‘State of the Art’ in Using Agile Methods for Embedded Systems Development

Jayakanth Srinivasan, Radu Dobrin, Kristina Lundqvist
Mälardalen University
{jayakanth.srinivasan, radu.dobrin, kristina.lunqvist}@mdh.se

Abstract

Agile methods hold a significant promise to reduce cycle times and provide greater value to all key stakeholders involved in the software ecosystem. While these methods appear to be well suited for embedded systems development, their use has not become a widespread practice. In analyzing the state-of-the-art, as captured in published literature, we found that there are technical issues (requirements management, and testing), as well as organizational issues (process tailoring, knowledge sharing & transfer, culture change, and support infrastructure). In this paper, we build preliminary guidance for firms around these six areas and presented as a framework that will enable understanding the expected adoption trajectory.

1. Introduction

Embedded systems are pervasive in the world we live in today, the extent to which is illustrated by Eggermont [1], who notes that the average home today has over 50 embedded systems. Organizations involved in the lifecycle (design, development, and sustainment) of these systems, have to manage the growing operational and environmental complexity that these systems face. Increased competitive pressures are driving compressed schedules to enable the organization to meet time-to-market goals, while, at the same time, teams face daunting budgetary challenges. Surviving, in this context, is no longer sufficient, and organizations must fundamentally change the way they do systems developed to thrive and grow.

Graaf, Lormans, and Teotenel [2] studied seven European firms to determine the state of the practice in embedded software engineering. One of the key findings in the study was that systems engineering decisions are largely being driven by hardware constraints, which then impact software efforts two stages later in the lifecycle when software requirements at the component level are developed. This development partitioning into two independent streams of hardware and software development occurring in parallel, resulted in the integration becoming the critical point of failure, the nexus in the classical ‘V’ model of software development. It is important to note that, in the context of embedded systems development, there are multiple instances of software development that need to be synchronized with hardware development, at the latest at integration testing.

As Dutt and Choi [3] point out, the designer has to trade-off between *flexibility* and *efficiency* in selecting the right hardware platform. When the choice of platform is made at the system architecture level, the impacts on software are not really known, and, more importantly, may lead to significant changes in requirements when the developed software does not meet performance requirements. As Karsai et al. note [4], the development of software for embedded systems is difficult because these systems are part of a physical environment whose complex dynamics and timing requirements have to be satisfied. Furthermore, these projects often lack an effective software development methodology [5].

In discussing the nature of agile development, Highsmith [6] identifies three elements of the ecosystem that constitute agile methods: barely sufficient methodology, collaborative values, and chaordic perspective. Going further to discuss the class of problems that are best suited for agile methods, he notes that the more volatile the requirements, and the more experimental the technology, the more agile approaches improve the odds of success. Considering that most embedded systems development projects (including those with a co-design [7] flavor) have those characteristics, it begs the question of why agile methods are not the development approach of choice.

In this paper, we review the current state of the literature, and develop guidance to support the adoption of agile methods in embedded systems development. Our focus is on enabling firm-level adoption with an implicit emphasis on enterprise-level agility [8-10].

2. Effective Systems Development

Ronkainen and Abrahamsson [11] frame the development of embedded systems as 'hardware-related' software development, and identify the four problem areas that need to be addressed as being: meeting hard real-time requirements; supporting safe experimentation; generating sufficient documentation; and supporting test driven development. Analyzing the discussion topics at the workshop on agile development for embedded software [12], we see emergent themes to be in the areas of domain characteristics, tool support, verification & validation, and project management.

