# Foundations of Agile Decision Making from Agile Mentors and Developers

Carmen Zannier[1], Frank Maurer[1]

[1] University of Calgary, Department of Computer Science
2500 University Drive NW, Calgary, AB, CAN
{zannierc, maurer}@cpsc.ucalgary.ca
http://ebe.cpsc.ucalgary.ca/ebe

**Abstract.** There are few studies of how software developers make decisions in software design and none that places agile in the context of these decision making processes. In this paper, we present results of interviewing agile software developers and mentors to determine how design decision making aligns with rational decision making or naturalistic decision making. We present results of twelve case studies evaluating how agile professionals make design decisions, comparing mentor perspectives to developer perspectives. We describe our interview technique, content analysis used to analyze interview transcripts, and the interpretation of our results, to answer the question: how do agile designers make design decisions? Our results show that naturalistic decision making dominates design decision making but is supported by rational decision making.

## 1 Introduction

In this paper, we examine how agile software designers make software design decisions. There are three reasons to examine this topic. Firstly, little work exists concerning how decisions are made in software design [5][11], and none of this work focuses on agile methodologies, but evidence shows the ubiquity of design decisions in software development and the significant impacts of these decisions on software development [21][11][12]. As a result, there is a strong call and need to examine software design decisions [1][5][11][12][21][24]. Secondly, this topic provides insight into the important behavioral dimensions surrounding software design. Our work underscores the idea that, "the major problems of [software design] work are not so much technological as sociological in nature" [6], as well as the agile value of People and Interactions over Processes and Tools [2]. Lastly, by understanding the way that designers work and think, we can evaluate existing design processes and metrics against the way designers actually work and think, and we can motivate design processes and metrics suited to inherent work and thought processes.

Our multi-case study of twelve members of the agile community looks for consistency between agile mentors' ideas about design decisions and agile developers' practices in making design decisions. We define a design decision as the selection of an option among zero or more known and unknown options concerning the design of

a software application [26]. We say zero or more because making no choice is still making a choice. We define an agile decision as a decision occurring in an agile environment. We define an agile *developer* as an interview subject who discussed a design decision that they championed when s/he was a member of a design team. We define an agile *mentor* as an interview subject who discussed design decisions as an abstract concept, based on the culmination of design experiences with software development teams where s/he was a coach or paid consultant. More than one design decision was discussed briefly in mentor interviews, as opposed to a detailed discussion of one design decision, in a developer interview. The abstract level at which mentors discussed design decisions allowed us to compare general understanding of agile work (e.g. agile literature and rhetoric found in the agile community) to actual practice of agile work, as reported by members of the agile community. We do not evaluate the quality of the decision.

Our empirical study provides two results in the area of agile design decision making. Firstly, we find agile design decision making includes elements of both rational decision making (RDM) and naturalistic decision making (NDM) [14][17]. The current state of decision making literature suggests these decision making approaches are independent of each other. For example, fire-fighters use NDM [14], and operations researchers use RDM [17]. Our results show that in agile design, decisions are made using aspects of both, concurrently. This impacts the area of agile design by challenging traditional views of decisions, making agile research a forerunner in design decision making research. Our second result shows much agreement between agile developers and mentors. This impacts the area of agile design by strongly suggesting that what agile developers say they do is closely aligned with what they are seen doing. Such agreement in the agile community is a qualitative indicator of the effectiveness of agile literature and rhetoric.

Section 2 provides the background work and Section 3 describes our methodology. Section 4 describes results that emerged from our interviews. Section 5 compares quotes from agile mentors and agile developers to show similarities and differences. Section 6 discusses validity and Section 7 concludes this work.

## 2 Background

### 2.1 Decision Making

We use the concepts of rational and naturalistic decision making to provide insight on software design decision making. Rational decision making (RDM) is characterized by consequential choice of an alternative [17] and an optimal selection among alternatives. To select an optimal alternative, three features are required. First, alternatives are represented by a set of possible courses of action and potential outcomes for each action. Second, a utility function assigns a value to each possible action based on the attributes of its outcome. Third, a decision has probabilities for which outcome will occur given the selection of an alternative. Consequential choice is the analysis of

alternatives and potential outcomes, typical of rational decision theory [16][17]. While consequential choice is a main factor of RDM, three other assumptions are also important. The first is the possible courses of action and the probability of specific outcomes are known. The second is a decision maker pursues optimality. The third is the large amount of time calculating alternatives is acceptable [14][19].

