Agile Approaches to Software Maintenance: An Exploratory Study of Practitioner Views

Warren Reyes, Deakin University, Burwood, Victoria 3125, Australia
Ross Smith, Deakin University, Burwood, Victoria 3125, Australia; E-mail: ross.smith@deakin.edu.au
Bardo Fraunholz, Deakin University, Burwood, Victoria 3125, Australia; E-mail: bardo.fraunholz@deakin.edu.au

ABSTRACT

Whilst there has been some research into the application of agile approaches to the world of software maintenance, in this paper it is argued that there has not been a coherent investigation that focuses on the collection and analysis of the views and perceptions of agile software maintenance approaches held by experienced software maintenance professionals. In this paper, we report such an exploratory investigation, which has seeded the development of a simple framework for classifying collected views and perceptions. Specifically, a matrix framework has been introduced, to facilitate comparison of the levels of understanding of the issues affecting an agile adoption decision, and the extent to which an agile approach has been implemented. Examples of organizations operating in all four cells of this matrix have been presented.

Keywords: Software Maintenance, Agile Methods, Software Development

INTRODUCTION

Software maintenance has been long-recognized as placing before IT management a critical challenge (Martin & McClure, 1983). Indeed, it can be argued that maintenance decisions can be more critical to system users than decisions taken during software development. Further, with the ever-increasing rate of software development the burden of maintenance increases. Previously developed systems must be maintained to ensure their value and sustainability within an organization. Martin & McClure (1983, p.3) defines software maintenance as "an activity which imposes changes to a software system after its release to the user or customer" and provides varied rationale including: correction of errors, improving the overall design, interfacing a program to other programs and making necessary enhancements.

Many potential solutions or "silver bullets" have been proposed over the years to ease the burden of software maintenance, some boldly proclaiming a dramatic reduction of the burden. While the adoption of some of the "silver bullets" has addressed aspects of maintenance efforts, the burden of maintenance still remains (Bennett, 2000; April et al., 2005). April et al. (2005) even argues that the maintenance burden has been compounded by some of the proposed solutions.

Agile approaches have emerged as a challenge to the status quo as they propose a substantially different, radical philosophy and process for developing software. They are a collection of methodologies, processes and tools for the creative process, that anticipate the need for flexibility and apply a level of pragmatism to the delivery of a finished software product. They seek to deal with the limitations of traditional development approaches, especially the inability to cope with an unstable and rapidly changing requirements environment.

Whilst there has been some research into the application of such agile approaches to software maintenance, we argue that there has not been a coherent investigation that focuses on the collection and analysis of the views/perceptions of agile software maintenance approaches held by experienced software maintenance professionals. In this paper we report such an exploratory investigation, which has seeded the development of a simple framework for classifying aspects of the collected views and perceptions - a framework that may well underpin future studies.

BACKGROUND

In order to situate the present research it is necessary to explore the definitions of both maintenance and agile approaches.

The term "maintenance" has been used since the early 1960s to describe the delivered modification of software on an implemented system. Terms such as "change" or "modification" commonly described activities carried out by personnel participating in the original development, while maintenance usually implies the involvement of personnel who were not party to the original development (Chapin et al., 2001). As maintenance becomes increasingly complex, (including modifications and announcements, adaptive modifications, changes reflecting shifts in processing and environments), a more sophisticated definition of software maintenance is required. Sousa and Moreira (1998, p. 265) in part address this when stating that software maintenance can be viewed as "the modification of the software product after its delivery to the customer, to correct errors, to improve its performance or other attributes, or to adapt the product to a modified environment". For our purposes this is a suitably broad definition.

It is interesting to note that compared to the software development process, research into the maintenance of software is comparatively sparse (April et al., 2005). This may well be a consequence of the so-called software cost "iceberg" (Chapin et al., 2001). Costs and issues associated with software development are explicit and visible. Software maintenance costs surface gradually, later in the system lifecycle, and as such are less visible to management. This has been long argued. Swanson (1976) for example suggests that the metaphorical "iceberg" infers that "much goes on here that does not currently meet the eye, and further that our ignorance in this regard is, in a sense dangerous". Software maintenance is performed in response to software failures, environmental changes and in response to change requests made by users. These activities can be classified as Corrective Maintenance, Adaptive Maintenance and Perfective Maintenance. Yip (1995) suggests that the maintenance component could be as high as 70-75 percent of the overall life cycle cost. In the light of these figures it is perhaps surprising that software maintenance is often overlooked and that it has not been subject to the same intensive research as the software development process.