From the perspective of developing mobile software, the Mobile-D approach has been shown to be effective [46]. In discussing the challenges of creating a hybrid approach for print engine development, Dohmen and Summers [13] note that development effort really did not increase. While they attribute the lack of improvement mainly to learning curve effects, they found that using a visual language (in their case UML-RT) improved understandability at peer-review sessions. Fletcher et al. [14] report on their year long experience at Atomic Object in developing control software for autonomous warehouse vehicles. The two case examples of implementing speed control capability and monitoring battery charge level, demonstrate not only the challenges that they faced, but also provide lessons learned. The major pain points in the first project, besides manually generating mock objects were the lack of transparency into compiler assumptions; poor simulator capability; and lack of automation for system and regression testing. They addressed some of the pain points by automating system testing in the second project by creating a more effective test rig, and equally important, a trustable, more powerful tool chain (including support for automatically generating mocks). Manhart and Schneider's [15] use of agile methods at DaimlerChrysler was motivated by the fact that 58% of all functions were rejected atleast once, with six potential rejection paths leading to rework. Their challenges revolved around automated testing, and changing the developer mindset from develop-document-test to test-develop-document. In an orthogonal view, Schatz et al. [16] show that approaches such as model-based development (which are becoming increasingly more popular for embedded systems development) are orthogonal to the software development process used, be it agile or plan-based. Similarly, using component-based development (for example [17]) or service-oriented architecture [18], does not preclude the use of agile methods – for instance, instead of stories, the development unit can be effectively measured as components or services.

3. Analysis of the Literature

3.1. Requirements Management

Agile approaches are best suited for environments in which requirements are evolving. When the software development is outsourced (which is often the case when software is a part of a larger system such as an aircraft or an automobile), there is a tension between innovation-driven value created by the designer (on the client side), and the value added through implementation (carried out by the supplier). When the client is not well versed in software specification, novel requirements gathering approaches are needed to ensure that the voice of the customer is accurately captured. Puschnig and Kolagiri [19] highlight the use of goal oriented brainstorming and workshops as potential strategies to support agile requirements gathering. Memmel, Reiterer and Holzinger [20] experiments on visual specifications to support human-computer interaction and to provide interesting insights into using hi-fidelity prototypes as partial replacements for solely textual specifications, leading to a greater emphasis on requirements evaluation rather than just generation. Whether the organization decides to use the actual construct of stories [21] or not, it is important to note that the right abstraction mechanism makes requirements gathering and analysis easier.

3.2 Test-Driven Development [22]

Van Schooenderwoert and Morsciato [23] note that testing of embedded software is bound up with the testing of hardware that crosses professional and organizational boundaries. On reflecting on three years of experience in the project, they note that only 50 something bugs made it undetected to integration testing, and at any given point in time, the list of open bugs never exceeded 2. They attribute their success to having multiple testing strategies that ranged from unit testing to domain testing. An important enabler is being able to use approaches such as mock objects [24], and testing frameworks [14]. Tsai et al. [25] developed an approach for scenario-based testing that is compatible with formal analysis techniques such as model checking, as well as for reusing test script templates. For embedded applications that have safety-critical implications, such approaches [25, 26] can provide the necessary additional assurance needed. The challenge for any organization that is attempting to use agile methods is finding the right tools and techniques to support test-driven development without compromising on the efficiencies already gained with homegrown tool suites. Karlesky, Bereza and

Erickson's [27] guidance on TDD provides a first step towards creating more generalizable guidance.

3.3. Process Tailoring

Boehm [28] lays out the planning spectrum ranging from 'undisciplined hacking', on one end, to the heavyweight 'inch-pebble ironbound contract' on the other end, with milestone risk-driven approaches such as spiral development [29] in the middle. In analyzing the risk exposure as a function of the probability of loss and the size of the loss, he notes that the sweet spot is a function of the 'homeground' of the development approach. From the perspective of adopting agile, this essentially translates into the organization assessing the needs of the project, and matching the project needs with the organizational culture to find the best tailoring. This notion of requisite tailoring is implicitly discussed in Vanhanen, Jartti, and Kähkönen's [30] study of agile adoption in a large telecom company. Abrahamsson et al. [31] analyze the family of agile methods note that while universal solutions dominate the literature, more work is needed to support situation appropriate adoption.

