

A Design Change Metric Derived From Extreme X-Machines

Christopher Thomson, Mike Holcombe

University Of Sheffield
Regent Court, 211 Portobello Street, Sheffield S1 4DP, UK
{c.thomson, m.holcombe}@dcs.shef.ac.uk

Abstract. Change management during software development remains a difficult process. Whilst there are methods which claim to manage and predict changes they often have poor performance or are empirically untested. We propose a metric which is easy to collect when Extreme X-Machines are used in the design process. Furthermore we present limited empirical evidence in the form of an explorative observational cross-sectional study. It showed that the frequency of these changes overtime follows a Norden/Rayleigh curve that was derived from recording the changes functionality proposed at client meetings. We hope that future research will show that tools developed by others to predict this curve for man power will also work in this special case.

Keywords: Lightweight formal model, empirical, software engineering, software lifecycle management, X-Machine, Extreme X-Machine, XXM.

1 Introduction

The management of the introduction of changes to a software development project is one of the factors in the success of a project. There is however a lack of theory in this area [1], therefore there is little firm advice on how to proceed. In this paper we explore a change classification metric that could allow further research into the prediction of changes as made to software development projects.

To investigate the way in which changes are made to software development projects we collected various data from a student development project at the University of Sheffield, known as the Software Hut over a three year period. In each case a client from an external company provided the real problem to be solved and met the teams on a weekly basis. The best of the four solutions was later used by the client as part of his business. The development project ran over an eleven week period during the spring semester, thus giving us change data over 10 weeks.

When we analyzed the data collected we found that the frequency of changes proposed in client meetings over time followed the Norden/Rayleigh curve [2,3] a well known distribution [4]. The Norden/Rayleigh curve is normally used to describe the effort used during a project, so we expected the pattern to appear across the measurements of change that we made. However this pattern was not found in measurements taken of the lines of code that changed each week, suggesting that the

reported changes were made at the design level. We therefore looked to the design of the system to verify this.

In this paper we explore the design changes as made to software projects by the use of a lightweight formal model known as Extreme X-Machines (XXM) [5,6], which are based on the work of Eilenberg and Holcombe [7,8]. The students were provided with a tool that captured their models [6], which were then analysed for change. As a formal model, XXM allow changes to be described as transformations; these either take the form of refactorings [9] or functional changes. These represent a measure of design change and were made for teams using Extreme Programming (XP) [10] an agile development process. Previous work in this area predicted that there should be large spikes of changes towards the end of the first and third quarters of the development [11].

An XXM model describes the functionality of the software without defining exactly how this functionality is achieved. This perspective allows an analysis of functional change whilst excluding specific implementation or requirements issues. In this paper we propose a metric that can be used to measure changes in an XXM model. The scheme is based on a set of proposed operations that were validated through observation, although several theoretical changes were not observed.

To show that the metric was correlated to the changes recorded from the meetings a statistical analysis is used. To show this we used a prediction [4] and applied three tests of statistical correlation. These showed that the frequency of functional change as measured by the XXM metric was correlated to the frequency of functionality changes recorded in meetings between the developers and clients.

2 Literature Review

Handling the changes requested remains one of the major challenges facing the software engineering community [12,13]. Indeed the changes made to requirements pose a difficult and costly problem [14]. It has been observed that if changes can be anticipated and catered for, then the value of software can be increased [15]. But “there is a lack of theory and practice in the area of requirements change” [1]. In particular, tools and techniques can be built to monitor and trace changes so that developers can be notified of important changes related to their code [16,17], but these tools are limited both by lack of data [18], and also conversely when there is too much data [17].

Simple code based predictions can be flawed as for instance object orientated code at first appears particularly prone to changes [19,20] but a ‘single line change’ only has a 4% probability of introducing further errors [21]. Indeed simple measurements on the code may record additions, deletions and changes, but provide little further information that allows reasoning about the changes [22].

To investigate and predict changes we may therefore need more complex measurements. Changes can be measured by size, complexity and developer expertise [23], and the size of a change can be measured in terms of a delta or churn [24].

One proposed alternative measurement of change uses object orientated code [11]. The system design instability metric gives a measurement of change at the class level.

It is calculated by combining the measurement of the number of classes: renamed; added; removed; and moved in the hierarchy. However, it is hard to see how this measurement can be applied to non-object orientated code. We can treat these more generally as a form of refactoring where additional functionality is also added [25], Refactoring is a process that allows change to be engineered efficiently [9]; this is achieved by moving code around the structure of the software, in order to create a better design based on design patterns [26].

