The Waterfall Model in Large-Scale Development

Kai Petersen[1,2], Claes Wohlin[1], Dejan Baca[1,2]

[1] Blekinge Institute of Technology, Box 520,
SE-37225 Ronneby, Sweden
kai.petersen@bth.se,claes.wohlin@bth.se,dejan.baca@bth.se
[2] Ericsson AB, Box 518,
SE-37123 Karlskrona, Sweden
kai.petersen@ericsson.com,dejan.baca@ericsson.com

Abstract. Waterfall development is still a widely used way of working in software development companies. Many problems have been reported related to the model. Commonly accepted problems are for example to cope with change and that defects all too often are detected too late in the software development process. However, many of the problems mentioned in literature are based on beliefs and experiences, and not on empirical evidence. To address this research gap, we compare the problems in literature with the results of a case study at Ericsson AB in Sweden, investigating issues in the waterfall model. The case study aims at validating or contradicting the beliefs of what the problems are in waterfall development through empirical research.

1 Introduction

The first publication on the waterfall model is credited to Walter Royce's article in 1970 (cf. [1]). In literature there seems to be an agreement on problems connected to the use of the waterfall model. Problems are (among others) that the model does not cope well with change, generates a lot of rework, and leads to unpredictable software quality due to late testing [2]. Despite the problems identified, the model is still widely used in software industry, some researchers are even convinced that it will be around for a much longer period of time (see [3]). The following trends can be seen in research. First, the model seems to be of very little interest for researchers to focus on as it seems to be old-fashioned. Instead, recent studies have much more focus on agile and incremental development. Secondly, there is very little empirical research backing up what we believe to know about the waterfall model. In order to identify the evidence provided by empirical research on the waterfall model we conducted the following search on Inspec & Compendex:

– (”waterfall model” OR ”waterfall development”) AND (”empirical” OR ”case study” OR ”industrial”)

Inspec & Compendex was selected as it integrates many full-text databases in computing and thus is considered a good starting point. The search resulted in

33 publications where none of the publications had an explicit focus on studying the waterfall model in an industrial setting. Thus, most of the problems reported on the waterfall model are mainly based on researchers' beliefs and experience reports. Consequently, in order to provide substantial evidence on the usefulness of the waterfall model in industry empirical studies are needed. Evaluating the usefulness empirically aids decision making of whether to use the model in specific context (here large-scale-development).

To address this research gap we conducted a case study focusing on identifying issues in waterfall development and compare them to what has been said in literature. Furthermore, the issues identified are ranked based on their criticality. The case being studied is a development site of Ericsson AB, Sweden. The waterfall model was used at the company for several years. The case study has been conducted according to the guidelines provided by Yin (see [4]). The case study makes the following contributions to research on waterfall development: 1) Illustration of the waterfall implementation in practice within large-scale industrial software development, 2) Identification of issues related to the waterfall model and their prioritization showing the most critical issues, and 3) Comparison of case study results with state of the art (SotA).

The remainder of this paper is structured as follows: Section 2 provides an overview of related work. Thereafter, Section 3 illustrates the waterfall model used at the company. Section 4 presents the case study design. The analysis of the collected data is provided in Section 5 (qualitative analysis) and Section 6 (quantitative analysis). Section 7 presents a comparison of the case study findings and state of the art. Section 8 concludes the paper.

2 Related Work

Literature identifies a number of problems related to the waterfall model. An overview of the problems identified in literature is shown in Table 1. In addition to the identified articles we considered books discussing advantages and disadvantages of the waterfall model.

The waterfall model is connected to high costs and efforts [2][5]. That is, it requires approval of many documents, changes are costly to implement, iterations take a lot of effort and rework, and problems are usually pushed to later phases [2]. Few studies are explicitly focused on the waterfall model and some reasons for the failures of the waterfall approach have been identified. One reason mentioned by several studies is the management of a large scope, i.e. requirements cannot be managed well and has been identified as the main reason for failure (cf. [7] [9] [8]). Consequences have been that the customers' current needs are not addressed by the end of the project [7], resulting in that many of the features implemented are not used [9].

