2010

Agility versus Maturity: Is There Really a Trade-Off?

Vishnu Vinekar
Fairfield University, vvinekar@fairfield.edu

Christopher Huntley
Fairfield University, chuntley@fairfield.edu

Agility vs. Maturity: Is there really a tradeoff?

Dr. Vishnu Vinekar, PhD
Dolan School of Business
Fairfield University

Dr. Christopher Huntley, PhD
Dolan School of Business
Fairfield University

Two developers walk into a bar. The bartender asks, "why the long faces?"

Developer 1: "I love coding, but I spend all my time doing all these unnecessary documentation that even my boss can't even read. And I have to work 100 hours a week because some bozos in marketing decided on all these milestones we can't meet with silly requirements that no one's going to use"

Developer 2: "My job doesn't involve any of that because we're doing something called "agile" - it's nice because I get to do what I want. But now I'm wishing we had decided on the architecture up front; I have to do all this unnecessary rework. And management hates us because they have no clue what we're doing"

Advocates of agile and formal methods aver that their chosen method is superior to the other, with a few contending that their method is a silver bullet applicable in all situations and contexts. However, others believe that each have their 'home grounds'. For example, Boehm & Turner (2004) believe that agile is more suited for projects that experience a lot of change and that projects that are relatively stable can benefit from the 'Big Design Up Front' of formal methods. Some advocates of Agile insist that it is an 'all-or-nothing' approach, i.e., you have to follow all their principles otherwise your methodology isn't Agile. However, it is becoming increasingly clear that such polarized notions do not reflect the reality of systems development

organizations. For example, in a conference between several practitioners of agile methods, it became clear that most were not following all the principles advocated (Lindvall et al 02). It also became clear that the methods followed were not the rigid 'waterfall' approaches that agile advocates frequently use a a straw-man.

So if the majority of systems being developed don't completely follow agile methods, nor do they completely follow formal methods, what might these emerging methods be? In this article, we suggest that there may be two 'modified' approaches to systems development emerging in the industry.

To postulate what these new approaches may be, we have to first analyze why agile or formal methods are not being followed as traditionally conceptualized. Boehm & Turner (2004) suggest that practitioners chose to go either agile or formal depending on the project. If it is a project that is rapidly changing, they follow agile methods. If the project is relatively stable they chose formal methods. However, it may not be that easy to switch between agile and formal methods. Systems development organizations may be constrained by other forces that prove very difficult to change if they want to switch between a purely formal approach and a purely agile approach. This is because there may be another factor that can affect the suitability of agile methods: the culture of the organization. Organizational cultures vary between more flexible 'organic' cultures and more structured 'mechanistic' cultures (Burns & Stalker, 1961). When practitioners discussed issues about changing from formal methods to agile methods, they agreed that the most difficult aspect was that it requires changes in organizational culture (Lindvall et al 02). Organizational culture is one of the most difficult things to change in an organization - organizational culture is more difficult to change than the organization's strategy, structure, processes, or tools (Adler 1989), and can take years. In a recent survey, non-adopters of agile quoted "rigid-culture" as the biggest reason for not switching to agile. (DDJ, 2009). Unfortunately, agile advocates do not see culture as a real impediment, and even dismiss it as an "excuse". In addition, it's not just the IS development organization that has to have this culture – the client organization also has to have a similar culture. For example, a bureaucratic organization might not be able to handle a project without

set milestones, deadline, plans, schedules, budgets, etc. They might not want to be bothered with a constant stream of deliverables upsetting their carefully planned schedules.

Mechanistic	Organic
Individual specialization: Employees work separately and specialize in one task	Joint Specialization: Employees work together and coordinate tasks
Simple integrating mechanisms: Hierarchy of authority well-defined	Complex integrating mechanisms: task forces and teams are primary integrating mechanisms
Centralization: Decision-making kept as high as possible. Most communication is vertical.	Decentralization: Authority to control tasks is delegated. Most communication lateral
Standardization: Extensive use made of rules & Standard Operating Procedures	Mutual Adjustment: Face-to-face contact for coordination. Work process tends to be unpredictable
Much written communication	Much verbal communication
Informal status in org based on size of empire	Informal status based on perceived brilliance
Organization is a network of positions, corresponding to tasks. Typically each person corresponds to one task	Organization is network of persons or teams. People work in different capacities simultaneously and over time

Table 1: Mechanistic and Organic Cultures (Borgatti, 2001)

From this it seems that agile methods are more suited for organic cultures and projects that have a high degree of change. On the other hand, formal methods may be more suitable for mechanistic organizations and stable projects. However, what if the organization is organic but the project is relatively stable? What if the organization is mechanistic but the project keeps

changing? We may need some sort of hybrid approach for these two scenarios. To examine what these two approaches may be, we look at the Agile Manifesto:

1. individuals and interactions over processes and tools,
2. working software over comprehensive documentation;
3. customer collaboration over contract negotiation;
4. responding to change over following a plan.