Other approaches have proposed building architectural models of the software and performing change analysis on the models. One solution describes the software architecture using a formal logic; transitions are then defined in order to describe the differences between versions of the software [27]. A similar approach uses a graph to describe the software based on the object orientated structure; conditional rewriting rules are then used to describe the changes [28]. A final approach associates change management information with architectural concepts, this captured information [29]. However, none of these approaches have been applied to real projects.

In this paper we show how these approaches can be combined by using a light weight formal model known as the Extreme X-Machine (XXM) [5,6]. XXM are a state based model, they are intended to be used by developers as a method to design their systems at the top level, but you can also express hierarchical structures and lower level detail [5]. Each model typically consists of a set of states which correspond to screens in the final system and functions which link the screens together. The functions are typically labeled with an enabling action such as "click_ok" which corresponds to a user clicking the OK button. The metric we propose describes changes to these models, thus encompassing a formal model and a design metric.

Lastly previous research showed that the frequency of changes as discussed in the project meetings tends to follow the Norden/Rayleigh curve over time [4]. The Norden/Rayleigh curve has been previously used to represent and predict the amount of effort required during development [2,3,30], as used in COCOMO [31]. This model is not without its critics as no theory has been developed to explain why the Rayleigh curve fitted the observed data [32]; moreover, project stress was more closely correlated with staff loading [32,33] than elapsed time [32]. Later work used the Gamma distribution [35] as a more general solution [36]. A similar Rayleigh model has also been found useful in modeling the number of defects introduced and fixed during development [37].

3 Experimental Design

The Software Hut module is typically taken in the second year by students on the Computer Science and Software Engineering undergraduate degree programs [38,39]. An important characteristic about this project is that external business clients specify a business problem and the student teams analyze the issues and develop working solutions which are delivered to the client, who chooses the best and uses it in their business. The data used relates to all of the teams in the module in one year. The teams were self-selected and contained four or six members (six teams of each), the

choice of projects was based on the client who responded to local adverts. In all cases the students delivered a working system to the client, who selected the best version, that team then received a small prize. The students choose which client they wished to work for by ranking the clients in order of preference. The projects progressed through a full software lifecycle, using the Extreme Programming method [10], over a period of eleven weeks. The teams met a lecturer each week for a management meeting; likewise, they met their client in each of the first five weeks and then again from the ninth week onwards to deliver the software. The four teams each worked with one of three clients on the same problem. The problems were: the development of a comprehensive web site for a real estate company specializing in overseas properties; a database to support an occupational health unit; and a web based recruitment agency database.

The collection and analysis of data from student projects is well established in the literature where 87% of experiments are conducted with student subjects [40], where it has been noted that the students are an acceptable sample of the population of novice developers [40]. The students taking this project are competent developers but do not yet have the level of knowledge that would be expected from an industrial team [41]. The Software Hut is the students' second large-team based software development project, the first being more constrained and based around 'toy' problems. Prior to the experiment all the students had studied and passed modules in the following subjects: Introduction to Programming, Object Oriented Programming, Systems Design and Testing, Requirements Engineering, Functional Programming, Systems Analysis and Design and Database Technology.

This method has good internal validity as all the participants had similar experience and training and the teams all had to deliver similar deliverables to the same deadlines. However differences between the projects and client may have had an impact as different programming languages were used (MS Access, PHP, C#, Java, and JavaScript) and of course the projects differed. However this variability points towards the enhanced generalizeability of the results.

4 Proposed Metric

XXM is a state based model and changes can be described as mutations to this model (table 1). The memory and function specifications were ignored as these were not used by the students.

Only one year's worth of data was available (Software Hut 2004-5) to assess the model. To make use of the data single instances of changes were identified as examples (see appendix) to confirm the validity of the proposed scheme. For the detailed analysis that follows the occurrences of each kind of change in each project week were counted (figures 1 and 2).

No evidence was found of either reordering or redirecting (see appendix). This is not an immediate concern due to the small sample size. The counts of these actions show that there was a ratio of 3:2 functional to refactoring change (figure 1). Figure 1 also shows that the functional changes have a clearer trend, although for both classes of data the final weeks seem to contain no trends. The detailed data shown in figure 2

does not make the final weeks any clearer, but it does show that in the initial weeks there are two dominating components: insert and additional exit.

Table 1. Changes that relate to X-Machines.

