

Improving Estimation Practices by Applying Use Case Models

Bente Anda¹, Endre Angelvik² and Kirsten Ribu¹

¹Simula Research Laboratory, P.O. Box 134,
NO-1325 Lysaker, Norway
{bentea, kribu}@simula.no

²Mogul Norway, Drammensveien 134,
NO-0277 Oslo, Norway
endre.angelvik@mogul.com

Abstract. An estimation method based on use cases, *the use case points method*, has given promising results. However, more knowledge is needed about the contexts in which the method can be applied and how it should be adapted to local environments to improve the estimation process. We applied the use case points method to several projects in a Scandinavian software development company as the first activity in a software process improvement project on improving estimation. The second activity of the improvement project was to conduct interviews with project managers and senior developers about how to obtain continued and more widespread use of the method in the company. Based on the interviews, we propose a tailored, potentially improved version of the method and suggest how estimation practices can be improved by applying it. We believe that these experiences may be of interest to other companies that consider applying use case models as part of their estimation practices.

1 Introduction

A use case model describes the functional requirements of a system to be constructed, and use case models are frequently used as input to the process of estimating software development effort. An estimation method based on use cases, the use case points method, was introduced by Karner [9]. This estimation method has been evaluated in several software development projects with promising results [2,3,12]; it was considered easy to use and performed similar to or better than teams of very experienced software developers. Nevertheless, more knowledge is needed about how to apply the method and tailor it to a specific organization.

We evaluated the use case points method on three projects as the first activity in a software process improvement project on improving estimation in a Scandinavian software development company, Mogul [3]. Since then, the company has also applied the method on a couple of other projects with success.

The improvement project is conducted as part of a Norwegian software process improvement project, PROFIT, with the Universities of Oslo and Trondheim, SINTEF and 12 software development companies as partners. The goal of Mogul's improvement project is to develop an estimation method based on use case models that is simple to use and that is supplementary to expert estimates.

The second activity in the project was to conduct interviews with project managers and senior developers to

1. understand the ordinary estimation process in the company,
2. find out how the method for estimation based on use cases can be tailored to the company, and
3. establish the necessary context for applying the method successfully.

It is often difficult to sustain software process improvement projects beyond the initial phase, so the interviewees were also asked about how a supplementary method could obtain continued and widespread use in Mogul.

This paper describes Mogul's ordinary estimation process and its current practices for use case modeling. Then, contrasting the ordinary estimation process with evaluated best practices for estimation [8], areas of Mogul's estimation process are identified that may be improved by applying the use case points method. The paper also discusses requirements on the use case model that must be fulfilled for the use case points method to be applicable.

Context information is often missing when new or improved estimation methods are reported. The work described in this paper may provide a background for other companies that wish to improve their estimation practices applying use case models.

A major result of the interviews is a proposed modification of the use case points method, which includes, for example, an alternative way of measuring the size of a use case and modified adjustment factors.

The remainder of this paper is organized as follows. The use case points method is described in Section 2. Section 3 describes the context of the study. Section 4 describes estimation practices in Mogul and how they can be improved. Section 5 describes current practices for use case modeling. Section 6 suggests how the use case points method can be tailored to the company. Section 7 concludes and suggests future work.

2 The Use Case Points Method

The use case points method was initially developed by Gustav Karner [9]. It is based on the function points method [1], and the aim was to provide a simple estimation method adapted to object-oriented projects. This section gives the steps of the method as described in [13]. The method requires that it should be possible to count the number of transactions in each use case. A transaction is an event occurring between an actor and the system, the event being performed entirely or not at all.

1. The actors in the use case model are categorized as *simple*, *average* or *complex* depending on assumed complexity. A weight is assigned to each actor category:

- Simple actor – another system with a defined API: weight = 1
- Average actor – another system interacting through a protocol: weight = 2
- Complex actor – a person interacting through a graphical user interface or web-page: weight = 3

The total *unadjusted actor weight (UAW)* is calculated counting the number of actors in each category, multiplying each total by its specified weight, and then adding the products.

