Comparing Decision Making in Agile and Non-Agile Software Organizations

Carmen Zannier[1], Frank Maurer[1]

[1]University of Calgary, Department of Computer Science, 2500 University Drive NW, Calgary, AB, CANADA, T2N 1N4
{zannierc, maurer }@cpsc.ucalgary.ca

Abstract. Our ability to improve decision making in software development hinges on understanding how decisions are made, and which approaches to decision making are better than others. However, as of yet there are few studies examining how software developers make decisions in software design, especially studies that place agile approaches in the context of decision making. In this paper, we present results of a multi-case study of design decision making in three software organizations of varying levels of agility. We show an agile organization produced a culture that supported communication and debate about alternatives to design decision more than 2 organizations of lesser agility.

Keywords: Consequential Choice, Serial Evaluation.

1 Introduction

We present an emergent multi-case study in which we compare the use of consequential choice [11, 12] and serial evaluation [9] in three small software organizations of varying levels of agility. Consequential choice is defined as the *concurrent* comparison and trade-off evaluation of more than one option in a decision [11, 12]. Serial evaluation is the *sequential* evaluation of options (n.b. no tradeoff evaluation and no concurrency) in a decision [9]. The results of our observations strongly suggest that small agile environments lead to more use of consequential choice (rather than serial evaluation) than small non-agile environments. This was a surprising result because consequential choice is rooted in "rational" approaches to decision making, typical of operations research, economic theory, and other seemingly "non-agile" fields of study. Our results show an agile environment was implicitly able to foster rational design decisions by emphasizing direct collaboration.

We conducted this study to continue learning how design decisions are made in software development. It is a relatively unexamined topic, despite recognition of its importance [1, 4, 8, 14, 18]. We also conducted this study to determine if agile methods were beneficial or detrimental to decision making, a question that has not been addressed empirically in the agile literature, to the best of our knowledge.

We present background work in Section 2 and describe our empirical study in Section 3. Results are presented in Section 4 and validity is described in Section 5. In Section 6, we conclude this work.

2 Background

We discuss four topics: decision making, problem structuring, qualitative studies of designers at work, and our past study on design decision making. First, we use rational and natural decision making as the conceptual frameworks to evaluate software design decision making processes. Rational decision making (RDM) is characterized by consequential choice of an alternative and an optimal selection among alternatives [11, 12]. Natural decision making (NDM) is characterized by serial evaluation of alternatives in dynamic and often turbulent situations [9]. *One alternative at a time* is evaluated and a satisficing goal is applied [9]. There are numerous perspectives from which to view design decision making (e.g. real options theory [3]) and we are in the process of comparing our data with different perspectives. For now we focus on the difference between consequential choice and serial evaluation provided by RDM and NDM.

Second, software design is a problem structuring activity accomplished throughout the software development lifecycle [5, 6, 7]. A well-structured problem (WSP) is a problem that has criteria that reveal relationships between the characteristics of a problem domain and the characteristics of a method by which to solve the problem [15]. An ill-structured problem (ISP) is a problem that is not well structured [15]. Most of problem solving is problem structuring, converting an ISP to a WSP [15].

Third, a survey of software design studies shows that five related factors impact software design: expertise, mental modeling, mental simulation, continual restructuring, and group interactions. Expertise is the knowledge and experience software designers bring to design [1]. Existing studies showed higher expertise resulted in an improved ability to create internal models and run mental simulations [1, 16]. Mental modeling is the creation of internal or external models by a designer. A mental model is capable of supporting mental simulation [1]. Mental simulation is the "ability to imagine people and objects consciously and to transform those people and objects through several transitions" [9]. Mental simulations occurred throughout the software design process at varying levels of quality dependent upon the skill of the designer and the quality of the model on which the mental simulation ran [1, 4, 5, 6]. Continual restructuring is the process of turning an ISP to a WSP. Group interactions are the dynamics of group work in software design. The terms "distributed" and "shared" cognition suggest that individual mental models coalesce via group work, resulting in strongly overlapping – mental model [5, 18].