	Operation	Description
Refactoring	Relabelled	Either a function name or a state can be relabelled.
	Pull up	A state (or group of) is inserted on a function.
	Push down	A state (or group of) is removed from the diagram and replaced with a single function.
	Automate	A previous user choice is automated.
	Make specific	States or diagrams are split into a more specific format.
	Make generic	States or diagrams joined into a more generic format
	Choice	A previously automated choice is given to the user.
Functional	Reorder	States moved around and entered in a different order.
	Insert	An extra state added.
	Remove	A state or function is removed.
	Additional exit	A state can have another function originate from it.
	Redirect	A function can point at a new target or source state.

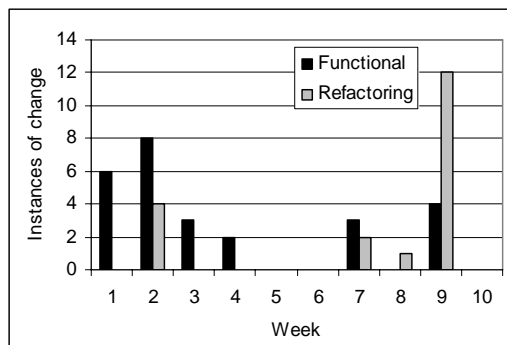


Fig. 1. Comparing functional and refactoring change.

These measurements broadly follow the observations made by Alshayeb and Li as there are periods of high change at the ends of the first and third quarters (weeks 2-3 and 8-9 in figures 1 and 2) [11], however more changes seem to have occurred in week 9, later than the previous observations predict. As this measurement was made on the XXMs, as opposed to the code based measures used by Alshayeb, it could be that the students updated the code first.

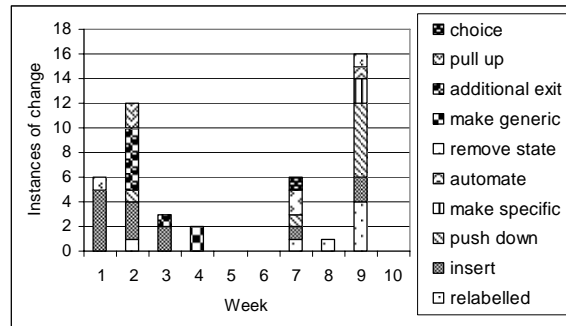


Fig. 2. Functional and refactoring changes as measured by changes to the developer's XXM.

5 Experimental Method

This experiment assessed if the changes recorded in the students XXM models by the metric above followed a similar Norden/Rayleigh curve to the one observed in the change reports. To do this the frequency of each change operation was counted across all the teams (12) and a total was calculated for each week, in total 45 changes were found (26 functional and 19 refactoring). As there was only a small amount of data it was combined into three data sets: functional; refactoring; and combined changes (table 2). To estimate the Norden/Rayleigh curve the gamma probability density function was used (GPDF), with the parameters taken from other experiments which estimated the level of functional change [4].

Table 2. The number changes made to Extreme X-Machines in each week.

	Functional	Refactoring	Total
<i>Week 1</i>	6	0	6
<i>Week 2</i>	8	4	12
<i>Week 3</i>	3	0	3
<i>Week 4</i>	2	0	2
<i>Week 5</i>	0	0	0
<i>Week 6</i>	0	0	0
<i>Week 7</i>	3	2	5
<i>Week 8</i>	0	1	1
<i>Week 9</i>	4	12	16
<i>Week 10</i>	0	0	0

The GPDF [35] is derived from the gamma function. It is a continuous probability distribution that is characterised by a shape and scale parameter. The function is defined between two points on the x-axis and can be non-symmetrical. The shape of curve given by the GPDF can be interpreted in the same way as the Rayleigh distribution as used by Norden and Putnam and has been shown to have some

advantages [36]. To estimate known deviations from the curve two further factors were considered: the week after the Easter vacation and the weeks on which deadlines occurred. The final formula for the estimation is shown as (3) where: a - is the scale factor (3.1); b - is the shape factor (0.7); c - is a factor to represent the amount of change (72); p_1, q_1 - a factor to indicate a week following a vacation ($q=-2.9$); p_2, q_2 - a factor to indicate a week with a deadline ($q=7.5$); x - the project week for which we are making the estimate; y - the amount of change.

$$g(x, a, b) = \begin{cases} \frac{x^{b-1} e^{-x/a}}{\Gamma(b)a^b} & x \geq 0 \\ 0 & elsewhere \end{cases} \quad (1)$$

$$\Gamma(b) = \int_0^{\infty} x^{b-1} e^{-x} dx \quad (2)$$

$$y = g(x, a, b) \times (c + (c \times p_1 \times q_1) + (c \times p_2 \times q_2)) \quad (3)$$

The Pearson's, Spearman's Rho and Kendall's Tau(b) correlation tests were used to assess the statistical significance of the relationship between the predicted curve and that measured by the XXM [5,6]. The Pearson correlation test is parametric and therefore required the data to conform to the normal distribution, and this was checked by a one sample Kolmogorov-Smirnov test as summarised in table 3 [42]. In all cases the data was found to conform to the normal distribution ($P>0.05$). All statistics were calculated using SPSS 13.

