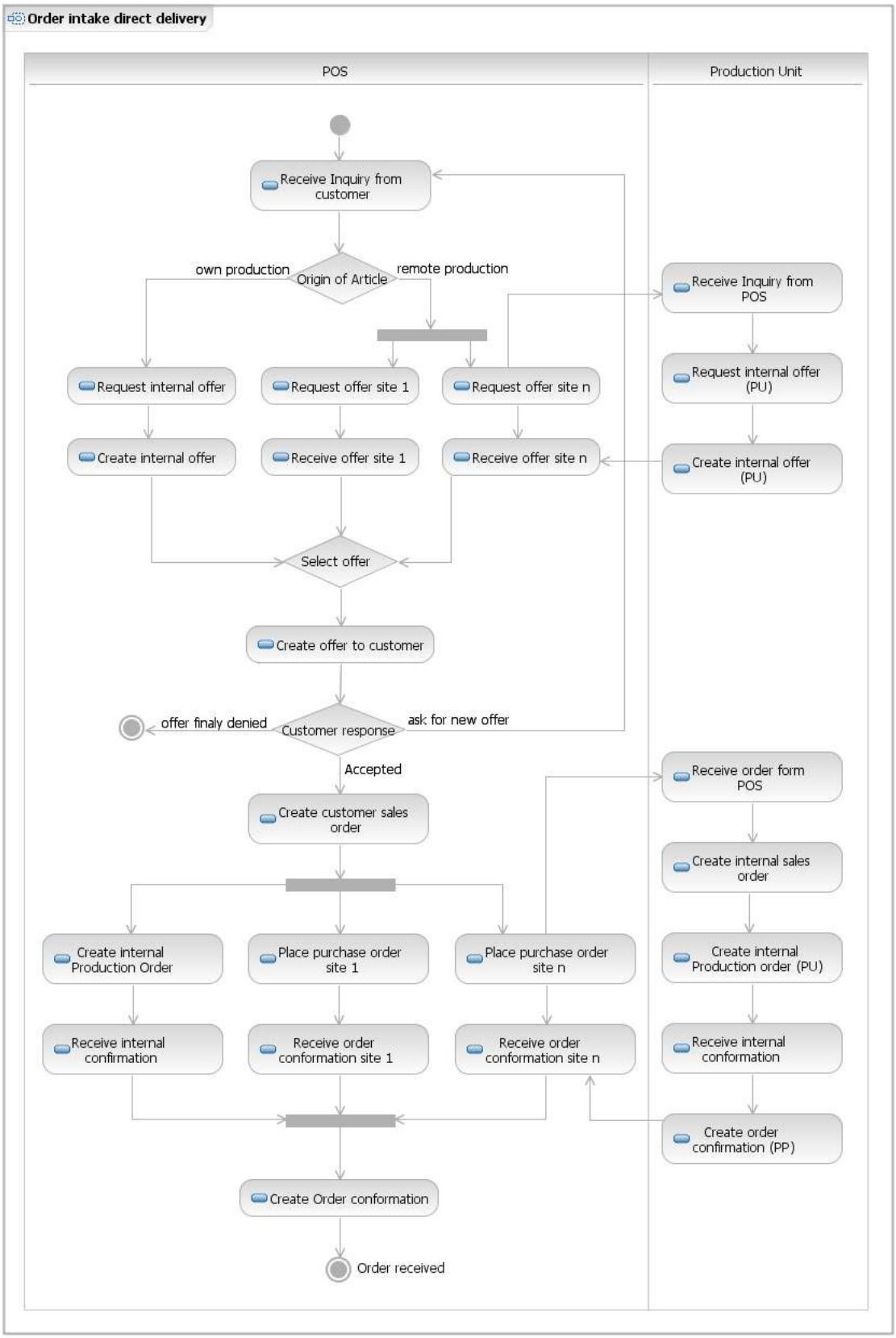

Figure 5.2.1-1 Activity diagram for order intake

- The POS receives the offer from the plant. Only one offer from the plant can be received for each released inquiry to the plant but more than one offer from the plant are possible for each article included in the inquiry from the customer.
- The POS selects the offer from the plant, which will be used for the offer to the customer.
- The POS sends the offer to the customer, based on the information received in the offer from the plant.
- The customer reviews the offer and can either deny it, which ends the process, request a new offer, which restarts the offering process, or accept the offer.
- If the customer accepts the offer, the POS receives the purchase order from the customer.
- The POS creates the sales order from the customer.
- The POS sends the order to the plant, in the case of a remote production, or uses the customer order for the local production planning.
- In the case of remote production, the purchase order is received in the production unit proceeded as a local sales order.
- The remote production uses the customer order for the local production planning.
- The remote production sends the order confirmation to the POS.
- The POS receives the order confirmation from the plant, in the case of a remote production, or receives the information for the confirmation from the local production planning.
- The POS sends the order confirmation to the customer, based on the information received in the order confirmation from the plant or the local production.

The sales order and the production order are related to a received purchase order from an external customer and the order intake is completed successfully. Therefore, the order is considered as received and is part of the order book for the local reporting on the level of the POS. In addition, this order is considered as received and is part of the order book for the high-level group reporting.

5.2.1.2. Make-to-stock Production

Beside the make-to-order production, the sales process has an important variation, which is the make-to-stock production. The following describes the main sub-processes of the order intake process to sell products from the stock and the prior purchase to the stock. In this case, the process steps are similar to the sub-processes of the make-

to-order production, but the timely order of the main sub-processes is divided into two different parts of the time lines. The first part, the purchase of the goods into the stocks is taken place before the second part of the process starts, where the goods are sold from the stock.

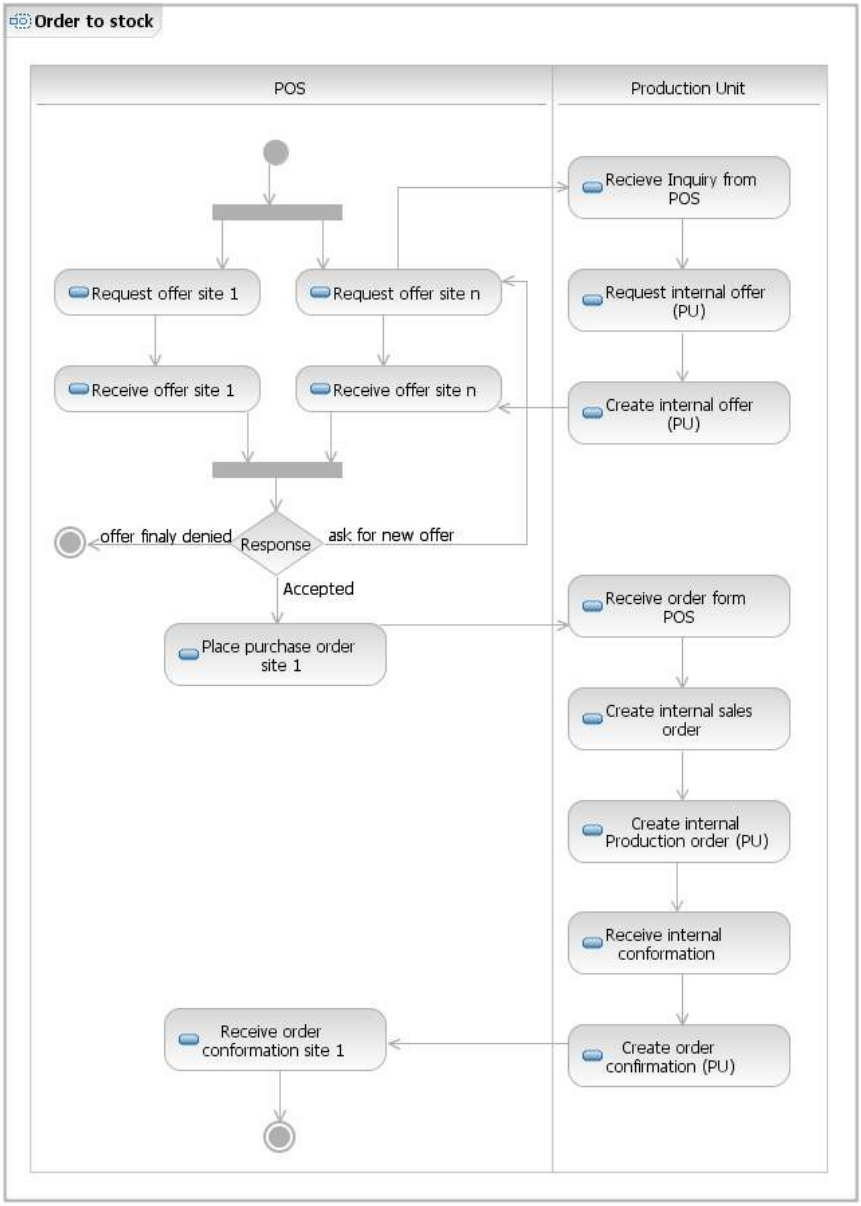

Figure 5.2.1-2 Activity diagram for purchase order to stock

In the case of local production, where the POS and production is located in the same company, the production into stock is not part of the sales process, but part of the production planning. For the case, where the production is done by an external supplier, the process of purchasing the goods into stock is part of the purchase process

and therefore not part of the sales process. Only the case, where the production is not located in the same company as the POS, but in one of the production plants of the group, is of further interest for the support of the operative work and for the monitoring of the process. The process flow for the purchase to the stock is shown as activity diagram in Figure 5.2.1-2.

- The POS sends the inquiry to the plant, which is a remote production. More than one inquiry to the plant is possible different production units can be asked for quotes.
- The remote production receives the internal inquiry from the POS.
- The remote production sends the calculated internal offer to the POS.
- The POS receives the offer from the plant. Only one offer from the plant per inquiry to the plant is possible.
- The POS reviews the offer and can either deny it, which ends the process, request a new offer, which restarts the offering process, or accept the offer.
- If the offer is accepted, the POS sends the order to the plant to the remote production.
- The purchase order is received in the production unit and preceded as a local sales order.
- The remote production uses the customer order for the local production planning.
- The remote production sends the order confirmation to the POS.
- The POS receives the order confirmation from the remote production.

The production order in the production unit is not related to a received purchase order from an external customer. Therefore, the order is considered as received and is part of the order book for the local reporting on the level of the production unit. Otherwise, this order is not considered as received and is not part of the order book for the local reporting on the level of the POS and for the high-level group reporting.

The process flow for the second part of the process, the sales from stock, is shown as activity diagram in Figure 5.2.1-3.

- The POS receives the inquiry from the customer.
- The POS sends the offer to the customer, based on the information of the goods on stock.
- The customer reviews the offer and can either deny it, which ends the process, request a new offer, which restarts the offering process, or accept the offer.

- If the customer accepts the offer, the POS receives the purchase order from the customer.
- The POS sends the order confirmation to the customer, based on the information of the stock.

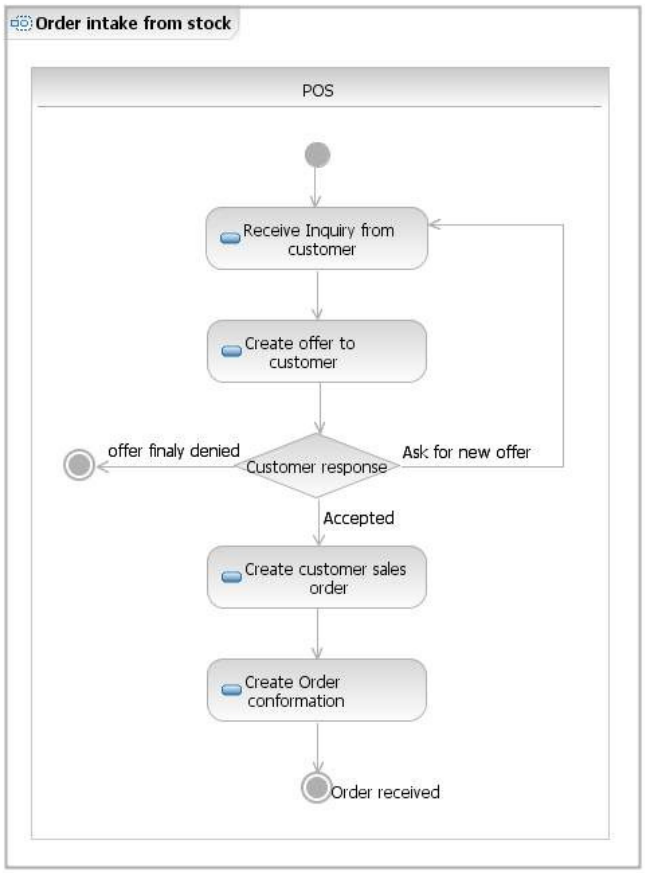

Figure 5.2.1-3 Activity diagram for sales order from stock

The sales order is now related to a received purchase order from an external customer and the order intake is completed successfully. Therefore, the order is considered as received and is part of the order book for the local reporting on the level of the POS. In addition, this order is considered as received and is part of the order book for the high-level group reporting.

