COMPARING TRADITIONAL AND AGILE DEVELOPMENT APPROACHES: THE CASE OF EXTREME PROGRAMMING

Dr. Mary Helen Fagan, University of Texas at Tyler, mfagan@uttyler.edu

ABSTRACT

Some adherents of agile development feel their approach goes against the predominant tenets of existing traditional approaches to software development, and thus can invoke fear and anxiety in practitioners. One way to understand if and how agile methods differ from existing approaches is to explore one agile approach in depth and, if possible, its philosophical underpinnings to see in what way it presents a new and different view of the software development process. This paper works to understand some of the philosophical underpinnings of Extreme Programming (XP), an agile software development approach. The study uses a qualitative data analysis approach to address the following research questions: 1) what philosophy underlies XP, and 2) does XP's philosophy differ from the dominant software development paradigm? The paper finds that Extreme Programming reflects many pragmatic philosophical viewpoints and that many of its tenets conflict with the rationalistic approach that underlies traditional software development discourse.

Keywords: Extreme Programming, XP, Agile development methods, pragmatism

INTRODUCTION

There are a variety of approaches that make up the agile software development movement in information systems. The Agile Alliance manifesto and principles provide insight into the common views shared by the diverse group of agile development practitioners [1]. Some adherents of agile development feel their approach goes against the predominant tenets of existing traditional approaches to software development, and thus can invoke fear and anxiety in practitioners [2]. One way to understand if and how agile methods differ from existing approaches is to explore one agile approach in depth, and, if possible, its philosophical underpinnings, to see in what way in presents a new and different view of the software development process.

This paper works to understand some of the philosophical underpinnings of one of the agile approaches known as Extreme Programming (XP). By studying writing on XP [2] and work that is given credit as contributing to its philosophy [5], this paper addresses the following research questions: 1) what philosophy underlies XP, and 2) does XP's philosophy differ from the dominant software development paradigm? The analysis is organized as follows. First, the study's methodology is described. Then the results of the analysis are outlined in the findings section. Next, some implications of the findings are explored in the discussion section, and finally, conclusions are drawn.

METHODOLOGY

This study uses a qualitative data analysis strategy based upon the work of Miles and Huberman [8]. The research questions drove the initial selection of data for analysis: Beck [2] and Coyne [5]. First, the works were read in detail and codes were noted along with reflections in the margins. Then, in an effort to facilitate the coding process, material from Coyne's chapters that contained themes, phrases, and concepts relevant to the research questions was data entered and open coding was performed using the qualitative data analysis software package, Atlas/TI. The main concepts were identified and tables were developed to compare and contrast key concepts regarding the rationalistic and pragmatic approach to design from Coyne [5]. Then Beck [2] was re-analyzed to identify commonalities and differences and these findings were written up in analysis memos. Further material was analyzed at a high level relating to XP [3] and another agile approach [4] to see if more material that illustrated the pragmatic approach to design could be identified. This review concluded that sufficient material was available in Beck [2] related to XP to form the basis of analysis for this initial study. Then a search was conducted to find additional material related to John Dewey, a key figure in philosophical pragmatism [6, 7, 9, 10]. Work related to Dewey's pragmatism, especially in the area of experience and education, was reviewed to discern common concepts and themes between these works, Beck [2], and Coyne [5]. The review of work related to Dewey opened up fruitful areas of analysis as it reinforced themes that had not been stressed in the initial coding and analysis process.

The Agile Alliance manifesto was used as an organizing device for relating the findings of the qualitative data analysis. The manifesto states, "We are uncovering better ways of developing software by doing it and helping others to do it. Through this work we have come to value

- **individuals and interactions** over processes and tools
- **working software** over comprehensive documentation
- **customer collaboration** over contract negotiation
- **responding to change** over following a plan" [1].

FINDINGS

In the bibliography of his book on XP, Beck [2] indicates that the Coyne's [5] discussion of the differences between modernist and postmodernist thought is a theme that is important to XP. Coyne [5] describes the traditional approach to systems development as modernist and based upon Cartesian rationalism. Rationalism could be described as the dominant paradigm in information technology (IT) development since "rationalism is a discursive practice we are all caught up in and it imbues understanding of technology and of design" [5, p. 19]. Other authors have also noted "rationalistic styles of discourse and thinking have determined the questions that have been asked and the theories, methodologies, and assumptions that have been adopted" in IT [12, p. 16]. However, although the influence of rationalism can readily be seen in the IT arena in the design methods movement and the empirical research literature, the actual operating philosophy that informs the practices of the IT world is pragmatism, a postmodernist approach according to Coyne [5]. In fact, rationalism, with its emphasis on the primacy of theory over practice stands in opposition to pragmatism, which focuses on the importance of human action and sees theory as just another form of practice.