Table 3. A One-Sample Kolmogorov-Smirnov test of the data shows all the data sets are normal.

		Functional changes	Refactorings	Combined XXM changes	XXM Predicted
Normal Parameters	Mean	.087	.083	.085	.10
	Std. Deviation	.093	.16	.10	.17
Kolmogorov-Smirnov Z		.70	.97	.66	1.0
Asymp. Sig. (2-tailed) (P)		.7	.30	.78	.26

6 Analysis

The correlation between each of the three XXM data sets and the model (3) was found to be significant using Spearman's Rho test (Table 4). Kendall's Tau(b) test was not significant for the refactorings data set and the Pearson test only found the functional changes to be significant. This implies that the refactoring changes are less related to the estimation and that only the XXM functional changes are linearly related to the prediction. Spearman's Rho test also shows a highly significant correlation for the

combined data set suggesting that some elements of the refactorings are important for describing functional change.

Table 4. Comparison of the gamma distribution prediction to the observed data.

		Functional changes	Refactorings	Combined XXM changes
Pearson	Correlation Coefficient	.74 ⁽¹⁾	.31	.59
	Sig. (2-tailed) P	.013	.39	.071
Kendall's tau_b	Correlation Coefficient	.63 ⁽¹⁾	.49	.64 ⁽¹⁾
	Sig. (2-tailed) P	.015	.067	.011
Spearman's rho	Correlation Coefficient	.74 ⁽¹⁾	.66 ⁽¹⁾	.77 ⁽²⁾
	Sig. (2-tailed) P	.014	.04	.010

The scatter plot (figure 3) shows that although the variance between the measures is high the linearly related functional changes follow the predicted trend. The bar graph (figure 4) also shows that the predicted changes for weeks 2 and 7 are higher than observed, whereas they are lower than observed in weeks 1 and 9. In these weeks the functional changes are more closely related to the predictions but there is still a large discrepancy in week 2 which is unexplained.

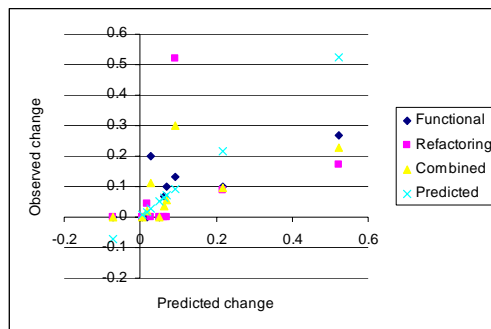


Fig. 3. Scatter plot showing the observed data against the prediction.

As the predictions are based on the student's reports of changes and the measurements on actual changes this suggests that the changes reported in week 7 were not implemented in the XXM until week 9. Therefore the design changes if made directly to the code may have more closely followed Alshayeb and Li's observations [11]. There is also a negative prediction in week 6, which suggests that the model requires further tuning before use.

¹ Correlation is significant at the 0.05 level (2-tailed).

² Correlation is significant at the 0.01 level (2-tailed).

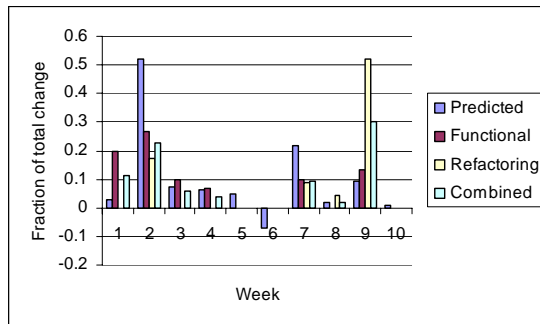


Fig. 4. Bar graph plot showing the observed data against the prediction.