5.3. The given Support from IT Systems

Before the start of the project, the support from IT systems to assist the workflow of the sales process and to enable an overall reporting on commercial numbers and process monitoring was incomplete.

5.3.1. IT Support for Process steps

The support from IT systems for the different steps during the sales process is based on the standard functionality of the ERP solution. Like any other ERP, also Navision includes in the sales modules the support for customer offers and orders and in the purchase modules the use of purchase requests and purchase orders. Since all the subsidiaries are using local installations of the ERP system, some of them have extended functionalities with customized additional programming to broaden the scope of the use of the system. However, these functions are restricted to the standard article information like description, number of pieces and price. For the internal information flow between different companies of the group, the transfer of more detailed information about the technical parameters and the commercial figures of the products are requested. The situation of the different process steps at the beginning of the implementation project is summarized at the following.

- Inquiry from the customer

The support for this process step is implemented in some ERP systems but also in these systems not used for all inquiries, which are received from the customers. Where this step is supported, there is no verification in place, to prove the incoming request, if the customers supplies sufficient data to enable an article calculation. Without this check of data, additional communication with questions and answers is started, to collect missing data from the customer. No information for reporting purposes is collected.

- Inquiry to the plant

No special support for this process step is implemented in any ERP system. The work is done outside the ERP system and the communication is beside ERP system with e-mail or fax communication. No system check for completeness of the transmitted data is used. No information for reporting purposes is collected.

- Offer from the plant

No special support for this process step is implemented in any ERP system. The work is done outside the ERP system and the communication is beside ERP system with e-mail or fax communication. No system check for completeness of the transmitted data is used. All data for reporting purposes is collected manually beside the ERP system. A definition of the necessary data exists as a working guideline from central sales.

- Offer to the customer

The support for this process step is implemented in some ERP systems but in these systems, it is not used for all offers. There exists no system control, if the data from the offer from the plant is used correctly, or if substantial information was changed for the offer to the customer. Because of the data transfer outside the ERP system, all data from the offer from the plant is keyed in manually. Since the information requested for reporting purposes is more comprehensive than the ERP systems supplies, it is collected manually beside of the ERP system.

- Order from the customer

The support for this process step is implemented and used in all ERP systems.

- Order to the plant

The support for this process step is implemented and used in any ERP system. Communication is done beside ERP system with e-mail or fax communication.

- Order confirmation from the plant

No special support for this process step is implemented in any ERP system. The communication is done beside ERP system with e-mail or fax communication. Since the information requested for reporting purposes is not fully covered by standard reports from the ERP system, there are difficulties to extract all the data needed for reporting.

- Order confirmation to the customer

No special support for this process step is implemented in any ERP system. There exists no system control, if the data from the order confirmation from the plant is used correct, or if substantial information was changed for the order confirmation to the customer. Since the information requested for reporting purposes is not fully covered by standard reports from the ERP system, there are difficulties to extract all the data needed for the reporting.

5.3.2. IT Support for Reporting

The reporting standards cover two different views on the sales process. At first, it summarizes and consolidates the commercial data, delivered from the different instances of ERP systems. The purpose is to provide information about the value of the sales and the order intake for a period of time and the existing orders at a defined reporting day. In addition, it is requested to cover the process flow through different

instances of ERP systems to monitor the process itself. The situation of the IT support for the reporting at the beginning of the implementation project is summarized in the following.

- Sales Reporting

To receive information covering all subsidiary companies of the group, the reporting needs to be done in a two-step process. At first, the data is reported on the level of each single unit. Afterwards the data on the group level view needs additional manual work on grouping, consolidating and summarizing the figures. In addition, because of the missing data exchange between the different ERP systems, the same sales case with an external customer can be displayed in different process steps in the POS and the production unit. These circumstances increase the risk to show inconsistent data in sales reporting.

Beside the different status on one sales case, the involved companies are using different customer master data, since each of the different instances of ERP systems uses its own local data store. If one sales case with one external customer is handled by two different companies for POS and production unit, this sales case is displayed with different local customer data. Therefore, any customer related report needs additional manual work to verify the connected customers through different ERP systems.

- Process Monitoring

To monitor the duration of all sales cases through the different process steps is not possible, since not all process steps are implemented in the ERP systems. Moreover, for those steps, which are implemented, only fragments of the necessary data are collected inside the different ERP systems. Reporting on manually collected data is not usable for process monitoring anyway.

5.4. Organizational Aspects

Beside the operational workflow and the reporting view on the sales process, also some organizational aspects are influencing the project.

5.4.1. Global Ownership

The global owner of the sales process, and thus recipient of all reporting information, in detail and overview, is the central sales department. The responsibilities of the

owner are the definition of the rules of the process for process management and process monitoring, formally as well as regarding the contents for individual sales cases. Finally, the measurements are used for continuous process optimization. Furthermore, the central sales department is responsible for the general communication with the customer, which also includes solving any problem for the customer, and the pricing of the products. These tasks are the duty of the sales manager, who is responsible for the customer.

5.4.2. Local Ownership

Local owners of the sales process are all the different POS. The responsibilities of the local owners are the process execution according to the global rules and the reporting on all aspects of the process execution.

The execution of the sales process is done by the employees in customer service, who are responsible for the usage of the ERP system to maintain all data in connection with the customer and sales cases, communication with the different possible production units, and the creation of all documents for the customers, which are related to their tasks of sale.

The local sales offices report directly to the central sales department concerning all aspects of the executed sales process.

5.4.3. Area of Customers Responsibility

Each POS is given its geographical area of responsibility by the head office. Customers are assigned to POS per default through their legal address, by checking if residing inside the area of responsibility. However, the head office may define exceptions to this geographic rule, these exceptions leading to a specific assigning of a customer to a sales office. Anyway, one customer is managed by only one sales office, orders of this customer may not be managed by other sales offices.

5.4.4. Market Related Aspects

Since the customers of the group are also competing against each other on their customer side, it frequently happens that one single end-user project and therefore inquiry may lead to multiple inquiries coming into different POS from different customers. Two different cases are possible. In one case, the different customer

requests are received in the same POS or in the other case, the requests are targeted to different internationally distributed POS's. This double entry of one single sales change makes it difficult to create correct sales forecasts. The existing standard ERP solutions do not support the possibility to track this kind of information to find such issues. Therefore, there is a need to identify those multiple inquiries as such, and recognize the underlying inquiry or project from the final end user. This identification can be performed through gaining knowledge concerning the identity of the end user, so the customer of the customer, and the final place of installation.

5.5. Characterization of the Sales Process

To summarize the situation before the project started, the given sales process over all the various subsidiary companies is characterized by the following:

- Distributed: the process starts in many POS geographically distributed over many countries all around the world.
- Cooperative: the POS creates all offers to the customer based on internal offers from production units.
- Interlinked: every sales office can request internal offers from any production unit.
- Managed: there are organizational, financial and process-related guidelines applying to this process.
- Partly standardized: there is a different ERP implementation for each sales office and if existing, integration with local production planning.
- Lacking IT support: the global process is not supported by any global electronic system, but its local aspects are managed in local ERP systems.
- Awkward to monitor: Central sales management covers all sales cases, worldwide, without proper IT support.

5.6. Goals for the Implementation Project

To improve the given situation with an extended IT support the, goals for the objective implementation project have been defined as following.

5.6.1. General Goals

To improve the given situation, the following general goals for the implementation project have been defined to support the future sales process.

- Transparency

The status of each sales case, including all relevant data in each sales office has to be visible for sales management with minimal delay. The longest acceptable time lag under normal circumstances is 24 hours. Additionally, the history of each ongoing sales case shall be retrievable, and all closed sales cases shall be stored in the local ERP system for evaluation and comparison purposes for a defined period.

- Comparability and IT-suitability

The sales cases of all sales offices and thus, out of all ERP systems shall be comparable as the status information indicates an identical, standardized progress step in the process. Therefore, the set of information collected out of each ERP system must at least contain an identical core. Differences of additional information are to be discussed, weighting advantages of more information against increased costs and complexity.

- Support of Cooperation

The interlinked, cooperative work in generating offers and handling orders between POS and production units is supported only locally by IT systems. It is only supported if taking place inside the same ERP, but not across the worldwide organization.

- Reducing Possibility of Errors

The manual entry of data in different locations produces inconsistent data across different ERP installations and non-matching details on processes and data between ERP systems in the POS and the according production unit. The data has to be the same in all involved ERP systems, and the status of the workflow has to be synchronized between different ERP installations.

- Reducing Administrative Efforts

The multiple manual entries of data and multiple verification of this data in different POS and production units has to be replaced by an electronic data transfer to reduce manual work on process steps, which creates no additional value.

- Supporting Market Structure

Different customers may sell into the same project of one final customer. Therefore a possibility to identify projects from final customers is requested.

- High Efficiency of Monitoring

The monitoring and reporting activities shall be based on a single interface for the user, therefore it needs to be supported by a central IT system that contains all information required to fulfill this task.

5.6.2. Specific Goals

To realize the general goals for the improvements on support for the global sales process and reporting, the following specific goals for extended features and functions in each of the existing ERP systems are defined.

- Coverage of Sales Process in ERP System

The ERP system Navision will be the leading system for all work and documents related to the sales process and will cover all sub-processes. No additional work or documents outside this operative platform will be used for commercially relevant communication from the POS to the customer. The support of the process starts with the entry of the information from the initial customer request and covers all internal and external used documents.

- Collection of Additional Data

All data where a report is requested on will be collected inside the ERP system, which will be the single point of data entry. No additional data collection outside of the ERP system will be used for commercial reporting. Therefore, more detailed and additional information about the technical parameters and the commercial figures of the products are requested in the operative platform. The support of the data entry starts with the information from the initial customer request. Additional mandatory information will added for each sub-process to covers all internal used figures.

- Data Collection during Operational Work

To collect the data for the extended sales and process reporting on each step of the sales process it is necessary to do this automatically inside the ERP systems during operational work. As the data is used during the workflow from one sub-process to the other, the data will be entered into the operative ERP system as a side effect. No additional user action will be requested to receive the information for the sales reporting.

- Consistent Information

For any sales case, which is represented in more than one ERP system, each of the local collected reporting data needs to show consistent information about the progress in the sub-process.

- Automatic Reporting

All data and information that are collected in one of the local ERP systems will be transferred to the central sales reporting automatically. No additional user interaction will be necessary to prepare a local report or rework on the local data.

5.6.3. Administrative Rules

The given situation shows existing administrative rules concerning the sales process, which are not fully supported or implemented by the existing ERP systems. To enforce the following rules influences the implementation project.

- Complete Documentation of Sales Cases

All relevant steps of the sales process must be represented in the local ERP system. This means that for each customer order an offer must exist and for each offer, an inquiry must exist in the local ERP system. Only the order handling is implemented and used in every ERP system. Where the possibility for inquiry or offer handling in the ERP systems is implemented, there is no control in place, if the document of the predecessor process step was created. After the implementation, each ERP system allows the creation of a document for a process step only based on the document of the predecessor document.

- Article Master Data

Articles have always been managed individually. Because of the direct connection to production planning, the creating system is the leading system having the data sovereignty for the respective articles. For group reporting purpose a centrally defined set of article groups is used. Each of the articles needs to be member of one of the listed groups. After the implementation, each ERP system will stay with the own article master data, but the connection to the centrally defined set of article groups will be implemented as mandatory data for the articles.

- Customer Master Data

Customer Data have always been managed individually. This results in different data entries in different ERP systems for the same external customer. For each customer a leading ERP system is defined by the responsibilities as POS. The data in the ERP system of the POS is always the reference for customer data. This administrative rule will not be technically implemented in the ERP systems. After the implementation, each ERP system will stay with its own customer master data.

5.7. Solution Design

After the analysis of the existing situation about the support from IT systems and the definition of the goals of the project, the following conceptual design for the solution was developed.

This complete solution design includes the use of a new central sales reporting database, but the reporting database is not part of any of the ERP systems. This additional reporting tool is implemented outside of the ERP solution, since it is not used for the operational workflow. The reporting database collects all data, which were created during the work. This system pulls all relevant data from all involved ERP instances without any additional involvement of the user and provides an overall view on the sales process.

The setup and planning for the reporting tool is not in the content of the investigated implementation project, but to deliver the requested information the following additions to the ERP system Navision are required.

5.7.1. Business Case

To enable the work on a sales case that is processed in different ERP systems and different organizational units a new data model called Business Case was developed. The business case identification is used as a unique identifier between POS and production unit. Furthermore, it builds the container for all connected steps and documents and delivers the possibility to track them through all organizations. A new instance of a Business Case is created in the POS, where the initial customer request is collected. All further work on the documents for this sales case are only possible inside this new instance and therefore always related to it.