Fourth, our previous study examined how agile developers make decisions [20]. We found that agile developers used NDM to a) recognize a design decision needed to be made, b) recall past experiences in design and c) apply external goals (e.g. marketing pressures) to the decision problem. We also found that agile developers used RDM to a) apply consequential choice in unstructured decisions and b) manage time pressure in decision problems. What was unclear from our study was when and how consequential choice was applied. We were only able to note that consequential choice was used in unstructured decisions [20]. However, exactly half of our interview subjects discussed the use of serial evaluation. Given the important role consequential choice and serial evaluation have in RDM and NDM respectively, and the frequency of each approach in our interviews, we pursued this specific topic via observations, to determine if our results were as mixed as in our first study.

3 The Empirical Study

The purpose of this empirical study is to address the question: Does consequential choice occur more, less or with equal frequency in small agile and small non-agile software development organizations? Our qualitative study used observations conducted from January to April 2006 and used convenience sampling [13]. At one organization we had a personal contact, at the other two organizations we advertised on a professional newsgroup. We spent 1-3 days with each developer depending on availability and environment. Developers in Company B pair programmed, thus we did not observe one developer at a time, as in Companies A and C.

Company A develops point of sale systems for the restaurant industry. There were 3 developers and development tasks were assigned to individual developers that specialized in certain components of the system: one developer focused on the back-end, another focused on the user interface. The third developer was also a manger and developed wherever was needed. Company A was not familiar with Agile methods, but due to the size of the organization they exhibited some traits of Agile methods such as direct verbal communication over heavy documentation [2].

Company B develops logistics software for the supply-chain industry. There were 6 developers and the breakdown of tasks was group-based. Company B followed Agile methods – specifically Extreme Programming – in a regimented fashion. Pair programming was performed, with 5 of the 6 developers sitting at two tables together.

Company C develops integration products for information technology operations management. There were between 11 and 14 developers (hiring occurred during our observations). Each developer was assigned their own mini-project and was matched with someone (called a "shadow") who had previous experience in the area or could take over the task should the primary developer not be available. Company C used a few agile practices such as standup meetings, story cards and iteration planning.

Table 1 provides a description of each company. At Company B we observed 6 developers plus 1 person who played the role of customer contact. This person was not a developer but interacted with the developers so often that it was impossible not to observe his role in the discussions. Table 2 describes the agile practices we observed at each company. We were unclear about the agility of Company C. They claimed to be agile and used some of the practices [2]. However, we did not observe traits we would expect of an agile culture. For example, we observed numerous developers who were *repeatedly* apologetic about interrupting work when asking questions of their colleagues; we observed developers who were so focused on their *own* tasks that they were unwilling to volunteer for new tasks or assist colleagues with new tasks; we observed a separation of development tasks among team members to the extent that an integral member of the team calmly referred to their development as a factory. The culture of Company C was so significantly different than the culture of Company B (whose agility was extremely apparent) that we report on the concepts presented in Table 3. Table 3 describes our observations of each company's ability to fulfill the principles of the Agile Manifesto [2]. We recognize the risk of using the word "culture" and the subjective nature of our classification of agile culture. Yet it is imperative that we express the difference between Company C's claim to agility and our observations. For Company C, where we found an agile culture was not supported, we have provided quotes from our field notes for support in Table 3.

Table 1: General Description of Each Company

Parameter	Company A	Company B	Company C
Breakdown of People Observed	2 Developers 1 Manager/Developer	5 Developers 1 Manager/Developer 1 Customer Contact	3 Developers 1 Lead Developer
Number of People Observed	3	7	4
Total Number of Developers at Organization	3	6	11-14
Days Spent with Each Participant	3	1-2	2
Days Spent at Organization	10	9	9
Division of Labour	Individual	Group-based	Individual