3.4. Knowledge Sharing and Transfer

Agile development is about people, as Cockburn and Highsmith note [32]. The approach is based on the idea that the development team is more effective when responding to change if it can reduce the cost of moving information between people, and if the time elapsed between making a decision and seeing the consequences of the decision is minimized. The use of collocation, brainstorming facilities are present in embedded systems development, through the notion of integrated product teams. This construct of creating organizational environments is referred to by Nonaka, Toyama and Konna [33] as 'Ba' in the SECI-Ba framework. However, the shared team mentality (since work is partitioned into hardware and software and often executed independently – this also has the negative impact of the teams not know what their customers want) is missing, as is the construct of working incrementally. Chau, Maurer and Melnik [34] note that agile development relies heavily on socialization through communication and collaboration to access and share tacit knowledge within the team. Kettunen [35] maps the embedded software project knowledge contexts for both inputs and outputs, to either share knowledge using planning templates, or through the use of a sharing chart. While creating communities of practice, such as those in Nokia [36], are a promising approach, we recommend that organizations build on their current collaboration (for instance MACE [37]) and knowledge management strategies.

3.5. Developing Supporting Infrastructure

As we noted in earlier work [38], transitioning to agile methods is not easy. It is critical to train both the people involved in the process, as well as the senior leadership that manages the process. The concepts and practices of agile methods are simple to grasp but very difficult to institutionalize without the proper support structure. As Lindvall et al. [39] noted, the greatest challenges associated with agile adoption are integrating them with the existing environment. In addition to supporting tailoring, the supporting infrastructure has to support cross-team communications, harmonize existing organizational monuments such as a change control board with practices such as refactoring, and put in an effective governance system. The governance structure doesn't necessarily mean 'burning your Gantt charts' [40] , but rather about finding a mechanism that best supports both the project team and the project manager [41].

3.6 Managing Culture Change

One of the critical elements to successful iterative development is the active involvement of the customer. As Hirsch [42] pointed out, a commitment issue will lead to a project failure. From the stakeholder perspective, it is essential to note that the software team is both a customer of (since they provide feature requests etc to), and a supplier to (providing working software) the hardware team. When organizations work in a distributed fashion, the problem of managing commitments and risks gets exacerbated [43].

4. Recommendations

There are two broad areas that need to be addressed to effectively adopt agile methods in the context of embedded systems development: technical issues (requirements, and testing); and organizational issues (process tailoring, knowledge sharing & transfer, culture change, and support infrastructure development). Our recommendations emerge from our understanding of the coupling across the factors, as shown in Figure 1.

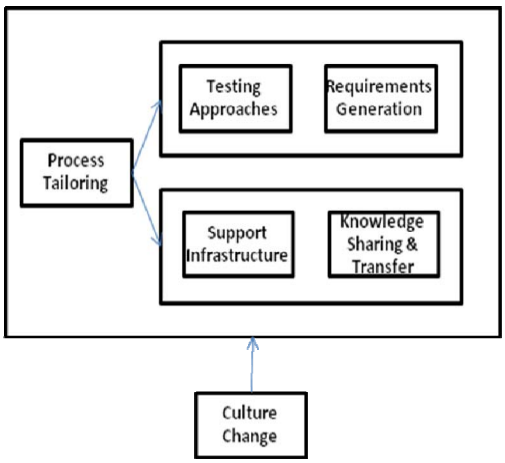

Figure 1. Framework for Understanding Challenges of Embedded System Adoption

Recommendation 1: Determine both the organizational and project 'homegrounds' and tailor agile practices to fit within the larger organizational context

One of the challenges that firms developing embedded systems face in adopting agile methods, is that the existing codified knowledge (books, papers etc) does not explicitly address domain specific characteristics. Firms should tailor the process such that it specifically addresses industry and product specific characteristics. The goal of tailoring is not to just hire an external 'methods expert' (also referred to as a consultant) who defines the process for the organization - rather the process should be developed and specified by members of the organization, with the method expert serving in the role of a mentor.