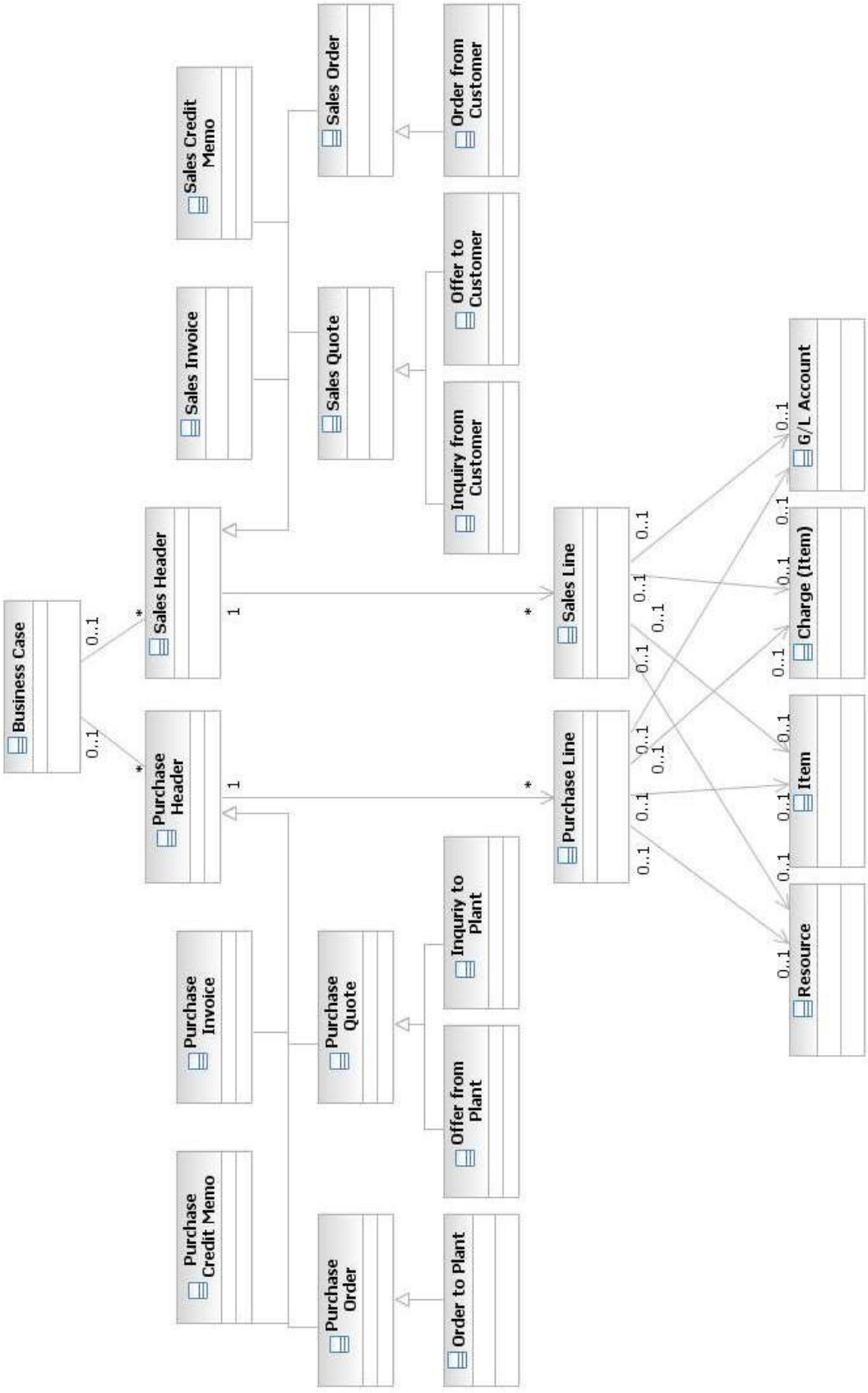

Figure 5.7.1-1 Part of Class Diagram for Business Case

In addition, the Business Case container is used to collect market related aspects about the project of the final customer. The collected project information helps to find similar requests in different POS to enable the track of sales opportunities. With the transfer of documents between the involved sites, the business case information is transmitted together with the document and the ERP system of the production unit shows the same information about the project like the ERP system of the POS. The different users of both involved companies have now a common reference and common knowledge about the customer project. This common knowledge supports the coordinated work on a distributed sales case.

Furthermore, the business case provides the additional possibility to collect information in case of lost projects. If a business case was not successful because the customer does not place the order, the ERP system provides the possibility to enter information received from the customer like the competitor, who got the project and what was the reason to lose the project.

Beside the use of this new object during the operational work inside of the ERP, the business case object includes a link to a certain folder on a fileserver outside of the ERP. In the referenced folder on the fileserver, related documentation like customer drawings or technical specifications can be stored. This common file storage enables the share of information between POS and all involved production units.

Figure 5.7.1-1 shows the fraction of the class diagram with the new Business Case element and the purchase and sales documents.

5.7.2. Additional Documents

Since the existing standard documents for sales and purchase only cover quotes and orders, the set of documents was extended to cover also inquiries and therefore realize the support of all steps of the sales process. The additional inquiry forms will cover the initial sub-process and create a possibility to enter the data of the initial customer request. Based on the existing data model the standard documents for sales and purchases quotes are used to design the needed documents for inquiry and offer. The purchase quote is extended to build the documents for the inquiry to plant and the offer from plant. The sales quote is extended to build the documents for the inquiry from customer and the offer to customer.

To complete the set of documents, the standard documents for sales and purchase orders are used to design the order documents, where the sales order is extended for the order from the customer and the purchase order is extended for the order from customer. With these additional documents in place, it is possible to cover all sub-processes inside the ERP system.

The order confirmation was not implemented as an additional document, since it is more related to the data inside the order from customer. Nevertheless, this sub-process was covered by the implementation of additional mandatory data.

Figure 5.7.1-1 shows the fraction of the class diagram with the new Business Case element, the standard purchase, the standard sales documents and the extended set of documents.

5.7.3. Control on Documents

In the Navision standard solution, the creation of new orders or offers is not connected with special controls. It is possible to use an offer to create an order, but there is no obligation to use the predecessor document.

To ensure that the workflow is followed, the creation of a new document is only possible in consistency with the administrative rules. As mentioned above, the new container business case builds the frame for each sales case. Only inside of this business case, the work with the sales and purchase documents is enabled. To start the workflow, it is possible to create the document inquiry from customer for make-to-order production, or to create the document inquiry to plant for make-to-stock production. Starting with these two documents, the control on the document creations provides only functions to instance the document for the successor process step. This enables the control of the workflow and the connection and reference between the initial document and all successor documents.

5.7.4. Additional Data

To cover all the data entries for the requested figures, the sales and purchase line was extended for additional fields. According to the progress in the sales process, it was defined which of these data is mandatory for which document. This enables the control of the data in connection to the proper use of it.

One of these additional data was used to implement the order confirmation. Because of the controlled workflow and the collected information regarding the connection between the documents, the order confirmation is just the delivery of one data, the confirmed delivery date. When the production unit knows the delivery date of the products, this data will be added in the order from customer at the site of the production unit and transferred to the POS. In the POS, this data is added to the order to the plant and to the related order from customer. For the users of the POS it is not possible to enter this confirmed delivery date manually into the documents. Therefore, this transfer of data replaces the use of an additional document for the order confirmation.

5.7.5. Data Transfer

The communication between the systems is requested to forward all collected information between the different locations. Therefore, each Navision installation was extended to export and import XML-Data. The technical availability to use XML ports is included in the standard Navision functions, but the definition of the data is a custom extension. The use of this structured set enables the transfer of the documents, like inquiries, offers and orders between the systems. Because of the legal and tax issues regarding electronic invoicing, the invoices and credit memo documents are not included in the implementation of the interface. The XML data transfer distributes the information of the business case together with the documents.

The transferred data includes the codes and references of each of the ERP systems for documents, customers and articles. This publishes the information from one data store to the other and delivers all referencing information from any involved remote ERP system to the users. As explained with the administrative rules for the solution, each instance of the ERP solution will stay with its own set of article data and codes and with their own set of customer data and codes. This transfer of reference information brings now the possibility to connect the different customers and articles based on their use during operational work in the sales process. Over time, the collected data in the central reporting can be used to build up translation tables between the codes of the different systems.

Beside the operational data transfer, the XML interface is used for the distribution of some central setup configurations. This includes especially the product groups, which

are used for the central reporting. One company is defined, where the central setup is possible. Only in this single company, the work on the setup data is allowed. Afterwards this setup data is distributed to the other instances of the ERP system with the XML interface.

5.7.6. Status Information

It was necessary to implement additional status information on the new object business case as on each of the used sales documents to enable the coordination of the documents between different instances of the ERP system. Furthermore, it was important to define dedicated points in time to collect data for sales reporting and process monitoring.

5.7.6.1. Business Case Status

The status on the business case is steered by the use of the different documents inside the business case. Depending on the used document, the business case instance changes from on status to the other. This behavior is shown as state machine diagram in Figure 5.7.6-1 for a business case in a POS.

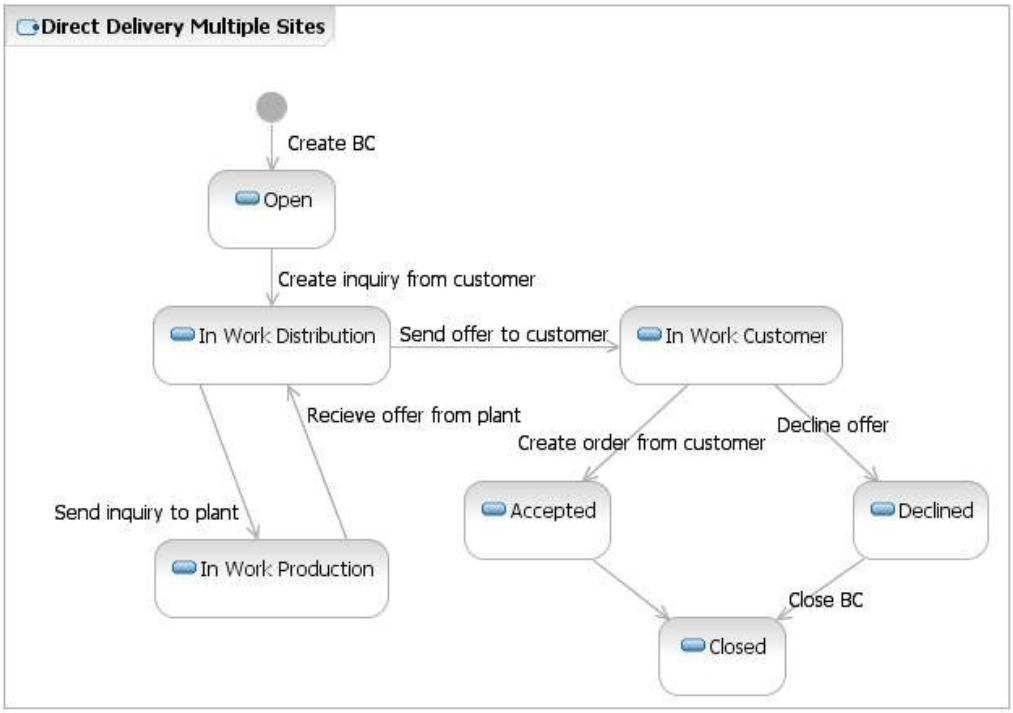

Figure 5.7.6-1 State Machine Diagram for Business Case

Possible statuses for the business case object are the following:

- Open: After the creation of a new instance
- In work Distribution: After creating the document inquiry from customer
- In work Production: After sending the document inquiry to plant
- In work Distribution: After receiving the document offer from plant
- In work Customer: After sending the document offer to customer
- Accepted: If the customer places the order, and the document order from customer was created.
- Declined: If the customer rejects the offer and does not place the order.
- Closed: If a declined case will be finally closed, or for an accepted business case, the order is completed and all articles are delivered and invoiced.

5.7.6.2. Document Status

Similar to the overall status of the business case object, the status on the document level is related to the progress of the work. However, the change from one status to the other for the documents is initiated only manually by the user. The possible actions and changes in the data of the documents depend on the actual status of the document. In addition, the change of the status starts a procedure to verify the mandatory data in the document. Only if the check of the data is successful, the change of the status is fulfilled. This status change triggers at the same time the collection of information for reporting. Finally, based on the correct document status the system allows the transfer of the document to a remote ERP system.

This behavior is shown exemplarily as state machine diagram for the offer to customer in Figure 5.7.6-2.

Possible statuses for the offer to customer object are the following:

- Open: Based on the data from an offer from plant, located in the same business case, the new offer to customer is created with the status open. This is the status, where the user from customer service is able to work in the document. After the changes in the document are completed, the user initiates the change of the status. This request for status change starts the verification of the mandatory data. If the required data is correctly filled in the offer to customer, the status change is completed successfully.