The items on the right (processes, tools, documentation, contracts and plans) can be of two types: Those that relate to upfront design, and those that relate to upper management's control over the development team. Upfront design may be needed in more stable projects, while management oversight may be required by more mechanistic cultures. However, when taken in conjunction with the items on the left, the first two principles of the manifesto relate more to the organizational culture, while the third and fourth principle relate more to the amount of change on the project.

Organic organizations will have very definite preferences on the first two statements of the manifesto. In line with the first statement, organic organizations would prefer not to be bound by rigid, formal processes, they'd rather have their people decide what is necessary. Mechanistic organizations, on the other hand, may not prefer the ambiguity of letting their people decide everything. They would rather have set guidelines to follow, fixed processes and responsibilities.

In line with the second statement, organic organizations do not like copious documentation. They'd rather get the job done and dispense with the formalities. On the other hand, mechanistic organizations need everything written down and documented so that there is a paper trail to follow, superiors can check if all guidelines were followed, etc.

The third and fourth statements of the manifesto, however, deal more with rapidly changing projects. The third statement, on frequent customer feedback, becomes more important for changing projects rather than stable ones. When projects keep changing, meeting with customer becomes critical to project success. On the other hand, if projects are more stable, a lot of optimization can be achieved and rework mitigated by designing more upfront.

The fourth statement, on responding to change, is also meant more for changing projects. Agile methods are iterative, incremental and adaptive for this reason. On the other hand, a linear, phased structure with upfront design may be more suitable for projects that are relatively stable.

Now we can examine the two earlier scenarios: 1) Mechanistic organizations that have changing projects and 2) organic organizations that have relatively stable projects (Figure 1)

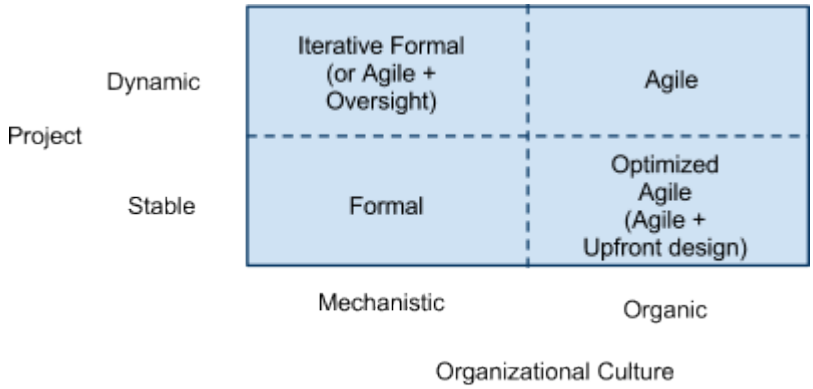

Figure 1: The four methods

1) Mechanistic organizations that have changing projects: These organizations will find it very difficult to follow the first two principles of the Agile manifesto, as this would involve changing their culture, as they need structured processes and tools, comprehensive documentation, as dictated by their organizations policies. However, they can benefit from the third and fourth principles of the agile manifesto to handle the higher amounts of change in their project. In line with the third principle, they can plan for frequent customer meetings to review work done so far, ascertain changes to the project, and plan the next iteration. In line with the fourth principle, these organizations can have project management process that is more iterative, more adaptive and more emergent. This will help them deal with the amount of change in the project. A recent survey indicates that such iterative methods are not only as popular as agile methods, but also have as much success as agile methods (DDJ 2008). Even though these methods have the iterative, incremental, and emergent as well as the customer collaboration characteristics of agile projects, they can be distinguished by a heavier reliance on processes, tools and documentation. For example, the oxymoron "agile tools" was created to meet the needs of this market. Demand for these tools is growing (Goth 2009), especially because of the needs of top management to gain visibility and control into the working of the

adaptive development teams below them. Another example is a blend of the Rational Unified Process and agile methods called the Agile Unified Process (Ambler, 2006), which is process-heavier than most agile methods, yet iterative enough to handle change. Christou et al (2009) describe an organization where cultural/political issues were the main barrier to agile, but the Agile Unified Process (AUP) was formal and structured enough for the organization to accept. 2) Organic organizations that have projects with some level of stability. While it may be impossible to accurately predict every single requirement accurately at the beginning of the project, it is not impossible that at least a few requirements remain relatively stable through the duration of the project For example, embedded systems, regulatory compliance requirements, some requirements for mission critical projects, requirements from a few stakeholders in projects that require several stakeholders, non-functional requirements for enterprise systems, etc. may be examples of such requirements. While some projects contain very few such requirements, other may contain more. With latter projects, agile teams can gain several advantages by doing up-front architecture and design with the predictable requirements while leaving the others to emerge. In fact, a recent survey by Scott Ambler indicates that 89% of agile teams do some sort of up-front requirments and 86% do some sort of up-front architecture/design (Ambysoft, 2009). In this case, the organic organizations will still have to follow the first two principles of the manifesto, as it is part of their culture to depend on people instead of depending on processes (principle one). In addition, extensive documentation and following guidelines is not part of their culture. However, they can change the project management process instead of following principles three and four in the manifesto. The third principle need not be followed as much as the project is relatively stable, so they need not meet the customer too often. The fourth principle also need not be followed as much. As the project is relatively stable, the project can have fewer and longer iterations, with more design up front instead of being more emergent and adaptive. The increase in upfront design may enable them to optimize the architecture and reduce rework and integration issues from adding requirements. For example, Madison (2010) describes several projects in which agile methods incorporated upfront architectural design to achieve several benefits for the development team, the project, and the client. Madison calls this "Agile Architecture" (Madison, 2010). Similarly,

