Agile Ecosystems

A Position Paper for the Workshop: Are Agile Methodologies Really Different?

by Mary Poppendieck

Jim Highsmith identifies seven agile software development ecosystems[1]:

1. Scrum
2. DSDM
3. Crystal Methods
4. FDD
5. Lean Development
6. XP
7. Adaptive Software Development

One of the big differences between these approaches is the amount of specific guidance vs general guidance embodied in each. Scrum, DSDM, FDD, and XP give specific guidance in the areas they cover, while Lean Development, Adaptive Software Development, and Crystal Methods present a theoretical basis for agile practices.

Taking on the theoretical methods first, each has a somewhat different focus: Crystal concentrates on communication and varying practices based on project size and risk. Agile Software Development focuses on emergence, while Lean Development emphasizes the traditional lean concepts of value and flow. All three emphasize the primary importance of the development team. None of these three methods focus on specific practices, so they do not find themselves in conflict with the four agile methods that offer more specific practices, or with each other, for that matter.

The four other agile methods: Scrum, DSDM, FDD, and XP offer specific practices which are generally expected to be followed as a package until one is expert enough in the method to modify the practices. DSDM and Scrum focus on project management and are agnostic about the underlying technical approach. Both emphasize business value, fixed timeboxes, significant customer involvement, developing high priority items first, and stopping when you run out of time. DSDM is the higher ceremony approach, while Scrum spends less time on project initiation activities.

FDD is a complete approach covering both technical issues and project management. It is different from the other agile practices because it does not advocate common code ownership, but rather assigns class ownership. This rather large technical difference underlies many of its other differences from other agile practices. Because of its comprehensiveness and different technical approach, FDD would not co-exist well with other agile approaches, but would be used separately.

---

[1] *Agile Software Development Ecosystems,* Addison-Wesley, 2002.

Scrum and XP are often used together because their practices are more or less disjoint: Scrum focuses on project management while XP focuses on developer practices. Both of these approaches are often billed as a complete set of practices that are supposed to be used without modification, at least until the users become skilled enough to make appropriate changes. However, when used together, a few differences have to be resolved, including recommended iteration length, specific planning details including release planning and stories or backlog items.

However, for all of their differences, I find the most interesting difference to be the way leadership is viewed in the various practices. Here is how I perceive the way each practice deals with leadership.

1. XP

XP has the role of a coach, but this role is not necessarily considered a leadership role. I perceive that XP is in part a reaction against bad management, and thus there has been a reluctance to create a role that can degenerate into a bad manager. In addition, XP makes little distinction between an XP coach who is a consultant that comes from time-to-time to give advice, and a coach who is involved in the day-to-day activities of the team. XP expects a 'customer on site', and expects this customer role to provide all leadership in terms of definition of business needs and guidance of team priorities.

2. Scrum

Scrum has the role of a Scrum Master, a person who organizes the daily Scrum meeting, runs interference with management, and helps break down barriers. Outside consultants are not generally considered Scrum Masters, they assist Scrum Masters. A Scrum Master is not allowed to help the team figure out what should be done, and does not serve as a technical lead. Scrum also has a role called a product owner, generally outside the team. This person (and the product owner is expected to be one person) defines backlog priorities.

3. DSDM

DSDM identifies specific roles, a key one being the technical coordinator. The technical coordinator is responsible for system architecture and technical quality, as well as basic practices such as configuration management. This role is quite different from the coach of XP or the Scrum Master of Scrum, where technical leadership is left to the team as a whole. DSDM also has a role called 'Visionary', someone similar to Scrum's Product Manager, but possibly less involved. DSDM also has an Ambassador User role, a role similar to the 'customer on site' role in XP.

4. Crystal Methods

I can't say much about Crystal Methods as I am not very familiar with them.

5. FDD

FDD has the following roles:

Project Manager – Administrative lead, responsible for securing funding and other resources and for reporting progress. This role is not like a Scrum Master, but has some similarities.

Chief Architect – Responsible for system design, but more by providing guidance than direction. Similar to the technical coordinator of DSDM.

Development Manager – Responsible for day-to-day development activities. I'm not really sure what this person does, and do not see an equivalent in other agile ecosystems.

Chief Programmers – Provide technical leadership to small teams. Sort of like mini-chief architects for sub-portions of a large project.

Class Owners – Each class is owned by a single developer, who makes all changes to it.

Domain Experts – Well, every ecosystem has to have users.

6. Lean Development

Lean Development looks for several kinds of leadership – Master Developer (similar to the Chief Architect of FDD or the technical coordinator of DSDM), expertise leadership (leadership in specific technical areas such as GUI design, database development, security, etc.) and project leadership (similar to the role of a Scrum Master).

7. Adaptive Software Development

Adaptive software development promotes the leadership-collaboration model as opposed to the command-control mode. Leadership-collaboration focus is on work states rather than process, on creating a collaborative environment, and on creating accountability for results. I'm not sure how to compare this to other roles in other ecosystems.

Conclusion:

True to form, the higher ceremony methods are also more formal on defining leadership roles. The methods which focus more on self-directing teams seem to be more adverse to defining leadership roles, probably worrying that a leader will interfere with team decision-making. I think that there is a lot to be learned from operations management, which has developed good theories on how to balance team independence with good leadership. A few lessons on how to effectively manage a high volume call center or a large retail store might do us all some good.

Mary Poppendieck