Naturalistic decision making (NDM) is defined by six characteristics [14]. A naturalistic decision appears in dynamic and turbulent situations. It embodies fast reactions to changes and embraces ill-defined tasks and goals. A naturalistic decision is resolved under the guidance of knowledgeable decision makers. It uses situation assessment and has a goal of satisficing design alternatives, instead of optimizing them. Situation assessment is the evaluation of a single alternative. A decision maker exercises this alternative after determining it is "good enough" [14]. Satisficing is the acceptance of a satisfactory alternative (e.g. "Good Enough Software") [14][4].

### 2.2 Doing Software Design

Software design is a problem structuring activity accomplished throughout the software development lifecycle [7][8][9][10]. 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 [22]. An ill-structured problem (ISP) is a problem that is not well structured [30].

A survey of software design studies, [1][3][5][7][8][9][10][18][21][23][24], shows that six related qualities impact software design: expertise, mental modeling, mental simulation, continual restructuring, preferred evaluation criteria and group interactions. While we do not consider this to be an exhaustive list of what impacts software design, the qualities and the studies give us some background about the way designers work. Expertise is the knowledge and experience software designers have in design [1]. Existing studies showed expertise is fundamental to design productivity [5], and that higher expertise resulted in an improved ability to create internal models and run mental simulations [1][3][23]. 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, finally picturing them in a different way than at the start" [21]. 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][5][7][8]. Continual restructuring is the process of turning an ISP to a WSP. The term "preferred evaluation criteria" refers to the minimal criteria a subject adopts to perform continual restructuring [14]. It occurred on an individual level or group level [7][24][5]. 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 a common model [7][24].

## 3 Methodology

We discuss our research methodology in terms of data collection and data analysis. We interviewed software designers about critical design incidents, and the decisions made concerning software design, using a critical decision method (CDM) [13]. The CDM studies "cognitive bases of judgment and decision making." [13]. The CDM contains questions regarding cues, knowledge, goals, options, experience, and time pressure surrounding a decision but the CDM is not indicative of NDM or RDM. Such indications are generated from our interpretations in Section 4. Table 1 lists the themes covered during the interview, and an example of how we asked the question. In practice the interview question was tailored to the context of the interview.

The software designers interviewed include recognized experts in successful software design, both as developers and as mentors. All of the interviewees presented in this paper were familiar with and used agile methods. In order to find new subjects to interview, we used snowball sampling "(getting new contacts from each person interviewed)" [20, p.194]. We did not restrict by age, experience, mentor versus developer roles, or by any other demographic characteristic. The comparison between 6 mentors and 6 developers emerged from a larger set of 25 interviews. Such emergence is indicative of certain formes of qualitative inquiry [20]. The interview subjects discussed varying types of software systems and we did not distinguish among the different types. The interview subjects in this paper discussed web based business applications, and/or small software systems (e.g. ~3-15 developers). Thus far we have not found any consistent differences in approaches to decision making, between web-based applications and non-web based applications.

**Table 1.** Design Decision Making Interview Questions

| |
|---|
| **Decision**<br>Describe how you make a design change to a system, and how you make the decision to make the change. |
| **Cues**<br>What do you see, hear, discuss, that suggests a change needs to occur? |
| **Knowledge**<br>Where do you acquire the knowledge to make the change? |
| **Options**<br>Discuss the extent to which you consider options in making a design change. |
| **Experience**<br>To what extent do specific past experiences impact your decision? |
| **Time Pressure**<br>How does time pressure impact decisions in design changes? |
| **Externals**<br>How do external goals impact decisions in design changes? |

Consistent with naturalistic inquiry [14], we examined each critical design incident as an explanatory case study of the context and circumstances surrounding one or more design decisions [25]. We considered each case separately in order to allow the

participant viewpoint to speak for itself. From a single case, we then made initial statements about software design decision making and then revisited the initial case and examined new cases to continuously shape the statements [25]. This pattern continued, incorporating more cases, until our theoretical propositions about design decision making were able to explain all cases [25]. This multiple-case design allows researchers to develop general knowledge about social phenomena from both the induction of data, and the deduction from theory [25].

We used content analysis [15], to identify recurring themes in the interviews to validate our theoretical propositions. Content analysis places words, phrases, sentences or paragraphs into codes which can be predefined or interactively defined [15]. Finally, for each question we used interpretations in Table 2 to decide if interview answers followed RDM or NDM.