Of the research reported, a driving focus has been the role of maintenance as a means of resolving software failure (Dekleva, 1992). In addressing this, however, there have been many foci of research interest, including: the quality of the software and its documentation (Lientz & Swanson, 1981; Dekleva, 1992; Yip, 1995; Sousa & Moreira, 1998); coordination and management (Lientz, 1983; Yip, 1995, Sousa & Moreira, 1998); testing of software modifications (Dekleva, 1992; Martin & McClure, 1983); and the domain-specific nature of software (Sousa & Moreira, 1998). To address some of these problems a number of approaches have been proposed, including the adoption of technologies such as relational databases, fourth generation programming languages, object-oriented programming techniques, structured programming techniques, reuse of modules, metrics and computer-aided software engineering environments. All of these technologies, activities and processes have the capacity to reduce, in part at least, the burden of software maintenance.

It is important, however to recognize that the above address only a subset of identified maintenance problems. In order to facilitate a holistic approach to

software maintenance some researchers (e.g. Svenssen & Höst, 2005; Poole & Huisman, 2001; Schuh, 2001) have suggested that agile approaches may have something to offer.

"Agile" or "light weight" software development approaches have emerged over recent years. Proponents suggest that these approaches are revolutionary, and as such have stimulated passionate debate within the industry. The core characteristics and benefits of agile approaches are their emphasis on highly engaged and frequent communication between project participants and clients. This facilitates frequent and intuitive releases of products, which can be evaluated immediately. A further characteristic is the claimed reduction in price to produce quality products in a short period of time, without having to resort to short cuts (Avison & Fitzgerald, 2003). Examples of agile software development approaches include: eXtreme Programming (XP); Scrum; Feature Driven Development (FDD); and Crystal (Beck, 1999; Cockburn, 2002).

As a means of characterizing such approaches, the Agile Alliance (2006) has enunciated the core values underpinning agile approaches, as:

- A means of uncovering better ways of developing software by doing it and assisting others to do it;
- Valuing individuals, interactions, working software, customer collaboration and responding to change as items of most value to practitioners or teams who apply an agile approach; and
- Such considerations are valued over other considerations such as processes, tools, comprehensive documentation, contract negotiation, and the strict adherence to a plan.

Agile approaches have also been characterized in terms of the techniques/methods that typically feature, including: *Incremental Development*; *Time Boxing; MoSCoW Rules; JAD workshops; Prototyping; the roles of Sponsor & Champion; and the adoption of supporting Toolsets* (Avison & Fitzgerald, 2003).

AGILE APPROACHES AND SOFTWARE MAINTENANCE

Lientz (1983) identifies the user-oriented nature of software maintenance as one of the most critical challenges facing IT management. The advocacy of constant and timely communication, coupled with ready feedback and iterative releases, by proponents of agile approaches may, as such, be advantageous to software maintainers. Maintainers can seek to address problems through collaboration and communication with users, thus reducing the potential to introduce further problems (Cockburn, 2002).

Agile approaches as an alternative to the traditional waterfall approach in maintenance have been studied by several researchers. E.g. Poole and Huisman (2001) demonstrate that an agile approach, XP, might be introduced into an organization as a maintenance tool. However they have identified a strong correlation between effectiveness and customer commitment to communicate with the maintenance team. Schuh (2001) suggests, however, that agile approaches might not be a blanket solution to problems faced by the development and maintenance functions. Svenssen and Höst (2005) reinforce this view in their empirical study, suggesting that agile approaches need to be adjusted or adapted to suit an organization's circumstances and situation, and that following each of the processes suggested verbatim can be a recipe for disaster.