Table 2: Agile Practice Present in Company

Agile Practice	Company A	Company B	Company C
Iteration Planning	No	Yes	Yes
Pair Programming	No	Yes	No
Versioning System	No	Yes	Unsure
Stand Up meeting	No	Yes	Yes
Unit Test	No	Yes	Yes
Unit Test First	No	Yes	No
Collective Code Ownership	No	Yes	No
Move People Around	No	Yes	No
Integrate Often	No	Yes	No
Refactoring	No	Yes	No

We used content analysis to examine our field notes [10]. First we made case study summaries of 28 decision events we found in the field notes from our observations. Next, we used content analysis to classify a case study summary as using consequential choice or serial evaluation. Content analysis places phrases, sentences or paragraphs into codes (i.e. classifications) which can be predefined or interactively defined [10].

4 Results

The primary result of our study is that when observing design decision making, we observed more conversations in the small agile software team than in two small non-agile software teams. This led to more use of consequential choice in the small agile teams than in the small non-agile teams. We present quantitative and qualitative results to support this. Table 4 presents all our case studies showing the type of decision problem, the number of people involved and the approach to making a decision. The case study ID is indicated with an A, B or C to identify the company, and a number of the order in which it is shown.

Table 3: Agile Principle Present in Company

Agile Principle	Present in Company A	Present in Company B	Present in Company C
Individuals & Interactions over Processes & Tools	Clear	Very Clear	**Not Clear** *O1: "[Developer] says they've just started the shadowing idea, but …' it requires time and everyone is really busy. So it's the first thing to go.'"* *O2: "Assign big things to people and they go off to do individual planning. … An Iteration Planning Meeting ends. ... No one shared what tasks they had or that they needed help with anything. E.g. [Developer] had her tasks on a piece of paper with yellow stick-its and they aren't on the [main] whiteboard. So those are tasks for her no one knows about."*
Working Software over Comprehensive Documentation	Very Clear	Clear	**Not Clear** *O3: Development team has a technical writer who writes all the documentation.* *O4:"During the meeting [Developer] reminds people of the documentation process. [Manager] adds that they should use [tracking system] and should also communicate [with technical writer]."*
Customer Collaboration over Contract Negotiation	Did not Pursue Topic	Did not Pursue Topic	Did not Pursue Topic
Responding to Change over Following a Plan	Clear	Very Clear	**Not Clear** *O5: "[Developer] … said, 'Everybody is expected to meet their schedule. There is no room to fall behind.'"* *O6: "'One more day [of research phase].'"*

First, Company A (non-agile) used consequential choice in 4 of the 12 decision events we observed (33.3%). Company B (agile) used consequential choice in 9 of the 11 decision events we observed (81.8%). Company C (self-claimed agile, observed non-agile) used consequential choice in 2 of the 5 decision events we observed (40%).

We emphasize that our observations showed consequential choice when more than one developer contributed to the decision. In the 15 decision events where consequential choice was used, 14 of these involved more than 1 person contributing to the decision. In Company A more than one person was involved in decision making in 3 of the 12 decision events we observed. In Company B more than one person was involved in decision making in 9 of 11 decision events we observed. In Company C more than one person was involved in decision making in 1 of the 5 decision events we observed.

We acknowledge that there are numerous potential contributors to the approach to making a decision (e.g. the type of decision problem, the number of people involved, the culture of the organization). Our previous work has found the more structured the decision problem was (e.g. debugging), the less consequential choice was used [21].

