A Multiple-Case Study of Software Effort Estimation based on Use Case Points

Bente Anda, Hans Christian Benestad and Siw Elisabeth Hove

Simula Research Laboratory,
P.O. Box 134, NO–1325 Lysaker, Norway
Tel. +47 67828306
{bentea,benestad,siweh}@simula.no

Abstract

Through industry collaboration we have experienced an increasing interest in software effort estimation based on use cases. We therefore investigated one promising method, the use case points method, which is inspired by function points analysis. Four companies developed equivalent functionality, but their development processes varied, ranging from a light, code-and-fix process with limited emphasis on code quality, to a heavy process with considerable emphasis on analysis, design and code quality.

Our effort estimate, which was based on the use case points method, was close to the actual effort of the company with the lightest development process; the estimate was 413 hours while actual effort of the four companies ranged from 431 to 943 hours. These results show, that the use case points method needs modification to better handle effort related to the development process and the quality of the code.

1. Introduction

Use cases are often used as input for estimating software development effort. Several studies show that a particular estimation method based on use cases, the use case points method, performs well early in a project [3-5,7,17,19]. This method requires little technical insight and little effort. However, the studies reveal a need for more scientific evaluation of the method.

We investigated the use case points method in a multiple case study in which 35 companies responded to a tender for a web-based system. The estimates from the companies that bid for the system ranged from 78 to 654 hours, with a mean of 275 hours. Four companies were chosen to develop the system. The size and qualifications of the development teams from the different companies were very similar. The Software Engineering Department at Simula Research Laboratory was the client for the system, the purpose of which was to handle information about the studies conducted by the department.

The four companies implemented equivalent functionality, but they followed different development processes. The processes ranged from a very light, code-and-fix approach with little focus on the quality of the code, to a heavy process with much emphasis on front-end activities such as analysis, design and project management and focus on quality in both product and project. The companies recorded effort on each of the use cases for the system daily. The effort data was validated by a research assistant. The detailed effort data allowed us to evaluate the number of transactions as a measure of the size of the functionality of the use cases. It also allowed us to investigate how well the use case points method handles effort related to satisfying quality requirements on the code and on the development process. To the authors' knowledge, there are no other reported studies in software engineering in which development teams from different companies have developed systems based on the same functional requirements specification.

The results from this study show that the effort spent on realizing the individual use cases was strongly related to the number of transactions of the use cases. This supports the use of transactions as a measure of the size of the functionality.

The use case points method estimated necessary effort for this system to be 413 hours based on the use cases that describe the functionality of the system and the assumptions that the non-functional requirements were trivial and that the development teams were well-qualified for the task. The actual effort spent on the

0-7803-9508-5/05/$20.00 ©2005 IEEE

project ranged from 431–943 hours for the four companies. The minimum effort was spent by a company that did not have a defined development process and that had little focus on the quality of the code.

The use case points method adjusts the size of the functionality of the system based on a number of technical and environmental factors. The technical factors are related to non-functional requirements on the system, while the environmental factors characterize the development team and its environment. The influence of these factors on the estimate was in this case much smaller (a 16% increase in the estimate) than the increase in actual effort spent by the companies that emphasized the development process and the quality of the code (an increase in actual effort of more than 100%).

The remainder of this paper is organized as follows. Section 2 describes the use case points method and experiences with it, and gives a brief overview of related work on size measures and estimation methods. Section 3 describes the multiple-case study, i.e., the requirements and the four development projects. Section 4 describes the research method. Section 5 discusses transactions as a measure of size of the use cases. Section 6 describes how the use case points' estimate was produced and compares the estimate with actual effort expended on the four systems. Section 7 discusses the scope and validity of the results. Section 8 concludes and gives directions for future work.

2. The Use Case Points Method

This section describes the use case points method, previous experiences with it, and gives a brief overview of related work on size measures and estimation methods.