- Released: In this status, the content of the offer to customer is verified as correct by the user and the system. Therefore, the document enables the function to print the information. The printout of the document is not allowed in predecessor status open, since the data inside was not verified. The printout triggers the collection of the reporting data and changes the status to transferred.
- Transferred: In this status, the offer to customer allows the conversion to an order from customer. The conversion to an order is not allowed in any of the predecessor status, since the offer to the customer was not transferred to the customer. The conversion triggers the final data collection for sales reporting.

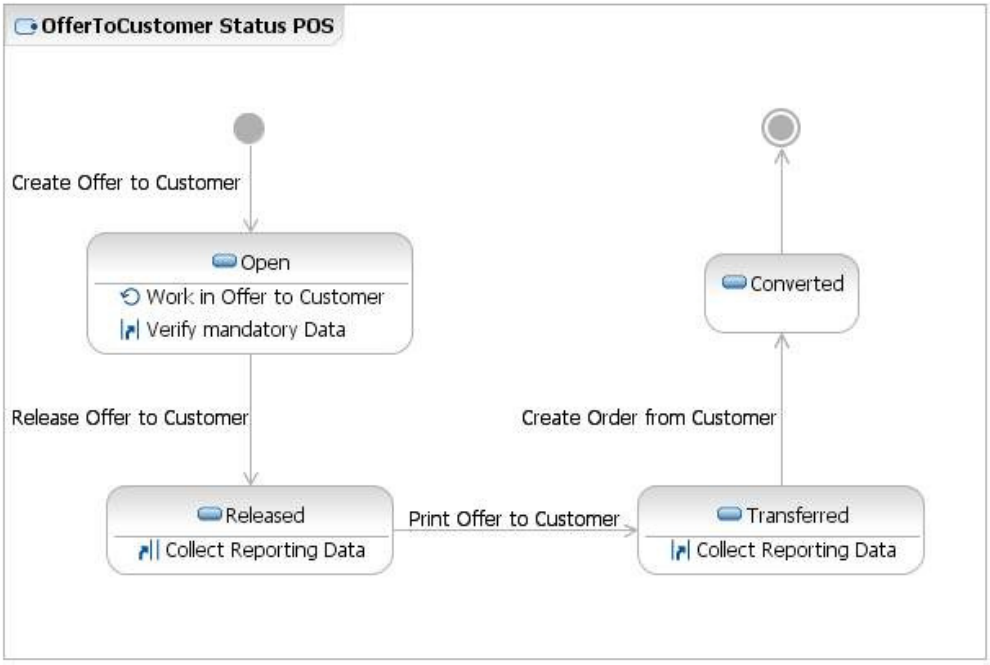

Figure 5.7.6-2 State Machine Diagram for Offer to Customer

To support the distributed work on one business case in different locations, the XML interface is used to deliver feedback on status changes from one ERP system back to the other involved ERP system. Therefore, also the users from the remote site are able to follow the track of the ongoing progress on the work of the document. This increases the transparency of the sales process.

5.7.7. Collected Reporting Data

For the central sales reporting it is crucial to receive data from each of the ERP instances and to combine this into overall reports and views. Therefore, the collected information needs to be suitable for the common data model. The above-mentioned

additional data is used as a source of information. In addition, the described control on the status changes is triggering this collection of data.

If one of the triggers initiates the process to collect the reporting data, the procedure creates a snapshot of the observed document. The information of the header and the lines are combined together and stored as data set for each line. This ensures the collection of complete information for each line in a document. It creates for one document with many lines redundant data, since the header information always shows the same customer.

Nevertheless, this mechanism builds a solid base for all further reporting. Based on this collected data, the central reporting database is able to calculate differences, between different versions of the document to calculate changes for a certain period. The collected snapshots also deliver verifiable data for views on a specific valuation date. Furthermore, all these single pictures can be tied together to support the monitoring and optimization of the process.

6. Research Questions and Evaluation Concept

To evaluate the approach of requirements modeling with use cases, a case study was performed to compare this way of documentation to the traditional approach, which was used to implement the first pilot installation.

For each existing installation of the ERP system, one local Navision Solution Centers is responsible as supplier for the support and development. These multiple external partner companies need also to take care to implement the new workflow, in coexistence with the already used functions, data and legal needs. The initial documentation was developed together with the users for the Austrian instance of the Navision ERP system.

The target is to investigate the necessary additional effort to explain the defined requirements to the different audiences. The users of the system need to understand the process flow during their daily work. They need to be aware how the work on the single process step will be changed with the additional features. In addition, the developers from the external partners need to understand the requirements to implement the additional functions in the different local instances. Beside the effort to explain the existing requirements, the investigation focus on the additional implementation of the sub process for changing sales orders after the initial order intake.

6.1. Solution Criteria

The case study will investigate the impact of requirements modeling with use cases and the additional use of selected UML notations on the implementation and change process. It is assumed that the introduction of use cases requirement modeling creates smaller packages, which are well connected to the different sub-process steps of the overall global sales process. This increases the traceability between business process and requirements documentation. The following criteria are formulated to analyze the approach.

- Improvement of the understandability of distributed sales processes
- Effort needed for users to identify the business process in the use case requirement model documentation

- Effort needed to update use case requirement model for change requests in consistency with the initial project goals
- Improvement of the understandability of the additional features implemented in the ERP system for the developing partners
- Time needed to solve evaluation tasks in context with distributed sales process implementation
- Reduction of misunderstandings due to different domain knowledge and languages

6.2. Research Questions

In the context of the process, characteristics and criteria mentioned above, the following research questions arise.

- To what extent can a use case requirement model reduce effort and time which is necessary to transfer sufficient knowledge about implementation requirements?

The transfer of sufficient knowledge is a major factor for successfully solving a distributed implementation task. Time and effort are critical cost factors.

- To what extent can a use case requirement model reduce the number of errors and misunderstandings about implementation requirements?

Especially in the start up phase of the implementation, errors and misunderstandings lead to higher costs and time effort and increase the risk of missing the project goals.

- Which impact has a use case requirement model on solving change requests?

The avoidance of errors during implementation of change requests due to an unambiguous documentation and improved understanding prevents inconstancies between different local instances.

- To what extent can a use case requirement model be used to implement requirements tracing?

Requirements tracing enables a connection between requirements through a number of various artifacts. The representation of the tracings in suitable form plays an important role to ensure high quality software development.

7. Case Study

Since the whole project covers the implementation for six different instances of the ERP system Navision, the installation in the Austrian database was selected for the pilot realization. The Austrian environment is used for the operative work of two companies, where both roles as POS and production unit are processed. This installation shows all relations, which are included in the distributed work on the sales process, but reduces the program work, because of the use of the same database. Therefore, this location was suitable to start the project, since it is a small model of the whole network of ERP systems. For the work on all other instances the documentation, codes and experience can be reused, after the new extension was going live for these two companies.

The analyses and definition of the requirements were accomplished with selected key users of the system together with the external partner, who is responsible for this Navision installation. At the start of the implementation project, it was defined to use the traditional approach for the documentation of the software requirements. After the definition of the first basic set of requirements, a prototyping was realized, to increase the understanding of the new functions for the involved users. With the prototype in place the requirements documentation was finalized and the external programmers implemented all requested features and functions. Based on the requirements documentation, a test plan was developed. Together with the users, the test of the new implementation was executed and finally the extension was integrated in the productive system.

This existing set of requirements documentation is the starting point of the analyses for this case study. The case study will compare the traditional approach with the use case requirements model in detail on an extension of the initial functionality. This extension implements an additional sub-process for changes in a customer order after the initial order intake was completed.

7.1. Analyzes of Requirements Documents

A set of the documents was written to describe the extensions of the ERP system. The main part of the documentation was realized in plain text. For different elements of the planned solution, a single document was used to describe the requested changes and

extension for this element. For each of the documents for the software requirements, the following template for the document structure was utilized.

7.1.1. Document Structure

The used template is divided into six main parts to structure the description of the required changes for the ERP system. The components of the template are the following.

- Target: What should be achieved by the IT changes?

The first paragraph is used to describe the specific goals, which will be realized with the described features und functions. With the description of the goals, the scope of the document is defined.

- Processes: Which changes of workflow cause changes of the IT environment? Who is responsible for each task?

The section for the process description defines the lay out of the workflow and the interaction of the users on the data inside the system. The processes are related to the scope of the document only.

- Masks and Reports: Do the changes of processing cause changes in the user masks or are further reports used?

The extension on the functions and data of Navision requests in addition an adjustment on the user interface and on the reports in Navision. This segment for masks and reports lists all changes on the user interface and the changes on the reports, to enter and display the extended set of data.

- Functions: Which new data connections, analysis, etc. are necessary to support masks and reports?

This part of the document is mainly filled from the implementation partner and gives information on how a solution may be implemented or has to be implemented. This paragraph shows most of the functional requirements.

- Data: Which changes on the data model are requested to support the changes to the functions?

The section for the data changes summarized all requests for extended, changed or additional data in the data model of Navision.

- Administrative Information

Some additional information was summarized in an administrative segment, to show the name of document, date of last changes and the name of analyst, who developed the document. For the further use on the project planning, this section includes also information about the implementation priority, the planned duration of the implementation, desired implementation term and costs. Finally, the document closes with the acceptance of the project members and the responsible manager.

7.1.2. Set of Documents

During the requirements definition for the project, the documentation was divided into several documents. Each of the documents provides the requirements for a fraction of the whole implementation. The complete set of documents consists of the following.

- Sales Process Requirements

This is the central document of the requirements definition and includes the documentation on the new data model of the businesses case and the description of each of the sub-process of the sales process. It explains the data structure and function for the business case. Furthermore, the work inside the business case on each of the different documents in relation to the sub-process is explained in detail.

- Sales Process Details

The scope for the document was an extension for the central document for the sales process requirements. The goal for this document was the delivery of a more detailed view on the data involved in the sub-processes. Therefore, it describes all data fields, the standard fields and the extended information, in relation to the flow in the process step. This includes the details, on which step of the sales processes the data will be collected and what is the behavior of it in the following steps of the workflow. Will the data change during the process or will it stay unchanged? For the further use of the documentation, this detailed description on the data field was further extended to define possible test data and explain expected test results.

- Sales Reporting Requirements

The document for the sales reporting covers all requirements, which are related to the additional data collection for the central sales reporting. As mentioned in the solution design the data is systematically collected in the ERP system during the work on the

documents. The standard features in Navision do not cover this automatic creation of historical data. Therefore, this function was defined as customized extension with the possibility to configure the dependencies on the document, status and data information, which triggers the event of creating a historical copy of the document.

- Inquiry Data Requirements

The requirements document for the inquiry data describes an extension related to the first step of the sales process, where the customer sends the initial request to the POS. The ERP system is also used for the support on the collection of the necessary data from the customer. The system provides the users a list of parameters, which are either mandatory or additional information. This assistance is configurable in one central ERP system and distributed with the XML interface.

- Collection Closing Reasons Requirements

This requirements document covers features to collect information in case of lost projects. If a business case is set to the status closed by a user, the ERP system provides the possibility to enter information about the reason of the loss of the project and the competitor, who got the project. The used categories are configurable in one central ERP system and distributed with the XML interface.

The entire set of requirements document was developed in an iterative and incremental process. After the initial collection and the development of the first draft, the status of documentation was used as input to create the prototype of the new system. The prototype was then used during the further sessions, to increase the understanding on the new features and functions for the users. The final set of the documentation includes the feedback on the experience with the prototype from users and implementers.

With the requirements in place, the implementation in the Austrian database was completed, tested and finally set productive in the real database for the two Austrian companies. The existing set of requirements document was also delivered to the others external partner companies, which are in charge for the different international distributed Navision installations. During the operational work with the complete sales process in the ERP system, new requests for changes or additional features come up from the users. Moreover, with the work on the planning of the international

implementation, the Navision partners raised additional questions, which are not fully covered from the original documentation.

7.2. Use Case Requirements Model

Starting from the existing set of documents, additional iterations for analyzing the solution are initiated to build the requirements based on the object-oriented use cases. At first, the following template for the use case description was defined.

7.2.1. Use Case Template

The use case template delivers a detailed structure for the description of each use case. The template shows all sections, which are maybe necessary to fill for a use case. Therefore, it is possible for some smaller use cases, that not each of the paragraphs will be filled. For the iterative process of defining the content of the use case, the components are ordered by their use during the definition process.

The parts of the template for the initial use case description are the following.

- Use Case name and unique ID: where the ID is important for the follow up, since the name can slightly change during the work on the description.
- Purpose Overview and Scope: shows a briefly description of the use case itself, the purpose and the scope of it.
- Primary Actor and Secondary Actors: the primary actor is responsible for the initiation of the use case, and the secondary actors are all other involved users or systems.

These initial sections on the use description are used to create the initial version for each single use case. This detail on information is sufficient to build the initial version of the use case diagram. The use case diagram will then further develop with the more detailed specification in each use case.

The following paragraphs of the template are utilized for the base use case description.