Additionally, there is a problem in integrating the overall system in the end and testing it [10]. A survey of 400 waterfall projects has shown that the software being developed is either not deployed or if deployed, it is not used. The reasons for this are the change of needs and the lack of opportunity to clarify

Table 1. Issues in Waterfall Development (State of the Art)

ID	Issue	Reference
L01	High effort and costs for writing and approving documents for each development phase.	[2][5]
L02	Extremely hard to respond to changes.	[2][5][6]
L03	When iterating a phase the iteration takes considerable effort for rework.	[2]
L04	When the system is put to use the customer discovers problems of early phases very late and system does not reflect current requirements.	[1] [2] [7]
L05	Problems of finished phases are left for later phases to solve.	[2]
L06	Management of a large scope of requirements that have to be baselined to continue with development.	[8] [7] [9]
L07	Big-bang integration and test of the whole system in the end of the project can lead to unexpected quality problems, high costs, and schedule overrun.	[10][1][11]
L08	Lack of opportunity for customer to provide feedback on the system.	[10]
L09	The waterfall model increases lead-time due to that large chunks of software artifacts have to be approved at each gate.	[12]

misunderstandings. This is caused by the lack of opportunity for the customer to provide feedback on the system [13]. Specifically, the waterfall model fails in the context of large-complex projects or exploratory projects [3].

On the other hand, waterfall development comes with advantages as well. The waterfall model is predictable and pays attention to planning the architecture and structure of the software system in detail which is especially important when dealing with large systems. Without having focus on architecture planning there is a risk that design decisions are based on tacit knowledge and not explicitly documented and reviewed [14]. Thus, the probability of overlooking architectural problems is high.

3 The Waterfall Model at the Company

The waterfall model used at the company runs through the phases requirements engineering, design & implementation, testing, release, and maintenance. Between all phases the documents have to pass a quality check, this approach is referred to as a stage-gate model (see for example [15]). An overview of the process is shown in Figure 1.

We explain the different phases and provide a selection of checklist-items to show what type of quality checks are made in order to decide whether the software artifact developed in a specific development phase can be passed on to the adjacent phase.

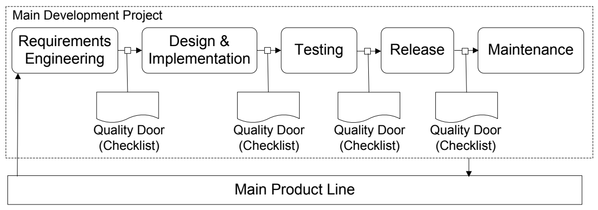

Fig. 1. Waterfall Development at the Company

Requirements Engineering: In this phase, the needs of the customers are identified and documented on a high abstraction level. Thereafter, the requirements are refined so that they can be used as input to the design and implementation phase. The requirements (on high as well as low abstraction level) are stored in a requirements repository. From this repository, the requirements to be implemented are selected from the repository. The number of requirements selected depends on the available resources for the project. As new products are not built from the scratch, parts from the old product (see main product line in Figure 1) are used as input to the requirements phase as well. At the quality gate (among others) it is checked whether all requirements are understood, agreed upon, and documented. Furthermore, it is checked whether the relevant stakeholders are identified and whether the solution would support the business strategy.

Design and Implementation: In the design phase the architecture of the system is created and documented. Thereafter, the actual development of the system takes place. The developers also conduct basic unit testing before handing the developed code over to the test phase. The quality gate checklist (among others) verifies whether the architecture has been evaluated, whether there are deviations from the requirements compared to the previous quality gate decision, and whether there is a deviation from planned time-line, effort, or product scope.

Testing: In this phase the system integration is tested regarding quality and functional aspects. In order to make a decision whether the the system can be deployed, measures of performance (e.g, throughput) are collected in the test laboratory. As the company provides complete solutions (including hardware and software) the tests have to be conducted on a variety of hardware and software configurations as those differ between customers. The outcome of the phase is reviewed according to a checklist to see whether the system has been verified and whether there are deviations from previous quality gate decisions in terms of quality and time, whether plans for hand-over of the product to the customer are defined according to company guidelines, and whether the outcome of the project meets the customers' requirements.

Release: In the release phase the product is brought into a shippable state. That is, release documentation is finalized (e.g. installation instructions of the system for customers and user-guides). Furthermore, build-instructions for the system have to be programmed. Build-instructions can be used to enable and disable features of the main product line to tailor the system to specific customer needs. At the quality gate (among others) it is checked whether the outcome meets the customers' requirements, whether the customer has accepted the outcome, and whether the final outcome was presented in time and fulfilled its quality requirements. A post-mortem analysis has to be performed as well.

