A Pilot Study of Comparative Customer Comprehension between Extreme X-Machine and UML Models

Christopher Thomson, Mike Holcombe, Tony Cowling, Tony Simons, George Michaelides
University of Sheffield
Department of Computer Science
Regent Court, 211 Portobello, Sheffield S1 4DP
+44 114 222 1980
{c.thomson, m.holcombe, a.cowling, a.simons, g.michaelides}@dcs.shef.ac.uk

ABSTRACT
Many design notations are used during software development to help the developers better understand the required system. However they are infrequently shown to clients, partly because developers believe that clients don't understand them. In this study two popular UML diagrams (activity and use case) and Extreme X-Machines diagrams (a type of state diagram developed to support Extreme Programming) were shown to three clients for whom we had recently delivered the software that was represented. The clients were given some simple guidance on interpreting them and asked to evaluate how well they understood them. This pilot study found that all the diagrams studied were equally understood, but further experiments are required to evaluate their usefulness.

Categories and Subject Descriptors
D.2.2 [**Software Engineering**]: Design Tools and Techniques - *Object-oriented design methods, State Diagrams.*

General Terms
Measurement, Documentation, Experimentation, Human Factors.

Keywords
Extreme X-Machines, XXM, empirical, formal method, testing, customer, requirements elicitation, requirements validation, requirements verification.

1. INTRODUCTION
It is typical in traditional software development processes that the customer provides a list of requirements that is agreed with the developers who develop and return a functioning system, which may or may not then be found to fit with the customer's vision [8]. The agile movement has begun to strengthen the relationship between developers and customers through practices such as the on site customer in Extreme Programming [2]. However this style of relationship is not applicable to all projects and may lead to poor documentation that can cause problems during maintenance [12].

In traditional style development diagrammatic models are often used to document the system. However these diagrams are often not maintained after their original construction due to the pressure to complete a system that is changing. As a result there have been many calls for automated systems to generate these diagrams from code or vice versa. Agile methods typically abandon such techniques due to this problem; however this has led to accusations of a hacker mentality [3].

The underpinnings of the agile manifesto encourage developers to be more productive by casting aside those parts of a process which are not useful. Therefore any modeling technique used in a agile process should deliver a reasonable return on the time investment made. Whilst this is a specific goal of the agile philosophy it is also a fundamental business axiom, which perhaps explains the recent rush to agile [7]. Therefore any model produced should have a specific return on investment.

A potential return on investment is achievable if documentation that is quick and easy to construct and can be used to verify a technical requirement with a customer with little or no technical training. Such a method has the potential to avoid unnecessary development costs and ensure the right solution is produced.

We have demonstrated in previous papers that the use of Extreme X-Machine (XXM) diagrams is beneficial to the development process [10, 11]. XXMs belong to a class of state machines known as X-Machines as defined by Eilenberg and later investigated by Holcombe [5, 6]. In the context of XXM they are partially defined and used to show a high level model of the entire system. They are suited to agile methods, as in common with other stream X-Machine models they can be used to generate test sequences to give complete functional tests. However it is unclear how useful they are to clients.

The use of design models with clients is not uncommon in practice. A previous survey of UML practitioners by Dobing and Parsons showed that Use Case Narratives, Activity Diagrams and Use Case Diagrams were used by more than 70% of the respondents to verify the design with their clients [4]. They also found that 50% of respondents had use UML Statecharts with clients; Statecharts are similar in representation to XXM diagrams. Whilst this result suggests that such diagrams have some use when shown to the client (rated as being "moderately successful"[4]) it did not address how well understood they were by the clients. To address this we posed our research question as: do clients understand software engineering diagrams?

Conference'04, Month 1–2, 2004, City, State, Country.

In this paper we present a pilot study that investigates this question with three customers who had just received delivery of a custom software application. The three models were Extreme X-Machines (XXM), UML use case diagrams and UML activity diagrams[1].

In this pilot study we produced both the UML and XXM models for three different industrial customers who each received three software systems which were intended to meet their requirements. Each of the software solutions was produced concurrently by teams of 3-5 students, whilst one author produced the models and another verified them to ensure consistency. We asked the customers to evaluate the models based on how useful they would have found them had they been presented with them at the start of the project.

2. EXPERIMENTAL DESIGN

The results reported here were obtained from a study where nine self-selected teams with similar and adequate development experience worked on one of three customer supplied projects competitively. Each student was directed to spend 180 hours on the project. Each of the three projects was provided by an industrial customer who later used the software in his/her business. Two projects were database driven websites and one an e-learning environment. We assumed that the project characteristics cannot confound comparisons between solutions developed with the same customer, but that between customers there is such a possibility.