Table 4: Quantitative Indicators of Consequential Choice in Agile Teams

Case ID	Motivator to Design Change	Number of Developers	Approach to Decision
A1	Bug Fix	1	Serial Evaluation
A2	Bug Fix, if done, halts release.	2	Consequential Choice
A3	Feature Request	1	Serial Evaluation
A4	Bug Fix	1	Serial Evaluation
A5	Feature Request	2	Consequential Choice
A6	Feature Request	1	Serial Evaluation
A7	Bug Fix	1	Serial Evaluation
A8	Bug Fix	1	Serial Evaluation
A9	Feature Request	1	Serial Evaluation
A10	Feature Request	1	Consequential Choice
A11	Feature Request	2	Consequential Choice
A12	Feature Request	1	Serial Evaluation
B1	Feature Request	1	Consequential Choice
B2	Bug Fix, if done, halts release.	4	Consequential Choice
B3	Feature Request	1*	Consequential Choice
B4	Bug Fix	1**	Serial Evaluation
B5	Feature Request	1	Consequential Choice
B6	Feature Request	2	Consequential Choice
B7	Feature Request	2	Serial Evaluation
B8	Feature Request	2	Consequential Choice
B9	Feature Request	3	Consequential Choice
B10	Bug Fix	2	Consequential Choice
B11	Feature Request	2	Consequential Choice
C1	Feature Request	1	Consequential Choice
C2	Feature Request	1	Serial Evaluation
C3	Bug Fix	1	Serial Evaluation
C4	Chose Programming Language	4	Consequential Choice
C5	Bug Fix	1	Serial Evaluation

*developer asked entire room open question. **a 2nd developer helped part-way through task.

Our results here are consistent with that. In 8 decision events that involved bug fixes (and as per [21], were well structured), only 1 decision event involved the use of consequential choice. These do not include decision events A2 and B2 listed in Table 4, which were bug fixes that had significant impact on the release of the software product. These decision events were not as well-structured as the other 8 bug fixes.

We provide excerpts from our field notes to show the different approaches to making a decision, in the different companies and to highlight that there was a cultural quality *not* present in Company A and Company C, but present in Company B. The cultural quality was an openness and willingness to challenge ideas. In Company A and Company C, more often than not, there was simply no one around to challenge an idea a developer had. Excerpts from our field notes are as follows:

"[Developer] has coded an idea like this before so he is going to copy and paste and modify what he needs. He says, 'It's faster, why waste my time?'" ***Company A.***

"[Developer1] tells me they're looking at code from another project with another customer to see how they did something more complex but same sort of

problem.…'We can just take this [code from previous project]. Just change the table.' [Developer2] says. They look through what they need and talk out loud about what they need. They read a large section of code and talk about changes. … [Developer1] says, 'How else are we going to do it?'… They're looking at how to sort days and months. ... Before they've just left it but 'it's a bit silly,' they say.…[Developer3] comes over and says it's ok to put the number before it, like 01Jan. [Developer2] laughs like he's not impressed. … [Developer3] agrees [to a suggestion from [Developer1]] but also says they've shown it to the customer in one way so if they can stay somewhat consistent with that it'd be good. … [Developer1] says "Each row is going to be a customer, month, no, … each row will be customer by day." He explains the "no" to [Developer2]. [Developer1] raises a question, if you ship an order over 2 days, it'll double count. [Develoepr2] asks "That happens?" [Developer1] says yes, [Developer1] says distinct count then. [Developer1] says need a month dimension. Create a fact table, it'll be a bit slower, [Developer1] says as an idea. …[Developer2] proposes a solution … and [Developer1] agrees." ***Company B.***

"[Developer] says he has never run into a situation where there was no SDK. If he can't find out that there ***is*** *an SDK, the company is going to need a consultant."* ***Company C.***

5 Validity

We address the validity and repeatability of our results. First, we selected observations for this study because it was the second step in a larger study on design decision making [21]. We recognize that our observations were only a glimpse into the life of an organization. While our observations occurred for 9-10 days, factors that affect our conclusions may have occurred before or after our arrival, without our knowledge. Future long-term studies are needed to validate our results. We addressed repeatability of our coding using a consensus estimate for our measure of inter-rater reliability [17]. We compared results of analysis performed by an external coder for 8 of the 28 case studies (29% of our data). We achieved a consensus estimate of 76%, comfortably above the recommended rate of 70% [17]. Lastly, we compared our results of this paper with the results presented in our previous work [20]. We found in our previous study, of the six agile developers who said they used serial evaluation in the decision event they described, only one described other people present during this decision. The other 5 agile developers described decisions they made by themselves. Thus, our theory that agile environments that lead to communication lead to the use of consequential choice, still applies. In comparison to our previous results [20, 21] this study shows multiple people involved in decision making leads to RDM in decision making, whereas fewer people involved in decision making leads to NDM.