Maintenance: After the product has been released to the customer it has to be maintained. That is, if customers discover problems in the product they report them to the company and get support in solving them. If the problems are due to faults in the product, packages for updating the system are delivered to the customers.

4 Case Study Design

The context in which the study is executed is Ericsson AB, a leading and global company offering solutions in the area of telecommunication and multimedia. Such solutions include charging systems for mobile phones, multimedia solutions and network solutions. The company is ISO 2001:2000 certified. The market in which the company operates can be characterized as highly dynamic with high innovation in products and solutions. The development model is market-driven, meaning that the requirements are collected from a large base of potential end-customers without knowing exactly who the customers will be.

4.1 Research Questions

The following main research questions should be answered in the case study:

- *RQ1: What are the most critical problems in waterfall development in large-scale industrial development?*
- *RQ2: What are the differences and similarities between state of the art and the case study results?*

The relevance of the research questions can be underlined as follows: The related work has shown a number of problems related to waterfall development. However, there is too little empirical evidence on the topic and thus more data points are needed. Furthermore, the criticality of problems is not addressed in any way so far, making it hard to decide in which way it is most beneficial to improve the model, or whether the introduction of a new way of working will help in improving the key challenges experienced in the waterfall model.

4.2 Case Selection and Units of Analysis

The case being studied is one development site of Ericsson AB. In order to understand the problems that occurred when the waterfall model was used at the company, three subsystems (S1, S2, and S3) are analyzed that have been built according to the model. The systems under investigation in this case study have an overall size of approx. 2,000,000 LOC (as shown in Table 2). The LOC measure only includes code produced at the company (excluding third-party libraries). Furthermore, the number of persons involved in building the system are stated. A comparison of the system considered for this study and the size of the Apache web server shows that the system being studied is considerably larger and thus can be considered as large-scale.

Table 2. Units of Analysis

	Language	Size (LOC)	No. Persons
Overall System		>5,000,000	-
S1	C++	300,000	43
S2	C++	850,000	53
S3	Java	24,000	17
Apache	C++	220,000	90

4.3 Data Collection Procedures

The data is collected through interviews and from process documentation.

Selection of Interviewees The interviewees were selected so that the overall development life cycle is covered, from requirements to testing and release. Furthermore, each role in the development process should be represented by at least two persons if possible. The selection of interviewees was done as follows:

1. A complete list of people available for the system being studied. Overall 153 people are on this list as shown in Table 2.
2. For the selection of persons we used cluster sampling. At least two persons from each role (the roles being the clusters) have been randomly selected from the list. The more persons are available for one role the more persons have been selected.
3. The selected interviewees received an e-mail explaining why they have been selected for the study. Furthermore, the mail contained information of the purpose of the study and an invitation for the interview. Overall, 44 persons have been contacted of which 33 accepted the invitation.

The distribution of people between different roles is shown in Table 3. The roles are divided into "What", "When", "How", "Quality Assurance", and "Life Cycle Management".

- *What:* This group of people is concerned with the decision of what to develop and includes people from strategic product management, technical managers and system managers.
- *When:* People in this group plan the time-line of software development from a technical and project management perspective.
- *How:* Here, the architecture is defined and the actual implementation of the system takes place. In addition, developers test their own code (unit tests).

- *Quality Assurance:* Quality assurance is responsible for testing the software and reviewing documentation.
- *Life Cycle Management:* This includes all activities supporting the overall development process, like configuration management, maintenance and support, and packaging and shipment of the product.

Table 3. Distribution of Interviewees Between Roles and Units of Analysis

	S1	S2	S3	Total
What (Requirements)	2	1	1	4
When (Project Planning)	3	2	1	6
How (Implementation)	3	2	1	6
Quality Assurance	4	3	-	7
Life Cycle Management	6	4	-	10
Total	18	12	3	33

Interview Design The interview consists of five parts, the duration of the interviews was set to approximately one hour each. In the first part of the interviews the interviewees were provided with an introduction to the purpose of the study and explanation why they have been selected. The second part comprised questions regarding the interviewees background, experience, and current activities. Thereafter, the issues were collected through a semi-structured interview. To collect as many issues as possible the questions have been asked from three perspectives: bottlenecks, rework, and unnecessary work. The interviewees should always state what kind of bottleneck, rework, or unnecessary work they experienced, what caused it, and where it was located in the process.