The teams were instructed to use a modified form of Extreme Programming (XP) [2, 9] to capture the requirements as story cards and implement the systems. To ensure that a high quality working solution was delivered to the customer 50% of the overall marks were awarded by the customer for the delivered product and user documentation. The remaining marks were awarded by the academic staff for the process followed by the students.

Prior to the delivery of the final system each of the teams demonstrated and explained their solution to the researcher, who used this information to construct Use Case, Activity and XXM diagrams for each of the systems. The diagrams were constructed to have a common look and feel, level of detail (to represent the top level of the system), and accuracy across all nine teams, this normalization process was to ensure a fair comparison that was not dependant on the quality of software developed or the skills of the developers.

The diagrams were delivered to the customers after they had time to evaluate all three of the systems that they had each received. The customers were also supplied with a one paragraph description on the interpretation of the diagrams:

XXM: *"This type of diagram shows the order in which things can occur in a software system. In this diagram the bubbles represent screens in your system, but occasionally a bubble represents several screens with a similar purpose. The first screen is shown by a small (green) bubble, after which the movement between the screens is shown by arrows. Each of the arrows is named to denote a user action (e.g. "click()") or a system test (e.g. "[valid]"), if the action is made, or the test succeeds then the arrow may be followed. The arrow indicates that the system is doing something as described by the name."*

Use case: *"This type of diagram shows who can do what in a software system. The stick men on this diagram represent different types of user. The bubbles represent actions in the system. When bubbles are linked to a stick man this indicates that the action is available to that user. When a bubble is linked to another bubble this indicates that the action in the bubble uses the action in the other bubble. Lines ending in a triangle indicate that the bubble or stick man on the other end of the line includes all of the functionality of item at the triangle end."*

Activity: *"This type of diagram shows the order in which things can occur in a software system. A solid spot indicates where the system starts; follow the arrows from this position to explore the activities available to you. Each bubble represents an activity in the system, in some cases this may be prefixed with "<<name>>", this indicates that this activity has a wider purpose as indicated. The diamond boxes indicate a choice between options, on the arrows from a diamond are tests, if this test is fulfilled that path can be followed."*

The customers were given the opportunity to comment on the diagrams and to rate each diagram using the following questions on a five point Likert scale from "Not at all" to "Completely":

1. How well do you understand this diagram?
2. How well does it represent the system produced?
3. How much did you like this diagram?
4. How easy is it to locate faults in the system which are also in the diagram?
5. How easy is it to locate faults in the diagram which are not in the system?

3. RESULTS

The customer responses to the questions are presented in table 1. The Wilcoxon signed ranks test was computed with SPSS 13 to compare each the responses to each question and diagram to the same questions asked about the other diagrams (as some of the sets are clearly not normal - i.e. for Use Case questions 4 and 5 all the results are 1). This revealed that none of the questions asked delivered significantly different responses. The most significant was between Q4 in XXMs and Use Cases with a significance of 0.08. Further data will be required to investigate this further.

Table 1: The customers' responses to the questions.

Question / Customer	XXM					Use Case					Activity				
	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
A	2	2	1	1	1	3	2	3	1	1	3	2	2	1	1
A	3	3	3	2	1	4	4	4	1	1	4	4	4	1	1
A	2	2	2	2	1	3	3	3	1	1	3	3	3	1	1
B	2	4	2	1	1	2	4	1	1	1	2	4	2	1	1
B	2	3	2	1	1	2	3	1	1	1	2	4	2	1	1
B	2	4	2	1	1	2	3	2	1	1	2	2	2	1	1
C	4	3	3	2	2	2	3	3	1	1	2	3	3	1	1
C	2	2	1	1	1	1	1	1	1	1	4	3	3	2	2
C	3	3	3	1	1	2	2	1	1	1	2	3	2	1	1

All of the diagrams scored similarly for the questions so this would suggest that none of them offered any more value to the customer than any other.

Question 1 showed that the customers felt that they had a basic understanding of the diagrams presented (mean: 2.5). This supports the previous findings [4]. It is possible that this rating

would be improved if the customer would see such diagrams throughout the project. We checked with an additional question to see if any of the clients had previously encountered any of the diagrams and found client A had seen state based models similar to both XXM and Activity diagrams but not recently. A Wilcoxon signed ranks test confirmed that his answers were not significantly different to those of the other customers (A-B 0.09, A-C 0.22), however the mean was slightly higher.

