A Case Study on Naked Objects in Agile Software Development

Heikki Keränen[1,2] and Pekka Abrahamsson[1]

[1] VTT Technical Research Centre of Finland, PO Box 1100, FIN-90571 Oulu, Finland
{heikki.keranen,pekka.abrahamsson}@vtt.fi
[2] Department of Information Processing Science, FIN-90014 University of Oulu, Finland

Abstract. Naked Objects and agile software development have been suggested to complement each other. Very few empirical studies to date exist where a product has been developed using the Naked Objects technologies in an agile development environment. This study reports results of a case study where a mobile application was developed using the Naked Objects Framework. Qualitative and quantitative data was collected systematically throughout the project. The empirical results offer support for the argument that the Naked Objects approach is suitable for agile software development. The results also reveal weaknesses in the current Naked Object Framework, namely, that it is not yet mature enough for applications that require intense database operations. The results also show that the development team was able to create an operational user-interface just in five hours, which demonstrates the applicability of the Naked Object Framework in practical settings.

1 Introduction

Naked Objects [4] is an architectural pattern which exposes core business objects to the user. Naked Objects Framework is a software framework which supports implementing this pattern. Pawson and Wade [5] have proposed that the use of the Naked Objects architectural pattern complements the Extreme Programming's (XP) set of practices. Developers can, for example, make use of the Naked Object Framework to render the requirements in a concrete form that is immediately usable by the end-users [5]. Originally, Pawson and Wade suggested that only exploration phase would benefit from the use of the Naked Objects Framework. Yet, the long term goal is to develop the Naked Objects Framework to a state that the development of the working prototype can be continued to a full working release.

Very few empirical studies exist today where a product has been developed using the Naked Object technologies in an agile development environment (i.e., XP or others). The lack of empirical studies hinders the ability of other practitioners to evaluate the proposed approach and makes it difficult for a researcher to pinpoint the weaknesses and strengths of Naked Objects pattern and framework. This study reports the results of a case study where a mobile application, enabling users around the world access the Helsinki Stock Exchange for trading and viewing stock market development, was developed. Fig. 1 shows a part of the user interface.

The project involved student developers working 8 weeks in calendar time and using a total development effort of 810 hours. While this is the first of the kind empirical study on the Naked Objects Framework, we characterize the study as of the 'empirical exploratory' type. No hypotheses were made to be tested. The results produced

H. Baumeister et al. (Eds.): XP 2005, LNCS 3556, pp. 189–197, 2005.

Fig. 1. Part of the stock application user interface: Main menu, branch view and stock view

in this study will therefore set some references for other developers and researchers, which can be tested or refuted. This paper concentrates on the process aspect of the development. In addition, the weaknesses and strengths of the Naked Objects based development are proposed including lessons learned from the project.

2 Naked Objects

In the Naked Objects Framework, the core business objects encapsulate all business data and behaviour. They implement the Naked Object Java interface and obey some simple coding conventions. The framework has an Object Viewing Mechanism (OVM) which autogenerates a desktop user interface based on information in the business objects. Core interfaces implemented by the application and the Java reflection mechanism are utilised to do that. [4] Due to the abstract nature of Naked Objects, it is possible to create OVM's for different kind of devices. In this project, we developed an OVM for 'Java Mobile Information Device Profile (MIDP) 2.0'-capable mobile phones called MIDP-OVM. The Naked Objects Framework also contains a set of Object Stores, which provide automatic persistence for the business objects. In this project, we used the XML Object Store, which saves the business objects into XML files, because it was considered to be the most mature object store. Due to the automatic user interface generation and object stores, the implementation of an application is supposed to be rapid.

3 Research Design

The research method utilized was a controlled case study approach [6], which is an approach drawn from the action research, case study research and experimentation. In a controlled case study, the development environment is a laboratory setting. Yet, the approach strives at an industry-like business and delivery pressure where the development of a particular system is of the highest importance.

