Agile Software Project Management Methodologies

Prof. Constanța-Nicoleta BODEA, PhD
Economic Informatics Department, Academy of Economic Studies, Bucharest

Successfully project planning, coordinating and controlling in order to deal effectively with projects sponsors, customers, unexpected risks and changing scope are difficult tasks even for the most experienced project managers. Different surveys indicated that about half of the software projects were considered total failures and only a few of them were successful. The tight deadlines, volatile requirements and emerging technologies are the main reasons for this lake of performance. This agile project environment requires an agile project management. The paper presents the main characteristics of the agile software project management approaches such as: MSF for Agile Software Development, Extreme Programming, Scrum, Crystal, Feature Driven Development, DSDM.
Keywords: *software development, project management methodology, agile project management, XP, MSF for Agile Software Development.*

1 Software project management methodologies

Methodologies impose a disciplined process upon software development with the aim of making software development more predictable and more efficient. We can consider a methodology containing ten basic elements: techniques, tools, deliverables, teams, roles, skills, activities standards, quality measures and project values [1].

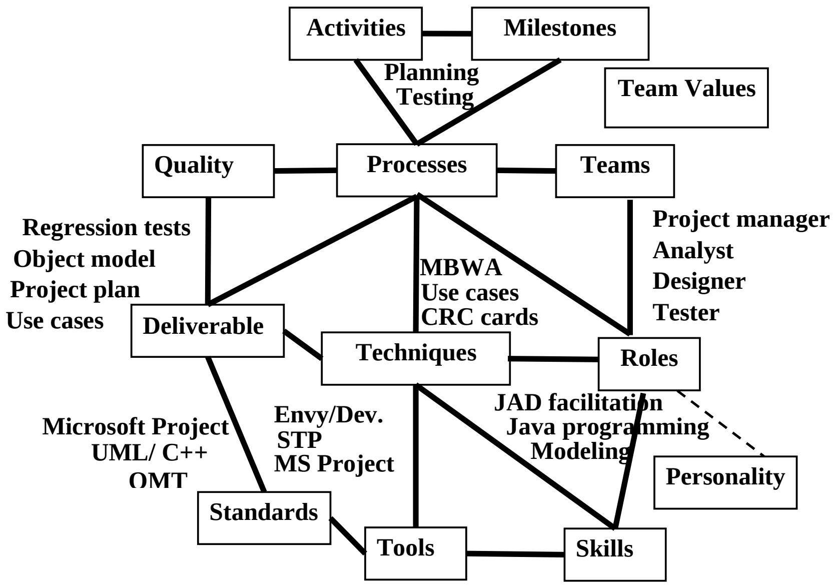

Fig. 1. Components of a project management methodology

A specific methodology is needed depending on the project size (number of people being coordinated), the criticality of the systems being created and the priorities of the project. For any point in the size/criticality space, a scope of concerns to address is selected (which project roles, activities, deliverables, and standards to cover) and optimization criteria are selected. Methodologies therefore differ by the size, criticality, scope and optimized quality. A larger methodology (with more control elements) is needed when more people are involved. Communication load raises as the number of people involved increases. Since methodology is a matter of coordinating the people and managing the communication, its size must also rise, as the number of roles and deliverables types increase [2].

Considering the project deliverables criticality the following four zones we can identify:

- *Loss of comfort* means that with a system failure, people will have to go and do more work by hand, or call each other and repair a miscommunication. Examples might include purchase support systems and corporate infrastructure programs.
- *Loss of discretionary moneys* zone if the loss of money or related valuables is merely uncomfortable
- *Loss of irreplaceable moneys zone* if the loss of moneys or related valuables has effect corresponding to going bankrupt.
- *Loss of life zone* if people are likely to die from a system malfunction.

For a project with higher criticality more visible correctness (greater density) is required. Density means more precision in the artifacts, with tighter reviews and less tolerance.