Of the all questions 4 and 5 scored the lowest on average (mean: 1.1). Indicating that the diagrams neither illustrated the faults of the system nor were easily comparable to the systems to locate faults in the diagrams. This is unexpected as the customers indicated that they had some understanding of the diagrams (Q1, mean: 2.5). There are three explanations, their understanding is not enough to identify faults, the types of faults detected were not those that mattered to the client, or the wording of the question was poor. It is possible to interpret the questions to mean that the customers had to identify a fault to score highly whereas the intension as to measure if they could identify a fault if it existed, thus this should be corrected in the next study.

Question 2 showed that the customers could see some relationship between the diagrams and the delivered systems (mean: 2.9). Contrasting this to the responses to questions 4 and 5 suggests that this understanding is in response to the major functionalities of the system or flow rather than specific details.

Question 3 shows no significant distributions of the scores so the data does not favor any diagram. We also asked the customers to select their preferred diagram overall. They each selected a different diagram. One customer (B) commented that "[XXMs] seem to have fewer states and links, so easier/quicker to get to grips with. [Use cases] has too much information presented in the same way - hard to see what is important", the customer (C) said "I would probably still go for diagram type [Activity] overall as I think it would be generally easier for more complex applications", whereas the final customer (A) felt that "[Use cases] was the type of diagram I could best understand". These responses suggest that the clients were looking for different desirable features in the diagram. Whilst this may help to refine the presentation both of these questions will need to be made more precise to obtain comparable results.

Lastly customer B commented: "All three diagrams are hard to read intuitively - I feel they will be of more use to the design team than to me as the customer". Dobing and Parsons also found that many of the clients that worked with the practitioners surveyed had UML training [4]. This suggests that further work is needed to refine the presentation of the diagrams to better match customer knowledge.

4. DISCUSSION AND FUTHER WORK

The data collected so far suggests that all three diagrams (XXM, UML Use Case and Activity) are equally understandable and useful for a customer. As XXMs are found to be as understandable as the UML diagrams it suggests that they are equally useful for use with clients based on the previous literature [4]. Nevertheless it was disappointing that none of the diagrams were identified by the customer to be useful in finding faults. To make full use of a diagram with a customer this remains an elusive and desirable attribute.

The low scores collected for the questions regarding fault finding may have been due to the phrasing of the questions. Therefore a follow on study to this one will repeat the experiment but will revise these questions with the aim of collecting enough data to achieve significant results. As an alternative a task could be designed for the customer to complete, where they would be required to identify a seeded fault. As one of the customers identified that the diagrams were hard to interpret further thought must be given to the presentation of all the diagrams types so as to improve this.

If we assume that the data will be found to be normal once more is gathered and the questions revised then a power analysis can be used as a guide to the number of samples required. For a paired sample t-test with a hypothetical mean difference of .33 (⅓ of the comparisons have delta>=.33) and of a hypothetical SD=.5 (⅓ of the comparisons have an SD=<.5) then a sample of 20 is needed to achieve a significance level=0.05 and power=0.8.

5. ACKNOWLEDGEMENTS

This work was supported by an EPSRC grant: EP/D031516 - the Sheffield Software Engineering Observatory.

6. REFERENCES

[1] Unified Modeling Language (UML), Object Management Group, http://uml.org.

[2] Beck, K. and Andres, C. *Extreme Programming Explained: Embrace Change*. Addison-Wesley Professional, 2004.

[3] Constantine, L.L. 2001. Management Forum. *Software Development*, *9* (6). 16-28.

[4] Dobing, B. and Parsons, J. 2006. How UML is used. *Commun. ACM*, *49* (5). 109-113.

[5] Eilenberg, S. *Automata, Languages and Machines*. Academic press, 1974.

[6] Holcombe, M. and Ipate, F. *Correct Systems - building a business process solution*. Springer-Verlag, 1998.

[7] Krasteval, I. and Llieva, S., Rush into Agile - Analytical Framework for Agile Practices Applicability. in *Agile Manufacturing, 2007. ICAM 2007. IET International Conference on*, (2007), 229-238.

[8] Sommerville, I. *Software Engineering*. Addison-Wesley, 2007.

[9] Thomson, C. Defining and Describing Change Events in Software Development Projects, Department of Computer Science, University of Sheffield, 2007.

[10] Thomson, C. and Holcombe, W., Applying XP Ideas Formally: The Story Card and Extreme X-Machines. in *1st South-East European Workshop on Formal Methods*, (2003), South-East European Research Centre, 57-71.

[11] Thomson, C. and Holcombe, W., Using a formal method to model software design in XP projects. in *2nd South-East European Workshop on Formal Methods*, (2005), South-East European Research Centre.

[12] Turk, D., France, R. and Rumpe, B., Limitations of Agile Software Processes. in *Third International Conference on eXtreme Programming and Agile Processes in Software Engineering (XP 2002)*, (2002).