Studies conducted thus far have, in a sense, focused upon technical and procedural activities and benefits, as opposed to building a realistic understanding of how the broader philosophy of agile approaches might assist software maintainers and users. We argue that there is a gap in substantial research, capturing the views and perceptions of front line maintenance staff as to the potential capacity of agile approaches to assist them in the performance of their day-to-day software maintenance activities. The present study describes an exploratory study that lays the groundwork for addressing this gap.

RESEARCH APPROACH

To identify practitioners' views and perceptions of the applicability of agile approaches to maintenance, and to understand the factors that influence those views and perceptions, we chose an exploratory, qualitative research approach, administered through face-to-face semi-structured interviews. Eight participants were chosen from a pool of maintenance practitioners, working in some seven different organizations. All practitioners were working as maintenance officers and had at least two years experience – some substantially more. Due care was taken to ensure a range of different systems were represented, thus avoiding domain or software-specific selection. Participants came from organizations of various sizes and types, ranging from small businesses, maintaining web-based applications, to large organizations operating sizable ERP systems and software. Table 1 presents a summary of the participant profiles.

Table 1. Summary of participant profiles

Participant Code	Organization Type	System Maintained
P1	Large banking institution.	Customer information system
P2	Medium-to-large commercial software company.	Large, commercial ERP software system.
P3	Large banking institution.	Distributed Lotus Domino applications.
P4	Small contractor.	Miscellaneous software systems (research to marketing systems).
P5 & P6	Small to Medium Enterprise (SME).	Web-based, Microsoft.NET application.
P7	Small contractor.	Websites and web-based applications
P8	Medium to large international commercial software vendor.	Large, commercial ERP software system.

We acknowledge that the number of participants have limited the validity of findings. However, being a preliminary exploratory study, the scope of this research was set to finding some indicatory insights from practitioners, so as to provide a platform for launching into further research investigations. Therefore, this small sample was rendered sufficient.

INDUSTRY STUDY

To set the scene, the following statement from participant P2 characterizes the view held by all concerning the challenge of software maintenance, as they live it, day to day:

"... We have to deal with history and archaeology. And we recognize it, but time, resources, and money constraints don't allow you to re-architect the entire portfolio in one hit. So you're constantly battling the weight of the old product with all of the measures of some of the newer modules. And that's a tug of war that our sort of business has to wrestle with."

This characterization captures the essence of software maintenance, as a struggle between legacy systems and the unrelenting need and demand for change and progress. Further, an understanding of a system's past is essential, to assist in determining the future viability and applicability of a software system.

In characterizing the views/perceptions of the software maintainers we initially report 4 primary findings. Subsequently (next section), we present a simple framework which has been helpful in practically classifying the situations thus observed in the participant organizations. Theoretically, these classifications may be used for similar studies and could also be modified in accordance with the further situational findings.

Finding 1: Software Modifications and Enhancements vs. Software Corrections and Adaptations

Consistent with the extant literature, the participants confirmed that software enhancements or modifications to an implemented software system are the most significant maintenance activity they face (Lientz, 1983; Yip, 1995). For example, P3 indicated that a high volume of requests is for the provision of upgrades and new functionalities. This was corroborated by P5, P6&P7. It is noteworthy, however, that Sousa and Moreira (1998) identify adaptive maintenance activities as the most costly software maintenance activity, at odds with the present findings.

Finding 2: User/System Knowledge and Knowledge Management are Crucial

Lientz and Swanson (1983) identify knowledge, programmer effectiveness, product quality, time availability, machine requirements and system reliability to be the factors of most pressing concern to maintainers. Consistent with the primacy of knowledge in this list, participants in the present study emphasized the need for user knowledge and system knowledge, and that the lack of documentation and/or the ability to transfer knowledge, were key issues. This problem is further compounded by a user's/customer's lack of understanding of the difficulties and the issues surrounding the performance of software maintenance. P4 provided insight by suggesting that:

"...from a customer viewpoint, they often say there's a problem in the software, it doesn't do what I want it to do… (and) they would probably classify them as software defects, but of course, if the functionality wasn't in the original requirements specification; it's not a defect, it's a modification."