Process Documentation Process documentation has been studied to gain an in-depth understanding of the processes. Documentation for example includes process specifications, training material for processes, and presentations given to employees during unit meetings.

4.4 Data Analysis Approach

The problems related to the waterfall model at the company have been identified conducting the four steps outlined below. The steps are based on more than 30 hours of interview transcriptions and have been executed by the first author over a three month period.

1. *Clustering:* The raw data from the transcriptions is clustered, grouping statements belonging together. For example, all statements related to requirements engineering are grouped together. Thereafter, statements addressing similar areas within one group (e.g,. all areas that would relate to requirements engineering lead-times) are grouped.

2. *Derivation of Issue Statements:* The raw data contains detailed explanations and therefore is abstracted by deriving problem statements from the raw data, explaining them shortly in one or two sentences. The result was a number of problem statements where statements varied in their abstraction level and could be further clustered.
3. *Mind-Mapping of Issue Statements:* The issue statements were grouped based on their relation to each other and their abstraction level. For example, problems related to requirements lead-times are grouped within one branch called "long requirements lead-times". This was documented in form of a mind-map. Issues with higher abstraction level are closer to the center of the mind map than issues with lower abstraction level.
4. *Validation of Issues:* In studies of qualitative nature there is always a risk that the data is biased by the interpretation of the researcher. Therefore, the issues have been validated in two workshops with three representatives from the company. The representatives have an in-depth knowledge of the processes. Together, the steps of analysis described here have been reproduced together with the authors and company representatives. For this a subset of randomly selected issue statements have been selected. No major disagreement has been discovered between the workshop participants on the outcome of the analysis. Thus, the validity of the issue statements can be considered as high.

After having identified the problems they are prioritized into A-problems (critical), B-problems (very important), C-problems (important), D-problems (less important), and E-problems (local). The actual limits on the classes is based on the results. The main objective of the classification is to systematize and structure the data and not to claim that these classes are optimal or suitable for another study.

A. The problem is mentioned by more than one role and more than one subsystem. Moreover, the problem has been referred to by more than 1/3 of the respondents.
B. The problem is mentioned by more than one role and more than one subsystem. Moreover, the problem has been referred to by more than 1/5 of the respondents.
C. The problem is mentioned by more than one role and more than one subsystem. Moreover, the problem has been referred to by more than 1/10 of the respondents.
D. The problem is mentioned by more than one role and more than one subsystem. Moreover, it has been referred to by 1/10 of the respondents or less.
E. The problem is only referred to by one role or one subsystem and thus considered a local or individual problem.

4.5 Threats to Validity

Threats to the validity of the outcome of the study are important to consider during the design of the study allowing to take actions mitigating them. Threats to validity in case study research are reported in [4]. The threats relevant to the study are: construct validity, external validity and reliability.

Construct Validity: Construct validity is concerned with obtaining the right measures for the concept being studies. One threat is the selection of people to obtain the appropriate sample for answering the research questions. Therefore, experienced people from the company selected a pool of interviewees as they know the persons and organization best. From this pool the random sample was taken. The selection by the representatives of the company was done having the following aspects in mind: process knowledge, roles, distribution across subsystems, and having a sufficient number of people involved (although balancing against costs). Furthermore, it is a threat that the presence of the researcher influences the outcome of the study. The threat is reduced as there has been a long cooperation between the company and university and the author collecting the data is also employed by the company and not viewed as being external. Construct validity is also threatened if interview questions are misunderstood or misinterpreted. To mitigate the threat pre-tests of the interview have been conducted.

External Validity: External validity is the ability to generalize the findings to a specific context as well as to general process models. One threat to validity is that only one case has been studied. Thus, the context and case have been described in detail which supports the generalization of the problems identified. Furthermore, the process model studied follows the main principles of waterfall development (see Section 3) and thus can be well generalized to that model. In addition, the outcome is compared to state of the art.

Reliability: This threat is concerned with repetition or replication, and in particular that the same result would be found if re-doing the study in the same setting. There is always a risk that the outcome of the study is affected by the interpretation of the researcher. To mitigate this threat, the study has been designed so that data is collected from different sources, i.e. to conduct triangulation to ensure the correctness of the findings. The interviews have been recorded and the correct interpretation of the data has been validated through workshops with representatives of the company.

