Association for Information Systems
AIS Electronic Library (AISeL)

SAIS 2008 Proceedings | Southern (SAIS)

3-1-2008

Issues with Incorporating Regulatory Compliance into Agile Development: A Critical Analysis

Sushma Mishra
Virginia Commonwealth University, mishras@vcu.edu

H. Roland Weistroffer
Virginia Commonwealth University

ISSUES WITH INCORPORATING REGULATORY COMPLIANCE INTO AGILE DEVELOPMENT: A CRITICAL ANALYSIS

Sushma Mishra
Virginia Commonwealth University,
Richmond, VA
mishras@vcu.edu

Heinz Roland Weistroffer
Virginia Commonwealth University,
Richmond, VA
hrweistr@vcu.edu

ABSTRACT
Agile development methodology is widely used for software development in organizations. Incorporating regulatory compliance aspects in development process is important. This paper discusses various issues in considering compliance aspects into development process. An analysis of different aspects of compliance related issues is presented.

Keywords Agile, compliance, regulations, audit

INTRODUCTION

Agile development is an approach to software engineering that explicitly champions an active role for the customer. A number of software development methods such as extreme programming (XP), feature-driven development, crystal clear method, scrum, dynamic systems development, and adaptive software development, fall into this category of agile development methodologies. Indeed, agile philosophy especially XP method, strives to include a real user(s) in the team who is located 'on-site' with software developers (Sharp el. al., 2006). Unlike the traditional methodologies, agile methodologies deal with unpredictability by relying on people and their creativity rather than on processes (Nerur et. al, 2005). Agile approach is characterized by short iterative cycles of development driven by product features, periods of reflection and introspection, collaborative decision making, incorporation of rapid feedback and change, and continuous integration of code changes into the system under development. A project is broken down into sub-projects, each of which typically involves planning, development, integration, testing, and delivery. Developers work in small teams with customers (representing system users) as active team members.

The features to be implemented in each development cycle are jointly decided by the customer and the rest of the development team. Collaborative decision making involving stakeholders with diverse backgrounds and goals is thus a characteristic of agile development. The management style followed by agile methodologies favors a leadership-and-collaboration collaboration style where the project manager's role is that of a facilitator or coordinator. The deliverable of each development cycle is working code that can be used by the customer. Agile methods discourage documentation beyond the code. Product knowledge, therefore, becomes tacit. Rotation of team membership ensures this knowledge is not monopolized by a few individuals. Further, the iterative strategies that characterize agile methodologies are best supported by OO technologies. The evolutionary- delivery model used by agile developers provides a meaningful framework to guide agile software development (Nerur et. al., 2005). Agile methodology encourages extensive collaboration and communication that provides the basis for collective action. Various stakeholders including developers and end users go through repeated cycles of thought-action-reflection that foster an environment of learning and adaptation. Team members, empowered with more discretionary and decision-making powers, are not confined to a specialized role. This increases the diversity/variety of the teams and enables them to self-organize and respond with agility to emergent situations.

Followers of agile methodology claim that the above mentioned benefits in software development are unique to agile. It is debatable if agile benefits outweigh various difficulties that this process introduces in

a project team and the organization. From organizational perspective, there are several issues associated with agile development methodology but clear solutions to deal with such issues. Project management in agile way is different from traditional ways of developing software. Issues, such as compliance with regulations or rules, software quality control, standardization with standards such as ISO or CMM, security and change management, paralyze the widespread growth and adoption of the methodology industry wide. Still being in a nascent stage of development, agile methodology, it's manifesto and users do not provide any clear cut answers to such concerns. This paper attempts to highlight some of these issues and point a plausible direction to solutions. The rest of this paper is arranged as follows. The next section presents a description of agile, unified process and waterfall methodologies of software development. A brief description is if followed by a comparison of three methodologies. Section three presents a discussion on some of the relevant issues pertaining to agile development. Finally, in the concluding section, some future research directions are suggested for agile development methodology.