Faber (2010) and Blair et al (2010) describes experiences with providing architecture as a 'service' to agile development teams. In this conception, architects are responsible for non-functional requirements while developers are responsible for functional requirements, and 'sets up rules, but helps break them'. This approach had several advantages over earlier agile projects that did not incorporate up-front architecture. These advantages include avoiding costly application design approaches, validating crucial performance requirements earlier, and less application code (Faber, 2010). In this approach, it is important that architects are 'servant leaders' and not bottlenecks between developers and stakeholders (Blair et al 2010). Embedded systems also require considerable deviations from agile to fit their context, including comprehensive functional and non-functional requirements gathering at the beginning of the project (Smith et al 2009).

Conclusion

Agile and Formal methods are not two ends of a continuum, but rather, vary on two dimensions. The first relates to the amount of upfront design, while the second relates to control and oversight from top management. Upfront design is needed when requirements are stable - Most projects have some requirements that are relatively stable, and agile methods can be 'optimized' by doing some upfront design for these requirements. Oversight from management is needed by organizations that have more mechanistic cultures - Developers in these organizations can handle changing requirements either by making formal methods more iterative, or by adding processes and tools to agile methods to give more visibility to top management. Emerging empirical evidence shows that most development teams actually follow one these two approaches - Most agile teams use some upfront design (Ambysoft, 2009), and most formal methods are iterative (DDJ 2008). This indicates that the arguments against agile ("agile has no architecture") and formal methods ("formal methods can't respond to change) are misplaced, and both have similar success rates (DDJ 2008). It also indicates that both research and practice need to focus more on supporting optimized agile (i.e., agile with some upfront design), and iterative formal (iterative methods or agile with 'visibility' tools).

References:

1. Adler, P. S. (1989). CAD/CAM: Managerial challenges and research issues. IEEE Transactions on Engineering Management, 36(3), 202-215.
2. Ambler, Scott W. (2006) The agile unified process. Retrieved from http://www.ambysoft.com/unifiedprocess/agileUP.html; current as of Feb 2, 2010
3. Ambysoft (2009) Agile project Initiation Survey. Retrieved from http://www.ambysoft.com/surveys/projectInitiation2009.html. Current as of Feb 2, 2010
4. Blair, S; Cull, T; Watt, R (2010) Responsibility driven architecture. IEEE Software, forthcoming.
5. Boehm, Barry & Turner, Richard (2004) Balancing Agility and Discipline. Addison-Wesley.
6. Borgatti, Stephen P. (2001) Organic Vs Mechanistic. Retrieved from http://www.analytictech.com/mb021/, current as of February 14, 2010
7. Burns, Tom, and George M. Stalker, The Management of Innovation, Tavistock Publications (1961)
8. Christou, I; Ponis, S; Palaiologou, E (2009) IEEE Software, forthcoming.
9. DDJ (2008) Software development project success rates survey 2008. Retrieved from http://www.ambysoft.com/surveys/success2008.html. Current as of Feb 2, 2010.
10. DDJ, (2009) State of the IT union survey, November 2009. Retrieved from http://www.ambysoft.com/surveys/stateOfITUnion200911.html. Current as of Feb 2, 2010
11. Faber, Roland (2010) Architects as service providers. IEEE Software, forthcoming.
12. Goth, Greg (2009) Agile tool market growing with the philosophy. IEEE Software, 26(2) pp 88-91.
13. Lindvall, M., Basili, V., Boehm, B., Costa, P., Dangle, K., Shull, F., Tesoriero, R., Williams , L., & Zelkowitz, M. 2002. Empirical Findings in Agile Methods. Paper presented at the Extreme Programming and Agile Methods - XP/Agile Universe, Chicago, IL, USA
14. Madison, J. (2010) Agile Architecture Interactions. IEEE Software, forthcoming.

15. Smith, M; Miller, J; Huang, L; Tran, A (2009) A more agile approach to embedded systems development. IEEE Software 26(3) pp. 50-57