5 Qualitative Data Analysis

In total 38 issues have been identified in the case study. The majority of issues is categorized in class E, i.e, they are only referred to by individuals or are not mentioned across subsystems (see Table 4). Furthermore, the distribution of issues between the phases requirements engineering (RE), design and development (DI), verification and validation (VV), release (R), maintenance (M), and project management (PM) is shown. The distribution of issues is further discussed in Section 7.

In the analysis of the issues we focus on classes A to D as those are the most relevant ones as they are recognized across roles and systems. Thus, they have a visible impact on the overall development process. However, this does not imply

Table 4. Number of Issues in Classification

Classification	RE	DI	VV	R	M	PM	No. of Issues
A	1	-	1	-	-	-	2
B	-	-	2	-	-	-	2
C	1	2	-	-	1	1	5
D	1	1	2	-	-	-	4
E	1	1	2	3	8	10	25
Sum	4	4	7	3	9	11	38

Table 5. Issues in Waterfall Development

ID	Class	Process Area	Description	SotA
P01	A	Requirements	Requirements work is wasted as documented and validated requirements have to be discarded or reworked.	L02, L03, L08
P02	A	Verification	Reduction of test coverage due to limited testing time in the end.	L07
P03	B	Verification	Amount of faults found increases with late testing.	L05
P04	B	Verification	Faults found later in the process are hard and expensive to fix.	L07
P05	C	Requirements	Too much documentation is produced in requirements engineering that is not used in later stages of the process.	L01
P06	C	Design	Design has free capacity due to long requirements engineering lead-times.	L09
P07	C	Design	Confusion on who implements which version of the requirements.	-
P08	C	Maintenance	High effort for maintenance (corrections released to the customer).	L04
P09	C	Project Mgt.	Specialized competence focus of team members and lack of confidence.	-
P10	D	Requirements	The impact of requirements on other parts of the system are not foreseen.	L06
P11	D	Design	Design is overloaded with requirements.	-
P12	D	Verification	High amount of testing documentation has to be produced.	L01
P13	D	Verification	Problems in fault localization due to barriers in communication.	-

that local issues are completely irrelevant, they just have little impact on the overall development process and thus are not recognized by other roles. Table 5 shows an overview of the identified issues in classes A to D and their mapping to literature summarized in Table 1.

5.1 A Issues

P01: The long lead-times of the requirements engineering phase led to the need to change requirements or discard already implemented and reviewed requirements as the domain investigated (telecommunication) is very dynamic. Furthermore, the distance to the customer caused misunderstandings which resulted in changed requirements or discarded requirements. Due to the complexity of the scope to be defined the number of requirements was too high for the given resources which resulted in discarding requirements (and sometimes this was done late in the development process). Furthermore, the interviewees emphasized that the decision what is in the scope and what is not takes a lot of time as a high amount of people that have to be involved.

P02: Test coverage in waterfall development was reduced due to multiple reasons. Testing is done late in the project and thus if there have been delays in development, testing has to be compromised as it is one of the last steps in development. Furthermore, too much has to be tested at once after the overall system has been implemented. Additional factors are that testing related to quality is often given low priority in comparison to functional testing, trivial things are tested too intensively, and test resources are used to test the same things twice due to coordination problems.

5.2 B Issues

P03: The later the testing, the higher the amount of faults found. The number of faults and quality issues is influenced negatively when using waterfall development. The main cause for this is late testing after everything has been implemented. This provides far too late feedback from test on the software product. Furthermore, basic testing is neglected as there has been low interaction between design and testing, resulting in lack of understanding of each other in terms of consequences of neglecting basic testing. Also due to communication issues, testing started verifying unfinished code which led to a high number of false positives (not real faults).

P04: Having late testing results in faults that are hard to fix, which is especially true for issues related to quality attributes of the system (e.g. performance). These kinds of issues are often rooted in the architecture of the system which is hard to change late in the project.

5.3 C Issues

P05: The interviewees emphasized that quite a lot of documentation is produced in the requirements phase. One of the reasons mentioned is limited reuse of documentation (i.e., the same information is reported several times). Furthermore, the concept of quality gates requires producing a lot of documentation and checklists which have to be fulfilled before passing on the requirements to the next phase. Though, in waterfall development the quality gates are required as they assure that the hand-over item is of good enough quality to be used as input for all further development activities.

