

**IT Executive Exchange Meeting, 9/16/2005****TOPIC: New Programming Methodologies**

Summarized by Prof. William McHenry

**EXECUTIVE SUMMARY**

On September 16, a number of regional IT Executives and faculty from the University of Akron's College of Business Administration held the next regular meeting of the IT Exchange, an informal discussion group that meets about every six weeks on current and leading edge topics of most relevance to today's IT executives. During this session, the focus was on "lightweight" programming methodologies, emphasizing Agile Programming (AP). Topics discussed included:

- how to best make use of consultants when starting an AP trial or program
- how AP can deliver improved estimates for project duration and staffing
- the role of "iteration managers" and Program Management Offices, and how management changes when using AP
- the crucial role of testing and quality assurance people with AP
- how maintenance fits in the AP paradigm
- other issues regarding AP, "lightweight" methodologies, the influence of Sarbannes-Oaxley, and achieving CMM certification.

The firms that have begun using AP have had positive results so far. While it was agreed that the traditional "waterfall" method works very well for projects with well-known and well-understood specifications, AP is a highly promising technique to deal with the necessity of changing specifications in a dynamic business climate.

**COMPLETE NOTES ON SESSION**

During this session, the main focus was on Agile Programming.

One of the regular participant firms (Firm A) has a large Agile trial going on, represented by two programming teams. Other teams in this group use a lightweight methodology. A considerable amount of time was spent discussing this firm's experience. Its Agile experiment started about a year ago. The firm hired a consulting firm to train them, but did not accept all of the firm's recommendations. The consulting firm presented to them a "pure" version of Agile, but Firm A did not want to shelve all that it had been doing previously, and so insisted on adapting Agile to some of its processes that were working well. Firm A also tried as much as possible to ensure that the technology was transferred from the consulting firm to them, and then it proceeded without the consultants.

The essence of Agile as practiced by Firm A is to describe the high level requirements as a series of stories, each one fitting on a small card. Experienced managers read these stories and assign them points according to their relative difficulty. Points do not equate

to weeks or resources needed. They are just a way of ordering the stories by perceived difficulty. Once this is done an “iteration manager” and business representative choose which stories will be implemented in a series of very short iterations, on the order of 2-3 weeks. The goal of each iteration is to provide a complete build of the system as it stands at that time. Test plans are written for the stories before the stories themselves are implemented, and no story is added to the build unless it passes its tests. Each time a story is added all tests are run. Firm A has automated much of the testing to facilitate these procedures. Programmers are alerted as to whether their own additions to the build have failed the test in the form of their screens going red, and of course no one likes this form of public reproach. (Firm D wrote users manuals before writing code, which was a form of a story and test plan.)

Over time, Firm A has come to understand the relationship between the points added and how many actual resources are needed to implement them. This was an unexpected benefit of using Agile, and the managers feel that the estimates they can now do using this method are a lot more accurate. (Another firm at the table said that they often will go over the allotted time frame by 30%, and one recounted a story in which the project was over the time frame by a factor of 10!).

Firm C said that budgeting does not seem to change much using Agile. It is still necessary. Estimates are still needed in order to get projects approved, although there may be less rigidity and getting away from a yearly cycle. Firm C does not yet have a feel for how good its estimating is using their version of Agile. Firm B noted that Agile can be compared to lean manufacturing where people and machines are seen as sunk costs. The point is once you see them this way, then how do you get the most out of them.

Are there “phase gates” with Agile? Firm D, which still uses more of a waterfall process, has go/no-go decisions after certain phases. Firm C, using a form of Agile that involves three-week iterations, still has some phase gates in place. Its environment includes a corporate parent body that uses stricter phase gates, and in that case there is no question of not using them. In Agile the phases are more like chunks of functionality. Large chunks of functionality still have to be prioritized.

A new work space has been created for Agile at Firm A. In a room that might be described as a large bullpen, all involved parties sit at desks that are not separated by any barriers. This means that QA (Quality Assurance) people, programmers, the iteration manager, the architect, and at least one business representative are seated side-by-side and can engage in continuous consultations. Daily stand-up meetings are held early in the morning to quickly review everyone’s progress (A second firm at our table reported that they also hold such meetings in the afternoon every day).

The results at Firm A are so good that many other programmers now want to join Agile teams, and the overall manager wants to move many teams in his group to this methodology. The work time is intense. There are even periods when programmers are paired in front of one screen, watching each other work. This means constant attention is focused on work. While the eight hours are intense, the programmers can usually go

home after this period and relax. Rarely do they have to stay late or work on weekends, except very close to deadlines.

The group spent some time discussing the role of high-level requirements. The way Agile works at Firm A is that users really must pick the things that they actually want. Many requirements never get built because their story cards never make it high enough in the pile. That's fine with the leader, because he thinks that users often ask for far more than they really want and/or need. Then what happens to high level requirements? It seems as though they are defined little by little with Agile, although there has to be an overall vision and architecture to ensure that all the pieces fit. Since each build is a complete build each time, though, there is never the question of different teams building components that ultimately will not work together at integration time. They are not exactly defining as they go, but the process is one of continuous evolution and small incremental changes that are highly tuned to users' needs.

The group was conceptualizing what Agile is like. One participant suggested that it is like requirements are being turned into objects. Each requirement is developed it is entirety, rather than simultaneously trying to work on all parts of the system from the bottom up. (This may be like building a house room by room rather than building the whole foundation, then the whole first floor, then the whole second floor, etc. However, you do know if you are building a ranch house, a split-level, a "MC-Mansion," etc.) Firm D sees Agile as a way of modifying processes to the project at hand.