2. The use cases are correspondingly categorized as *simple*, *average* or *complex*, depending on the number of transactions, including the transactions in alternative flows. A weight factor is assigned to each use case category:
 - Simple use case – 3 or fewer transactions: weight = 5
 - Average use case – 4 to 7 transactions: weight 10
 - Complex use case – more than 7 transactions: weight 15

The *unadjusted use case weights (UUCW)* is calculated counting the number of use cases in each category, multiplying each category of use case with its weight and adding the products. The UAW is added to the UUCW to get the *unadjusted use case points (UUPC)*.

3. The use case points are adjusted based on the values assigned to a number of technical factors (Table 1) and environmental factors (Table 2). These factors are meant to account for effort that is not related to the size of the task. Each factor is assigned a value between 0 and 5 depending on its assumed influence on the project. A rating of 0 means that the factor is irrelevant for this project; 5 means that it is essential.

The *technical complexity factor (TCF)* is calculated multiplying the value of each factor in Table 1 by its weight and then adding all these numbers to get the sum called the *TFactor*. Finally, the following formula is applied:

$$TCF = 0.6 + (.01 * TFactor)$$

The *environmental factor (EF)* is calculated accordingly by multiplying the value of each factor in Table 2 by its weight and adding all the products to get the sum called the *Efactor*. The following formula is applied:

$$EF = 1.4 + (-0.03 * EFactor)$$

The *adjusted use case points (UCP)* are calculated as follows:

$$UCP = UUCP * TCF * EF$$

Table 1. Technical complexity factors

Factor	Description	Wght
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 2. Environmental factors

Factor	Description	Wght
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

4. The number of person hours per use case point for a project estimate is determined by the environmental factors because these are considered to have a large impact on the actual effort [13]. The number of factors in F1 through F6 that are below 3 are counted and added to the number of factors in F7 through F8 that are above 3. If the total is 2 or less, 20 person hours per UCP is used; if the total is 3 or 4, 28 person hours per UCP is used. If the number exceeds 4, it is recommended that changes should be made to the project so the number can be adjusted, or alternatively that the number of person hours should be increased to 36 per use case point.

A spreadsheet is used to implement the method and produce an estimate. The method provides an estimate in total number of person hours.

The use case points method can be criticized from a theoretical point of view as has the function points method. The addition and subsequent multiplication of ordinal values, for example, is theoretically invalid [10]. However, the function points method has shown to predict effort reasonably well for many types of systems.

There are several other methods for use case based estimation. The methods differ in that size and complexity of the use cases are measured differently, and in that different technical and environmental factors are considered. Two alternative methods for estimation based on use cases are described in [6,14]. The method described in [6] maps attributes of the use case model into function points. In [14] a certain number of lines of code is assumed for each use case, and the total number of lines of code is used as a basis for the estimate. *Tassc:Estimator* is a commercial tool for estimation based on use cases [17]. A metric suite for use case models, which can be used for estimation, is suggested in [11], but a complete estimation method is not presented.

3 Context of Study

This section gives some characteristics of the company we studied, presents the results from the former case studies conducted to evaluate the use case points method, and describes the interviews with senior personnel of the company.

3.1 The Company

Mogul is a medium sized Scandinavian software development company located in Norway, Sweden and Finland. In Norway there are approximately 180 employees. The business area is software development for public and private sector, in particular banking and finance. Mogul's projects can roughly be divided into two types: traditional software development projects based on a three-layer architecture and web- projects, that is, intranet, internet or extranet solutions. The web-projects often consist in adapting existing systems to a web-environment. The company takes responsibility for complete projects or sell hours as consultants or mentors on methods and architecture. Mogul gives courses on the Rational Unified Process (RUP), which is also used in their own projects whenever possible.

3.2 Results from Case Studies

The use case points method has been evaluated in 3 development projects in Mogul. The estimates produced with the use case points method were compared with expert estimates and actual effort. The results were promising in that the estimates provided by the method were as accurate as the average estimates of the projects in the company. Table 3 shows some characteristics of the case studies. Table 4 gives the results.

3.3 The Interviews

The interviewees, 1 administrative manager, 7 project managers and 3 senior developers, had from 6 to 26 years experience with software development, and were chosen because they were very experienced estimators.

The interviews were semi-structured with open-ended questions to allow the respondents to speak more freely of issues they felt were important. They were conducted by one or two interviewers, lasted from 45 – 60 minutes and were tape recorded.