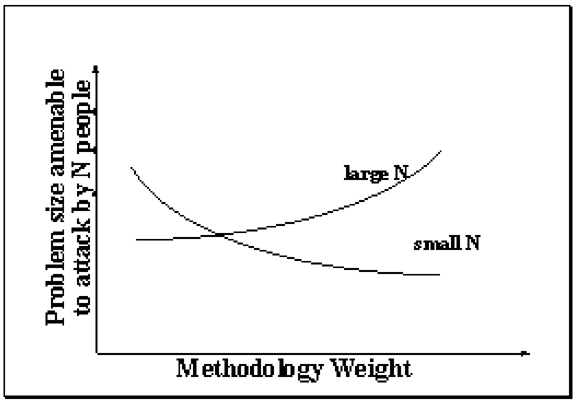

Fig. 2. Methodology weight and problem size

"Weight is cost": a relatively small increase in methodology size or specific density adds a relatively large amount to the cost of the project. With fewer people, less methodology is needed; with less methodology, the people work more efficiently. Working more efficiently, they can successfully address a larger problem. When more people are put onto a project, they need a heavier methodology to coordinate their work. The heavier methodology lowers their productivity, so more people are needed on the project. Since methodology size grows slower than project size, eventually they get to a point where they can solve the problem and manage the coordination activities. This does not mean that a small team can necessarily solve a larger problem than a large team. It does mean there may be an area of overlap, where a small team with a light methodology can solve the same problem as a larger team with a heavier methodology (figure 2).

2. The Agile Approach

The agile approach started in 1994 with some trials of *semi*-formal agile methodologies, such as RAD, DSDM, XP, Crystal, Scrum. These methodologies are based *on agile methods*. Agile methods are adaptive rather than predictive. Engineering methods tend to try to plan out a large part of the software process in great detail for a long span of time, this works well until things change. So their nature is to resist change. The agile methods, however, are waiting for change. Agile methods are people-oriented rather than process-oriented. The goal of engineering methods is to define a process that will work well whoever happens to be using it. Agile methods assert that no process will ever make up the skill of the development team, so the role of a process is to support the development team in their work.

The declaration of principles and values in the agile approach is known as the *Agile Software Development Manifesto,* launched in 2001, after a two day workshop at Snowbird Utah (figure 3). A non-profit organization the Agile Alliance was set up to promote knowledge and discussion of all the agile methods.

Applying these principles creates the foundation for managing IT projects in an agile approach. The basic characteristics of this approach are the following:

- *Assume simplicity.* As the project evolves it should be assumed that the simplest solution is the best solution. Overbuilding the system or any artifact of the project must be avoided.
- *Embrace change.* Since The stakeholder understanding of the requirements will change over time. Project stakeholders themselves may change as the project makes progress. Project stakeholders may change their point of view, which in turn will change the goals and success criteria of the project management effort.

• *Incremental change* – the pressure to *get it right the first time* can overwhelm the best project manager. Instead of futilely trying to develop an all encompassing project plan from the start, put a stake in the ground by developing a small portion of the system, or even a high–level model of a larger portion of the system, and evolves this portion over time. Or simply discard it when you no longer need it in an incremental manner.

• *Maximize stakeholder value.* The project stakeholders are investing resources (time, money, facilities) to have a system deployed that meets their needs. Stakeholders expect that their investment to be applied in the best way.

"We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

: Individuals and interactions over Processes and Tools.

: Working software over Comprehensive documentation.

: Customer collaboration over Contract negotiation.

: Responding to change over Following a plan.

That is, while there is value in the items on the right, we value the items on the left more."

Fig. 3. Agile Software Development Manifesto

• *Manage with a purpose* Identify a valid purpose for creating the artifact and the audience for that artifact. This principle also applies to a change to existing artifacts.

• *Rapid feedback.* The time between an action and the feedback on that action must be minimized. Work closely with the stakeholders, to understand the requirements, to analyze those requirements, and develop an *actionable* plan, which provides numerous opportunities for feedback.

• *Working software is the primary goal of the project.* The goal of any software project is to produce software that meets the needs of the project stakeholders. The goal is not to produce extraneous documentation, management artifacts or models of these artifacts.

3. Some Agile Software Project Management Methodologies

The agile approach focuses on: talent & skill (fewer better people), proximity (direct and face-to-face communication), less paper, more tacit / verbal communication, just-in-time requirements and design, frequent Delivery (incremental development), reflection, quality in work. So, the people are very close related to the agile methodologies (figure 4).