AGILE DEVELOPMENT METHODOLOGY

The "Agile Manifesto" provides a good overview of the intent of Agile Methods. The following values express the tenor of the principles employed: individuals and interactions over process and tools, working code over comprehensive documentation, customer collaboration over contract negotiation and responding to change over following a plan. Agile has different methods such as scrum, extreme programming, there are various differences and nuances of each methodology, but their commonalities are more significant than the differences. There are six common features to the various Agile Methods (Coram and Bohner, 2005): collaboration, code reviews, small teams, short release schedules, time-boxing, and constant testing. All Agile Methods are highly collaborative, both inside of the development group and outside of it. Agile Methods rely on informal communication rather than voluminous documentation to rapidly spread information throughout the team and to other stakeholders. Without a highly collaborative environment, any Agile Method is doomed to fail. This means that a primary responsibility of the project manager is to ensure a highly collaborative environment. Agile Methods also encourage, if not require, code reviews. Code reviews allow for dissemination of key information. For example, in XP, code reviews are continuous through pair programming where two developers share a single computer**.**

Agile Methods also encourage small teams and small numbers of teams per projects. Small teams are required to foster collaboration, are more likely to require less process and planning to coordinate team members' activities. Agile Method Release schedules can be as short as two weeks or as long as six months. At the end of each release, a functional product is released to the customer that allows for the product to be evaluated and for changes to be made in the priorities of features to be added to subsequent releases. Time boxing technique the release length is fixed but the features are not. This is the reverse of the "traditional" development mentality where the features are fixed and the delivery date flexible. It helps in fixing the focus on customer. To offset the potential for short releases to significantly degrade product quality, agile methods put a high degree of emphasis on testing the product throughout its lifecycle. The notion of test-first offsets the risks of a mindset of just writing the code. Agile Methods require integration testing throughout the development process. This testing must be automated with daily builds and regression tests to ensure all functionality works adequately. Since Agile teams involve experienced staff with sizeable responsibility, a mentor or coach leadership approach is most effective. Team leads must be willing to enable members to take initiative.

Section 3: Issues with agile methodology-an assessment

This section identifies some of the issues with agile development methodology. An assessment of the literature reveals that there is an awareness of such issues in practioner as well as research world. Though such awareness is necessary for work in this area to progress, it is not sufficient to identify useful solutions. More research in this area is required. To adopt agile methodology successfully, there are certain issues that need to be considered seriously. These are:

Governance issues: Organizations that have more command based structure would have issues in adopting a collaborative kind of development technique such as agile. Organizational culture has a significant impact on the social structure of organizations, which in turn influences the behavior and actions of people The values, norms, and assumptions of an organization are stabilized and reinforced over time,

and are reflected in the policies embodied in organizational routines. Culture exerts considerable influence on decision-making processes, problem-solving strategies, innovative practices, information filtering, social negotiations, relationships, and planning and control mechanisms (Nerur et. al., 2005). Neither culture nor mind-sets of people can be easily changed, which makes the move to agile methodologies all the more formidable for many organizations. The organizational form that facilitates this shift needs the right blend of autonomy and cooperation to achieve the advantages of synergy while providing flexibility and responsiveness. Elaborate methodologies, tools, and practices have evolved to manage an out-of-control world. But tools fail when neat linear tasks don't easily accommodate dynamic processes and when neat schedules require frequent updating to reflect changing circumstances. Agile requires adaptive management style (Augustine et. al, 2005).

Knowledge Management issues: Knowledge management is of vital importance to organizations. Traditional development approaches create much documentation. Such records serve as useful artifacts for communication and traceability of design. Agile methodologies, on the other hand, encourage lean thinking and cutting down on overhead, particularly documentation. Much of the knowledge in agile development is tacit and resides in the heads of the development team members. This can make the organization heavily dependent on the development teams and can potentially shift the balance of power from the management to the development teams (Nerur et. al, 2005). Such a situation may not be acceptable to many organizations. A possible solution to such a problem could be resolved by determining which knowledge should be codified and what may remain tacit. Agile development relies on teamwork, as opposed to individual role assignment that characterizes traditional. The central aspect of successful and mutually beneficial coexistence of agile project learning and organizational learning is the constant collaboration between the project teams and organizational level. The empirical evidence reveals that without *support* from organizational level a majority of the improvements agreed within project teams cannot be implemented (Salo and Abhrammson, 2003).

Project manager's role: Efficiency requires the existence of project management activities to enable the proper execution of software development tasks. Project management is thus a support function that provides the backbone for efficient software development. Therefore, a project management perspective can be seen as a relevant dimension in the evaluation of agile software development methods (Abrahamsson et. al., 2003). Agile philosophy demands that the project manager's traditional role of planner and controller must be altered to that of a facilitator who directs and coordinates the collaborative efforts of those involved in development, thus ensuring that the creative ideas of all participants are reflected in the final decision. Incorporating such a change is an already command based environment can be real challenging (Nerur et. al, 2005). In agile management, the "manager as leader" must perform the following: establish a sense of urgency, form a powerful guiding coalition, create a vision, communicate the vision, empower others to act on the vision, plan for and create short-term wins, consolidate improvements and produce still more change, and institutionalize the new approaches (Anderson et. al., 2003). Software development teams need leaders who set direction, align people, obtain resources, coordinate efforts, run interference, and provide a motivating environment. New team members need to be recruited and trained. Standards must be set and various areas of expertise require nurturing. In complex systems, expert designers are important; if they are not appointed, they will emerge (Anderson et. al, 2003).