7 Conclusions

Firstly this paper proposed a design change metric based on an analysis of an XXM model. Secondly the Pearson, Spearman's Rho and Kendall's Tau(b) correlation statistics were used to compare the frequency of changes made to some XXM model to a prediction defined by the changes by the developers and client at their meetings, as devised in previous work [4]. The correlation between the model and the changes from the XXM was found when just the functional changes were considered independently. The refactoring changes were not significantly correlated although this may have been because they were few in number and skewed towards the end of the project.

The data was also found to exhibit periods of high change in our prediction at similar times to those observed by Alshayeb and Li [11]. This suggests that their observation is valid for teams using XP and modelling their design using XXM models. However it was also observed that the changes to the XXM models appeared to lag behind the changes as reported by the students, perhaps indicating that the XXM models were used as documentation as opposed to a planning technique.

The predictive model did not give a perfect match to the observed data but showed some noticeable differences, including a negative prediction. These flaws may indicate that this result is not practically significant. Future research should collect further data to explore the relationship between reported change and observed change in the XXM models.

Reflection on the nature of the XXM model suggests why analysis of its changes should correlate with the changes as reported by the students. The XXM model has the advantage of giving a high level of abstraction to the change that is closer to ideas behind the changes than the implementation. As a high level model the XXMs produced by the students were few in number but were constructed quickly. As the diagrams were constructed quickly no changes due to progressive implementation were picked up. These factors combined led to a small number of changes being made, which better reflected those reported by the students.

The external validity of this explorative research is relatively low as we only studied one cohort of students, in one university. This limits the generality of the results, which must be verified in further more formal and larger experiments, both at Sheffield and other institutions and companies.

Future work should encourage developers to systematically update diagrams of this nature, or derive them from the project code, so that sufficient data from individual teams can be collected. Such work would allow for refinements in the model at the team level, and perhaps to redistribute or confirm the high number of changes in the final week. The previous work conducted on using the Norden/Rayleigh curve as a predictor for effort should be investigated to see if it can be reapplied in this context [31, 36]. If it can be then it is likely to be a useful tool to predict and manage functionality change in a development project when combined with XXMs.

Acknowledgements: This work was supported by an EPSRC grant: EP/D031516 - the Sheffield Software Engineering Observatory. During the period of research Thomson was also supported by an EPSRC studentship.