2.1 The Method

The use case points method was proposed by Karner, who also validated it on three projects [13]. The method is an extension of *MKII Function Points Analysis* [23]. Table 1 gives a brief overview of the steps of the method. In step 4, there are 13 technical factors, which are basically non-functional requirements on the system, see Table 2. There are also eight environmental factors that relate to the efficiency of the project in terms of the qualifications and motivation of the development team, see Table 3. The weights and the formula for technical factors are borrowed from the Function Points method proposed by Albrecht [1]. Karner himself proposed the weights and the formula for the environmental factors based on interviews with experienced developers and some estimation results.

In step 6, the adjusted use case points (UCP) is multiplied by a productivity factor. The literature on the UCP proposes from 20 to 36 person hours per use case point (PHperUCP) depending on the values of the environmental factors [13,20].

Table 1. The UCP estimation method

Step	Rule	Output
1	Classify actors: a) Simple, WF (Weight Factor) = 1 b) Average, WF = 2 c) Complex, WF = 3	Unadjusted Actor Weights (UAW) = ∑(#Actors * WF)
2	Classify use cases: a) Simple- 3 or fewer transactions, WF = 5 b) Average- 4 to 7 transactions, WF = 10 c) Complex- more than 7 transactions, WF= 15†	Unadjusted Use Case Weights (UUCW) = ∑ (#Use Cases * WF)
3	Calculate the Unadjusted Use Case Point (UUCP).	UUCP = UAW + UUCW
4	Assign values to the technical and environmental factors [0..5], multiply by their weights [-1..2], and calculate the weighted sums (TFactor and EFactor). Calculate TCF and EF as shown.	Technical Complexity Factor (TCF) = 0.6 + (0.01 * TFactor) Environmental Factor (EF) = 1.4 + (-0.03 * EFactor)
5	Calculate the adjusted Use Case Points (UCP).	UCP = UUCP * TCF * EF
6	Estimate effort (E) in person-hours.	E = UCP * PHperUCP

†The number of scenarios has been proposed as an alternative measure of the complexity of the use cases [18]. Analysis classes may also be taken into account when determining the complexity of a use case [8].

Table 2. Technical factors

Factor	Description	Weight
T1	Distributed system	2
T2	Response or throughput performance objectives	2
T3	End-user efficiency	1
T4	Complex internal processing	1
T5	Reusable code	1
T6	Easy to install	0.5
T7	Easy to use	0.5
T8	Portable	2
T9	Easy to change	1
T10	Concurrent	1
T11	Includes security features	1
T12	Provides access for third parties	1
T13	Special user training facilities are required	1

Table 3. Environmental factors

Factor	Description	Weight
F1	Familiar with Rational Unified Process	1.5
F2	Application experience	0.5
F3	Object-oriented experience	1
F4	Lead analyst capability	0.5
F5	Motivation	1
F6	Stable requirements	2
F7	Part-time workers	-1
F8	Difficult programming language	-1

2.2 Experiences with the Use Case Points Method

Our research group has conducted several studies in which the use case points method was evaluated and adapted to different industrial projects [3-5,17,19]. Table 4 offers an overview of these projects. The studies investigated the assessment of the size of the use cases, the effect of the technical factors and the productivity factor. All of these projects used variations of Rational Unified Process [14], and Java was the main programming language in all the projects, except in Study 6 where it was C++.

The number of transactions of the use cases was used to calculate use case points in all projects, but Study 5. In that study the use cases were not detailed out with transactions, and therefore the project manager assessed the complexity of each use case. Use case points estimation was performed by a researcher with access to requirements specifications and team members who could answer questions about the technical and environmental factors. The studies showed promising results for the use case points method, and the method also showed promise in a study in which the method was used to estimate effort in incremental development [17]. Nevertheless, in these studies we were unable to evaluate the different steps of the method in detail. This motivated a study with detailed collection of time data, and in which some factors were stable over several projects while others varied.