Change management issues: A cooperative social process characterized by communication and collaboration between a community of members who value and trust each other is critical for the success of agile methodologies (Nerur et. al, 2005). For programmers accustomed to solitary activities or working with relatively homogeneous groups of analysts and designers, the ideas of shared learning, reflection workshops, pair programming, and collaborative decision making is a challenge. Agile requires small and organic teams (Augustine et. al., 2005) where self organization and emergent order are due in part to complex interactions or flows among agents. The richness of the interaction among team members depends largely on their openness to the exchange of information. Developing such an open an interactive environment in an organization accustomed to work differently could be a challenge.

Process specific issues: Traditional processes are compliance-driven and activities-and measurement-based, aimed at providing assurance. Agile methodologies rely on speculation, or planning with the understanding that everything is uncertain, to guide the rapid development of flexible and adaptive systems

of high value. They stress the importance of assessing as opposed to measuring, and are highly tolerant of change. One of the biggest barriers to migration is the change in a process model from a life cycle model to one that supports feature based development using evolutionary and iterative development (Nerur et. al, 2005). Such a change entails major alterations to work procedures, tools and techniques, communication channels, problem-solving strategies, and roles of people. Agile practioners believe that software development is a dynamic and open system. An open system interacts with its environment, receiving inputs and providing outputs, but does not control it. A dynamic system changes and evolves its behavior in response to its inputs (Augustine et. al, 2005). Agile Development processes cannot rely on these traditional techniques of business analysis as these techniques do not take into account how systems change over time. Agile users should search for other more compatible techniques like systems dynamics (Rand and Eckfeldt, 2004).

Technological issues: Tools play a critical role in successful implementation of a software development methodology. Organizations planning to adopt agile methodologies must invest in tools that support and facilitate rapid iterative development, versioning or configuration management, JUnits, refactoring, and other agile techniques. Of course, tools alone cannot make software development successful. People must be trained to use them correctly.

***Compliance issues*:** Compliance with regulations such as Sarbanes Oxley Act (SOX) requires accurate financial reporting of the development costs of the project and extensive documentation of the process, both of which seem difficult to achieve in agile. Cost of software development is hard to estimate since infrequently records of internal software development costs are kept and there are few defined and accepted formulae available. Even if the records are there, costs are rarely integrated into identifiable asset categories in accounting systems. This makes compliance issues, such as internal control assessment in Sarbanes-Oxley Act, more difficult (Armour, 2005). Another approach to asset valuation determines value based on how much we could get if we sold the asset. This can also present a problem since, for example, systems are specific to an organization and there might not be a buyer for it.

There are problems even if we consider software development projects as investments. Software projects consume resources and generate returns. They may represent a very significant percentage of the allocated assets of a company. The concept of fiduciary responsibility that governs the management of corporate assets, such as bonds, does not seem to extend to software development projects. It is easier to computer returns on bonds than of these projects. It may be that, in conforming to SOX, companies will find they have to develop answers to these questions and construct systems that assess, track, and report the risk on their software projects (Armour, 2005). Intrinsically high-risk projects would have to be justified by equivalent high return. Companies could set up hedge funds against changing requirements and spread the legitimate risks that could be incurred when software systems are created.

Assurance activities require independent third party assessment. Even compliance with regulations such as Sarbanes Oxley, requires that, independent, outside auditors, attest the compliance plan (Beznosov and Kruchten, 2004). Independent auditors are expected to inspect, analyze, validate, test, and then certify a more or less finished product. This also helps in keeping developers and testers at a distance i.e. segregation of duty, as required by SOX. In keeping with the agile philosophy if efforts were failing, the team should refactor the approach to managing the project in an effort to minimize the costs of compliance without adopting more risk to ensure success (Cunningham, 2005).