6 Conclusion

We presented a multi-case study of software design decision making, examining the use of consequential choice and serial evaluation in small agile and non-agile

software companies. We show a small agile company using consequential choice often while two small non-agile companies applied mostly serial evaluation. Our theory is that agile environments produce more open communication among developers, which results in developers challenging each other's ideas for solutions to design problems, thereby increasing consequential choice. This result is surprising given that the reduced amount of documentation produced by agile teams is often associated with a less rational approach to software design. However the approach to decision making we observed is arguably rational. We are also now motivated to ask: which approach results in better software design? We believe consequential choice provides an opportunity to find "better" alternatives to a design decision and serial evaluation limits this opportunity. Future work includes long term empirical studies examining the impact of design decisions.

References

1. Adelson B, et al. "The Role of Domain Experience in Soft. Design"; *IEEE Trans. Soft. Eng* 11 11 Nov. 1985.
2. Agile Manifesto www.agilemanifesto.org (12/06/2005)
3. Boehm B et al.; "Software Economics, A Roadmap"; *Int. Conf. on S/W Eng. Proc. Conf. Future of S/W Eng*., Limerick, Ireland Pages: 319 – 343, 2000
4. Curtis B, et al. "A Field Study of the Soft. Des. Process for Large Systems", *Comm. ACM*, 31 11 Nov. 1988.
5. Gasson S; "Framing Design: A Social Process View of Information System Development"; *Proc. Int. Conf. Information Systems*, Helsinki Finland, 224-236; 1998.
6. Guindon R; "Designing the Design Process" *HCI* 5, 305-344; 1990.
7. Herbsleb J, et al. "Formulation and Preliminary Test of an Empirical Theory of Coord.in Soft. Eng."; *Eur. Soft. Eng. Conf./ACM SIGSOFT Symp. Found. Soft. Eng*; 2003.
8. Highsmith J; Agile Project Management; Add-Wesley; 2004
9. Klein G; Sources of Power, MIT Press Camb., MA; 1998
10. Krippendorff; Content Analysis; V5 Sage Pub. Lond. 1980
11. Lipshitz R; "Decision Making as Argument-Driven Action"; In: Klein et al, Decision Making in Action; NJ: Ablex Pub. Corp.; 1993.
12. Luce et al Games & Decisions John Wiley & Sons NY 1958
13. Patton M.Q; Qualitative Research & Evaluation Methods 3rd Ed.; Sage Pub, CA; 2002.
14. Rugaber S et al. "Recognizing Design Decisions in Programs" *IEEE Software*; 1990.
15. Simon H; "The Structure of Ill Structured Problems"; *AI* V4, 181-201; 1973
16. Sonnetag S; "Expertise in Professional Soft. Design" *J. App. Psych*. 83 5 703-715 1998
17. Stemler SE. "A Comparison of Consensus, Consistency and Measurement Approaches to Estimating Interrater Reliability" Practical Assessment, Research & Evaluation, 9(4). Retrieved October 30, 2006 from http://PAREonline.net/getvn.asp?v=9&n=4.
18. Walz D.B, et al. "Inside a Software Design Team"; *Comm. ACM*, V.36 No.10 Oct 1993.
19. Yin, R.K; Case Study Research: Design & Methods 3rd Ed. Sage Publications, CA, 2003
20. Zannier et al; "Foundations of Agile Decision Making from Agile Developers & Mentors" 7th Int. Conf. Ext. Prog.& Agile Proc. in Soft. Eng.; Springer; Finland, June 2006.
21. Zannier C, Chiasson F, Maurer F; "A Model of Design Decision Making based on Empirical Results of Interviews with Software Designers" Understanding the Social Side of Soft. Eng. A Special Issue of the J. of Info. and Soft. Tech., to appear Spring 2007.