References

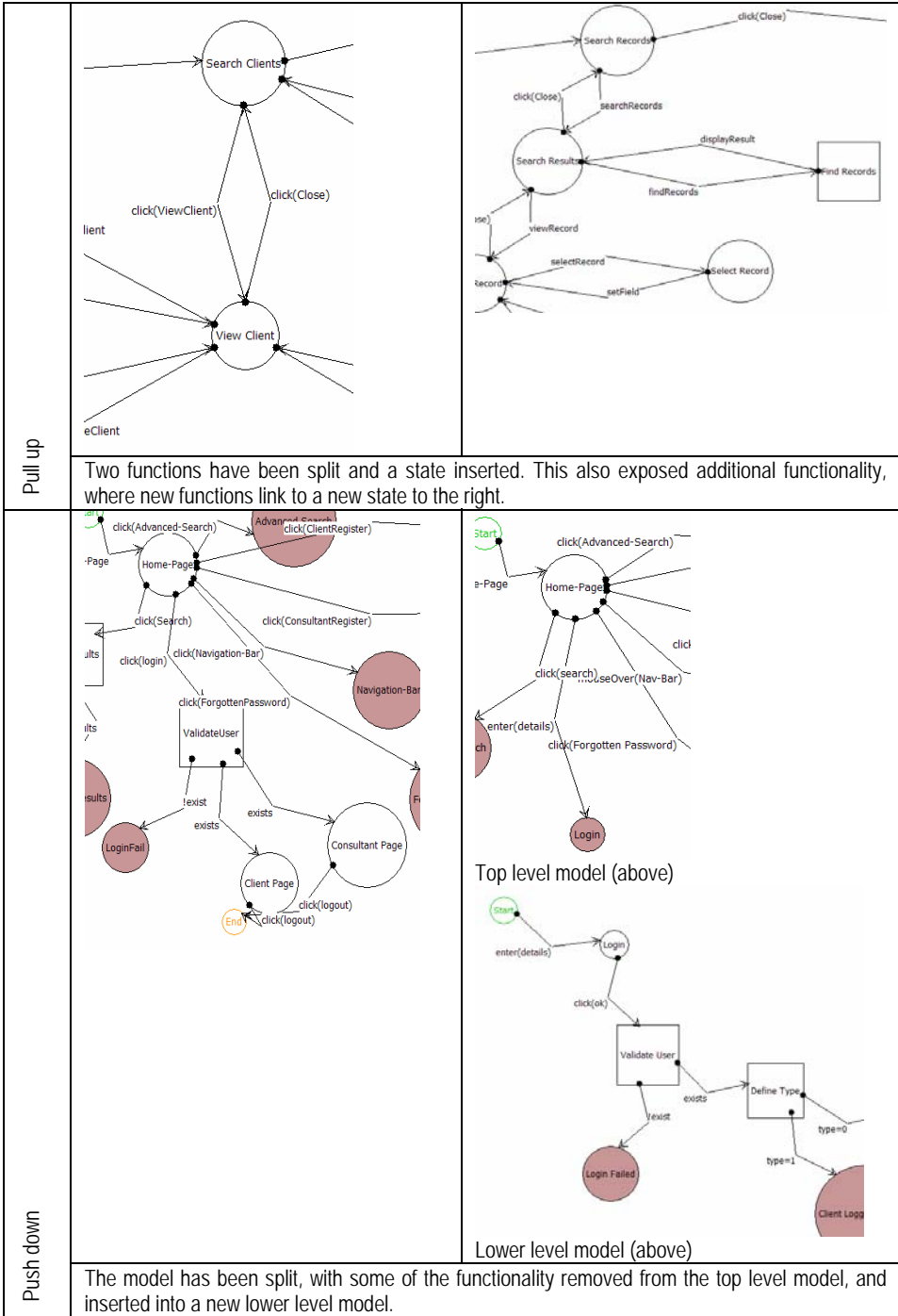
1. Lam, W., Shankararaman, V.: Requirements change: A dissection of management issues. In the 25th Euromicro Conference, volume 2, 2244--2251 (1999)
2. Norden, P.V.: Management of Production. In Useful tools for project management, 71--101, Penguin, Baltimore (1970)
3. Norden, P.V.: Project lifecycle modelling: Background and application of the lifecycle curves. In the Software lifecycle management workshop, Airlie (1977)
4. Thomson, C.D.: Defining and Describing Change Events in Software Development Projects. PhD Thesis, University of Sheffield (2007)
5. Thomson, C., Holcombe, W.M.L.: Applying XP ideas formally: The story card and extreme X-Machines. In the 1st South-East European Workshop on Formal Methods, Drandis., D., Tigka, K. (eds.), 57--71, Thessaloniki, Greece (2003)
6. Thomson, C., Holcombe, M.: Using a formal method to model software design in XP projects. *Annals of Mathematics, Computing and Teleinformatics*, 1(3) (2006)
7. Eilenberg, S.: Automata, Languages and Machines, volume A. Academic press, New York (1974)
8. Holcombe, M., Ipate, F.: Correct Systems - building a business process solution. Applied Computing. Springer-Verlag (1998)
9. Fowler, M., Beck, K., Brant, J., Opdyke, W., Roberts, D.: Refactoring: Improving the design of existing code. Addison Wesley (2000)
10. Beck, K., Andres, C.: Extreme programming explained: embrace change. Addison-Wesley Professional, Reading Massachusetts, 2nd edition (2004)
11. Alshayeb, M., Li, W.: Empirical study of system design instability metric and design evolution in an agile software process. *Journal of Systems and Software*, 74(3):269--274 (2005)
12. Finkelstein, A., Kramer, J.: Future of Software Engineering, in chapter Future of Software Engineering. 3--22. ACM Press (2000)