To address this problem, and to introduce some form of knowledge management to maintenance activities, P1, P3, P5&P6 proposed the standardization of practices, and P4 suggested managing customer expectations by involving customers in both the development and maintenance processes.

Finding 3: Prominence of Agile (or at least Flexible) Approaches in Present Software Maintenance

Software maintenance, unlike software development is not requirements driven but is rather event driven, triggered by unscheduled or random external events (Kitchenham et al., 1999). Software organizations do not have defined processes for the conduct of their software maintenance activities, or at best software maintenance is depicted crudely as the final activity in their software development process (April et al., 2005). This view might suggest that software maintenance follows an ad hoc process in many organizations, reflecting at best some coarse steps similar to those depicted within the traditional "waterfall" model of software development. In the present study, this view was supported by P4.

In contrast, many of the participants in the present study indicated that they are using some form of agile or at least flexible process. P1, P5&P6 reported that the use of iterative and incremental development approaches is a means of delivering readily assessable and tangible maintenance benefits to users, coupled with a means of prioritizing requests to deliver the optimum benefits. P1, P3, P5, P6&P8 also emphasized extensive customer participation and frequent feedback in the processes they employed.

It should be noted, however, that the use of such methods, incorporating features commonly associated with agile approaches, was not common to all participants. Indeed P2 and P7 reported adherence to a more traditional "waterfall" based approach of eliciting maintenance requirements from users and executing change requests.

Interestingly, while the approaches above have proven to be relatively successful, the participants did not formally recognize these as involving the use of "agile methodologies". Indeed, as P4 suggested:

"Agile methodologies, in a software maintenance environment, don't translate. Not unless it's a major, like 30 percent of the software is being changed then ok, but if it's a minor software defect change… if it's changes to features… there are stages to be done, waterfall stages. Define, design, code, test, implement."

This view is supported by the studies of Svenssen and Höst (2005) who suggested that a relative degree of adaptation and selection is required in order to successfully apply agile approaches to a software maintenance environment. Cockburn (2002) and Beck (2005) have also suggested that organizations or practitioners, interested in pursuing an agile approach, should select processes or methods which they can successfully apply within their particular domain and undertake a process of assessment and refinement, introducing new methods and refining existing methods in order to elicit the most value and benefit from agile approaches given their implementation context. Based on the responses collected from the participants in this study, the impression is that the participants are either unaware of the principles that drive agile approaches or that they are ignorant to such drivers and principles. With responses such as "a bit more rigor in following the process" (P5&P6) and "jumping into the code and fixing it (without appropriate documentation or knowledge of the software system)" (P3), it is appropriate that there is a degree of skepticism as to the notion of adapting an "agile methodology" as a basis for maintenance. This was expressed most clearly by P4:

"The word agile methodology is thrown around very much in the press. Agile was a buzzword 15 years ago and every time I read a magazine from the IEEE about every 3rd one had agile on the front cover. Each time you look at it, it means something different."

Finding 4: Superficial Understanding of Agile Approaches to Software Maintenance

One of the most interesting findings in the present research, in particular given the above observation of widespread use of methods/techniques that are commonly associated with agile methods, is that most participants, when probed, actually had, at best, superficial formal understanding of agile methodologies. Some participants who claimed an awareness of such approaches, when probed demonstrated substantial misunderstandings of some of the basic tenets of agile development. Boehm (2002) has previously observed this, expressing a view that agile approaches appear less disciplined than they really are, with people almost equating them to undisciplined hacking. Consistent with this, P3 stated:

"… how do we get good design in an agile approach? Because an agile approach, certainly in our case, tends to be jumping in and writing the code."

As an example of such misunderstandings, P5&P6, and to a limited extent P7, stated that they attempt to be flexible in the performance of their maintenance activities but in exercising this flexibility, they employ little adherence to set practices or standards. In summary, they view agile approaches as less disciplined than they really are, equating them, in a sense, to undisciplined hacking.