We have not found other studies than these that have evaluated the use case points method in an industrial project. Use case points have, however, been found useful to measure system size [7]. Four estimation tools, three of them commercially available, implement the use case points method or variants of it [15,25-27]. A relationship between external use cases and source lines of code (SLOC) was found in projects with students [8]. An approximate number of hours to implement a use case, depending on the size of system, was suggested in [21]. The challenge of measuring the size of use cases that hide complex business logic, and suggestions for improving use case based estimation to handle this issue, can be found in [24]. Use cases are also often used informally in estimating effort [4].

2.3. Related Work on Cost Estimation Methods

Many measures have been proposed for the size of the functionality of software systems. Such measures are useful input to estimating software development effort, both when this is performed by experts and when it is performed using a formal estimation method.

If a use case model is available, use case points is a size measure that is simple to calculate and that can be used early in a project. Other examples of size measures are Function points and Object Points. Function points is the oldest and most widely used measure. It measures the amount of data that each function accesses [1,23]. There are several adaptations of Function points to object-oriented contexts, examples are [2,6]. Object points measure the size of object-oriented software and are derived from class structures, messages and use cases [22].

Table 4. Characteristics of projects evaluating the use case points method

Study	Ref.	Application domain	Team size	No. of use cases	Non-functional requirements	UCP estimate (hours)	Actual Effort (hours)
1	**[5]**	Finance	6	9	End-user efficiency	2550	3670
2	**[5]**	CRM for a bank	6	16	End-user efficiency, usability	2730	2860
3	**[5]**	Banking	5	11	As above	2080	2740
4	**[4]**	Internet shopping	8	22	End-user efficiency, usability	4086	3360
5	**[19]**	Banking	6	63	Short response time, usability and changeability	10831	10043
6	**[19]**	Real-time system	14	63	Short response time, changeability and security	14965	13933

Function points can be counted from the use cases of a system [10], and in those cases the number of function points of a system will most often be related to the number of actors and transactions, that is, to the number of use case points. A major challenge with use case points when compared with other size measures, is that there is no standard format for describing use cases. Consequently, the use of use case points requires caution in the description of the use cases.

The use case points method adjusts the use case points based on a number of technical and environmental factors. This is similar to MKII Function Points, which also adjusts the size based on a number of calibration factors [1,23]. These factors have been criticized for not improving the precision of the estimate [16]. The criticism relates both to the chosen set of factors and to their influence, but little investigation has been done on the effects on effort of the individual factors. Our experience is also that it is difficult to know which calibration factors to include when estimating a particular system, and to assign values to them [5].

In the multiple-case study reported in this paper, two factors varied among the development teams; the development process and the emphasis on the quality of the code. This allowed us to study the impact of these factors on the actual effort and to compare this with how effort related to these factors is handled by the use case points method.

3. The Development Projects

The system to be developed was a web-based system for handling information about the studies conducted by the Software Engineering Department at Simula Research Laboratory. Thirty-five Norwegian development companies voluntarily responded to a tender. The tender included a requirements specification with 3 actors and 9 use cases. The use cases were described at user goal level with transactions [9], and the business logic was simple. An example of a use case is given in Figure 1.

The non-functional requirements, as specified by the client, were inherent in the technology that was used to develop the system. The requirements specification did not include specific requirements on the quality of the code.

The bids included a firm price, ranging from 21 000 NOK to 559 500 NOK[1], an effort estimate, ranging from 78–654 hours with a mean of 275 hours, and a description of the company's development process. The bidding process is described in detail in [11].

Four companies (henceforth, A, B, C and D) developed individual systems. The companies were chosen based on both business and research criteria. The business criteria were price, experience with similar solutions, experience with the programming environment (Java), size of the company (in order to enable similar teams) and apparent understanding of the requirements.

[1] 100 Norwegian Kroner (NOK) is (Sept. 1, 2005) about 12.7 Euro.

Create or edit study

Preconditions:

1. Login has been executed, user is authenticated, and authorized as study administrator.
2. An existing study has been selected using the use case "Select and report study", or a command has been given to create a new study.