Table 3. Characteristics of three software development projects

Characteristic	Project A	Project B	Project C
Size	7 months elapsed time, 4000 staff hours	3 months elapsed time, 3000 staff hours	4 months elapsed time, 3000 staff hours
Software architecture	Three-tier, established before the project	Three-tier, known, but not established in advance	As project B
Programming environment	Java (Visual Café and JBuilder), Web Logic	MS Visual Studio	Java (Jbuilder), Web Logic
Project members	6 developers with 0 to 17 years experience	6 developers with 0 to 12 years experience	5 developers with 2 to 10 years experience, 4 consultants were involved part time.
Application domain	Finance	CRM (Customer relationship management within banking), part of a larger solution	Banking (support for sale of credit cards)

Table 4. Expert estimate, use case estimate and effort (in hours)

Project	Expert estimate	Use case estimate	Actual effort
A	2730	2550	3670
B	2340	3320 2730 ¹	2860
C	2100	2080	2740

4 Estimation Practices and Possible Improvements

This section describes current practices for estimation in Mogul based on the information from the interviews. The estimation practices are compared with best practices for estimation described in the literature to identify particular areas that may benefit from applying use case based estimation.

The two types of projects in the company are estimated differently, and are therefore treated separately below.

4.1 Estimating Traditional Software Development Projects

A project manager is responsible for producing a first estimate early in the inception phase. He/she may gather a team for the estimation process, but the actual developers are usually not allocated at this stage. The estimate indicates the need for resources, often together with a completion date. RUP gives generally good opportunities for

¹ The first estimate for project B, 3320 hours, was produced based on information about actors and use cases given by the project manager. In the second estimate, 2730 hours, several actors with a very similar user interface were generalized into one super actor, and included and extending use cases were omitted.

negotiating with the client about functionality; specified functionality is frequently changed and given new priorities along the way. It is also often possible to get more resources if necessary. The completion date, however, is often critical.

The estimate is typically based on a requirements specification from the client, possibly with a solution outline and some use cases. Several of the interviewees also develop a high-level use case model, based on the available information, which in turn is also used in the estimation process.

Some estimates are made in offer situations where Mogul is bidding to get a project. In such situations only the client's description of the functionality is available; and it is difficult to get more information. The company therefore depends on the clients' ability to describe what they actually want.

If the project mainly involves new development, Mogul's policy is to conduct a pre-project to clarify the requirements and construct a detailed use case model before committing to an estimate. However, the client often wants to know what kind of solution can be had for the price they can afford without paying for a pre-project, and it may therefore be difficult to avoid giving an early estimate based on insufficient information. One of the interviewees describes this situation using the analogy of buying a car: "You have all sorts of requirements for your new car, but you only have € 5000, so you wish to know what you can get for that amount of money".

The estimation process is bottom-up because the project is broken down into activities that are estimated separately, perhaps by different people. Sometimes two people are involved in estimating the same activity, either discussing to reach an estimate, or by letting an independent person go through the estimate afterwards. Mostly, however, estimation is done individually, and estimates for different parts are added to form the complete estimate. Several of the interviewees had their own methods or spreadsheets to help them in the estimation process.

The ability to identify risks is an important part of estimation. The interviewees claimed to be good at identifying technological risks, but believed themselves to be less good at identifying organizational risk.

The time for project management, in the order of 5-15%, is added to the estimate. The estimate must also take into account that much of the developers' time is spent on other activities such as meetings. The percentage of the developers' time believed to be available for development varied among the interviewees from 50% to 80%.

It may also be sensible to consider whether the client is in public or private sector. This may impact effort because more people tend to be involved in the decision process in the public sector. Expected lifetime for the system should also be considered because this has implications for the documentation and subsequently for the effort.

New estimates are usually produced in the elaboration phase, typically after the first iteration. The developers re-estimate their bits, for example, screens or modules and assess how much time is needed for completion.

Mogul does not keep track of the accuracy of its estimates, so it is impossible to assess the typical precision of their estimates. The interviewees stated, however, that the estimates are usually overrun.