The pragmatic orientation of agile developers can be discerned in the Agile Alliance manifesto, which states "We are uncovering better ways of developing software by doing it and helping others do it" [1]. The pragmatic focus in agile methods such as XP is on learning by doing, not by referring to the work of experts or to some theoretical literature on how design should be done. In fact, in practice, design is ad hoc and atheoretical, and "where design methods are now in use or even discussed at all, it is in the area of teaching and research rather than in design practice" [5, p. 27]. Because of the lip service given to the traditional design approach, developers may say one thing in regards to design, but actually do something else. Agile methods and XP are focused on stating values and practices based upon what has been found to actually work, based on practice, and not upon theory or research. Dewey, a key pragmatic philosopher, stressed the importance of engagement in experience, and believed "there is no such thing as genuine knowledge and fruitful understanding except as the offspring of doing" [5, p. 40]. This pragmatic perspective is in accord with the approach championed by XP. An exploration of how rationalistic and pragmatic viewpoints can be related to the four values outlined in the Agile Alliance manifesto [1] provides additional insight into the philosophical underpinnings of XP.

Manifesto value 1: individuals and interactions over processes and tools

Pragmatism "draws attention to the person engaged in a situation, rather than to the abstract world of data, information and knowledge" [5, p. xii]. In XP the focus in on individuals and their needs and desires, and not on the steps of a methodology or use of computer aided tools. XP appeals to programmers who are dissatisfied with the current approach to software development and want to see the following promises fulfilled: "they will be able to work on things that really matter, every day. They won't have to face scary situations alone. They will be able to do everything in their power to make their systems successful. They will make decisions that they can make best, and they won't make decisions that they aren't best qualified to make" [2, p. xvi]. In XP, human contact among team members is designed to reduce loneliness and the practices are designed to work with, and not against, short-term human instincts. For example, programmers are asked to take responsibility for estimating the time it will take to complete their own work. By granting the individual autonomy, XP helps to ensure accountability. But, the possibility of over or under estimating the nature of the work is mitigated by the fact that individuals are given feedback on their actual performance so their estimates should improve over time. XP, as a lightweight approach, differs from traditional approaches to development. In fact, as Tom DeMarco says in the introduction to a book on XP planning, the "principles of XP are not just another methodology, another process. They are the antithesis of process. They are means to make process irrelevant" [3, p. xii].

Manifesto value 2: working software over comprehensive documentation

Coyne describes how the design methods movement seeks "to capture design expertise in process diagrams, to objectify the design process and to make it explicit as an aid to collaboration and communication" [5, p. 21]. The rationalistic orientation of the traditional design methods movement is reflected in its assumption that design documents (such as diagrams) can adequately represent and communicate knowledge (and permit knowledge to "be transmitted from those who know to those who are ignorant" [5, p. 27]). Agile methods such XP

eschew the creation of design documents, except for those such as story and task cards that are temporary and only serve as a vehicle for planning and communication until they are translated into the actual code. For XP, the code tells the story and there is no need for a lengthy design phase to precede the programming stage of systems development. XP relies on "oral communication, tests, and code to communicate system structure and intent" [2, p. xvii]. The practices that are used to keep the development journey on track are simple and primarily manual. As expressed by one of the principles of the Agile Alliance [1], "the most efficient and effective method of conveying information to and within a development team is face-to-face conversation. Working software is the primary measure of progress".

Manifesto value 3: customer collaboration over contract negotiation

Pragmatism focuses on the social nature of human activity. Instead of focusing on method and the creativity of the individual designer, XP embodies the pragmatic perspective with its focus on community. XP sees all developers as capable of participating in the design of architecture (through story/metaphor development) and in design (through refactoring). This democratic perspective is extended to the customer, who is fully involved in the social process of design as a participant. XP recognizes that customers don't know what they want up front in the development process and that systems requirements emerge from "learning that can only come from experience. But customers can't get there alone. They need people who can program, not as guides, but as companions" [2, p. 19]. IT development approaches that embody a rationalistic perspective assume that needs can be identified up front and that designers can control the development process. Rationalism "favors a hierarchical and bureaucratic view of knowledge" where "professional experts can reason objectively and can see their domains more clearly than non-experts," and this leads to a minimization of the role of end-users in favor of experts who know best how to design [5, p. x]. XP, on the other hand, accepts the pragmatic perspective that design is "reflection in action" and that "needs are commonly identified in retrospect rather than at the onset of the design process" [5, p. 11]. Thus XP differs from traditional development approaches in the egalitarian role accorded to the customer and the process whereby their requirements are elicited through engagement in development.

Manifesto value 4: responding to change over following a plan

The pragmatic perspective "assumes holistic engagement beyond the exercise of some rules or principles" [5, p. 3]. XP does not propose an extensive methodology with phases and stages of development that produce design deliverables. Instead, in XP, the system development process is compared to the process of driving a car, in which constant small adjustments are made to keep the journey on track. XP affirms a design approach where activities don't have to be predetermined but emerge through the process of human interaction. The traditional design methods movement, on the other hand, displays the influence of rationalism in that it prescribes a formal step-by-step process that starts with a problem situation, determines specifications and creates software programs, after which the pieces are tested and assembled into a system that should solve the original problem statement. From a rationalistic perspective, IT development is based upon a "systems-theoretic view of design that seeks to enlist science and its methods to arrive at objectively valid solutions to problems" [5, p. 11]. XP displays a pragmatic perspective

in its vision of design as an experiment, a journey of exploration, and in this way XP differs from traditional IT development approaches.