## 4 Thematic Results

Using RDM and NDM to guide the generalizing of our interviewee approaches to decision making [25], we interpreted each interview question as it related to the attributes of RDM and NDM. We found differences between RDM and NDM in the *goal, method, effect of environment,* and the *nature of the knowledge* employed in the decision as shown in Table 2. In the following description of Table 2 the bold text represents data we collected during our interviews. Each of the four paragraphs contain propositions we formed, evaluated and modified as we analyzed each case.

If a decision maker's goal was to *optimize* design (rational), then **information cues** were considered to indicate *right or wrong* decisions. If the decision maker's goal was to *satisfice* design (naturalistic), then **cues** were used only to indicate *better or worse* outcomes.

If a decision maker followed *consequential choice* (rational), then s/he **discussed numerous options** surrounding the decision to make a design change. If a decision maker followed *singular evaluation* (naturalistic), then s/he **did not discuss options**.

If a decision maker was *unconcerned about time pressure* and the external environment (rational), then s/he was **unconcerned with computational overhead** and **external goals**. On the other hand, if the decision maker was *concerned about time pressure* (naturalistic), then *dynamic conditions, real-time reactions, ill-defined tasks and goals* and *situation assessment* allowed external goals to influence decision making, thus providing **little time and point** in considering detailed computations.

If the decision maker was *cognizant of all possible courses of action* (rational), then **experience, knowledge and explicit searches** were used to reach decisions. If the decision maker was *not* cognizant of all possible courses of action (naturalistic), then s/he relied on general accumulation of experience or knowledge. Given these general interpretations, more detailed results illustrate similarities and differences within and across cases.

Table 2. Foundations of Agile Decision Making

| Component | 1 RDM | 2 NDM |
|---|---|---|
| Decision Goal | (1.1) *Optimizing:* Cues are right or wrong, quantifiable | (2.1*) Satisficing*: Cues are better or worse, not quantifiable. |
| Decision Method | (1.2) *Consequential choice*: Options are considered. | (2.2) *Singular Evaluation*: Options are not considered. |
| Decision Environment | (1.3) *Not concerned with computation overhead*: Time pressure is not a factor in decision making. External goals do not impact decision making. Cues are quantifiable. | (2.3) *Dynamic conditions*: External goals impact decision making. Time pressure impacts decision making.<br>(2.4) *Real-time reactions*: Time pressure is an issue. Cues are from some trigger.<br>(2.5) *Ill-defined tasks & goals*: Externals impact a decision.<br>(2.6) *Situation assessment*: Cues are unquantifiable. |
| Decision Knowledge | (1.4) *Cognizant of all possible courses of action*: Specific experience based knowledge, explicit search of knowledge. | (2.7) *Tacit based knowledge*: Accumulation of knowledge.<br>(2.8) *Experience-based knowledge*: Accumulation of experience. |

## 5 Mentor and Developer Comparison

We examine each question from Table 1, across all cases, with respect to NDM or RDM and highlight similarities and differences between mentors and developers.

### 5.1 Cues

Every case showed that cues to design decisions were difficult to quantify, and thus needed to be qualified in some way. Feedback from people or a desire to make software code express what you want are examples of cues to a design decision. There was much agreement between the mentor perspective and the developer perspective regarding the qualitative nature of cues to a design change. For example,

*Q1: "You looked at [the data model] and the picture was scrambly. It was like spaghetti code, a spaghetti data model; there were lines everywhere… And that project failed. So we worked on that project to try to do it again and it was very interesting because at the end we had something that you could look at and it was aesthetically pleasing." Mentor*

*Q2: "But whenever a customer comes in and says we need a new fee based on this, it's a substantial amount of work to implement if it's not something that is already supported in the old system." Developer*

Given a large number of quotes such as these and our interpretations found in Table 2 (point 2.1) we conclude that cues to design decisions are more naturalistic than rational. We find consistent results between the agile mentors and agile developers.