4.2 Estimating Web-projects

The web-projects differ from the traditional development projects in that they are smaller, they more often build on an existing solution, and the functionality is less complicated. The most important part of these projects is establishing the information structure. According to the interviewees, 40% of the resources are typically used on this activity. An outline of a graphical design is a prerequisite for an estimate. The effort put into the graphical design will vary based on how much the client is willing to pay. A solution will also include a number of templates into which the users will fill in information. Each template typically takes one day to develop.

At present, estimating these projects is not difficult, but some of the interviewees expected the two types of projects to merge as traditional software projects start include advanced web interfaces.

4.3 Improving Estimation Practices

We have compared the ordinary estimation practices in Mogul with best practice principles for estimation [8] to identify how the use case points method can improve the estimation practices and thereby the accuracy of the estimates. Below we describe the best practice principles that are relevant in our context and how they can be fulfilled:

1. "Ask the estimators to justify and criticize their estimates."
A supplementary use case based estimate may, if it differs from the expert estimate, provide a basis for criticizing the expert estimate.
2. "Estimate top-down and bottom-up, independently of each other."
The company's expert estimates are made bottom-up. The use case points method, on the other hand, provides a top-down estimate. A top-down estimate is produced identifying some characteristics of the development project and using those as input to a complete estimate.
3. "Combine estimates from different experts and estimation strategies."
It has been shown sensible to combine models and human judgment [5], but more work is needed on how to best combine expert estimates and estimates produced with the use case points method.
4. "Assess the uncertainty of the estimate."
The spreadsheet used to produce an estimate with the use case points method makes it possible to vary the input both with regards to the number and size of the use cases and with regards to the different technical and environmental factors. This may help assess uncertainty due to unknown factors in the development project.

The use of an estimation method in combination with expert estimates can also lead to the avoidance of biases and large overruns, and estimation methods have been shown to perform better than expert estimators with little domain experience [7,8]. Therefore, the support given by an estimation method may make more people competent to take part in estimation.

5 Practices for Use Case Modeling

To be suitable as a basis for estimation, a use case model should be correct and described at an appropriate level of detail. This section gives a brief overview of how use case modeling is done in Mogul, and discusses challenges relating to correctness and level of detail of the use cases.

In Mogul, use case modeling is applied in traditional software development projects to identify and describe business logic. Use case modeling is usually not applied in web-projects because use cases lack the possibility to describe functionality where a web interface lets the user perform a function by switching among different web pages or where it is necessary to save current work and later resume it. Another problem is that the terminology in RUP is unfamiliar to several of the participants that typically take part in web-projects, for example, graphical designers.

Moreover, the use cases are perceived as belonging to and driving development projects; they are seldom maintained in the elaboration phase and never when the system has become operational. Therefore, the original use cases are often outdated and unsuitable as a basis for specifying modified functionality in maintenance projects.

5.1 Use Case Modeling Process

The use case modeling process in Mogul is as follows. In the inception phase, use case models may just be described at a high level without details. It may supplement the client's requirements specification or be derived from it.

A detailed use case model is usually constructed as part of a pre-project together with representatives of the client. The use case modeling process is a breadth-first process where the first activity is to identify actors and use cases, and then construct a use case diagram. Subsequently, the use cases are detailed out, possibly in several iterations. The participants from Mogul set up the structure, while the participants from the client fill in the details. The participants work individually on the different use cases and meet regularly to discuss them. The use cases may also be constructed solely by the clients. The use cases are often supplemented by screens and a domain model.

Pen and paper are often used to construct the use cases, and then Rational Rose is used to document the use case diagram and different templates, depending on the project, are used to document the use case descriptions. Some of the interviewees also use the add-on tools to Rational Rose, Requisite Pro or SODA, to document the use cases.

When the use case model is completed, the project participants, in particular those from the client, often read through the use case model to verify that the requirements are covered.

5.2 Correctness of the Use Case Model

A use case model should be correct in that the functional requirements of all user groups are included. The interviewees found the use case modeling useful because it helps focus on functionality from the point of view of the user and helps assure that the requirements of all the user groups are included. They also found the technique useful for obtaining a common understanding of the requirements and for reaching agreement with the client.

The use case modeling process can often be a maturity process for the clients; they are forced to think through what they actually want. One of the interviewees described it like this: “The clients’ domain expert thought she had a good overview of the requirements, but because of the use case modeling process we found out that not everybody agreed with her about what should be the functionality of the system.”