Steps:

1. The user inserts study information. Study responsible and publications may be selected from lists. Study material files may be attached to the study by uploading files accessible from the user's machine.
2. The user selects a command to save the study to the database.
3. The system validates the data, writes the information to the database, and displays a message that the study was stored properly.

Exception:

3.1 If validation fails, the system displays a message containing information that will help the user to complete registration. He may then start at step 2.

Post conditions:

1. The study has been saved to the database. Time and user has been logged.

Figure 1. Example of a use case

We wanted companies that were likely to complete the project in a satisfactory way, as failure to do that would invalidate our research.

The four companies developed the systems in parallel and used from nine to 13 weeks. Before acceptance, the systems were thoroughly tested by the client (us). The functionality developed by the different companies was very similar. It was emphasized that functionality additional to that specified in the requirements specification was not desired. One of the final systems can be found at http://www.simula.no/des

4. Research Method

The research method was a multiple-case study with four development projects developing the same system in parallel.

Case studies are needed to investigate and evaluate software engineering methods in realistic contexts. Investigating and evaluating estimation methods require access to detailed data from software development projects. Such data is often difficult to obtain, and it may also be difficult to ensure that available data is of sufficient quality. We solved this by contracting the companies. The organization of the project and the collection and validation of data are described below.

4.1. Organization of Research and Development Projects

In this project, the Software Engineering Department at Simula Research Laboratory was both researcher and client on the same project. This required a separation of concerns, consequently the people involved in the project were organized in two separate teams. One team had the role of the client and consisted of a project manager and user representative. They were both employed by Simula at the start of the project and had long experience from IT companies, but no research experience. The other team was responsible for the research and consisted of a researcher and a research assistant. In addition, an experienced, external consultant was hired to ensure that we behaved realistically in the role of client, and that the development projects were affected as little as possible by the research.

The requirements specification was developed by three people who were not directly involved in the rest of the project.

4.2. Choice of Companies

The companies' price and development process, as described in the bids, were used in the selection of companies. We chose four companies that differed on these issues, to ensure that their proposals would span the field of likely development processes for such a project. It was made clear to the companies that although the prices were fixed, we would not use it against them if they reported less than the estimated effort. That is, they should not feel obliged to expend all the estimated effort. Table 5 shows the price, estimate in hours and development processes of the four companies.

The following aspects of the development projects were kept as equal as possible:

- The *development teams* were similar in terms of size and qualifications to avoid large differences in effort due to these factors. All the teams consisted of one project manager and two developers. Some of the companies also used additional resources when needed. The developers

all had at least three years' experience with Java development.

- All companies used Java as the *development language*, but they used different development tools. They were asked not to use pre-existing software components besides standard open-source components for the java platform (like Struts and log4j), and company-specific libraries.
- The *communication* between our client team and the development teams was done with the issue tracking system Bugzero [28]. The client team made a significant effort to behave in a similar manner towards the different companies.
- The companies and the people involved knew that they were participating in a research project, and agreed to it. They were paid extra for participating in research activities such as interviews, and preparing and sending time sheets. Effort expended on these activities was recorded separately.

Table 5. Characteristics of the development projects

Company	Price (NOK)	Estimate (hours)	Development process
A	160 000	220	A rather light, mainly code-and-fix process (focus on visual design).
B	360 000	341	A heavy process with focus on analysis and design
C	70 000	100	Very light code-and-fix process
D	450 000	650	A heavy process with focus on analysis, design and project management.