The Naked Objects development team involved four students as the development resources. The team worked for 24 hours a week in the development facilities. Both quantitative and qualitative data were collected. The developers collected the used effort for each defined task with a precision of 5 minutes using paper/pen and an in-house tool. The size of the development work, i.e. logical lines of code, was collected on daily basis using automated counters. The qualitative data includes the developer team interview.

The development was guided by an adapted version of the Extreme Programming approach called Mobile-D [1]. The adaptation has been performed taking into account the specific demands of the mobile development environment. The development cycle involves 5 system releases having the duration of one to two weeks. Before the project started, a rough version of MIDP-OVM existed[1]. It was, however, only capable of browsing objects and needed a lot of development before a real application could be used through it. The idea was to refine it in the beginning of the project and after that develop an application on top of it. MIDP-OVM is application independent and was developed for the purposes of enabling other developers to use Naked Objects on mobile platforms.

4 Results

In this section the results of the case study are presented.

4.1 Effort Distribution

Effort distribution is presented in Fig. 2. The coding phase consists of tasks related to the implementation of a feature. The management includes the collection of metrics, daily meetings and the project management work. The section 'Other' includes the environment setup, studying, coaching and the documentation activities. The planning activities include the planning game in the beginning of each iteration, as well as the architectural planning during development iterations. The quality assurance includes the tasks for verifying the user stories and related tasks. The defect fixing includes the refactoring and bug fixing activities. The testing includes writing test cases and a pre-release testing session, which is performed prior to the release. The release formalities include the formation of baseline and the acceptance test performed by the customer.

As shown in Fig. 2, the development profiles are slightly different in the MIDP-OVM development as compared to the application development. A lot of defect fixing (15%) was done in the application construction phase. The management activities also took more effort in the application development phase than in the framework development phase. More testing (i.e. 10%) was required for the platform development than application (i.e. 5%) part.

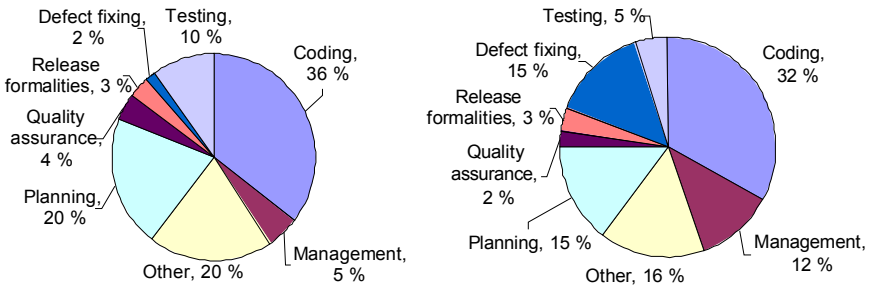

Fig. 2. Effort distribution of the MIDP-OVM construction phase (left) and the application construction phase (right)

[1] http://opensource.erve.vtt.fi/pdaovm/midp-ovm/

4.2 Estimation Accuracy and Precision

Estimation accuracy is presented in Fig. 3 using box plots[2]. The data used for drawing the box plots is based on the tasks that the developers' identified for the user story level implementation. The data below the thicker line indicates overestimation and data above the line refers to underestimation of the tasks.

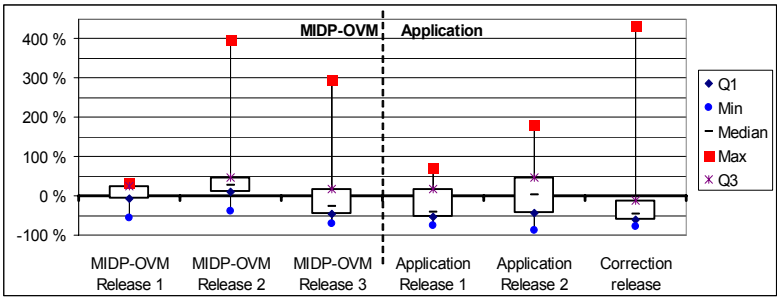

Fig. 3. Estimation accuracy

Overall, the data shows that the estimation error is a bit higher in the application creation than creating the MIDP-OVM. There is a high variance in the second and third releases of both MIDP-OVM and application. What is especially high, is the highest overspending in task time (Max-value), over 400 percent, which tells about unexpected problems in the development.