It may, however, be difficult to find end-users with sufficient competence and interest to participate in use case modeling. Some of the interviewees meant that use cases were too abstract for end-users. End-users may also be confused by the sequential description of the steps of the individual use cases and believe that the sequence must be followed strictly. They may also find it difficult to understand from the use case model how the individual use cases relate.

5.3 Level of Detail of the Use Cases

A balanced level of detail in the use cases is important when the use case model is to be used as a basis for estimation. If the use cases are unbalanced, there may be difficulties when measuring the size of the use cases with the use case points method.

The interviewees found it difficult to balance the use cases. In their opinion, use case descriptions tend to include too much detail. One of the interviewees described the problem in the following way: “The use cases tend to expand infinitely because to get complete descriptions we keep discussing unimportant details for a long time.” The proposed solution to this problem is to have good examples of use case models available, and to use tabular descriptions of the use cases to avoid too much text. Another solution may be to use specific guidelines in the use case modeling process as proposed in [4].

Since use cases describe functionality from the point of view of the end-users, they seldom provide sufficient architectural information, and the descriptions may hide complex business logic. These issues are described further in the next section.

6 Adapting the Use Case Points Method

The interviewees had experience from estimation based on use cases, and had suggestions for tailoring the use case points method, both with regards to measuring size (Section 6.1) and with regards to which technical and environmental factors were relevant in this particular company (Section 6.2). Section 6.3 discusses how to

estimate architecture when the use case points method is applied. Section 6.4 suggests how the use case points method can be more widespread in Mogul.

6.1 Assessing Size of the Use Cases

The use case points method takes the size of each use case as input. Size is measured in number of transactions in the use case descriptions. According to the interviewees, there are some problems with this measure:

- It is desirable to estimate with the use case points method in the inception phase, but at this stage the use cases may not sufficiently detailed out to show the individual transactions.
- When the use case descriptions are detailed out, they may be described at an unbalanced level of detail, which in turn may lead to skewed results due to inaccurate measure of size.
- The size measure does not capture complexity in the business logic and the architecture that may be hidden in the use case descriptions.

As a response to these difficulties, the interviewees suggested alternative ways of measuring size, for example, that weights could be assigned to each use case based on the intuition of the estimator or that the use cases could be used as a basis for identifying components to be estimated. However, these suggestions may contradict our goal of developing a method that requires little expert knowledge.

The following method was suggested by one of the interviewees as a supplement to counting transactions.

Consider for each use case what has to be done in the presentation layer, the persistence layer and the business layer:

1. The effort in the presentation layer will depend on the number of new screens, the number of transfers from one screen to another, the number of forms in the screens and the number of places where dynamic content must be generated.
2. The effort in the persistence layer will depend on the impact on the data model and persistent data, that is, on the number of new tables, the number of changes to table definitions, and the number of queries and updates on the tables.
3. The effort in the business layer is difficult to quantify as it may be anything from input to a database to complicated data processing, possibly also access to different back-systems. One of the interviewees described it this way: “The business logic may just be about transferring data, but you may find that you need a lorry to actually do it”. Our advice is, therefore, that the estimators should break down each use case sufficiently to form an opinion about the complexity of the business logic necessary for realizing it. If this is impossible, alternative estimates could be made for the most likely and the most pessimistic size of the use cases.

6.2 Adjustments Factors

In the use case points method, the estimate based on the size of the use cases is adjusted based on a number of technical and environmental factors. The method is inspired by the function points method, particularly the MkII function point analysis (MKII FPA) [15]. The two methods use several of the same technical factors. The technical factors of MkII FPA, however, have since been discarded [16]. They may be relevant early in a project's life-cycle when the requirements are stated in a general form, but when the requirements are detailed out, many of them will have influenced the functional requirements, so that adjusting the effort using the technical factors may lead to double counting. In [10] evidence is also presented that the adjustment factors applied in the function point method are unnecessary, particularly when the method is used in a single organization. In a case study, the use case points estimates for five projects were on average more accurate when the technical factors were omitted [12]. We therefore propose that the technical factors be omitted when the use case points method is applied to detailed use cases.