4.4. Data Collection

The following data was collected about the development processes and the effort spent on the project:

- Bids: The companies gave brief descriptions of their development process with their bids.
- Contract meeting: Contract meetings were held with the four companies; these meetings also provided information about their development processes.
- Curricula vitae: We received CVs for all the team members.
- Time sheets: The team members kept daily records of effort on use cases and activity during the development project. They were asked to record effort on use cases where appropriate, and instructed that effort on several use cases should be distributed over the appropriate use cases. The time sheets were sent to the researchers daily, and the research assistant checked that the recorded time was realistic. In case of anomalies, the team members were asked how effort had been expended.
- Interviews: The team members were interviewed about their qualifications, their development process, and their priorities with respect to non-functional requirements. The interviews were also used to validate the time sheets.

5. Measuring the Size of Use Cases

The number of transactions of the use cases is a much used measure of the size of the use cases, and the use case points method uses the number of transactions as a basis for assigning a number of use case points to each use case. Nevertheless, there has been little evaluation of how well the number of transactions of the use cases correlates with effort expended on implementing them.

Table 6 gives an overview of the use cases in this system and the number of transactions in each. The effort recorded on each use case is the total effort required to realize it, including analysis and design, coding, testing and error correction.

Two people were involved in describing the use cases. One person, not involved in the use case modelling, but with good knowledge of the requirements for this system and of use case modelling in general, assessed the number of transactions for each of the use cases. This involved determining which steps of the use cases were parts of the same transaction. It was a simple process, but for a couple of the use cases, knowledge of the requirements was used in the assessment together with the use case descriptions. It is usual to involve expert opinion when software size measures are used [12].

There was a strong correlation between the effort expended on use cases and the number of transactions of the use cases (Pearson correlation = 0,729, P-Value = 0,000). This supports claims that transactions are a good measure of the size of use cases, at least in situations in which the use cases are detailed and describe the business logic.

Table 6. Use case transactions and effort for A, B, C and D in hours

Use case	Trans.	A	B	C	D
1.Delete study	1	19	37	9	18
2.Create aggregated study report	1	22	52	5	27
3.Manage system properties	1	23	29	5	14
4.Grant administrative privileges	1	31	48	3	32
5.Manage system data	2	21	10	22	22
6.Login	2	37	36	22	35
7.Create csv-report	3	57	143	16	116
8.Select and report study	3	67	136	40	82
9.Create or edit study	4	157	148	99	147
Total on use case		434	639	221	493
Effort not assigned to use cases		153	304	210	336

Company C recorded less effort than did the other companies on use cases. We believe that this was because they worked in a less structured manner and consequently were less conscious about exactly which use case they were working on. There were variations in how much of the total effort the companies spent on each of the use cases. These variations can be due to the following:

- Use cases that were implemented early might have required more effort than use cases that were implemented later in the development project, and companies varied as to which use cases they implemented first.
- The amount of user interaction varies among the use cases. This variation combined with the different emphasis on visual design among the companies means that there might have been some variation on the effort spent on the user interface of the use cases.

6. Estimating the System with Use Case Points

Using the number of transactions of each use case as a basis for measuring the number of use case points, the functionality of this system corresponds to 57 unadjusted use case points. Furthermore, we initially set all the technical factors to zero since the non-functional requirements of the system, as specified by us, were trivial and inherent in the technology used. The first six environmental factors were correspondingly set to 5 because the teams were well qualified. The environmental factor "part time workers" was set to 3 since many of the team members worked on other projects in parallel. The environmental factor "programming language" was also set to 3 because we considered Java as an averagely difficult language. Figure 2 shows how the system is estimated.

The productivity factor is set to the minimum value of 20 since the values of the environmental factors indicate a simple project.

1 simple actor with weight 1 + 2 complex actors with weight 3
→ UAW = 7
+
8 simple use cases with weight 5 + 1 average use case with weight 10
→ UUCW = 50

→ UUCP = 57

T1 – T13 are assigned the value 0 → TCF = 0.6

E1 – E6 are assigned 5 and E7 – E8 are assigned 3
→ EF = 0.605

→ UCP = 57 * 0,6 * 0,605 = 20,619

PHperUCP (the productivity factor) is set to 20

→ Estimated effort (E) = **413 hours**

Figure 2. Estimating effort for realizing the system

The use case points method gives an estimate of 413 hours, meaning that for a qualified development team it should be possible to produce an acceptable system realizing these nine use cases in 413 hours.