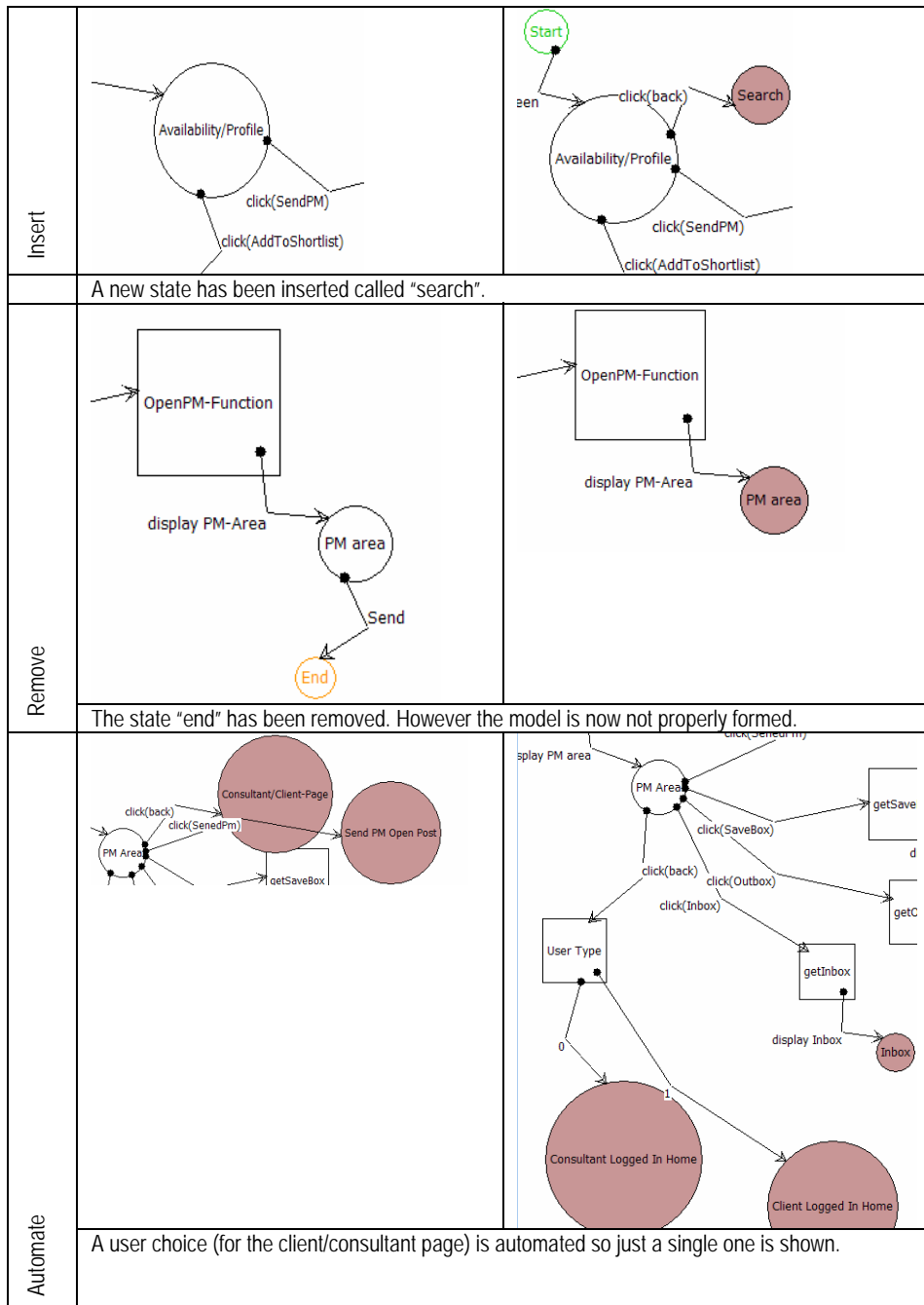
13. Bush, D., Finkelstein, A.: Environmental scenarios and requirements stability. In the international Workshop on Principles of Software Evolution, 133--137, New York, ACM Press (2002)
14. Harker, S.D.P., Dobson, J.E., Eason, K.D.: The change and evolution of requirements as a challenge to the practice of software engineering. In the IEEE International Symposium on Requirements Engineering, San Diego, California (1993)
15. Boehm, B.W., Sullivan, K.J.: Software economics: a roadmap. In the Conference on the Future of Software Engineering, 319--343, New York, ACM Press (2000)
16. Cronin, K.J., Linton, D.G.: A change prediction model for embedded software applications. In Southeastcon '98, 5--8. IEEE (1998)
17. Kowalczykiewicz, K., Weiss, D.: Traceability: Taming uncontrolled change in software development. In the IV National Software Engineering Conference, Tarnowo Podgorne, Poland (2002)
18. Ramil, J.F.: Algorithmic cost estimation for software evolution. In proceedings of ICSE 2000, 701--703, Limerick, Ireland (2000)
19. L. Li and A.J. Offutt. Algorithmic analysis of the impact of changes to object-oriented software. In proceedings of the International Conference on Software Maintenance, pages 171--184, Nov 1996.
20. Chaumun, M.A., Kabaili, H., Keller, R.K., Lustman, F., Saint-Denis, G.: Design properties and object-oriented software changeability. In the conference on Software Maintenance and Reengineering, p45, Zurich, Switzerland (2000)
21. Purushothaman, R., Perry, D.: Towards understanding the rhetoric of small changes. In the International Workshop on Mining Repositories, Edinburgh, Scotland, UK (2004)
22. Fenton, N., Pfleeger, S.: Software Metrics: A Rigorous and Practical Approach. Thomson Computer Press International, 2nd edition (1997)
23. Mockus, A., Eick, S.G., Graves, T.L., Karr, A.F.: On measurement and analysis of software changes. Technical report, National Institute of Statistical Sciences (1999)
24. Hall, G.A., Munson, J.C.: Software evolution: code delta and code churn. The Journal of systems and software, 54(2):111--118 (2000)
25. Counsell, S., Hassoun, Y., Johnson, R., Mannock, K., Mendes, E.: Trends in java code changes: the key to identification of refactorings? In PPPJ 2003, 45--48, Kilkenny city, Ireland (2003)
26. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns, Elements of Reusable Object-Oriented Software. Addison-Wesley (1994)
27. Lucena, C.J.P., Alencar, P.S.C.: A formal description of evolving software systems. Science of computer programming, 24, 41--61 (1995)
28. Mens, T.: Conditional graph rewriting as a domain independent formalism for software evolution. In the Applications of Graph Transformations with Industrial Relevance International Workshop, AGTIVE'99, p. 127, Kerkrade, The Netherlands. LNCS 1779, Springer (2000)
29. Hoek, van der, A., Mikic-Rakic, M., Roshandel, R., Medvidovic, N.: Taming architectural evolution. In ESEC/FSE-9: Proceedings of the 8th European software engineering conference held jointly with 9th ACM SIGSOFT international symposium on Foundations of software engineering, 1--10, New York, USA, ACM Press (2001)
30. Putnam, L.H.: A general empirical solution to the macro software sizing and estimation problem. IEEE Transactions on Software Engineering, 4(4):345--361 (1978)
31. Boehm, B.W.: Software Engineering Economics. Prentice Hall, New Jersey (1981)
32. Parr, F.N.: An alternative to the Rayleigh curve model for software development effort. IEEE Transactions on Software Engineering, 6(3):291--296 (1980)
33. Jeffery, D.R.: Time-sensitive cost models in the commercial MIS environment. IEEE Transactions on Software Engineering, 13(7):852--859 (1987)
34. Brooks, Jr., F.P.: The mythical Man-Month. Addison-Wesley (1995)