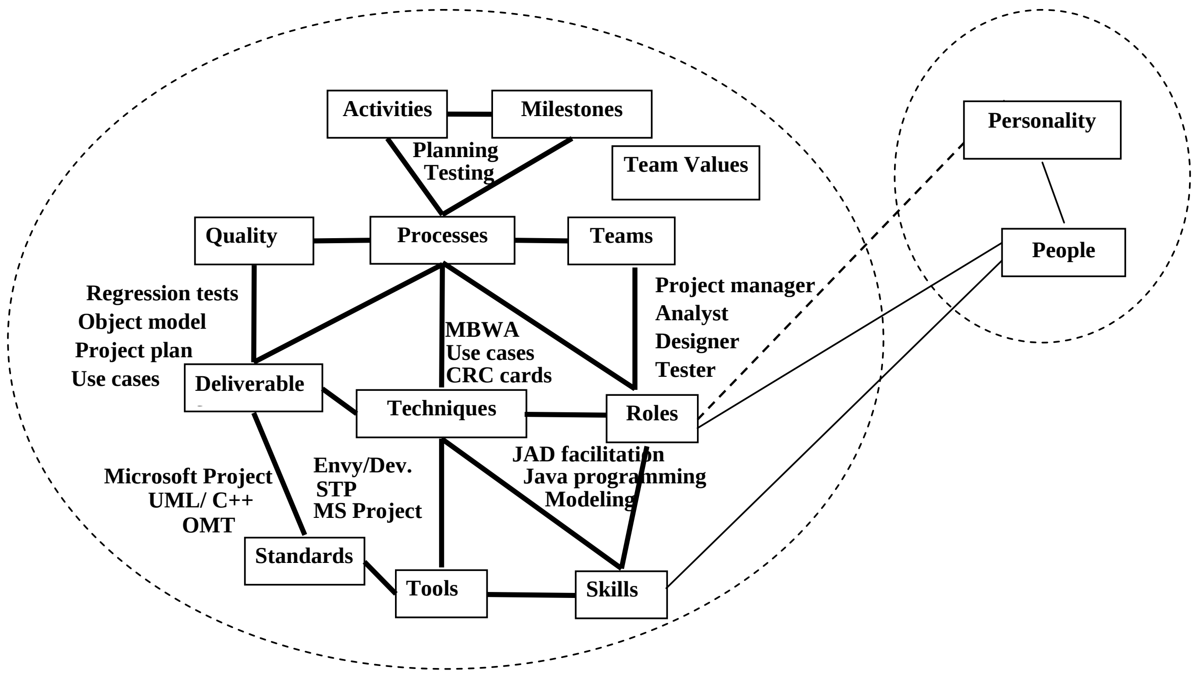

Fig. 4. Components of an agile project management methodology

3.1 Extreme Programming (XP) methodology

The roots of XP lie in the Smalltalk community, in the close collaboration of Kent Beck and Ward Cunningham in the late 1980's. Both of them refined their practices on numerous projects during the early 90's, extending their ideas of a software development approach that was both adaptive and people-oriented. The crucial step from informal practice to a methodology occurred in the spring of 1996. Kent was asked to review the progress of the C3 payroll project for Chrysler. The project was being carried out in Smalltalk by a contracting company and was in trouble. Due to the low quality of the code base, Kent recommended throwing out the entire code base and starting from scratch. The project then restarted under his leadership. XP begins with four values: Communication, Feedback, Simplicity, and Courage. It then builds up to a dozen practices which XP projects should follow. Many of these practices are old, tried and tested techniques, yet often forgotten by many, including most planned processes. As well as resurrecting these techniques, XP weaves them into a synergistic whole where each one is reinforced by the others. It is a strong emphasis on testing. While all processes mention testing, most do so with a pretty low emphasis. However XP puts testing at the foundation of development, with every programmer writing tests as they write their production code. The tests are integrated into a continuous integration and build process which yields a highly stable platform for future development.
On this platform XP builds an evolutionary design process that relies on refactoring a simple base system with every iteration. All design is centered on the current iteration with no design done for anticipated future needs. The result is a design process that is disciplined, yet startling, combining discipline with adaptivity in a way that arguably makes it the most well developed of all the adaptive methodologies.