The actual effort of the four companies was 587 hours (A), 943 hours (B), 431 hours (C) and 829 hours (D). This effort included effort on all activities in the project. The effort estimated by the use case points

method (413 hours) is close to the minimum effort for implementing the system (431 hours). Furthermore, it is closer to actual effort than is the expert estimate for all companies, except company D (Table 5). The use case points estimate is also closer to the actual effort than were most of the expert estimates from the companies that bid for the system.

There was a large variation in the actual effort among the companies, even though they developed the same functionality, used the same programming language and had similar qualifications. The companies followed different development processes, with varying emphasis on analysis, design and project management. The differences in development process entailed different assumptions on the non-functional requirements on the system with respect to the quality of the code. Before delivery for acceptance test the team members were asked in interviews how they considered the quality of their own code in terms of maintainability and reusability. Their responses are shown in Table 7.

Table 7. Subjective assessment of code quality

Company	Opinion on code quality
A	The quality is acceptable.
B	The quality is acceptable.
C	It is too costly to plan for changes, so we have not focused on changeability or reusability.
D	We have focused on code quality and expect this to be good.

The use case points method assumes an adapted version of RUP. The method is consequently not adapted to handling differences in effort due to different development processes.

Two technical factors in the use case points method handle effort related to the quality of the code, those are effort on implementing reusable (T5) and changeable (T9) code. Setting these factors to 3 (average importance)as was assumed by companies A and B, led to an increase in the estimate to 455 hours. Setting these to 5 (much emphasis), as assumed by company D, led to an increase in the estimate to 482 hours. Varying both of these from their minimum value (0) to their maximum value (5) increases the estimate by approximately 16%

In our opinion, these results indicate that the use case points method may support experts in estimating the necessary effort for an acceptable solution that satisfies the functional requirements of a system.

Nevertheless, the results also show that a heavier development process with an increased emphasis on the quality of the code led to a large increase in actual effort, and that the use case points method needs modification to handle effort related to such non-functional requirements. This supports previous results on effect of the complexity factors in similar estimation methods [16].

7. Scope and Validity of the Results

This section discusses the scope and the validity of the results from this study.

7.1. The Scope of the Results

The use cases were described with much detail and the business logic was simple. The development projects were small, although real projects, the non-functional requirements were trivial and the development teams were all sufficiently qualified for the task. The members of the teams that developed the systems were asked in interviews whether they found this development project realistic, that is, to what extent it resembled other projects with which they had experience. They all considered the project realistic, but some thought it to be most similar to developing a subsystem of a larger system. We therefore expect the results from this study to be applicable in similar industrial development projects, and also that they could serve as a basis for applying use case points in larger projects.

The use case points method may be particularly useful in situations where little technical expertise and experience from similar projects are available. An example of such a situation is that of a software client requesting a new system and facing the challenges of (1) adjusting the desired functionality to the available budget, and (2) choosing a software contractor requesting a realistic price.

7.2. The Validity of the Results

The expert estimates and actual effort with which the use case points estimates were compared in this study were obtained in a real bidding process and in real development projects. In our opinion the effort data with which the use case points estimate is compared, is of high quality. Effort was recorded daily by the team members in detailed time sheets, and was validated by a research assistant. Most of the teams recorded effort in more detail on this project than they usually did. They were, therefore, asked in interviews about their experiences with recording effort at this level of detail. Most of the team members had not

found this problematic, but some felt that it was difficult to be precise at this level of detail.

There are no fixed rules or guidelines for describing use cases. Consequently, the use cases for this system can be described in different ways, which could result in slightly different numbers of transactions and hence of use case points and estimate. Nevertheless, a standard use case format was applied based on [9], and the description and structuring of the use cases for the purpose of estimating was performed independently of the estimation. It was done by a person who was familiar with the system to be developed from the point of view of a client and who had little familiarity with the use case points method.