TAKING THE DECISION TO ADOPT AN AGILE APPROACH TO SOFTWARE MAINTENANCE

Reflecting upon the findings above, it is curious that whilst the maintainers studied saw their focus, somewhat conventionally, to be upon software enhancement and modification of implemented software systems (Finding 1) and argued that such tasks must be supported by substantial extant documentation and associated knowledge transfer mechanisms (Finding 2), they seemed to employ flexible approaches commonly associated with agile approaches which many saw as not supporting system documentation (Finding 3). Further, in many cases they displayed at best limited understanding of some of the basic tenets of agile methodologies (Finding 4).

This raises an interesting issue. To exercise an informed decision to adopt an agile approach in an organisational maintenance situation, it is reasonable to expect that the maintainer should understand agile approaches and the associated issues surrounding the operation of organisational maintenance processes. In the study however, this prerequisite knowledge does not seem to have uniformly been in place.

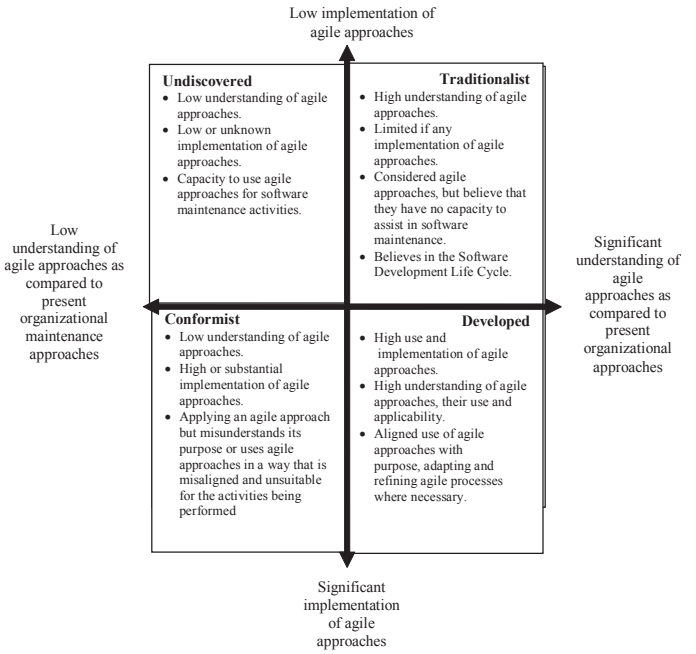
As a means of characterising the situations of the participants in the study, we introduce a matrix framework (Figure 1), to facilitate a comparison of the levels of understanding of the issues affecting an agile adoption decision and the extent to which an agile approach has been implemented.

The matrix involves two axes. The first reflects an organization's knowledge of agile approaches. An organization can possess varying degrees of knowledge of agile approaches, ranging from unaware or having a low understanding of agile approaches, to possessing substantial understanding and knowledge.

This axis also takes into consideration an organization's understanding of its present software maintenance processes, in particular how their present processes compare with agile processes.

The second axis records the level of implementation of an agile approach within the organization's software maintenance context, with organizations applying

Figure 1. The "Agile Understanding Matrix"



agile approaches or methods in varying degrees ranging from no (or low) implementation of agile approaches, to a situation where they apply a substantial implementation of agile approaches in support of performing their software maintenance activities.

As such, organizations can fall within four distinct quadrants characterised by different levels of understanding and different levels of implementation. These quadrants have been termed: undiscovered; traditionalist; conformist; and developed, for the purpose of this study, as shown in Figure 1.

Figure 2 provides a visual representation of the quadrants in which all 8 study participants fit, based on an analysis of the information collected. To illustrate the assessments made, one example for each quadrant is briefly presented.

CONCLUSION AND FUTURE RESEARCH

This research paper reports from an exploratory study that investigated whether agile approaches might have the capacity to assist software maintenance practitioners.

It was observed that the maintainers, somewhat conventionally, felt that their focus should be on software enhancement and modification of implemented software systems, which involves tasks supported by substantial extant documentation. Conversely, they seem to employ flexible agile approaches, which have been characterised as not supportive of such system documentation, with limited understanding of the basic tenets of agile methodologies.