Recommendation 2: Test-Driven-Development is sound engineering practice, but requires culture change

Given that software requirements for embedded systems are an outcome of the system architecture and hardware architecture definition processes, adopting a test-driven development strategy not only increases the overall quality of the development software, it also helps to elicit design decisions that were made as part of the hardware development process. As we noted earlier, test-driven development requires a fundamentally different workflow in the software development process since the model of develop-document-test now becomes test-develop-document. This requires documenting assumptions about the hardware (and other aspects of the software system that have not yet been explicated), providing better analyzability and evolvability.

Recommendation 3: Requirements development and management needs to be harmonized to support modifiability, maintainability and dependability

The use of approaches such as stories and metaphors for requirements elicitation and analysis provide mechanisms for capturing the 'voice of customer' by enabling them to articulate their needs within frames of reference with which they are comfortable. One of the limitations of using approaches such as stories is the increased challenges in modifiability and maintainability. When it comes to embedded systems modifiability is key as hardware if often changed (from a sustainability standpoint, hardware technology is easier to replace rather than to maintain), the systems true capabilities are derived from software. Since software evolves to deliver increasing capabilities, modifiability and maintainability are necessary but not sufficient characteristics. The third attribute of dependability has to be met by augmenting approaches such as stories with more detailed refinement in the form of use cases and/or specifications that are more formal.

Recommendation 4: Agile methods needs organizational support infrastructure ranging from new tools and methods, to refined communication/ collaboration approaches

Agile methods requires the development of the right organizational infrastructure, including the right tools to support the software lifecycle, as well as in creating an environment that supports communication and collaboration. The emergence of tools like instant messaging, wiki's and social networking tools require organizations to rethink the traditional documentation-based approaches to building software systems.

Recommendation 5: Agile methods require both mechanisms and incentives to support knowledge transfer and sharing

Knowledge transfer and sharing are foundational to successful agile adoption. It is very difficult for agile teams to operate in a 'closed' environment that does not incentivize open knowledge sharing and collective ownership of the developed software.

Recommendation 6: Transitioning to any new process requires the management of culture change, but it is even more critical in the case of agile methods, since it emphasizes the 'soft factors' governing organizational work

Implicit in the first five recommendations is the fact that agile methods represent a radically different way of doing the work associated with systems development. This difference in development processes has to be augmented with a robust change management strategy that will enable the organization to transition smoothly without undergoing either a severe loss in productivity or overall product quality.

5. Conclusions

Pereira and Carro [44] make an interesting point about the immediate challenge for any designer of a distributed real-time embedded system, being that of taking advantage of the latest technology – which in effect leads to changing software requirements. Kilpinen, Eckert and Clarkson's [45] studies have highlighted that emergent changes between systems and embedded software can unexpectedly increase the cost and duration of the design process. Agile methods provide a means of mitigating against the impact of emergent changes. This paper is a first cut at providing actionable guidance for the use of agile methods in embedded software engineering. The startling gap in the literature is the absence of published cases on failures with respect to agile adoption. Hence, we derived this guidance through an analysis of the state-of-the-art as found in the published literature. Traditional methods for embedded systems development work effectively when the requirements are stable, and the risks understood well enough to be managed effectively. When it comes to mission-critical systems in particular, the standards highlight the importance of creating the necessary artifacts to enhance trust in the implemented system, but do not necessarily specify the process to be followed to generate the necessary artifacts.

In our interviews with three aerospace and defense leaders, we found that they had in fact tried to use agile methods, but the results were not what they expected. The common reasons across the interviews were the lack of understanding of the change management required for introducing a new process, and being unable to address the myths of using agile methods. For instance, when the tenet of 'working software over comprehensive documentation' is interpreted as meaning no documentation, the developed software is neither maintainable nor reusable. The next phase in our research is to carry out case studies on these firms, to determine why the use of agile methods did not meet their needs. On addition to contributing to the growing body of knowledge on agile adoption, the findings will further validate the recommendations.