DISCUSSION

Coyne identifies five areas where pragmatism provides a different view from rationalism: "1) interests in the practical, 2) the social nature of design, 3) the physicality of the technology, 4) bodily engagement, and the 5) formative power of technology" [5, p. 36]. This study's qualitative analysis based upon Beck [2] found support for items 1 and 2, but did not identify all of these themes in Beck's discussion of XP. In part, this is due to the fact that Beck [2] focuses on the development of code and themes that are related to this core focus and does not address the uses to which the code is put in production. Thus, in regards to XP, the literature that was included in this analysis does not include in its scope material directly related to areas 3, 4 and 5. Further analysis of literature on XP and on other agile approaches may, in the future, find material that addresses all these pragmatic viewpoints in agile software development.

Furthermore, one area where common themes were expected to emerge from the analysis was not very fruitful. In his bibliography Beck [2] referred to the importance of Coyne's discussion of metaphor. It is clear that metaphor is critically important to XP since it forms the basis for common stories that help define the architecture and guide the development process. Likewise, it is clear that Coyne places key emphasis on the role of metaphor, stating that his book describes a transition from an approach to IT design "in which method holds sway to a radical position sustained by the complex and contradictory workings of metaphor" [5, p. 16]. However, despite the fact that Coyne argues that, "the idea of metaphor as a focus of design discourse belongs within the pragmatic theme" [5, p. 12], this analysis found very little in the Coyne's general discussion of pragmatic philosophy or of Dewey's work that related metaphor with pragmatism.

Finally, Cohen points out how the pragmatic orientation that sees design as a social process is "closely linked with the tenets of political and social liberalism" [5, p. 33] and especially with the work of Dewey in education [6]. This is a theme that merits further exploration in the future. In his philosophy Dewey rejected autocratic structures and Coyne suggests that, "Dewey's liberalism supports the elevation of the social nature of design and the breaking down of cliques of theoretical expertise" [5, p. 42]. XP certainly argues for a different management and organizational structure from that traditionally advocated for large IT development projects. In XP, a manager does not plan the project, estimate work tasks and then assign work to programmers. Instead, the developers work with the customer to plan, pick and estimate their work. In XP, the manager becomes a coach whose primary role is to step in and help steer the project when things go amiss. For those who are familiar with trends in education, it appears that the manager, like the teacher, has been taken from center stage and has now become the "guide on the side", a prospect which might concern those familiar with traditional project management approaches. It would be worthwhile to explore Dewey's writings further to see how his view of the appropriate educational environment for learning might correspond with the ideal software development environment envisioned by XP.

CONCLUSION

This paper investigated the philosophy that underlies XP and whether this philosophy differs from that of traditional software development approaches. The Agile Alliance manifesto [1] was used as a framework to explore concepts and themes drawn from Beck [2] and Coyne [5]. The analysis found that XP reflects many pragmatic philosophical viewpoints. Furthermore, the analysis supports the contention that traditional software development approaches are imbued with a rationalistic perspective, and that this viewpoint is at odds with the pragmatic perspective found in XP. Some aspects of the pragmatic perspective on IT design did not find support in XP based upon Beck [2] and one area, metaphor, which was expected to be a key concept, did not find much empirical support as an aspect of pragmatic philosophy. This analysis represents a first step toward understanding the philosophical underpinnings of one agile approach, XP, and can be expanded in the future to include additional literature and themes (e.g., the correspondence between XP development and the progressive educational experiences envisioned by Dewey).

REFERENCES

1. Agile Alliance. (2001). Manifesto and Principles. Retrieved February 15, 2005 from http://agilemanifesto.org/
2. Beck, K. (2000). *Extreme programming explained: Embrace change*. Boston: Addison-Wesley.
3. Beck, K. & Fowler, M. (2001). *Planning extreme programming*. Boston: Addison-Wesley.
4. Cockburn, A. (2002). *Agile software development*. Boston: Addison-Wesley.
5. Coyne, R. (1995). *Designing information technology in the postmodern age: From method to metaphor*. Cambridge, MA: MIT Press.
6. Dewey, J. (1939). *Intelligence in the modern world: John Dewey's philosophy*. New York: The Modern Library.
7. Hickman, L. A. (1990). *John Dewey's pragmatic technology*. Bloomington: Indiana University Press.
8. Miles, M.B & Huberman, A.M. (1994). *Qualitative data analysis: An expanded sourcebook.* Thousand Oaks: Sage Publications.
9. Robertson, E. (1992). Is Dewey's educational vision still viable? *Review of Research in Education, 18*, 335-381.
10. Thayer, H.S. (1973). *Meaning and action: A study of American pragmatism*. Indianapolis: Bobbs-Merrill Company.
11. Webb, J. L. (2002). Dewey: Back to the future. *Journal of Economic Issues*, *36*(4), 981-1003.
12. Winograd, T. & Flores, F. (1986). *Understanding computers and cognition: A new foundation for design*. Norwood: Ablex Publishing Corporation.