- Pre-Conditions: to describe the pre-conditions and system states before the flow of events in the use case are executed.
- Trigger: defines the event, which starts the use case.
- Main Success Scenario: this component is the main description of the flow of event in the use case. For each step the action from the actor and the related action from the system will be defined

- Included and Extended Use Cases: displays the reference to other use cases, which are either included or extended by the defined use case.
- Variation, Alternate Success Scenario: this part is only used, if a variation on single steps of the main success scenario or a complete alternate success scenario exists. It shows similar to the main success scenario the action from the actor and the related action from the system.
- Unsuccessful Scenario: this section is the description of the flow of event in the case of an unsuccessful end of the scenario. It shows similar to the main success scenario the action from the actor and the related action from the system. This paragraph is not filled for each use case, but to include it in the template forces the investigation on possible unsuccessful scenarios.
- Post Conditions and Success End Condition: describes the post and success end condition and system states after the flow of events in the success scenario are executed.
- Assumptions: if the description of the events is based on any assumptions, the documentation of these assumptions is mandatory. Otherwise, the use case model may show inconstancies.

These base descriptions for the use cases are the input for improvements on the use case diagram. For some use cases, it became evident during the work on the flow of events that they are to complex for one single use case. To reduce the complexity inside a use case, the flow of events can be divided into separate use cases. For other use cases, the short description of events shows obviously, that the merge of two use cases should be considered. In this phase the relation between the use cases are analyzed and possible include and extend associations are defined.

The following parts of the template are used for the elaborated use case description.

- Stakeholders and Interests: describes the interests of stakeholders of the system, which are no actors. These interests can be also business rules related to the flow of events.
- Frequency: shows an estimate the frequency of execution. This number is an additional input for the analyses of change requests.
- Hierarchy Level: to indicate the hierarchy level of the use case model, to which this use case is related to

- Failure end condition: defines the failure end condition and system states after the flow of events in the unsuccessful scenario are executed.
- Activity Diagram or Sequence Diagram: if necessary for better understanding on the flow of events, the description can include these UML diagrams.
- Special Requirements: provide the possibility to describe other issues, which are not covered from the above sections like performance issues, special security requests or user interface requirements.
- Other artifacts: show references to other documents. This paragraph will be utilized for further reuse of the use case in test plans or user guides.

The elaborated use case description is available, after most of the components of the template are completed. This provides a well-defined view on each of the use cases.

For all steps during the work on the description, an additional common section is available for notes and follow up related to the use case.

- Issues: collect all open questions related to the definition of the use case. During the work on the definition of the use case, any issue that needs to be clarified will be collected in this paragraph. In the finally released version oft the document, this section must be empty. Otherwise, the document cannot be considered as finished.
- To-do: needs to collect any follow-ups that remains to be done on this use case. The explanation for the to-do needs to include the responsible person and a planed due date.

With the template document for the use case description in hand, the work on the use case model started with analyzes of the existing requirements documentation.

7.2.2. Use Case Model

The existing documentation was reviewed and used as source to develop a use case requirements model. The uses cases are defined in strong correlation to the sales process. This simplifies the tracing between the business processes and the use cases. The textual description of the use cases was incrementally developed based on the template through iterative loops of reviews on the documentation.

The use case model was divided into different fragments to show the relation to the hierarchy of the work process and to the membership to different sub-systems.

The full requirements model describes the interaction of users during a sales case in different instances of the ERP system Navision. Therefore, the model includes requirements on the installation for the POS and the installation for the production unit as separated sub-systems. In addition, the work in each of these ERP databases is ordered in a hierarchical way.

The top level includes all use cases, which are related to the work on the new data structure of the business case. This level describes all manipulation of business cases. This component is an entire customized extension of the standard ERP functions. The use case diagram for the business case management in the POS and the production unit is shown in Figure 7.2.2-1.

The actors, which represent the users of the ERP in the POS, are the customer, the sales manager, and the customer service from the POS. These actors initiate the events in the use cases for the handling of the business case, like creating a new instance for a new customer project. Only the sub-system for the BC management in the POS steers the work on this top-level.

The actors for the file storage, the XML interface and the sales reporting represent technical systems, which are interacting with the ERP. The file storage and sales reporting are only recipients of information from the BC management. In contrast, the XML interface builds the active connection between the involved ERP instances.

The BC management in the production unit is controlled from the XML interface, and therefore dependent from the actions of the users in the POS. The creation of a new indirect BC is only possible, if the information about a new BC is transferred from the POS via the XML interface.

The other actors of the system in the production unit represent the users of the ERP, who fulfill the role of the customer service from the production unit and the production calculation. The diagram indicates the responsibilities for instances of a BC. The manipulation on the BC as container structure is only possible for actors in the POS. In the sub-system of the production unit, the same BC is only displayed with the according information. The actors are able to work in the documents inside the BC, but they are not able to work on the BC itself. The BC and the related status are always controlled from the POS and the action of the users in the ERP of the POS.

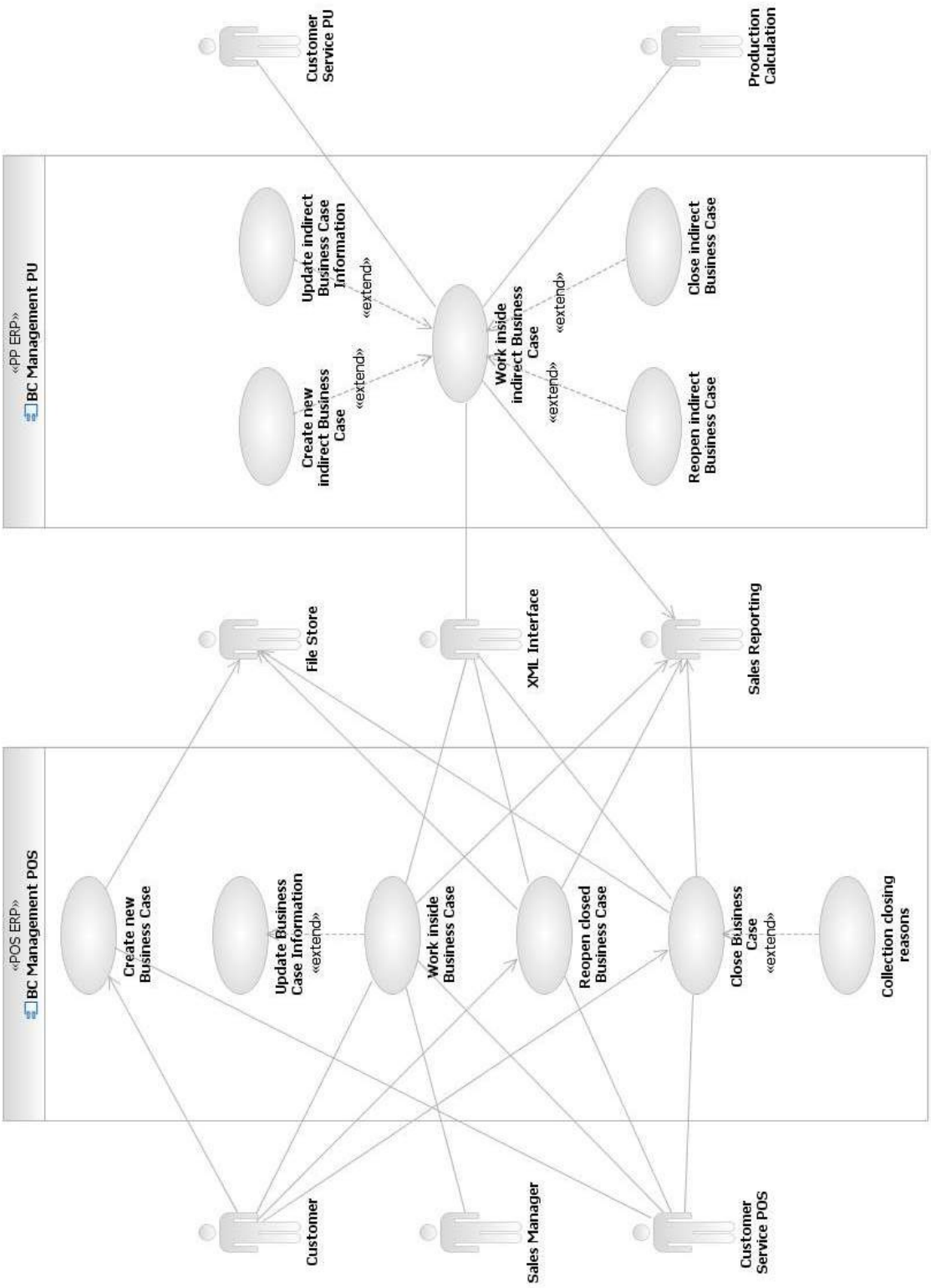

Figure 7.2.2-1 Use Case Diagram for Business Case Management

On the top-level, the work in the different documents of the sales process are summarized in the use cases "work inside BC" in the POS sub-system, and in the use case "work inside indirect BC" in the production unit sub-system.

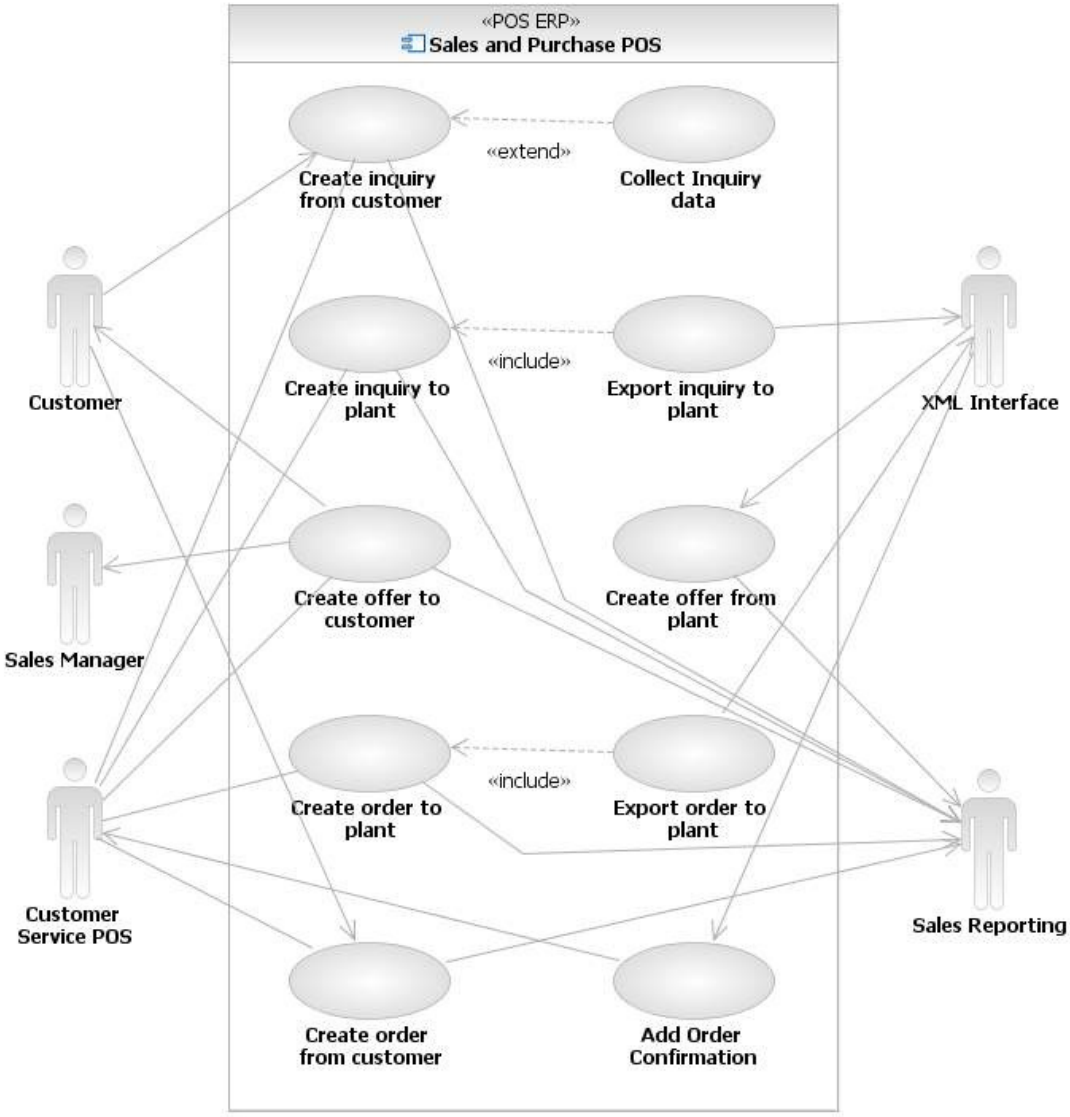

Figure 7.2.2-2 Detailed Use Cases in POS

The lower level of the document handling represents the work inside one instance of a BC. On this level the actors are in the sub-system "sales and purchase" of the ERP system from the POS or the production unit. This component is an extension of the existing sales and purchase granules from Navision, where existing documents are reused to implement the new functions and data. For the further explanation, the sub-system "sales and purchase" in the POS is described exemplarily. Figure 7.2.2-2 shows the according use case diagram.