6. References

[1] L. D. J. Eggermont, "Embedded systems roadmap 2002," *STW Technology Foundation, Utrecht (The Netherlands)*, pp. 90-73461, 2002.

[2] B. Graaf, M. Lormans, and H. Toetenel, "Embedded software engineering: The state of the practice," *IEEE software,* vol. 20, no. 6, pp. 61-69, 2003.

[3] N. Dutt, and K. Choi, "Configurable processors for embedded computing," *Computer,* vol. 36, no. 1, pp. 120-123, 2003.

[4] G. Karsai, J. Sztipanovits, A. Ledeczi *et al.*, "Model-integrated development of embedded software," *Proceedings of the IEEE,* vol. 91, no. 1, pp. 145-164, 2003.

[5] B. Greene, "Agile methods applied to embedded firmware development," in Agile Development Conference (ADC'04), 2004, pp. 71-77.

[6] J. Highsmith, *Agile software development ecosystems*, 2002.

[7] G. De Michell, and R. K. Gupta, "Hardware/software co-design," *Proceedings of the IEEE,* vol. 85, no. 3, pp. 349-365, 1997.

[8] J. Srinivasan, C. Norstrom, and K. Lundqvist, "Exploring the Sources of Agility in Software Organizations."

[9] P. Kettunen, "Adopting key lessons from agile manufacturing to agile software product development—A comparative study," *Technovation*, 2008.

[10] P. Kettunen, and M. Laanti, "Combining agile software projects and large-scale organizational agility," *Software Process: Improvement and Practice,* vol. 13, no. 2, 2008.

[11] J. Ronkainen, and P. Abrahamsson, "Software Development Under Stringent Hardware Constraints: Do Agile Methods Have a Chance?," *Lecture notes in computer science*, pp. 73-79, 2003.

[12] J. Grenning, J. Peeters, and C. Behring, "Agile development for embedded software," *Lecture notes in computer science*, pp. 194-195, 2004.

[13] L. A. J. Dohmen, and L. J. Somers, "Experiences and lessons learned using UML-RT to develop embedded printer software," *Lecture notes in computer science*, pp. 475-484, 2002.

[14] M. Fletcher, W. Bereza, M. Karlesky *et al.*, "Evolving into Embedded Develop," *AGILE 2007*, pp. 150-155, 2007.

[15] P. Manhart, and K. Schneider, "Breaking the ice for agile development of embedded software: An industry experience report." pp. 378-386.

[16] B. Schatz, A. Pretschner, F. Huber *et al.*, "Model-based development of embedded systems," *Lecture notes in computer science*, pp. 298-312, 2002.

[17] Y. Berbers, P. Rigole, Y. Vandewoude *et al.*, "Components and contracts in software development for embedded systems." pp. 219–226.

[18] W. T. Tsai, X. Wei, R. Paul *et al.*, "Service-oriented system engineering (SOSE) and its applications to embedded system development," *Service Oriented Computing and Applications,* vol. 1, no. 1, pp. 3-17, 2007.

[19] A. Puschnig, and R. T. Kolagari, "Requirements engineering in the development of innovative automotive embedded software systems." pp. 328-333.

[20] T. Memmel, H. Reiterer, and A. Holzinger, "Agile methods and visual specification in software development: a chance to ensure universal access," *Lecture notes in computer science,* vol. 4554, pp. 453, 2007.

[21] R. Mugridge, "Managing Agile Project Requirements with Storytest-Driven Development," *IEEE software*, pp. 68-75, 2008.

[22] D. Janzen, H. Saiedian, and L. L. C. Simex, "Test-driven development concepts, taxonomy, and future direction," *Computer,* vol. 38, no. 9, pp. 43-50, 2005.

[23] N. Van Schooenderwoert, R. Morsicato, A. Rules *et al.*, "Taming the embedded tiger-agile test techniques for embedded software." pp. 120-126.