### 5.2 Knowledge

The results from the knowledge question were extremely mixed. Regarding the knowledge a designer used in making a design decision, some developers reported having a general awareness of ideas, knowing only small things, or the absence of an actual search for knowledge. Given Table 2 (specifically points 2.7 and 2.8) knowledge seems to align with NDM. However, some mentors reported searching for knowledge or actively seeking it out, which aligns more with RDM (Table 2, point 1.4). There was not a clear disagreement between the mentor and developer perspective but in general the mentors spoke in a more positive fashion about actively searching out knowledge, than the developers did. For example,

*Q3: "I don't think people come to a design problem and then look it up in … [a] book. At least not when they're expected to." Mentor*

*Q4: "I'm appalled sometimes at how little people read, how little people go after knowledge … in organizations." Mentor*

*Q5: "I'm always amazed at the teams I work with that won't read. It drives me nuts… I don't see any commonality in what people look at in books. I don't see them looking anyway – which drives me nuts! It really does." Mentor*

*Q6: "I never use those books. … You talk with purists and they say … every programmer has read 30, 50, 100 books about this and every year there's new bibles coming out…I grew up in software, I guess it comes to me naturally." Developer*

*Q7: "I haven't read a lot of books but there are a few that I have read … You can read … but until you actually put it into practice you never know, sort of wonder, if what you read is right or if you even remember it." Developer*

*Q8: "I went out and bought a book, the first book on XSLT that came out, a very nice book, and I just started devouring that book and started trying it out, seeing what worked. … I bought the book to implement this design idea." Developer*

We found no pattern yet as to when the pursuit of knowledge aligns more with NDM or RDM. We found no pattern among the mentors and developers, suggesting a lack of understanding of the way that agile designers learn and pursue learning.

### 5.3 Options

The results from the options question were mixed as well. Regarding the extent to which a software designer used consequential choice, the six mentors reported considering options as an integral step in making a design decision. Given Table 2, point 1.2, this would suggest that design decision making is rational in nature. However, the six developers reported not considering options to any large degree, choosing an option they believed would work or choosing an option based on what was the easiest at the time. Given Table 2, point 2.2 this would suggest decision making is more naturalistic in nature. For example,

*Q9: "…whenever I start thinking about a problem, between the first moment the problem gets into my head and the moment my fingers hit the keyboard, I'm thinking about alternatives …." Mentor*

*Q10: "[The chosen option] was pretty close to the first one that … popped up on our radar when we started looking. [It] ended up working and it was the first one that I tried so [I] didn't really look at too many other options, no." Developer*

In general we found that the larger the decision, the more the decision maker considered options. One interpretation of the mentor perspective is that mentors deal more with larger software design decisions (e.g. architecture, design patterns) than the type of decisions a developer perspective would discuss (e.g. automated refactorings), which is why we have a split between our developers and mentors in their approaches to decision making.

### 5.4 Experience

Similar to cues, a software designer's reliance on past experiences aligned with NDM across all cases. The case studies suggested that software designers rely on a general sense of past experiences, but also rely on assessing current situations. None of the results discussed applying specific past solutions to current design problems. There was agreement between mentor and developer perspectives. For example,

*Q11: "In programming it seems to me as if everything I've ever learned is just in [my brain] … a lot of times I know a thing but I have no idea where it came from. My brain has turned into this much of ideas that are accessible to me more or less but I don't remember the sources, I don't associate them." Mentor*

*Q12: "[The design decision] was generally around the problem of refactoring, so I thought our code [was] too verbose and I thought we were using the wrong solution for the problem. So yes, they were reminders of that, that's pretty generic. Where you're working in the system and you say no, we're solving this problem in the wrong way. We could solve it in a completely different way that would be much better. … I had definitely experienced that before." Developer*

From our interpretations found in Table 2 (point 2.8), we conclude that a software designer's use of past experience aligns more with NDM than with RDM. We found consistent results between agile mentors and agile developers in our study.

### 5.5 Time Pressure & External Goals

Time pressure did not impact decision making for the majority of the cases reporting results on time pressure and there was much agreement between the mentor and developer perspectives. Given our interpretations from Table 2 (point 1.3), we conclude that time pressure and its impact on decision making align more with RDM than with NDM. Lastly, the impact of external goals on decision making produced a 9:1:2 split NDM:RDM:N/A for the case studies, with much agreement between the mentor and developer perspectives. Given Table 2, points 2.3 and 2.5, the impact of external goals aligns more with NDM than RDM. For both time pressure and external goals we found consistent results between agile mentors and agile developers.

### 5.6 Design Decision

We find RDM occurring in design decision making in the form of consequential choice in large design decisions, in the absence of computational overhead involved in decision making and sometimes in the pursuit of knowledge used to make a decision. While NDM is dominant in recognizing a decision needs to be made, in the use of past experiences and in the impact of external goals, NDM is supported by RDM.