Figure 4 presents the estimation precision development, i.e. how many actual hours the developers lost by faulty estimates. The thick line indicates a loss of zero hours. The data points below the line indicate that an implementation of the task took less time than expected. The data points above the thicker line indicate that a particular task took longer than expected.

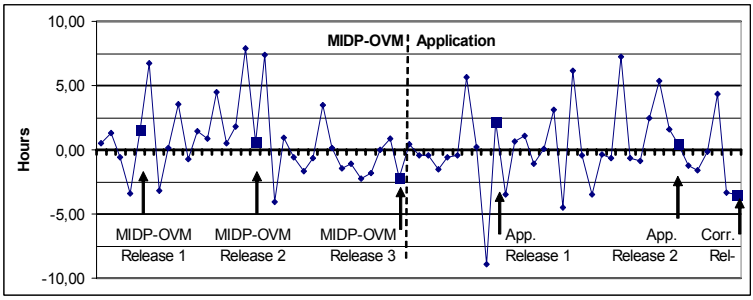

Fig. 4. Hours lost by faulty estimates

[2] A box plot diagram visualises the 5 number summary of a data set. Median value is the line in the shaded box area. A1 (first or lower quartile) shows the median of the lower 50% of data points. Q3 (third or upper quartile) shows the median of upper 50% of data points. The minimum value indicates the lowest and the maximum the highest values in the respective data sets.

By observing Fig. 4, we can see that in the implementation of the MIDP-OVM the estimation precision enhanced over time. On the other hand, in the application creation phase the implementation of the basic application went smoothly with very small task sizes, and thus, small errors in the absolute error estimates, but towards the end of the project, the estimates become less accurate.

4.3 Distribution of Task Sizes and User Story Effort

In the planning game, the team, together with a customer, identifies the user stories to be included in the iteration. The team divides each user story into a set of tasks preferably between 2-10 hours. The distribution of the actual task sizes are presented in Fig. 5.

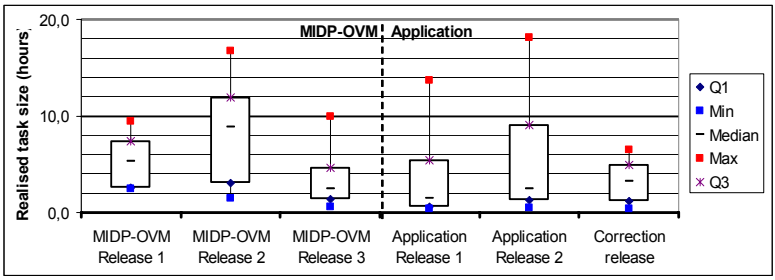

Fig. 5. Actual task sizes in each release

As can be deduced from the earlier figures, the estimation error caused some tasks to exceed the suggested limits. Yet, the data reveals that in spite of the complexity of creating OVM's, the team was able to split the user stories into reasonable sized tasks.

When creating the application, there were more tasks that took less than two hours, which indicates some difficulties in combining tasks into larger ones.

The user story effort correlates to how fast the project is able to generate visible results, meaningful to the customer. The user story effort is presented in Table 1. In the planning game of Application Release 1, there were first several user stories concerning the application requirements, introduced by the customer. As these were, however, considered trivial to be implemented using Naked Objects, the team decided to group these stories to one big story called “Create application” and define those stories as tasks.

Table 1. Actual effort used in the implemented user stories

Release	OVM R1	OVM R2	OVM R3	App. R1	App. R2	Corr. Rel.
User story effort (median, h)	13,7	49,1	9,7	0,6	9,7	9,8
User story effort (max, h)	9,5	98,2	24,7	2,4	23,7	14,5
# User stories implemented	2	2	5	6	9	2
# User stories postponed for next rel.	0	2	0	1	1	0

The size of this story was estimated at 8 hours and it was implemented in 5 hours, which is the size of a typical task in the normal development. To make it fair to com-

pare the implementation speed from the user point of view, in table 1 this "Create application" -story is split back into the original user stories.