We contribute the large differences in effort to differences in the development process and in the emphasis on the quality of the code. Several sources were used to determine the differences between the companies' with respect to these issues: Bids, contract meetings, time sheets and interviews. We made a significant effort to ensure that the conditions of the development teams were similar with respect to their size and competencies as well as programming language used. We also took measures to separate the roles of researcher and client and to behave consistently towards the four teams.

In the comparison of the effort on the four systems, we disregarded differences in usability, but there were no large differences between the systems.

8. Conclusion and Future Work

We have investigated the application of use cases in estimating software development effort in a multiple-case study, in which 35 companies bid for developing a system. Four of those companies were chosen to actually develop a system based on the same requirements specification. The teams from the four companies had very similar qualifications, and the functionality of the four resulting systems was almost equivalent. The teams followed different development processes and placed different emphasis on the quality of the code quality. We estimated the necessary effort for developing the system using a particular method for estimating based on use cases; the use case points method.

The results from this study support previous claims that the use case points method may support early estimation of software development effort. Necessary effort was estimated to 413 hours by the use case points method, and the minimum effort of the development teams was 431 hours.

A heavier development process and more emphasis on the quality of the code increased effort by more than 100%, to a maximum of 943 hours. Our results show, however, that the use case points method is not sufficiently adapted to handling increase in effort due to development process and quality requirements on the code. These results therefore motivate more studies on the impact of these factors on software development effort of requirements. More studies are also needed on how to apply use cases as a size measure.

The following activities are planned for future work:

- The code of the four systems will be analyzed to identify in more detail how they differ.
- The development processes followed by the four companies, and the effects of these processes, will be investigated in more detail. Interviews, time sheets and code will be used to describe the development processes followed by the four companies. In addition, effort data from Simula's client team will be used to investigate the effect on the client.
- The use case points method will be applied to more projects to provide improved guidelines for how to measure size and assess productivity in different kinds of project.

Acknowledgements

We gratefully acknowledge the following, without whom a research project of this scale would not have been possible: the companies for their participation and willingness to contribute to our research; Dag Sjøberg for initiating this work and obtaining the necessary funding, as well as for comments on this paper, Magne Jørgensen for laying the foundations for this work through his own research on the bidding process; Stein Grimstad for his job as Simula's user representative and tester for the system; Vigdis By Kampenes for her help in interviewing the development teams and testing the final systems; Gunnar Carelius for his help in choosing development companies and for technical support during the project, and Glen Farley for his advice during the whole project.

9. References

[1] Albrecht, A.J. Measuring Application Development Productivity. In *Proceedings of the IBM Applic. Dev. Joint SHARE/GUIDE Symposium,* Monterey, CA, USA, pp. 83-92, 1979.

[2] Abrahão, S., Poels, G., and Pastor, O. Assessing the Reproducibility and Accuracy of Functional Size Measurment Methods through Experimentation. The 2004 International Symposium on Empirical Software Engineering (ISESE), Redondo Beach, California, August 19-20, 2004, pp. 189-198.

[3] Anda, B. Comparing Use Case based Estimates with Expert Estimates. Proc. of Empirical Assessment in Software Engineering (EASE), Keele, United Kingdom, April 8-10, 2002.

[4] Anda, B., Angelvik, E. and Ribu, K. Improving Estimation Practices by Applying Use Case Models. The 4th International Conference on Product Focused Software Process Improvement (PROFES), Finland, December 9-11, 2002, pp. 383-397, LNCS 2559, Springer-Verlag,

[5] Anda, B., Dreiem, H., Sjøberg, D.I.K., and Jørgensen, M. Estimating Software Development Effort Based on Use Cases – Experiences from Industry. The 4th International Conference on the Unified Modeling Language, Concepts, and Tools (UML), Canada, October 1-5, 2001, pp. 487-502, LNCS 2185, Springer-Verlag.