Is Agile just another form of RAD (Rapid Application Development)? The group did not seem to be too familiar with just what RAD is, although it was identified as mainly having to do with a lot of prototyping. Firm C used a lot of prototyping for defining user interfaces.

What about support? Firm C uses the same people who are developers as support people. They do train some help desk people in the new software. They need to be deft about negotiating with users as to whether something gets fixed now or is deferred into the new version of the software.

What about multi-location teams, including offshore? It was agreed that Agile would be harder, and that this would require a very good architect and iteration manager to ensure that the pieces fit together. A Russian firm was mentioned that advertises its ability to perform Agile remotely.<sup>1</sup>

Another issue discussed at this meeting was the influence of Sarbannes-Oxley (SOX). Firms in the room that are private did not worry about it too much, but did keep its precepts in the backs of their minds because they might have to comply with it if they ever went public. One firm said that its processes were more or less fine, but SOX added the necessity of documenting them at a level they had not done previously. Plus they have

---

<sup>1</sup> Discussed in the following paper: Pries-Heje, Jan; Baskerville, Richard; Hansen, Galina Ianshina, "Strategy models for enabling offshore outsourcing: Russian short-cycle-time software development," *Information Technology for Development*, Volume: 11, Issue: 1, Winter 2005, pp. 5 - 30

to exert more oversight over whether the procedures are actually followed. Firm D has pretty strong divisions between requirements specifiers, developers, testers, etc., so they do not feel they have had problems with SOX.

Agile requires very good quality assurance people. QA people are not like developers. They may be diametrically opposed. Good QA people have to be able to take the heat that entails when they point out that an on-time project will no longer be on-time because of certain defects. They have to know the business well, know the application, be detail-oriented, and have backbone.

One thread of the discussion concerned the newness of Agile and its longer term potential. Even Firm A has only been using this for a year, and it chose highly qualified and motivated teams for the trial. Can teams that are more average in composition perform as well? It is possible that any new methodology will show initial high results because of the novelty and motivation involved. Another question was raised about how to keep everyone in the bullpen busy. It was agreed that this requires a very skilled iteration manager. There have to be enough tasks so that some multitasking is possible.

The group discussed how Agile fits in with maintenance. Firm C does not make any distinction between developers and maintenance programmers. Such a distinction inevitably leads to tensions on the team. So all developers do both types of tasks. Although it may not be possible to assemble the same original “bullpen” of people when maintenance issues arise, the straightforwardness of defect resolution probably does not require the same level of user-developer intimacy as required in the initial development.

### **Related Developments**

One of the firms represented (Firm B) is still mainly using a waterfall method. Firm B has implemented a web-based tracking tool that is “lightweight,” i.e. does not require huge amounts of documentation to be written. Each stage is tracked from the initial request through the workflow of the implementation steps. Projected ROI is recorded and developers are assigned. Then testers record issues found and developers record how they were resolved and closed out. Firm B plans an Agile trial next year. It was agreed that the waterfall method is best when you know exactly what it is you need to build.

Firm D had recently established a Project Management Office (PMO). Its previous IT organization had comprised a CIO and some Vice-Presidents, under which there were heads of IT in each functional area. This led to a lot of stovepipes and inability to share resources across them. The new PMO controls all the developers as a pool and assigns them to teams for projects in the functional areas as needed. With the PMO they can give a priority score for the first time to all projects across the firm. This has been a major culture change for them. They can move resources around if projects get in trouble—this was a major benefit that they had not foreseen. There are customer relationship managers who work with the functional areas as liaisons.

Firm E worked on implementing the Software Engineering Institute's CMM. It had a software product that it sold to customers. They had to make consistent changes from version to version, but also take into account modifications they had made for several different customers. So they needed very strong version control management. The CMM forced them to take better charge of the processes they were using. They hired a consulting firm to help them make the transition, and this firm later supplied the people who evaluated them. They passed Level 2. When he left this firm, they had reached many aspects of Level 3, but had not officially passed it. It took \$20-30K initially to get this going and then the consultants would come if for a few days here and there for additional work. Determining compliance took 3-4 days, which was facilitated by the consulting firm's familiarity with the environment.

### **References for further readings:**

Definitions of Agile on the Web

([http://en.wikipedia.org/wiki/Agile\\_software\\_development](http://en.wikipedia.org/wiki/Agile_software_development))

Manifesto for Agile Software Development

(<http://www.agilemanifesto.org/>)

Agile Alliance

(<http://www.agilealliance.org/home>)

Links below are good starting points for further reading on the Agile methodology and several of the concrete implementations of the principles.

<http://www.stsc.hill.af.mil/crosstalk/2002/10/highsmith.html>

<http://www.agilealliance.org/programs/roadmaps/Roadmap/index.htm>

A document from <http://www.cutter.com/research/freestuff/AgileObjections.pdf> outlines common objections to Agile along with responses to those objections.

The following link should take you to a page for a great series of articles dealing specifically with the XP implementation of Agile titled "Demystifying Extreme Programming".

<http://www->

[128.ibm.com/developerworks/views/java/libraryview.jsp?search\\_by=demystifying%20extreme%20programming](http://www-128.ibm.com/developerworks/views/java/libraryview.jsp?search_by=demystifying%20extreme%20programming)

The next couple of links deal with the Lean implementation of Agile. The second link below seems particularly useful in offering some concrete examples of the Lean principles as they apply to software development (e.g. what constitutes waste, etc.).

<http://www.poppendieck.com/lean.htm>

[http://www.poppendieck.com/pdfs/Lean\\_Software\\_Development.pdf](http://www.poppendieck.com/pdfs/Lean_Software_Development.pdf)