35. Scheaffer, R.L., McClave, J.T.: Probability and Statistics for Engineers. ITP/Wadsworth/Duxbury, 4th edition (1995)
36. Pillai, K., Nair, V.S.: A model for software development effort and cost estimation. IEEE Transactions on Software Engineering, 23(8):485--497 (1997)
37. Kan, S.H.: Modeling and software development quality. IBM Systems Journal, 30(3):351--362 (1991)
38. Holcombe, M., Parker, H.: Keeping our clients happy: Myths and management issues in 'client-led' student software projects. Computer Science Education, 9(3):230--241 (1999)
39. Holcombe, M., Gheorghe, M., Macias, F.: Teaching XP for real - some initial observations and plans. In Proceedings of 2nd International Conference on Extreme Programming and Flexible Processes in Software Engineering, Sardinia, Italy, 14--17 (2001)
40. Sjøberg, D.I.K., Hannay, J.E., Hansen, O., Kampenes, V.B., Karahasanovic, A., Liborg, N.-K., Rekdal, A.C.: A survey of controlled experiments in software engineering. IEEE Transactions on Software Engineering, 31(9):733--753 (2005)
41. Kitchenham, B.A., Pfleeger, S.L., Pickard, P.W., Hoaglin, D.C., El Emam, K., Rosenberg, J.: Preliminary guidelines for empirical research in software engineering. IEEE Transactions on software engineering, 28(8):721--734 (2002)
42. DeGroot, M. H. Ch. 9 in Probability and Statistics, 3rd ed. Reading, MA: Addison-Wesley (1991)

Appendix Examples of Changes Made to Extreme X-Machines

Operation	Before	After
Relabelled		
	A state has been relabelled.	
Additional exit		
	The "validate input state" has an additional function leaving it "!valid".	





Choice		
<p>An automated choice is removed. In this case there was only one exit from the automated state "search properties" so these two diagrams are functionally identical.</p>		
Make specific		
<p>The original single diagram that was generic has been broken down into two more specific versions.</p>		
Make generic		
<p>Two specific states ("ConsultantLogin" and "ClientLogin") have been combined into a single generic state ("SuccessLogin").</p>		