All events of any use cases are only executable inside an existing BC instance. The different steps of the workflow during the progress are defined as single use case. The diagram includes the same actors as the top level, but without the file store, since this is only related to the BC structure, but not to the one of the documents inside this object. Furthermore, the timely order in the sales process is indicated in the order of the use cases from top to down.

The top-level model shows also some included and extended use cases. The use case "collect inquiry data" has an extend relationship, because the request of detailed technical information on the article is not mandatory. The initiation for the request on this data depends on the product group. It is not reasonable for all product groups to fill the additional form.

The combination of the both levels of the diagram is shown in Figure 7.2.2-3. It shows the relation between the use cases of the two sub-systems in the ERP from the POS.

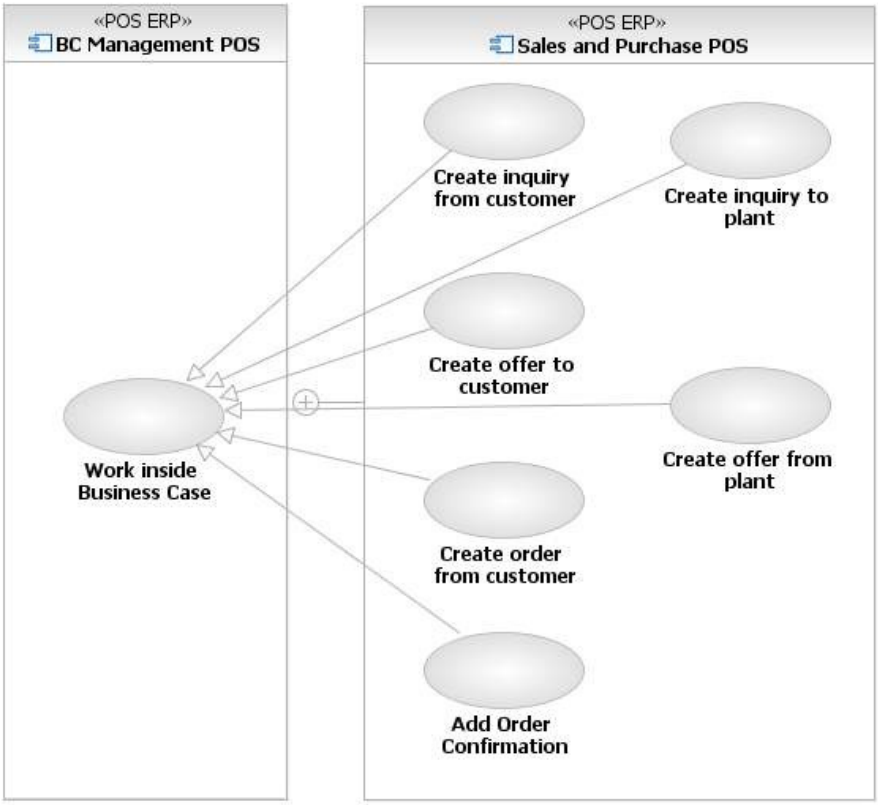

Figure 7.2.2-3 Generalization of Uses cases in POS

All the use case diagrams indicate already an important difference between the traditional requirements documentation and the use case model. The number of documents is increasing. For each use case, a single document will describe the details. To compare the differences in detail, the documents are investigated for an extension of the sales process.

7.3. Additional Sub-Process: Change Order

After the initial phase of the implementation project and the go-live in the first two companies in Austria, some change requests have been initiated. In the first months of operative use of the extended ERP systems, it became evident that one process step was missing in the initial process definition. Therefore, this sub-process was not supported by the new functions and features. This additional requirement was the change of a booked and confirmed order from customer.

At the start of the original analyses, the assumption was that changes to confirmed orders do not happen frequently. Therefore, the integration into the IT systems is not cost-efficient. Because of that, any further definition of details and procedures was not initiated.

As the experience with the extended ERP system shows, the assumption was wrong and the requirements definition started again for this change request. This part of requirements documentation will not be compared with the use case requirement model in detail. At first, the business requirements are defined.

7.3.1. Business Process Change Order

The business background of this change request is the long lifetime of sales orders. The production time depends on the type of article. Producing the article takes on average 8 to 10 weeks. Furthermore, the load of the production plants adds additional lead-time, since the production is already filled with other orders. Finally, the delivery of the goods can take additional four to six weeks, since the final place for the use of the products can be anywhere in the world. These circumstances together can add up the time between the purchase order from the customer and the availability of the articles at the final destination from 25 to 32 weeks. Because of this long lead-time, it happens frequently that the customer changes the article quantity or details of the technical specifications.

These changes happen inside of a booked order, which is already entered in the ERP data stores of the involved sites and are now strongly correlated with the extended features and controls. Therefore, it is important to implement this sub-process of order change similarly to the original sales process for the order intake. The changed IT support needs to enable the change of quantities in the different documents in the involved ERP systems.

7.3.2. Reporting Requirements

Beside the operational component of the order changes, the reporting factor creates additional requirements.

The main commercial figures for sales reporting are order book, orders received, and sales. The definition for these numbers is the following:

- Order book is the total sum of the value of all confirmed orders at a certain reporting date. The standard reporting date is the last day of the month.
- Orders received are the total sum of changes on the value from all confirmed orders for a period, where new orders or increases in value are counted positive and cancelations or decreases in value are counted negative. The standard reporting period covers one month from first to the last day of the month.
- Sales are the total sum of turnover for a period, where invoices are counted positive and credit memos are counted negative. The standard reporting period covers one month from first to the last date of the month.
- Confirmed orders include all lines in an order, where delivery date is confirmed from the production unit.

For sales and orders received, the period can be extended to year to date, where the amounts are summarized from the first day of the year to the last day of the reporting month. Based on the definition, the year to date number can be calculated as total of the monthly numbers.

Furthermore, the reporting has to calculate the same reporting figures on the level of the customer, for each reported product group and for the production unit. The customer is always directly connected to the document, since an order or invoice is for one customer only. The information about the product group and the according

production unit for the article is related to the line in the document only. Each line in the document may show different information on these two reporting dimensions.

Between the reported numbers, the following correlation exists:

$OB_{am} = OB_{lm} + OR_{am} - S_{am}$

- OB_{am} = Order Book actual month
- OB_{lm} = Order Book last month
- OR_{am} = Orders Received actual month
- S_{am} = Sales actual month

The formula shows, that the order book for the actual month can be calculated. It is the order book from the last month, increased by all new orders received in the actual month and decreased by all sales from the actual month. Therefore, if all numbers included in the formula are reported and calculated independently, the above-mentioned calculation needs to be true.

All these reporting requirements together force a close track on the value of orders. Especially, since the orders stay for many months in the ERP systems and therefore are included in many monthly reports.

In addition, the definitions for the confirmed order request a close synchronization of the status between the POS and the production unit. If an order is confirmed in the production unit, but not in the POS, it is not possible to find the correct consolidation for the reporting figures.

7.4. Additional Requirements Document: Change Order

To show the difference between the traditional approach and the use case requirements model, the documents for the additional process step to enable changes in an existing order are analyzed in detail. At first, the requirements description for the new sub-process is provided. The filled template for the traditional documentation shows the following content. The administrative section is not displayed, since this part is not related to any difference between the two approaches.

7.4.1. Traditional Requirements Document

- Target: What should be achieved by the IT changes?

The decrease of quantity (cancelations) and increase of quantity (negative cancelation) should be handled inside the business case and supported with the XML interface. Requests for price changes, which can become necessary for the production unit on a quantity change, should also be transferred. The cancelation of one document, done by entering the state "canceled", should lead to an automatic cancelation of all line-quantities. Thereby the same process should be used as it is done by the quantity-changes of single articles. This cancelation sets the quantities to zero.

- Processes: Which changes of workflow cause changes of the IT environment? Who is responsible for each task?

The following process steps should be modified:

1. Order from Customer

In business cases where POS and production unit are at the same site, the change of quantity starts in the order from customer. The entry of a quantity change will set the confirmation and modifies the quantity. Step 3 and 4 are not passed in this case. Previously the user has to verify the possibility of quantity changes with the production unit as production orders and their state will not be considered in this process.

2. Order to Plant - XML interface not used

In business cases where POS and production unit are on different sites and the communication between these sites is not using the XML Interface, the change of quantity starts in the order to plant. The entry of a quantity change will set the confirmation and modifies the quantity in the order to plant in the first step and afterwards, automatically, in the order from customer. Step 3 and 4 are not passed in this case. Previously the user has to verify the possibility of quantity change with the production unit.

3. Order to Plant - XML interface used

In Business Cases where POS and production unit are on different sites and the communication between these sites is using the XML interface, the change of quantity starts in the order to plant. Quantity changes in "order to plant" can be entered in the

column "canceled amount". After additional transfer of the "order to plant", the "order from customer" in the effected company will be modified on the entries described in step 4.)

4. Order Confirmation from Plant

The request for quantity changes in the effected "order from customer" in the production unit can be answered in a new column "confirmation column" as following:

- Confirmed with the entry "executable"
- Denied with the entry "not executable"
- Send back with the entry "executable on price correction"

The production unit has to name a new minimum article price for the quantity change. Afterwards this information is transferred to the originating "order to plant" in the POS.

Quantity changes with status "executable" and "not executable" end the communication between the POS and the production unit. The documents are completed in each company right after the data transfer. Customer service in the POS can immediately send a new order confirmation to the customer. The documents for the production can be finalized immediately in the production unit.

Quantity changes with status "executable on price correction" need additional action in the POS. The process starts again in the "order to plant" and the production unit can answer with the status set to "executable" or "not executable". Afterwards, the status in POS and production plant is according to the above-mentioned behavior.

On the import of the data in the "order to plant", the data is transferred to the "order from customer", too. The transfer at this moment is useful as changes in quantity are only valid for the "order from customer", if the production unit sends the status "executable" for the article.

- Masks and Reports: Do the changes of processing cause changes in the user masks or are further reports used?

The column "confirmation column" will be inserted in the documents "order from customer" and "order to plant". Documents, which are modified by changes of quantity, are displayed in bold (unread) in the overview.

- Functions: Which new data connections, analysis, etc. are necessary to support masks and reports?

Extension of the XML interface is necessary:

- For data transfer of the confirmation of the quantity change.
- For data transfer of the price change that can additionally occur on quantity changes.

The quantity will be modified automatically by the canceled amount, when the confirmation of the quantity change has the state "executable".

Modifications are done not only in "order to plant", but in the matching "order from customer", too.

While an article is waiting for the confirmation, the line cannot be booked.

Quantity changes can also be done on documents with state "transferred".

Extension of the view of not read documents: the view in bold has to be enlarged to show modified documents on quantity changes.

If the POS requests a quantity change by the production unit and the production unit recommends a price change this recommendation is send back to the POS. POS now has to confirm. Afterwards production unit has to confirm and changes on quantity and price will be done.

- Data: does the data model, the data amount have to be changed to meet the functions?

The following extension of the lines is necessary:

In the documents "order from customer" and "order to plant", a confirmation column has to be implemented in the line area. The following values are intended for this column: Empty, Executable, Not Executable and Executable on price change.

In the documents "order to customer" and "order to plant", a new column for the recommended price has to be implemented. The following values are intended for this column: Empty, Value of new price.

7.4.2. Use Case Document

Like the other documents, the definition for the additional sub-process covers the instances of the ERP for the POS and the production unit and describes the process crossing these system borders. For the use case model, this is now divided into two sub-systems. Therefore, two new use cases are introduced. Figure 7.4.2-1 shows the use cases for the new change order process.

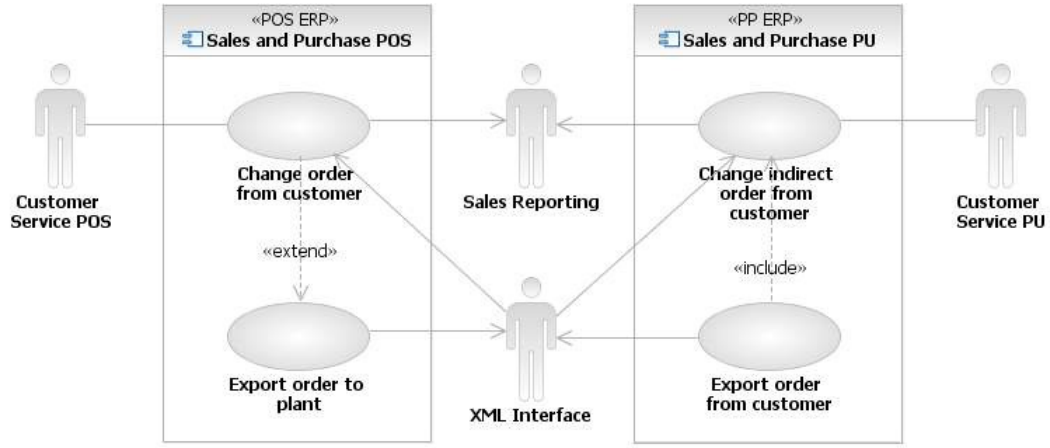

Figure 7.4.2-1 Use Case Diagram for Change Order Process

The use cases for the export of the documents are reused from the workflow of the initial order intake and are not investigated in detail. The textual description of the new use case is provided for comparison to the traditional documents. The filled template for the use case documentation shows the following content. The administrative and not utilized paragraphs are not displayed, since this part is not related to any difference between the two approaches.