P06: Design and implementation have free capacity, the reasons being that requirements have to be specified in too much detail, decision making takes a long time, or requirements resources are tied up due to the too large requirements scope. This has a negative impact on design, as the designers have to wait for input from requirements engineering before they can start working. As one interviewee pointed out *"For such large projects with so many people involved half the workforce ends up working for the rest"*. In consequence, the lead-time of the overall project is prolonged.

P07: From a design perspective, it is not always clear which version of the requirements should be implemented and by whom. The cause of this problem is that work often starts on unfinished or unapproved requirements which have not been properly baselined.

P08: Support is required to release a high number of corrections on already released software. This is due to the overall length of the waterfall projects resulting in very long release cycles. In consequence, the customers cannot wait for the corrections to be fixed for the next release, making corrections a time-pressing issue. Furthermore, the development model requires to handle parallel product branches for customer adaptations of the main product line. In this domain, products have a high degree of variability and thus several product branches have to be supported (see Figure 1).

P09: The competence focus of people in waterfall development is narrowed, but specialized. This is due to that people are clearly separated in their phases and disciplines, and that knowledge is not well spread among them. As one interviewee pointed out, there are communication barriers between phases. Furthermore, a lack of confidence has been reported. That is, people are capable but do not recognize their particular strength to a degree they should.

5.4 D Issues

P10: New requirements do not have an isolated impact, instead they might affect multiple subsystems. However, due to the large requirements scope, requirements dependencies are often overlooked.

P11: The scope of the requirements was too big for the implementation resources. In consequence, designers and architects were overloaded with requirements which could not be realized with the given resources. Furthermore, after the project has been started more requirements were forced into the project by the customer. In consequence, emergent requirements cannot be implemented by architects and designers as they already face an overload situation.

P12: Test documentation has been done too extensively as the documents became obsolete. The reason for the high amount of documentation was mainly that the process has been very documentation centric.

P13: When dealing with different subsystems, the fault localization is problematic as a problem might only show in one subsystems, but due to communication barriers not all subsystem developers are aware of the problem. In consequence, due to the lack of communication (see P09) the localization of faults reported by the customer is time consuming.

6 Quantitative Data Analysis

Table 6 shows the distribution of time (duration) in the development process. The requirements engineering phase takes very long time in comparison to the other phases. The actual implementation of the system seems to be the least time-intensive activity.

Table 6. Distribution of Time (Duration) over Phases (in %)

Req.	Impl.&Design	Verification	Release	Total
41	17	19	23	100

Furthermore, we measured the number of change requests per implemented requirement, the discarded requirement, and the percentage of faults found in system test that should have been found in earlier tests (function test and component test). The figures quantify the issues identified earlier. In particular, the high number of discarded requirements and the cause of change requests are related to issue P01. The long lead-times of requirements engineering increase the time-window for change requests and approximately 26 % of all requirements become obsolete. From a quality perspective the fault slip of 31 % is a symptom of P03 (increase of number of faults with late testing) and P04 (the types of faults found in system tests could have been found earlier and thus would have been easier to fix).

Table 7. Performance Measures

Measure	Value
CRs per implemented requirement	0.076
Discarded requirements	26 %
Fault slip to system test	31 %

7 Comparative Analysis of Case Study and SotA

Table 5 relates the issues identified in the case study to the issues mentioned in literature. If an issue from the case study is identified in literature the column SotA provides the ID of the issue identified in literature (listed in Table 1). Through this comparison it becomes apparent that four issues not mentioned in the identified literature have been discovered in the case study, namely P07, P09, P11, and P13. Vice versa all issues acknowledged in literature have been identified in the case study. Table 5 also shows that the highest prioritized issues (A and B) have all been mentioned in literature describing the waterfall model. In conclusion researchers and practitioners are aware of the most pressing issues related to waterfall development, while lower prioritized (but still important) issues have not been linked to the waterfall model to the same degree.