[24] M. Karlesky, G. Williams, W. Bereza *et al.*, "Mocking the Embedded World: Test-Driven Development, Continuous Integration, and Design Patterns."

[25] W. T. Tsai, R. Paul, L. Yu *et al.*, "Rapid pattern-oriented scenario-based testing for embedded systems," *Software Evolution with UML and XML*, pp. 222-262.

[26] W. T. Tsai, L. Yu, F. Zhu *et al.*, "Rapid embedded system testing using verification patterns," *IEEE software,* vol. 6, pp. 9, 2005.

[27] M. Karlesky, W. Bereza, and C. Erickson, "Effective test driven development for embedded software."

[28] B. Boehm, "Get ready for agile methods, with care," *Computer,* vol. 35, no. 1, pp. 64-69, 2002.

[29] B. W. Boehm, "A spiral model of software development and enhancement," *Computer,* vol. 21, no. 5, pp. 61-72, 1988.

[30] J. Vanhanen, J. Jartti, and T. Kahkonen, "Practical experiences of agility in the telecom industry," *Lecture notes in computer science*, pp. 279-287, 2003.

[31] P. Abrahamsson, J. Warsta, M. T. Siponen *et al.*, "New directions on agile methods: a comparative analysis," in International Conference in Software Engineering (ICSE03), Portland, Oregon, 2003, pp. 244-254.

[32] A. Cockburn, and J. Highsmith, "Agile software development, the people factor," *Computer,* vol. 34, no. 11, pp. 131-133, 2001.

[33] I. Nonaka, R. Toyama, and N. Konno, "SECI, Ba and leadership: a unified model of dynamic knowledge creation," *Long range planning,* vol. 33, no. 1, pp. 5-34, 2000.

[34] T. Chau, F. Maurer, and G. Melnik, "Knowledge sharing: agile methods vs. Tayloristic methods," *Twelfth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, 2003. WET ICE 2003. Proceedings*, pp. 302-307, 2003.

[35] P. Kettunen, "Managing embedded software project team knowledge," *IEE Proceedings-Software,* vol. 150, no. 6, pp. 359-366, 2003.

[36] T. Kahkonen, "Agile methods for large organizations-building communities of practice." pp. 2-10.

[37] T. Chau, and F. Maurer, "Knowledge sharing in agile software teams," *Lecture notes in computer science*, pp. 173-183, 2004.

[38] J. Srinivasan, and K. Lundqvist, "Organizational Enablers for Agile Adoption: Learning from GameDevCo."

[39] M. Lindvall, D. Muthig, A. Dagnino *et al.*, "Agile software development in large organizations," *Computer,* vol. 37, no. 12, pp. 26-34, 2004.

[40] M. Karlesky, and M. Vander Voord, "Agile Project Management (or, Burning Your Gantt Charts)," in Embedded Systes Conference, Boston, 2008.

[41] S. Augustine, B. Payne, F. Sencindiver *et al.*, "Agile project management: steering from the edges," 2005.

[42] M. Hirsch, "Making RUP agile," in OOPSLA Practitioner Report, 2002.

[43] J. Kontio, M. Hoglund, J. Ryden *et al.*, "Managing commitments and risks: Challenges in distributed agile development." pp. 732-733.

[44] C. E. Pereira, and L. Carro, "Distributed real-time embedded systems: Recent advances, future trends and their impact on manufacturing plant control," *Annual Reviews in Control,* vol. 31, no. 1, pp. 81-92, 2007.

[45] M. S. Kilpinen, C. M. Eckert, and P. J. Clarkson, "The Emergence of Change at the Interface of System and Embedded Software Design," in Conference on Systems Engineering Research, Hoboeken, NJ, 2007.

[46] P. Abrahamsson, A. Hanhineva, H. Hulkko *et al.*, "Mobile-D: an agile approach for mobile application development." pp. 174-175.