3.2 Crystal methodologies

Alistair developed this family of methodologies considering that different kinds of projects require different kinds of methodologies. The Crystals share a human orientation with XP, but this people-centeredness is done in a different way. Alistair considers that people find it hard to follow a disciplined process, thus rather than follow XP's high discipline; Alistair explores the least disciplined methodology that could still succeed, consciously trading off productivity for ease of execution. He thus considers that although Crystal is less productive than XP, more people will be able to follow it.
Alistair also puts a lot of weight in end of iteration reviews, thus encouraging the process to be self-improving. His assertion is that iterative development is there to find problems early, and then to enable people to correct them. This places more emphasis on people monitoring their process and tuning it as they develop.

3.3 Scrum

Scrum has been around for a while in object-oriented circles. It focuses on the fact that defined and repeatable processes only work for tackling defined and repeatable problems with defined and repeatable people in defined and repeatable environments.
Scrum divides a project into iterations (which they call sprints) of 30 days. Before you begin a sprint you define the functionality required for that sprint and then leave the team to deliver it. The point is to stabilize the requirements during the sprint.
However management does not disengage during the sprint. Every day the team holds a short (fifteen minute) meeting, called a scrum, where the team runs through what it will do in the next day. In particular they surface to the management blocks: impediments to progress that are getting in the way that management needs to resolve. They also report on what's been done so management gets a daily update of where the project is.
Scrum literature focuses mainly on the iterative planning and tracking process. It's very close to the other agile in many respects and should work well with the coding practices from XP.

3.4 MSF for Agile Software Development

MSF provides a customized and scalable set

of software development guidelines for application development improvement ([5]). MSF incorporates both agile and formal approaches, and then allows the user to select the most suitable path. MSF's flexible framework can be adapted to meet the needs of any project, regardless of size or complexity.
The MSF philosophy holds that there is no single structure or process that optimally applies to the requirements and environments for all projects. MSF provides this guidance without imposing prescriptive detail and allows the user to customize the content provided. MSF components can be applied individually or collectively to improve success rates for the many types of projects. MSF guidance focuses on managing the "people and process." Because the needs and practices of software development teams are constantly evolving, the materials gathered into MSF are continually changing and expanding to keep pace. Additionally, MSF interacts with Microsoft Operations Framework (MOF) to provide a smooth transition to the operational environment, which is a requirement for long-term project success.
With MSF, process is not just documentation. It also manifests itself as actual tool behavior changes. When you chose the process at project inception, you are also choosing the workflow and work products, which then drive how the system behaves. Support for the software development life cycle process (SDLC) is built-in, which makes for seamless workflow support. By integrating process into the tools team members use on a daily basis, MSF lowers the barrier to adopting process and enables the automatic collection of cross-functional project metrics without the overhead associated with manual reporting.
The following elements of MSF are customizable:

- Process Guidance
- Iteration structure
- Entry criteria and exit criteria views
- Work item type definitions and rules (activities and work products)
- Work item queries
- Source check-in policies
- Role clusters and security groups
- Document templates (Excel and Word)
- Microsoft Project templates
- Reports
- Project portal /SharePoint site template

MSF uses methodology templates to define the process that individual projects follow. There is no universal process that works for all organizations, or even all projects within an organization. To address this, MSF provides a flexible toolset that works with both agile and formal processes. Microsoft's Global Solution Integrator partners provide their own product consumable methodology templates; or, you can create your own. Process extensibility allows customization of work item types, check-in policies, custom reports and project management templates.

Conclusions

Getting projects faster is a universal desire of management. The reality of project management is that we never really have the time to create perfect plans, to analyze all the options. Agile approach provides some methods for project management to become more effective. These methods need to be taken and customized to the unique business environment of the project.

References

1. Alistair C. *A Methodology Per Project*, (arc@acm.org), Humans and Technology.
2. Harrison, N., Coplien, J, "Patterns of productive software organizations", Bell Labs Technical Journal, summer, 1996.
3. Jeffries, R., Beck, K., *Extreme Programming*, http://armaties.com/extreme.html.
4. Fowler M, *The New Methodology*, Martin Fowler.com
5. Microsoft, *Visual Studio 2005 Team System: Microsoft Solutions Framework*, 2004, www.Microsoft.com .
6. http://crystalmethodologies.org/