7.4.2.1. Change Order from Customer

- Scope: POS
- Level: Sales and Purchase
- Primary Actor: XML Interface
- Secondary Actors: Customer Service POS, Sales Reporting
- Stakeholders and Interests: Group Sales needs synchronized data and status between POS and PU, detailed tracking of changes in existing orders.
- Pre-Conditions: The order from customer is in the status transferred or confirmed in the POS.

For Alternate Success Scenario: Customer Service POS verified successfully the possibility of quantity change with the production unit.

For Variation: Customer Service POS verified successfully the possibility of quantity change with the production unit as production orders and their state will not be considered in this process.

- Trigger: Change request from the customer was received in the POS, correction on data entry errors in the POS.
- Extended Use Cases: Export order to plant.
- Main Success Scenario: In Business Cases where POS and production unit are on different sites and the communication between these sites is using the XML Interface.

1. Customer Service POS enters quantity changes in the line of the "order to plant" in the column "canceled amount".

2. Customer Service POS initiates the export of the "order to plant" to the XML Interface. The line is marked as blocked for bookings. Sales Reporting creates a copy of the document including the modified data.

3. XML Interface imports entries from the PU in the "order to plant" and in the "order from customer". Both documents are marked bold (unread) in the overview.

a. PU denies quantity change with the entry "not executable" in the "confirmation column". The quantity in "order from customer" is not modified; the line is marked as unblocked for bookings. Sales Reporting creates a copy of the document including the modified data

b. PU confirms quantity change with the entry "executable" in the "confirmation column". The quantity is modified in "order to plant" and "order from customer"; the line is marked as unblocked for bookings. Sales Reporting creates a copy of the document including the modified data. Customer service POS prints the paper document order confirmation for the customer.

c. PU requests a new price and set entry to "executable on price correction" in the "confirmation column".

    I. Customer Service POS confirms quantity and price change. XML Interface exports the "order to plant". The quantity is modified in "order to plant" and "order from customer"; the line is marked as unblocked for bookings.

Sales Reporting creates a copy of the document including the modified data. Customer service POS prints the paper document order confirmation for the customer.

II. Customer Service POS denies quantity and price. The quantity in "order from customer" is not modified; the line is marked as unblocked for bookings. Sales Reporting creates a copy of the document including the modified data

- Alternate Success Scenario: In business cases where POS and production unit are on different sites and the communication between these sites is not using the XML Interface.

1. Customer Service POS changes the quantity in the order to plant. The entry of a quantity changes will set the confirmation and modifies the quantity in the "order to plant" and in the "order from customer". Sales Reporting creates a copy of the "order to plant" and the "order from customer" document including the modified data.

- Variations: In business cases where POS and production unit are at the same site.

1. Customer Service POS changes the quantity in the order from customer. The entry of a quantity change will set the confirmation and modifies the quantity. Sales Reporting creates a copy of the changed data.

- Post Conditions, Success end condition:

1. Received entry "not executable": Quantity is not modified.
2. Received entry "executable": Quantity is modified

7.4.2.2. Change Indirect Order from Customer

- Scope: Production unit
- Level: Sales and Purchase
- Primary Actor: Customer Service PU
- Secondary Actors: Sales Reporting, XML Interface
- Stakeholders and Interests: Group Sales needs synchronized data and status between POS and PU, detailed tracking of modifications in existing orders.
- Pre-Conditions: The order from customer is in any state in the production unit.
- Trigger: Receive change request from the POS with XML Interface.
- Included Use Cases: Export order from customer.

- Main Success Scenario: In Business Cases where POS and production unit are on different sites and the communication between these sites is using the XML Interface, the change of quantity starts in the order to plant of the POS.

1. XML Interface imports quantity changes in the column "canceled amount" in the "order from customer". The document is marked bold (unread) in the overview. Sales Reporting creates a copy of the document including the modified data

2. Customer Service PU verifies the possibility of quantity change with the production unit. The answer is entered in the system

a. Customer Service PU confirms quantity change with the entry "executable" in the "confirmation column".

b. Customer Service PU denies quantity change with the entry "not executable" in the "confirmation column". Customer Service PU finalizes documents for the production.

c. Customer Service PU enters a new minimum price and set entry to "executable on price correction" in the "confirmation column".

3. Customer Service PU initiates the export of the "order from customer" to the XML Interface. Sales Reporting creates a copy of the document including the modified data. Customer Service PU set entry to "executable on price correction".

4. XML Interface imports answer from POS in the "order from customer". The document is marked bold (unread) in the overview. Sales Reporting creates a copy of the document including the modified data.

a. Customer Service POS confirms quantity and price change. The quantity is modified in "order from customer". Sales Reporting creates a copy of the document including the modified data. Customer Service PU finalizes documents for the production.

b. Customer Service PS denies quantity and price change. The quantity in "order from customer" is not modified. Sales Reporting creates a copy of the document including the modified data.

- Post Conditions, Success end condition:

1. "Confirmation column" entry "not executable": Quantity is not modified.

2. "Confirmation column" entry "executable": Quantity is modified.

7.5. Results of the Case Study

For the comparison of the two approaches, the first difference is already obvious in the structure of the documents. The traditional requirements documentation and the use case description are using both a template structure with different details in the paragraphs. The traditional document is more focused on the system and functional side. It includes one section for the definition of the process, but in this section, no structure exists to deliver guidance for the users or analysts. The use case description introduces a structured schema for the interaction between the users and the system. Furthermore, the paragraphs for pre- and post-conditions support the readers of the document to understand the scope of the single document.

The scope on the single document is the second important difference. The goal of the central sales process requirements documentation was to deliver a description for the data and functions for the complete sales process and for all involved ERP instances. The process description covers all sub-process, including possible variations and includes the requirements for the system from the POS and from the production unit.

This big amount of text makes it very hard for new readers to handle all the information. This was notable after the delivery of the documentation to the local partner companies. They started to work on the planning for the local implementation into Navision, but for each of them it was a big workload to collect the knowledge based on the set of requirement documents.

In contrast, the scope of one use case description does not cross the system border and is limited to one ERP system. Furthermore, it is related to one sub-system only, and includes a limited set of interactions, which are necessary to fulfill one process step. This break up into smaller peaces was an advantage for the understanding of the business process and the extended functions and features. The object-oriented use case approach forces this split into pieces.

Already the separated document for collection of the closing reason, and for the inquiry data have a strong correlation to the use cases "collect inquiry data" and "collect closing reasons". In the traditional approach, the requirements documentation for these two actions has been separated from the central documents. However, the reason for this was not related to the view on the work of the users, but on the system

related behaviors. The requested features are able to be separated from the central workflow and are implemented beside the standard features without any reuse of existing data models.

Together with the object-oriented approach, the use of selected UML notation and diagrams was introduced to support the textual use case description. Since UML is a language with a very broad scope not all of its modeling capabilities are necessary. For the actual model, only those parts of the language are used that are of direct interest. The activity diagrams became very useful for the documentation of the business process, whereas the state machine charts clarified the status on documents. Beside these UML notations, the class diagrams are used for the data structure.

For the basic UML concepts, it was necessary to provide some education for the users and the external partners. However, this additional effort creates an advantage by the better understanding of the requirements model. The visualization was an important support for the textual descriptions. It became evident, that UML helped through some misunderstanding because of different languages. For most of the users, English is not the first language. Therefore, the combination of UML diagrams and textual description together assist the transfer of the knowledge.

One of the goals for the implementation project was the collection of data for the new sales report system. Therefore, the requirements documentation has to deliver a description for the data and functions to receive all mandatory information. In the traditional documentation, this description was separated from the process of the workflow. In the use case requirement model, the actor sales reporting present the needs for the sales reporting. This improves the awareness of this role and the relation to the processes. The description of the flow of events in the use cases delivers a good possibility to track the collection of data during the process.

For the complete definition of the requirements, the use case descriptions and the UML diagrams were extended to cover user interfaces, reports in the ERP and printed documents. Especially for the new data containers "business case", a new user interface inside Navision was defined. The changes of the interface are now providing the entry point for all users, who are working in any role on a business case. This new data model builds now the main references for communication between POS and

production unit. Therefore, the printed documents and reports have been adopted, to show this new information. To describe all these requirements with UML and use cases was not possible. Only the data model for the documents is possible to include in the class diagram. The graphical layout of the user interface and the documents was described in additional specialized documents.

8. Discussion, conclusion, further work

The use case requirement model seems to be a useful approach to support the implementation of a distributed business process in different instances of an ERP system. The following subsection discusses the results of the case study in relation to the defined research questions. Finally, it shows the conclusion of the thesis and further work.

In the investigated project, the comparison between the traditional approach and the use case requirement model is based on the experience by the author and two colleagues. These three team members together build the core of the project team and are involved in the implementation work in all local ERP installations. The estimated effects are based on the personal experience during the progress of the project. One of the factors, which influenced this estimation, was the numbers of requested clarifications from other project team members.

A difficulty for the comparison is the circumstance, that the two different approaches have been used on different local ERP installations. The first two installations have been implemented with the documentation based on the traditional approach. Afterwards the use case requirement model was developed and used for the further implementations. Therefore, the direct comparison on the same ERP instance is not possible.

8.1. Discussion on Research Questions

- To what extent can a use case requirement model reduce effort and time, which is necessary to transfer sufficient knowledge about implementation requirements?

The main advantage of the use case requirement model is the brake up of a large and complex system into smaller pieces with the introduction of use cases. The validation of the descriptions inside a use case is more comfortable. The identification of the business process in the requirements model delivers a higher acceptance from the users of the system. Important is the combination with selected UML notations.

The effort to teach object-oriented modeling and to explain UML diagrams is additional to the work on the requirements. These additional tasks increase the costs at the start of the project. Nevertheless, during the project the used time and resources for the transfer of knowledge are decreasing with the utilization of the use case approach.

The experience during the ongoing project shows that the use cases requirements model together with the UML notations reduced workload by app. 33%.

- To what extent can a use case requirement model reduce the number of errors and misunderstandings about implementation requirements?

The combination of the textual use case description and selected parts of UML, the visualization supports the understanding for software engineers and users. The diagrams are able to build the common language for an international distributed implementation project. This reduces the errors and leads to lower costs and time effort.

The improvements for the understandability for all project members increase the chance to reach the project goals. The introduction of a common technical language UML reduces misunderstanding due to different languages. The experience during the ongoing project shows that in comparison to the traditional approach, the use case requirements model reduced the numbers of mistakes by app. 33%.

- Which impact has a use case requirement model on solving change requests?

Like many other software products, the ERP systems need to be continuously adapted and evolve. The request for changes can be related to adoptions in the implemented business process, or to the extension of the scope to cover additional tasks with the ERP solution. Additionally, the supplier of the software provides frequently upgrades to maintain the product.

These facts lead to the need for suitable maintenance and evolution process. Thereby, the understanding of the design, functionality and behavior of the customized elements in the ERP system play an import role.

The use case requirements model assists the maintenance process very well. Most of the requested changes are related to a variation in the business process. The close connection between the business processes and the use cases assists to localize the target of a change request. The structured description of the use case itself supports the verification of the change request. Each use case has well defined borders with the systems scope and the pre- and post-conditions. This reduces the complexity on the analyses of the implications of the planned changes.

The unambiguous documentation reduces the change of errors during implementation of change requests. Moreover, an improved understanding for all external partner companies prevents inconstancies between different local instances. This improvement in the requirements model reduced the need resources for the implementation of change requests by app. 50%.

- To what extent can a use case requirement model be used to implement requirements tracing?

The use case supports the trace from the business process to the requirements description. A change of the business process can be followed up on the use case model, either with adoption in existing use cases, or by introduction of new use cases. Because of the relation between use and business cases, the model assists also the validation and verification of the implementation.

The use case requirements model supports the implementation of requirements tracing. However, to maintain these traces manual will increase the risk of inconsistencies. Therefore, the use of a requirements management tool is recommended.

8.1.1. Cost effect

On the one hand, each of the first three research questions shows a positive effect on the implementation costs. On the other hand, the additional education for users and external partners to enable the work with UML increased the implementation cost. Furthermore, the initial creation of the use case requirements model produced additional expenses. To combine these positive and negative effects for the investigated implementation project, it is important to take the number of installations into account.