## 6 Validity

Given that we conducted case studies our external validity relies on generalization to theory (analytical generalization), and not statistical generalization [25]. All of our case studies align with our theory described in Section 5.6. For example, one agile developer (from Quote 8) found code “awkward” and “verbose” (qualitative cues) so he read up on XSLT as a solution to his problem (search for knowledge), “sold the idea” to business managers (external goals), and then prototyped his idea and fully implemented it once the prototype worked (singular evaluation of alternatives, satisficing). He felt little time pressure and was not reminded of specific past experiences.

## 7 Conclusions

We have presented results of a multi-case study with software designers concerning how they make design decisions. We conclude that NDM dominates design decision making, with support from RDM where conditions are suitable. Understanding the nature of software design decision making yields much insight into the way that people work, motivating the development of design processes and metrics tailored to our inherent approaches to decision making. For example, if a junior software designer follows a satisficing approach to his/her design decisions because s/he is under significant time pressure, and if the software engineering community cannot say that such an approach to design is right or wrong (i.e. it just _is_), then design metrics should evaluate the resulting design in light of the junior developer’s knowledge/expertise/experience/etcetera and the time pressure imposed on the design decision. Comparing abstract ideas about the nature of design decisions to specific experiences in design decisions qualitatively indicates that agile rhetoric is mostly effective. As a community we lack consensual understanding of how agile designers pursue learning. This work motivates design processes and metrics that incorporate the intrinsic nature of agile software design.

## Acknowledgments

We thank all of our interview participants who took time to speak with us.

## 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 (08/17/2005)
3. Ahmed S et al. “Understanding Differences Between How Novice & Experienced Designers Approach Design Tasks”; *Res. Eng. Design* 14, 1-11 2003.
4. Bach, J. The Challenge of "Good Enough Software", *American Programmer*, Oct, 1995
5. Curtis B, et al. “A Field Study of the Soft. Des. Process for Large Systems”, *Comm. ACM*, 31 11 Nov. 1988.
6. Demarco T, et al. Peopleware; 2nd Ed. Dorset House Pub. Co. NY; 1999.
7. Gasson S; “Framing Design: A Social Process View of Information System Development”; *Proc. Int. Conf. Information Systems*, Helsinki Finland, 224-236; 1998.
8. Guindon R; “Designing the Design Process” *HCI* 5, 305-344; 1990.
9. Guindon R; “Knowledge Exploited by Experts During Software Sys. Design”; *Int. J. Man-Mach. Stud*. 33, 279-304; 1990.
10. 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.
11. Highsmith J; Agile Project Management; Add-Wesley; 2004
12. Highsmith J; Agile Software Development Ecosystems; Addison-Wesley; 2003;
13. Klein G et al; “Critical Decision Method for Eliciting Knowledge” *IEEE Trans. Sys, Man and Cyber*.; 19, 3, 1989.
14. Klein G; Sources of Power, MIT Press Camb., MA; 1998
15. Krippendorff; Content Analysis; V5 Sage Pub. Lond. 1980
16. Lipshitz R; “Decision Making as Argument-Driven Action”; In: Klein et al, Decision Making in Action; NJ: Ablex Pub. Corp.; 1993.
17. Luce et al Games & Decisions John Wiley & Sons NY 1958
18. Malhotra et al “Cognitive Processes in Design” *Int. J. Man-Mach. Stud* 12 119-140 1980
19. Orasanu J et al; “The Reinvention of Decision Making”; In: Klein et al, Decision Making in Action; NJ Ablex; 1993.
20. Patton M.Q; Qualitative Research & Evaluation Methods 3rd Ed.; Sage Pub, CA; 2002.
21. Rugaber S et al. “Recognizing Design Decisions in Programs” *IEEE Software*; 1990.
22. Simon H; “The Structure of Ill Structured Problems”; *AI* V4, 181-201; 1973
23. Sonnetag S; “Expertise in Professional Soft. Design” *J. App. Psych*. 83 5 703-715 1998
24. Walz D.B, et al. “Inside a Software Design Team”; *Comm. ACM*, V.36 No.10 Oct 1993.
25. Yin, R.K; Case Study Research: Design & Methods 3rd Ed. Sage Publications, CA, 2003
26. Zannier C, et al.; “A Qualitative Empirical Evaluation of Design Decisions”; *Wkshp on Human & Social Factors of Soft. Eng.*; ACM Press: 2005