In the MIDP-OVM Release 2, there is only one huge story and another with zero effort because it became a side effect of the huge story. This huge story is due to complete rewriting of the MIDP-OVM. The implementation speed of the user stories in Application Release 1 is remarkable (median 0.6 hours). Most of the effort in Application Release 1 was used to a story that took over 30 hours and still had to be postponed for the next release.

4.4 Growth of the Code Base

The development of the code base is important, since it portrays how the project progressed over time in terms of actual product development. Figure 6 presents the code size development during the project.

The initial version of MIDP-OVM was roughly two thousand lines of code, but after the experiences of the first release, the team decided to discard the old version and rewrite the MIDP-OVM starting from scratch, which explains the amount of code going down to zero. Bugs in the MIDP-OVM had to be fixed and some features had to be added in it, which explains the little changes in the amount of code after the application implementation started.

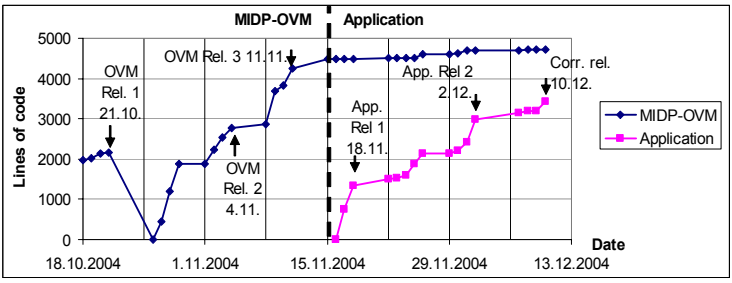

Fig. 6. Growth of the code base (LOC) during the project

4.5 Naked Objects from the Developers' Viewpoint

Any new software process innovation needs to be accepted by the developers. Otherwise the impact of the new technology is limited. In the interview, we asked the development team about the positive and negative experiences of using the Naked Objects Framework, as well as how easy or difficult they perceived the use of the framework and the pattern to be.

The programmers found the Naked Objects principle simple, but they felt that it requires quite little practice to learn how the framework actually works:

> "*For just writing the Naked Object application I think [it] is very easy, [it takes] just one or two days, and [a good way to do it is] writing couple of things and testing them if they work and if not try in another way and after three or four trials you get it, basically because there are new concepts, but they are not so many." [developer 1]*

Developers viewed that the implementation an OVM was especially challenging:

> "*But programming the MIDP-OVM is like getting really inside the framework, how it works, and not just using in [in a way] the ordinary programmer would do it." [developer 1]*

As a part of the application, the team had to do a module which periodically parses the stock data from the public web pages and feeds it into the application. Due to limited time and lack of examples of using application specific direct connection to an SQL database, the developers were consulted in using XMLObjectStore to store data. Other object stores were considered but they seemed either not to be compatible with the framework version we used, or were labeled as "proof of concept" -version. Also, there were no examples on how to make application tailored persistence of Naked Objects. In the application release 2, the developers told that the XMLObjectStore is not capable of handling a vast amount of automatic updates. There was no time to explore other approaches.

The developers also reported other problems with the Object Store concept. The integration of the web page parser to the Object Store was found problematic:

> "*There is just not a clean interface to do that and that is why we had so many problems with it." [developer 1]*

There were also less severe problems which were considered as bugs in the framework:

> "*We found out that although the book said we could disable some actions or [editing] associations, but at least this framework we used did not support those." [developer 3]*
> *"The framework itself uses a strange way to call all the classes in the application once in a while and load the objects. Finally, we managed to solve the problem by putting the actual method calls in the title-method, so we had to be creative." [developer 3] "At first we ended up in an endless loop." [developer 2]*

The developers' opinion was that Naked Objects suits well for agile development:

> "*I even think that its biggest advantage is that after a very short time of coding, maybe two or three hours, you already see the result." [developer 1]*

5 Discussion

As it has been stated a few times, there are very few case studies on the Naked Object technology that would be comparable to the study presented in this paper. The closest one on the Java technology and on a similar research setting can be found in [2] (i.e., the "eXpert" project) where a web based system was developed using the same cyclic approach and development rhythm. Also, the practices and tools are mostly the same as well as some of the support team members.