We argue that to exercise an informed decision to adopt an agile approach in an organisational maintenance situation, the maintainer should understand the basic tenets and associated operational issues. Based on the participant situations in the study reported in the paper, a matrix framework has been introduced, to facilitate a comparison of the levels of understanding of the issues affecting an agile adoption decision and the extent to which an agile approach has been implemented.

The characterisation of organisations taking decisions concerning the adoption (or non-adoption) of agile software maintenance approaches, as developed in this paper, may well provide a framework for on-going study of software maintenance practitioner views. Further, based on the indicatory results of this study, which is limited by the number of participants, structured empirical studies could be initiated calling for participation from specific industry sectors and organisations classified by size. As we have pointed out earlier, there is a gap in substantial research, capturing the views and perceptions of front line maintenance staff as to the potential capacity of agile approaches to assist them in the performance of their day-to-day software maintenance activities. The results of empirical studies seeded from this preliminary study could be of valuable contribution to the body of knowledge in this area, benefiting both academia and practice.

REFERENCES

Agile Alliance, 2006, "What is Agile Software Development?", Online Resource Site, Last Accessed: 26 September 2006, URL: <http://www.agilealliance.org/intro>.

April, A., Huffman Hayes, J., Abran, A., & Dumke, R., 2005, "Software Maintenance Maturity Model (SMmm): The Software Maintenance Process Model", Journal of Software Maintenance and Evolution: Research and Practice, Vol. 17, No. 3, pp. 197-223.

Avison, D., & Fitzgerald, G., 2003, Information Systems Development: Methodologies, Techniques and Tools, 3rd eds, McGraw-Hill, Berkshite, UK.

Boehm, B., 2002, "Get Ready for Agile Methods, With Care", Computer, Vol. 35, No. 1, pp. 64-69.

Chapin, N., Hale, J.E., Khan, K.Md, Ramil, J.F., & Tan, W., 2001, "Types of Software Evolution and Software Maintenance", Journal of Software Maintenance and Evolution: Research and Practice, Vol. 13, No. 1, pp. 3-30.

Cockburn, A., 2002, Agile Software Development, Addison-Wesley, Boston, USA.

Dekleva, S., 1992, "Delphi Study of Software Maintenance Problems", Proceedings. International Conference on Software Maintenance, Orlando, FL, USA.

Kitchenham, B.A, Travassos, G.H., Mayrhauser, A.v., Niessink, F., Schneidewind, N.F., Singer, J., Takada, S., Vehvilainen, R., & Yang, H., 1999, "Towards an Ontology of Software Maintenance", Journal of Software Maintenance: Research and Practice, Vol. 11, No. 6, pp.365-389.

Lientz, B.P., 1983, "Issues in Software Maintenance", ACM Computer Surveys (CSUR), Vol. 15, No. 3, pp. 271-278.

Martin, J., & McClure, C., 1983, Software Maintenance: The Problem and its Solution, Prentice-Hall, New Jersey, USA.

Poole, C.J., & Huisman, J.W., 2001, "Using Extreme Programming in a Maintenance Environment", IEEE Software, Vol. 18, No. 6, pp. 42-50.

Schuh, P., 2001, "Recovery, Redemption, and Extreme Programming", IEEE Software, Vol. 18, No. 6, pp. 34-41.

Sousa, M.J.C., & Moreira, H.M., 1998, "A Survey of the Software Maintenance Process", Proceedings. International Conference on Software Maintenance.

Svenssen, H., & Höst, M., 2005, "Introducing an Agile Process in a Software Maintenance and Evolution Organisation", Proceedings. Ninth European Conference on Software Maintenance & Reengineering, Manchester, UK.

Swanson, E.B., 1976, "The Dimensions of Maintenance", Proceedings. 2nd International Conference on Software Engineering, Long Beach, CA, USA.

Yip, S.W.L., 1995, "Software Maintenance in Hong Kong", Proceedings. International Conference on Software Maintenance, Opio, France.

Figure 2. Placement of participants within the "Agile Understanding Matrix"