Based on the experience with the two different approaches the total effect on the implementation costs has been estimated. The comparison assumes that the project for all local ERP installations was either fully implemented with the traditional approach or fully implemented with the use case requirements model. The costs are shown as multiples of costs for one installation with the traditional approach. The total cost depending on the number of installations is shown in Figure 8.1.1-1.

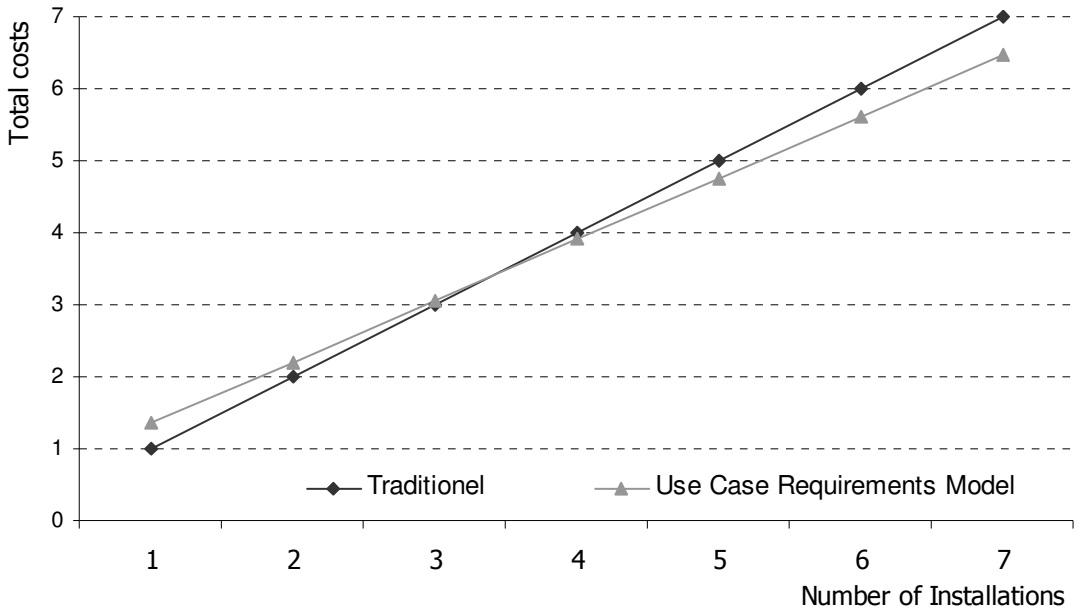

Figure 8.1.1-1 Total costs of implementation

This comparison of the total costs indicates that for the investigated project the utilization of the use case requirements model is more expensive for the first three installations. The one-time costs for the initial creation together with the additional education work increases the costs compared to the traditional approach.

The advantages of the use case requirements model appear as the number of involved installations grows. The reuse of the initial documents reduces the implementation costs per installation. The negative effect of the additional education task is more than compensated by the positive effect of cost reduction because of the better transfer of knowledge and the reduced numbers of errors.

8.2. Conclusion

Enterprise Resource Planning (ERP) solutions became the replacement for obsolescent legacy systems for many companies. Multinational enterprises have several units across separate geographical locations. Many of these enterprises do not use a central single instance of the ERP, instead they are utilizing local instance of the same ERP software. As the information used in managing a multinational enterprise originates in different locations, it is fragmented between these instances. With such an environment in place, it can be difficult to get a global distributed process mapped into the ERP systems.

This thesis proposed the use case requirements model as method to decrease the time needed to implement distributed business process in a multiple ERP environment. This improves the understandability of the process and the related function and features in the ERP systems. The benefit of the proposed solution has been shown with an evaluation concept for a case study.

The proposed model supports the object-oriented approach with the integration of UML diagrams. With the utilization of selected parts of UML, the visualization supports the understanding for software engineers and users. The diagrams are able to build the common language for an international distributed implementation project.

The main advantage of the proposed approach is the brake up of the large and complex system into smaller pieces with the introduction of use cases. The validation of the descriptions inside a use case is more comfortable. Furthermore, the trace from the business process to the use case and the implementation delivers a higher acceptance from the users of the system.

As the experience in the case study shows, to cover all and everything with use cases and UML is not possible. The suggested approach extends the object-oriented documents with the description on the implementation requests for user interfaces, reports in the ERP and printed documents.

8.3. Further work

Further work will concentrate on the usage of requirements tracing to support the development process and ensure the quality of the software solution. Identifying and maintaining dependencies between business processes, use case requirement model and implemented solutions are crucial to understand the interaction of parts in one of the implemented ERP-systems. The existence of a well-defined requirement model is much more essential to control the interactions between the different local instances.

Another important task is the introduction of tools to support modeling and describing the use case requirement model. A requirements management tool needs to be selected to support the process of requirements modeling. This software delivers a common workspace for analyst developers and users. The requirements management tool can be used to combine the UML diagrams and the textual use case descriptions. This assists the implementation and maintenance of tractability matrices.

Beside the introduction of the additional requirements management tool, the requirements model can be reused to build advanced test cases for the acceptance tests on the implementation in the international ERP instances.

Furthermore, the requirements model builds a good foundation to improve the user guides. The description of use cases and their scenarios are a solid starting point. Each use case describes to the user how the system behaves and how they interact with it. The use cases can be put together with screen shots of the user interface and combined with exercises to build the starting point for training materials. The use cases are also a "repository of knowledge" to store knowledge about the business which was collected during the implementation project.

BIBLIOGRAPHY

Armour, Miller 2001
Frank Armour, Granville Miller, "Advanced Use Case Modeling: Software Systems", Addison-Wesley, 2001

CMMI 2006
CMMI for Development, Version 1.2
http://www.sei.cmu.edu/publications/documents/06.reports/06tr008.html [2008-05-11]

IEEE 1984
Institute of Electrical and Electronics Engineers "IEEE Guide to Software Requirements Specifications", IEEE, Inc, 1984

INCOSE 2003
Guide to the Systems Engineering Body of Knowledge, International Council on Systems Engineering
http://g2sebok.incose.org/ [2008-06-21]

Jacobson, Booch; Rumbaugh 1999
Ivar Jacobson, Grady Booch, James Rumbaugh, "The unified software development process", Addison-Wesley, 1999

Lindvall 1994
Mikael Lindvall, "A Study of Traceability in Object-Oriented Systems Development", Linköping University, 1994

Loucopoulos, Karakostas 1995
Pericles Loucopoulos, Vassilios Karakostas, "System requirements engineering", McGraw-Hill, 1995

OMG 2007
Object Management Group, "OMG Unified Modeling Language (OMG UML),Superstructure, V2.1.2"
http://www.omg.org/spec/UML/2.1.2/Superstructure/PDF [2008-06-28]

Pinheiro 1996
Francisco A.C. Pinheiro, Joseph A. Goguen, "An Object-Oriented Tool for Tracing Requirements," IEEE Software, vol. 13, no. 2, pp. 52-64, Mar., 1996

Ramesh 1995
B. Ramesh, T. Powers, C. Stubbs, M. Edwards, "Implementing requirements traceability: a case study," re, p. 89, Second IEEE International Symposium on Requirements Engineering (RE'95), 1995

Schneider, Winters 1999
Geri Schneider, Jason P. Winters "Applying use cases: a practical guide", Addison-Wesley, 1999

Sommerville, Sawyer 1997
Ian Sommerville, Pete Sawyer, "Requirements engineering: A good practice guide", Wiley, 1997

Sutcliffe 2002
Alistair Sutcliffe, "User-centred requirements engineering", Springer, 2002

Wiegers 2000
Karl E. Wiegers, "When Telepathy Won't Do: Requirements Engineering Key Practices", originally published in Cutter IT Journal, May 2000
http://www.processimpact.com/articles/telepathy.html [2008-07-12]

Wiegers 2003
Karl E. Wiegers, "Software Requirements", 2nd Edition, Microsoft Press, 2003

Wiegers 2006
Karl E. Wiegers, "More About Software Requirements, Microsoft Press, 2006

Wieringa 1995
Roel Wieringa, "An Introduction to Requirements Traceability" 1995

ZUSAMMENFASSUNG

Ausgangspunkt dieser Diplomarbeit ist meine Erfahrung bei der Durchführung eines international verteilten Software Implementierungsprojektes.

Die Aufgabenstellung besteht darin, den Prozessablauf des Vertriebes zu beschleunigen, und die Qualität der Daten zur Ermittlung der Kennzahlen zu verbessern. Zusätzliche sollten weitere Daten gesammelt werden, die zukünftige Auswertungen über den Prozessdurchlauf ermöglichen. In jedem der beteiligten Unternehmen ist die Standard ERP Software Microsoft Dynamics NAV, Navision, im Einsatz. Innerhalb dieser bestehenden ERP Systeme, wird im Rahmen des Projektes ein zusätzliches Modul entwickelt, das den gesamten Vertriebsprozess vom ersten Kontakt des Kunden bis zur erfolgreichen Lieferung und Fakturierung unterstütz, sowie die Datenübertragung zwischen den beteiligten Systemen ermöglicht. Durch die einheitliche Entwicklung des zusätzlichen Moduls soll sichergestellt werden, dass der Prozess in allen Systemen nach den gleichen Regeln abläuft und die die Sammlung der Daten für das Reporting konsistent in den verschiedenen lokalen Installationen erfolgt.

Im Rahmen der Diplomarbeit soll untersucht werden, wie der Prozess der Anforderungsanalyse verbessert werden kann.

Als Vorgehensmodell wird das Software Capability Maturity Model herangezogen. Es soll geklärt werden, durch welche Maßnahmen die Qualität der Softwareentwicklung zu verbessern ist.

Als Schwerpunkt wird die Überarbeitung der Anforderungen überprüft, ob diese in UML zu einem besseren Verständnis der Aufgabenstellung führen. Weiters soll untersucht werden, ob diese Darstellung den Wissenstransfer zu den Implementierungspartnern erleichtert.

Im Detail wird dabei speziell auf die Erstellung eines erweiterten Use Case Templates eingegangen, sowie die genaue Ausformulierung einiger wichtiger Use Cases analysiert.

Das Ergebnis dieser Diplomarbeit richtet sich an Software Entwickler im Bereich der Anpassung von Standard Software.

ABSTRACT

Many Multinational enterprises do not use a central single instance of the Enterprise Resource Planning (ERP), instead they are utilizing local instance of the same ERP software. As the information used in managing a multinational enterprise originates in different locations, it is fragmented between these instances. With such an environment in place, it can be difficult to get a global distributed process mapped into the ERP systems.

Starting point of this Diploma thesis was my personal experience during such an international distributed software implementation project.

The task of this implementation project was the extension of existing standard ERP software to support the sales process to increase efficiency on the operational side, and to ensure consistent and comparable data for the sales reporting on the analytic side. Beside that, additional data for the monitoring of the process flow should be collected. The existing support of the process by IT systems has been found incomplete, especially concerning sales cases that transcend organizational units and are therefore represented in different ERP installations. Each of the involved companies uses the Standard ERP Software Microsoft Business Solutions Dynamics NAV, Navision. During this project an additional granule was developed, which supports the complete sale process, starting from the first inquiry from the customer until the final delivery and invoicing. Because of the central development of this granule, the implementation of this business process is based on the same rule set and the collection of the reporting data will be consistent in all systems.

This thesis wants to analyze the possibilities for improvements on the requirements engineering for similar projects, which are implementing a business process into an existing ERP solution.

The evaluation for the requirements is based on the Capability Maturity Model. The investigation wants to clarify, which practices are necessary to implement and will improve the quality of the software development process.

The focus of this work is the verification of the rework on the existing documentation, if the use of the Unified Modeling Language is able to improve the understanding of

the goals and tasks. It will also be investigated, if this representation in an UML model, in connection to the written requirements is able to improve the knowledge transfer to the international implementation partners. The focus will be the development of an extended use case Template, where some high important use case will be analyzed in detail.

The target audiences for this thesis are software developers, who are involved in the field of customizing standard software.

LEBENSLAUF THOMAS SEKULIN

Persönliche Informationen

- Nationalität: Österreich
- Geburtsdatum: 28.2.1970
- Geburtsort: Wien

Ausbildung

1994 - 2008 Universität Wien Wien
Magister der Sozial- und Wirtschaftswissenschaften
- Wirtschaftsinformatik

1990 - 1994 Universität Wien Wien
Erste Diplomprüfung Wirtschaftsinformatik

1984 - 1989 Höhere technische Bundeslehranstalt Wien 4
Matura Elektrotechnik
- mit ausgezeichnetem Erfolg

Berufserfahrung

Seit 5/2005 PPC Insulators Holding GmbH Wien
CIO

10/2001 – 4/2005 PPC Insulators GmbH Wien
- PPC Insulators Group Treasury and IT

9/1999 – 9/2001 Ceram Handels GmbH Wien
- Ceram Group Treasury and IT

9/1996 – 8/1999 PSA Finance Austria Bank AG Wien
- Treasury & Reporting

Wehrdienst

10/1989 – 5/1990