The issues in the case study are formulated differently from those identified in literature as the formulation is an outcome of the qualitative data analysis. Therefore, we explain how and why the issues of high priority from the case study and SotA are related to each other. The most critical issues are related to the phases of requirements engineering, and verification and validation (both identified in literature). We found that requirements often have to be reworked and or discarded (P01). The qualitative analysis based on the interviews explained the issue with long lead-times for requirements and large scope making responding to changes hard (related to L02), distance to the customer (related to L08), and change in large scope leads to high effort due to that many people are involved (related to L03). The quantitative analysis shows that 41 % of the lead-time is consumed for requirements engineering. Having to define a large requirements scope extends lead-time and thus reduces requirements stability. In consequence the waterfall model is not suitable in large-scale development in the context of a dynamic market. Regarding verification issue L07 identified in literature states that testing the whole system in the end of the project leads to unexpected quality problems and project overruns. This issue relates to the case study in the following ways: First, testing has to be compromised and thus test coverage is reduced when having fixed deadlines which do not allow for project overruns (P02). Secondly, the faults found late in the process are hard to fix, especially if they are rooted in the architecture of the system (P07).

The issues categorized as C are quite mixed, i.e. they include issues related to requirements, design, maintenance and project management. The issues categorized as D show a similar pattern as the most critical ones (A and B), i.e. they are related to requirements, and verification and validation. Furthermore, one issue is related to design. As mentioned earlier, less than half of the issues classified as C and D have been identified in literature before. An explanation of the issues not yet identified has been provided in the qualitative analysis (see Section 5).

It is also interesting to observe that a majority of local issues is related to project management and maintenance (see Table 4). Thus, it seems that there is a high number of issues which do not have such an impact on the process that knowledge about them spreads in the organization.

8 Conclusion

This case study investigates issues related to the waterfall model applied in the context of large-scale software development and compares the findings with literature. The results are that the most critical issues in waterfall development are related to requirements and verification. In consequence, the waterfall model is not suitable to be used in large-scale development. Therefore, the company moved to an incremental and agile development model in 2005. The comparison of the case study findings with literature shows that all issues found in literature are found in the case study. Though, the case study findings provide more detailed explanations of the issues and identified four new issues, namely 1) confusion of who implements which version of the requirements, 2) high effort for maintenance, 3) specialized competence focus and lack of confidence of people, and 4) problems in fault localization due to communication barriers.

References

1. Royce, W.: Managing the development of large software systems: Concepts and techniques. In: Proc. IEEE WESCOM, IEEE Computer Society Press (1970)
2. Sommerville, I.: Software Engineering (7th Edition). Pearson Eductation Ltd. (2004)
3. Raccoon, L.B.S.: Fifty years of progress in software engineering. SIGSOFT Softw. Eng. Notes **22**(1) (1997) 88–104
4. Yin, R.K.: Case Study Research: Design and Methods, 3rd Edition, Applied Social Research Methods Series, Vol. 5. Prentice Hall (2002)
5. McBreen, P.: Software craftsmanship : the new imperative. Addison-Wesley, Boston (2002)
6. Pfleeger, S.L., Atlee, J.M.: Software engineering : theory and practice. 3. ed. edn. Prentice Hall, Upper Saddle River, N.J. (2006)
7. Jarzombek, J.: The 5th annual jaws s3 proceedings (1999)
8. Thomas, M.: It projects sink or swim. British Computer Society Review 2001 (2001)
9. Johnson, J.: Keynote speech: Build only the features you need. In: Proceedings of the 4th International Conference on Extreme Programming and Agile Processes in Software Engineering (XP 2002). (2002)
10. Jones, C.: Patterns of Software Systems: Failure and Success. International Thomson Computer Press (1995)
11. Sametinger, J.: Software engineering with reusable components : with 26 tables. Springer, Berlin (1997)
12. Anderson, D.J.: Agile Management for Software Engineering: Applying the Theory of Constraints for Business Results (The Coad Series). Prentice Hall PTR (2003)
13. Cohen, D., Larson, G., Ware, B.: Improving software investments through requirements validation. In: Proceedings of the 26th Annual NASA Goddard Software Engineering Workshop (SEW 2001), Washington, DC, USA, IEEE Computer Society (2001) 106
14. Boehm, B.: Get ready for agile methods, with care. Computer **35**(1) (2002) 64–69
15. Karlström, D., Runeson, P.: Combining agile methods with stage-gate project management. IEEE Software **22**(3) (2005) 43–49