Security issues: Security assurance provides confidence in the security-related properties and functionalities, as well as the operation and ad-ministration procedures, of a developed solution. Conventional assurance methods can be roughly grouped into the use of best practices in a form of unofficial guidelines; design and architectural principles; use of appropriate tools and technologies; dynamic testing and static analysis; and, most importantly, internal and third-party review, evaluation, and vulnerability testing. Lack of documentation, which is a forte of agile methodology, presents a difficult situation for assurance purposes. Some of the ways of achieving this would be (Beznosov and Kruchten, 2004): Include a security engineer in the development team for assessing security risks, proposing security-related user stories, and for performing "real-time" security reviews of the sys-tem design and code through pair programming.

Some agile practices fit well with security assurance. For example, pair programming advocated by XP naturally facilitates internal design and code review, and motivates developers to follow coding standards**,** including standards for writing "secure code." Additionally, developers receive immediate feedback from

their peers, which could very well be on the principles and guidelines of secure design. The practice of pair programming could be further enhanced by involving a security engineer who can use this opportunity for reviewing the design and the code. A core activity of agile security assurance should be the early identification of the types (Beznosov and Kruchten, 2004) of design and code changes that are likely to cause security problems, and to use these as guidelines when iterating through-out the lifecycle.

Standardization issues: The CMMI is a model for process improvement which is intended to take an organization to a level where continuous improvement in productivity and quality is possible (Chrissis, 2003). The CMMI is both a capability model and a maturity model. The model can be implemented in a gradual and staged fashion. This is the idea that an organization achieves maturity level 2 across all relevant practices and then level 3 and so on. It can also be implemented in a continuous fashion where capability in individual practices is measured. An organization can achieve a high capability level in a particular practice whilst holding only a level 2 capability in other practices.

A key to achieving more agility with the CMMI is to realize that the practices are primarily advisory or indicative only. To meet a CMMI appraisal, an organization must demonstrate that the goals of a process area are being achieved (Anderson, 2005). This is done by identifying evidence of practices. However, the practices need not be the ones described in the CMMI specification. The organization is free to propose alternatives practices and appropriate evidence. It is acceptable to document the approach as conformance to process and the measurement mechanisms to be variation aware and defined within common cause variation limits (Anderson, 2005). This is the key which unlocks the possibility of a truly agile CMMI implementation across the entire lifecycle.

The CMMI, based on decades of software engineering experience, already includes many of these aspects. Creating appropriate environments, commissioning appropriate tools and initiating projects with sufficient vision, communicating that vision with launch events, and planning for the realization of value through deployment, are all part of the CMMI prescription. There is nothing particularly at odds with the agile philosophy in these things. The main difference is that they are spelled out explicitly. By doing so the process embraces the agile manifesto (Cunningham 2001) ideas such as valuing customer collaboration over contract negotiation and responding to change over following a plan. In addition, by embracing variation and accepting it as part of the process, the people aspect of individuals and interactions over processes and tools is embraced (Anderson, 2005). Through use of iterative cycles agreed at a cadence acceptable to the customer, the first half, working software over comprehensive documentation is embraced. Organizations need to understand that extensive documentation increases resource utilization. This translates in turn into higher cost and perhaps slower delivery of the system. Where possible, the development team should use tools that automate the generation of documentation to reduce the resource utilization (Theunissen et. al, 2003).

Section 4: Conclusion

Agile methods have gained tremendous acceptance in the commercial arena since late 90s because they accommodate volatile requirements, focus on collaborate on between developers and customers, and support early product delivery (Huo et. al, 2004). The fast growth and acceptance of agile methods shows how effective this methodology can be to organizations in software development business. But agile has issues that need to be considered before it is accepted by industry as method of choice.

Software is developed to meet business needs and provide advantage to organizations in handling its business processes. This objective should be incorporated in methodologies such that these development techniques provide more flexibility to businesses. So what happens when the software team can handle the change, but the business it is supporting can't? To handle this situation, extend the philosophies of agile development and merge them with systems thinking and holistic design aspects from other operations and management methodologies in order to create a more holistic solution**.** However, system dynamics offers a powerful way to make predictions about what parts of a system need improvement. The combinations of these methods for systems analysis and Agile Development offer an incredible opportunity for business improvement (Rand and Eckfeldt, 2004). Agile can present a cultural shift for some as they must be willing to share decision making authority (Coram and Bohner, 2005). Project managers are also more involved

with the customer collaboration, instead of usual focusing on defining deliverables and contracts. If the project manager considering an agile method is not capable or does not want such a role, selecting an agile method may not be appropriate. In this regulatory era of legislations such as Sarbanes Oxley Act, demand for conformance to process standards corresponds often has a correlation with the maturity or scale of the business. Large, mature businesses have established protocols and strict guidelines and demand compliance. This may be due to organizational, managerial or regulatory needs. (Theunissen et. al, 2003). Agile or any other methodology needs to address this business requirement for long term survivability.