The environmental factors are not taken into account by the detailed use case descriptions and should therefore be considered. Some environmental factors may, however, be irrelevant to this particular company, and it may be necessary to consider other factors. The environmental factors regarding the development team, F1 – F6, were all considered relevant by the interviewees. Nevertheless, they stated that it would be beneficial to specify productivity and availability for each team member, instead of having to calculate an average, because there are large differences in productivity among developers. The interviewees also felt that they were usually too optimistic about the productivity of the team members. Regarding availability, many of the company's projects are located at the clients, which means that they are "at the mercy of the clients" regarding their ability to provide people with necessary knowledge about the application domain and technological infrastructure. The environmental factors may also be useful to show the client the consequences of uncertainties and risks in the project.

Requirements stability, F7, was considered irrelevant when using RUP, because one of the primary motivations for using RUP is that it gives the possibility to continually change the requirements.

Difficulty of the programming language, F8, was considered difficult to assess and therefore irrelevant because the development projects now require that the developers have knowledge about the technology used at each layer in the architecture.

6.3 Functionality versus Architecture

The interviewees meant that architecture mostly should be estimated separately from functionality: "The whole project can be estimated based on use cases only if you know the customer and the architecture well from previous projects, but if there is much uncertainty, the architecture should definitely be estimated separately."

Our goal is to develop a method that can provide a complete estimate, which requires that it can estimate a new or modified architecture. We therefore propose that

if an architecture already exists, the impact on the architecture should be considered for each use case and be used to adjust the size measures based on number of transactions.

We also propose, as did one of the interviewees, that the environmental factor F7, could be used to assess the architecture. A value of 5 (meaning new architecture or major changes to existing architecture) assigned to F7 increases the estimate by approximately 60% compared with the estimate produced when the value of F7 is 0 (meaning existing and stable architecture). One problem with this solution is, however, that the percentage of effort added for architecture is the same independently of the size of the project. In the interviewees' opinion, the proportion of the effort required for the architecture compared with the effort required for the functionality varies with the size of the project; the larger the project, the smaller is the proportion of effort needed to establish the architecture. One of the interviewees explained that many of the activities to establish the architecture must be done whether there are 5 or 50 use cases. He also mentioned as an example a project that took 8 months, and where 1/3 of the effort was on architecture and 2/3 on functionality. In a smaller project that took 3 months, 1/2 of the effort was spent on architecture and 1/2 on functionality.

6.4 Widespread Use of the Use Case Points Method in Mogul

The use case points method has been applied to several projects in Mogul. Nevertheless, obtaining continued and more widespread use of the method remains a challenge. We therefore wanted the interviewees' opinion about the prerequisites for a successful use of the use case points method in a larger scale. Our interviewees tended to use various tools for use case modeling, and they also used various tools and spreadsheets in estimation. This may indicate that there is a culture for applying tools and methods in an ad-hoc way in the company. Some of the interviewees stressed that they wanted a tool to be applied when they themselves found it useful, not methods that they were forced to apply. Hence, it may be difficult to get the whole company to agree on applying the use case points method.

Nevertheless, the interviewees were positive towards applying the use case points method; they found it desirable to apply the use case models in more activities in the development projects because of the effort that is often put into making it. A method to supplement expert estimates was considered particularly useful in projects with much uncertainty.

Although we agree that the use of the use case points method should be voluntary in Mogul, more experience with the method is needed to make it generally applicable.

7 Conclusions and Future Work

As part of a former software process improvement work in the software development company Mogul, an estimation method based on use cases, the use case points method, was evaluated with promising results. This paper described a follow-up software process improvement work that included interviews with senior personnel of Mogul to establish how the use case points method could improve the company's estimation practices, the prerequisites for applying the method and how to tailor it to this particular company.

We found that the use case points method can improve estimation practices in Mogul in that it provides a supplementary estimate in addition to the expert estimate. Combining estimates from different estimation strategies, particularly combining bottom-up estimates with top-down estimates, is an evaluated principle for improving estimates. In addition, applying an estimation method may help avoid estimation biases and thereby large overruns.

We also found that even though Mogul has good knowledge of RUP and use case modeling, it is challenging to construct a use case model that forms a good basis for estimation in that it correctly describes the functionality of the system and that the use cases are balanced. In particular, it is difficult to find end-users with sufficient competence and interest to take part in use case modeling. Nevertheless, the interviewees found use case models superior to old, unstructured requirements specifications.