One of the greatest differences compared to the eXpert project is that in that project the maximum estimation error varies greatly: In their study, the highest estimation error in the release was always from 100% to 170%. In our case, there were releases

well under 100% and the three releases experienced over 290% estimation errors. One reason, we suspect, is the defects in the Naked Objects Framework. Another is the lack of documentation and code examples on extending the framework and integrating it with legacy software. The third reason might be the complexity of writing extensions to the framework due to the reflection properties and abstract behaviour - it is easier to hard code things, as it is done in traditional software development, than in this case making the MIDP-OVM application independent.

It seems that the creation of MIDP-OVM follows the characteristics of traditional agile software project in the sense that the task size estimations become more accurate. On the other hand, when creating an application using Naked Objects, it seems that the beginning of the project goes very fast, with very small errors in task time estimations, but after a while, the development slows down and the estimation accuracy deteriorates, due to the fact that some parts of the software need to be implemented without Naked Objects and integration of those parts to the Naked Objects application may be hard. Due to the very limited time to implement the application, we cannot predict if the estimation accuracy would enhance over time.

Compared to the results of Pawson [3], in this case, we did not need business agility in this project, as the desired functionality was pretty much fixed before the project started. This study suggests that although the business agility of the Naked Objects applications is good [3], difficulties integrating Naked Objects into the traditional systems might cause a risk in that sense, since when decision to use the Naked Objects technology is done, all the future business changes may not be known.

As a conclusion, we can identify three principal lessons learned:

- Based on this study, it seems that the Naked Objects Framework (version 1.2) is not yet mature for making serious business applications having a lot of objects and multiuser security requirements. Generally, the problem seems to be "*leaving those predefined paths the Naked Objects people were defining",* as one of the team members commented; this is often necessary because the framework is not designed to address all problems of all applications.
- The rapid development of the first versions of the Naked Objects applications makes it possible to use Naked Objects in an exploratory phase, as Pawson and Wade suggested [5]. In this project, the requirements for the application were pretty well known, so we used the default Mobile-D way of not creating code during the planning phase, but normally, the first versions could be created with close interaction with the customer.
- Naked Objects enables a fast realization of user stories, as the data clearly showed.

6 Conclusions and Future Work

This paper has presented a first-of-a-kind empirical case study on a project using the Naked Objects Framework. The results show that the current Objects Stores are not mature for applications that need a high number of objects and high throughput. The lack of documentation and knowledge on how to do sample code, and lack of time in the project made it impossible to try the application tailored persistence for Naked Objects. The ability to generate applications fast is a remarkable feature of the Naked Objects technology and it may change the software business, if the current problems can be solved.

The usability of the autogenerated user interface was outside the scope of this paper and needs to be studied. It should be noted that the current development of the Naked Objects Framework is addressing the problems found in this study.

References

1. Abrahamsson, P., Hanhineva, A., Hulkko, et al. Mobile-D: An Agile Approach for Mobile Application Development, OOPSLA 2004, Poster session, Vancouver, Canada, 2004
2. Abrahamsson, P. and Koskela, J., Extreme programming: A survey of empirical results from a controlled case study, ISESE 2004.
3. Pawson, R., Naked Objects. PhD thesis, Trinity College, Dublin, 2004.
4. Pawson, R. and Matthews, R., Naked Objects. 2002: J Wiley.
5. Pawson, R. and Wade, V., Agile Development Using Naked Objects, XP 2003, p. 97-103.
6. Salo, O. and Abrahamsson, P. Empirical evaluation of agile software development: The controlled case study approach, Profes 2004.