Currently, the existing agile project learning techniques seem to lack means to perceive the organizational software process improvement (SPI) aspect. For example, they do not address the important aspects of systematically defining, validating, packaging and storing the SPI results of agile projects (Salo and Abrahamson, 2005). Also, conformance to industry standards such as CMM is required for software building oragnziations. The solution is to provide a loose project plan which approximates the scope of a whole project and lays out a plan for a series of iterations that define approximately what will be developed in each one (Anderson, 2005). The secret is to avoid big planning up front. Also, we have seen that agile methods do have practices that have QA abilities, some of them are inside the development phase and some others can be separated out as supporting practices. Typically, we think of refactoring as the reorganization and refinement of our designs and code. However, the concept also applies to our organizations, our techniques, and our behaviors. Agile methodology has its promises and shortcomings but like any other tool, it should be upon the discretion of organizations to use it effectively to meet their requirements.

References:

1) Abrahamsson, P., Warsta J, Siponen, M. and Jussi Ronkainen. New Directions on Agile Methods: A Comparative Analysis. IEEE, 2003, 244-254
2) Anderson, L., Alleman, G., Blotner, J., Poppendieck, M. and Wirfs, R. Agile Management – An Oxymoron? Who needs managers anyway? OOPSLA'03, October 26–30, 2003, Anaheim, California, USA.
3) Armour, P. Sarbanes-Oxley and Software Projects. *Communications of the ACM. (*48:6), June 2005, 15-17
4) Augustine, S., Payne, B., Sencindiver, F. and Woodcock, S. Agile Project Management: Steering from the edges. *Communications of the ACM. (*48:12), December 2005, 85-89
5) Beznosov, K. and Kruchten, P. Towards Agile Security Assurance. NSPW 2004 Nova Scotia Canada, 47-54
6) Boehm, B., Port, D. (1999). Escaping the Software Tar Pit: Model Clashes and How to Avoid Them. Software Engineering Notes (24:1), 36-48.
7) Booch G., J. Rumbaugh, and I. Jacobson (1999) *The Unified Modeling Language User Guide*, Addison-Wesley Longman.
8) Clear, T. (2003). The Waterfall is Dead: Long Live the Waterfall!!. Computer Society. Inroads (Featured Column). Vol. 35, pp. 13-14.
9) Cunningham, J. (2005). Costs of Compliance: Agile in an Inelastic Organization. *Proceedings of the Agile Development Conference (ADC'05)*.
10) Nerur, S., Mahapatra, R. *and* Mangalaraj, G. Challenges of Migrating to Agile Methodologies. *Communications of the ACM. (*48:5), May 2005, 72-78
11) Noble, J., Marshall, S., Marshall, S. and Biddle, R. Less Extreme Programming. *Sixth Australasian Computing Education Conference* (ACE2004), Dunedin, New Zealand. Conferences in Research and Practice in Information Technology, Vol. 30., 217-226
12) Racoon, L. (1997). Fifty Years of Progress on Software Engineering. Software Engineering Notes (22:1), 88-103.
13) Theunissen, W., Kourie, D. and Watson, B. Standards and Agile Software Development. *Proceedings of SAICSIT* 2003, Pages 178–188
14) Lowell, L., and R. Jeffries. Extreme Programming and Agile Software Development Methodologies. *Information Systems and Management* **21**(3), 2004, 41- 52.
15) Kruchten, P. *The Rational Unified Process: An Introduction*, Addison Wesley Longman, Inc, Massachusetts, 2000.
16) Sharp, H., Biddle, R., Gray, P., Miller, L. and Patton, J. (2006). Agile Development: Opportunity or Fad? CHI 2006.
17) Anderson, D. (2005). Stretching Agile to fit CMMI Level 3 - the story of creating MSF for CMMI® Process Improvement at Microsoft Corporation. *Proceedings of the Agile Development Conference (ADC'05)*
18) Coram, M. and Bohner, S. (2005). The Impact of Agile Methods on Software Project Management. *Proceedings of the 12th IEEE International Conference and Workshops on the Engineering of Computer-Based Systems (ECBS'05.*