The use case points method requires that the use cases should be detailed out, that is, each event between the system and the actor should be described, but this is not always done. We therefore proposed how the assessment of size of each use case could be refined, and made some suggestions for how the technical and environmental factors in the use case points method can be applied successfully to estimate the company's projects.

Nevertheless, more work is needed on how to tailor the use case points method. The following activities are planned:

- Establishing a scheme for measuring improvement to the estimation process. The most obvious success criterion is the accuracy of the estimates. Another criterion may be the number of people in the company who are competent estimators.
- Conducting a follow-up study to evaluate the proposed modifications to the use case points method.
- Investigating further how estimates produced with the use case points method can be combined with expert estimates.
- Investigating how use case modeling can be applied in web-projects.
- Investigating how to measure the size of a change to a use case, enabling the use case points method to be used in maintenance projects.

Acknowledgements

We gratefully acknowledge the employees of Mogul who took part in the interviews. We also acknowledge Dag Sjøberg for valuable comments on this paper. This research is funded by The Research Council of Norway through the industry-project PROFIT (PROcess improvement For the IT industry).

References

1. Albrecht, A.J. Measuring Application Development Productivity. Proceedings of joint SHARE, GUIDE and IBM Application Development Symposium. 1979.
2. Anda, B. Comparing Use Case based Estimates with Expert Estimates. Proceedings of the 2002 Conference on Empirical Assessment in Software Engineering (EASE 2002), Keele, United Kingdom, April 8-10, 2002.
3. Anda, B., Dreiem, H., Sjøberg, D.I.K., and Jørgensen, M. Estimating Software Development Effort Based on Use Cases – Experiences from Industry. UML'2001 - 4th Int. Conference on the Unified Modeling Language, Concepts, and Tools, Toronto, Canada, October 1-5, 2001, LNCS 2185, Springer-Verlag, pp. 487-502.
4. Anda, B., Sjøberg, D.I.K. and Jørgensen, M. Quality and Understandability in Use Case Models. ECOOP'2001, June 18-22, 2001, LNCS 2072 Springer-Verlag, pp. 402-428.
5. Blattberg, R.C. and Hoch, S.J. Database models and managerial intuition: 50% model + 50% manager, *Management Science*, Vol. 36, No. 8, pp. 887-899. 1990.
6. Fetcke, T., Abran, A. & Nguyen, T-H. Mapping the OO-Jacobson Approach into Function Point Analysis. Technology of Object-Oriented Languages and Systems, TOOLS-23. IEEE Comput. Soc, Los Alamitos, CA, USA, pp. 192-202. 1998.
7. Jørgensen, M. An empirical evaluation of the MK II FPA estimation model, Norwegian Informatics Conference, Voss, Norway. 1997.
8. Jørgensen, M. Reviews of Studies on Expert Estimation of Software Development Effort. Submitted to Journal of Systems and Software.
9. Karner, G. Metrics for Objectory. Diploma thesis, University of Linköping, Sweden. No. LiTH-IDA-Ex-9344:21. December 1993.
10. Kitchenham, B. A. *Software Metrics: Measurement for Software Process Improvement*. Blackwell Publishers. 1996.
11. Marchesi, M. OOA Metrics for the Unified Modeling Language. In Proc. of the Second Euromicro Conference on Software Maintenance and Reengineering, IEEE Comput. Soc, Los Alamitos, CA, USA, pp. 67-73. 1998.
12. Ribu, K. Estimating Object-Oriented Software Projects with Use Cases. Masters' Thesis, University of Oslo. November 2001.
13. Schneider, G. & Winters, J. *Applying Use Cases – A Practical Guide*. Addison-Wesley. 1998.
14. Smith, J. The Estimation of Effort Based on Use Cases. Rational Software, White paper. 1999.
15. Symons C.R. *Software Sizing and Estimating, MKII FPA*. John Wiley and Sons, 1991.
16. Symons, C. Come back function point analysis (modernized) – all is forgiven! Software Measurement Services Ltd. 2001.
17. <http://www.tassc-solutions.com/>