[6] Antoniol, G., Lokan, C., Caldiera, G., and Fiutem, R. A Function point-like Measure for Object Oriented Software. *Empirical Software Engineering,* 4(3), pp. 263-287, 1999.

[7] Arnold, P. and Pedross, P. Software Size Measurement and Productivity Rating in a Large-Scale Software Development Department. The 20th International Conference on Software Engineering (ICSE), Kyoto, Japan,, April 19-25, 1998, pp. 490-493.

[8] Chen, Y., and Boehm, B. An Empirical Study of eServices Product UML Sizing Metrics. The 2004 International Symposium on Empirical Software Engineering (ISESE), Redondo Beach, California, August 19-20, 2004, pp. 199-206.

[9] Cockburn, A. *Writing Effective Use Cases*. Addison-Wesley, 2000.

[10] Fetcke, T., Abran, A., and Nguyen, T-H. Mapping the OO-Jacobson Approach into Function Points analysis. International Conference on Technology of Object-Oriented Languages and Systems (TOOLS-23). IEEE Comput. Soc, Los Alamitos, CA, USA, pp. 192-202, 1998.

[11] Jørgensen, M. and Carelius, G.J. An Empirical Study of Software Project Bidding. *IEEE Transactions on Software Engineering*, 30(12), pp. 953-969, 2004.

[12] Jørgensen, M. and Gruschke, T.M. Industrial use of Formal Software Cost Estimation Models: Expert Estimation in Disguise? Proc. of Empirical Assessment in Software Engineering (EASE), Keele, United Kingdom, April 11-13, 2005.

[13] Karner, G. Resource Estimation for Objectory Projects. Objective Systems SF AB. 17. September, 1993.

[14] Kructhen, P. *The Rational Unified Process. An introduction*. 3rd ed. Pearson Education, inc. 2004.

[15] Kusumoto, S., Matukawa, F., Inoue, K., Hanabusa, S. and Maegawa, Y. Estimating Effort by Use Case Points: Method, Tool and Case Study. The 10th International Symposium on Software Metrics, Illinois, September 14-16, 2004, pp. 292-299.

[16] Lokan., C. and Abran, A. Multiple Viewpoints in Functional Size Measurement. Proc. of the International Workshop on Software measurement (IWSM'99), Lac Supérieur, Canada, September 8-10, 1999.

[17] Mohagheghi, P., Anda, B. and Conradi, C. Effort Estimation of Use Cases for Incremental Large-Scale Software Development. Proc. of the 27th International Conference on Software Engineering (ICSE), St Louis, Missouri, USA, May 15-12, 2005, pp. 303-311.

[18] Probasco, L. Dear Dr. Use Case: What about Function Points and Use Cases? http://www-128.ibm.com/developerworks/rational/library/2870.html, 2002.

[19] Ribu, K. Estimating Object-Oriented Software Projects with Use Cases. MSc thesis, November 2001.

[20] Schneider, G. and Winters Jason P. *Applying Use Cases – A practical guide. 2nd ed*. Addison-Wesley, 2001.

[21] Smith, J. The Estimation of Effort Based on Use Case. Rational Software white paper. TP-171 10/99, 1999.

[22] Sneed, H. Estimating the Development Costs of Object-Oriented Software. Proc. of the 7th European Software Control and Metrics Conference, Wilmslow, UK, 1996.

[23] Symons, P.R. *Software Sizing and Estimating MK II FPA (Function Point Analysis)*. John Wiley & Sons, 1991.

[24] Vinsen, K., Jamieson, D. and Callender, G. Use Case Estimation – The Devil is in the Detail. The 12th International Requirements Engineering Conference (RE), Kyoto, Japan, September 6-10, 2004, pp. 10-15.

[25] http://www.duvessa.com

[26] http://www.tassc-solutions.com/

[27] http://www.sparxsystems.com/

[28] http://www.websina.com/bugzero/