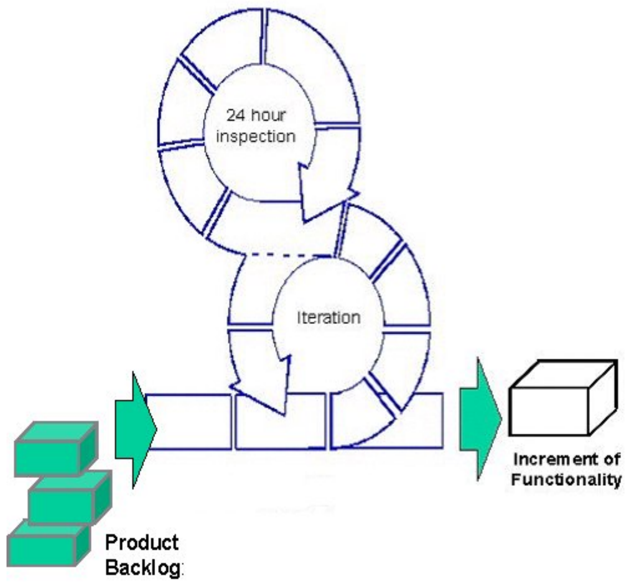
Are Agile Methodologies Really Different?

Ken Schwaber, ADM, for OOPSLA 2003

Agile processes have a common basis in that all of them accept the premise that software development is complex and that mechanisms other than predict and hope are required to manage the complexity. A common mechanism for addressing the complexity is iterative, incremental development, which I refer to as the agile skeleton. All agile processes share this skeleton, fleshing it out with different practices based on the orientation of the originators of the process and the domain in which it is mostly applied.



Scrum is based in empirical process control. The three mechanisms applied within Scrum are visibility, inspection, and adaptation. The skeleton supports these. A prioritized, emerging list of requirements is kept visible so that everyone understands what will be done next. Every twenty-four hours the work of the team is made visible and inspected by the entire team at a short meeting called the Daily Scrum. At the end of each iteration, the product, or increment, of the iteration is made visible and inspected by all stakeholders at a Sprint Review meeting. Every time an inspection is made, adaptation is called for. At the Daily Scrum, the team members adapt to each other’s progress. At the Sprint Review, the stakeholders adapt to the progress represented in the increment by selecting what they want the team to do next (stakeholders indicate what to do next, the team chooses how much of the what it believes it can turn into an increment).

Visibility implies that everyone knows and agrees upon what they are looking at. This introduces the concept of sashimi, a slice of the whole equivalent in content to all other slices of the whole. For the Daily Scrum, the slice of sashimi is a report that something is done. “Done” implies accepted engineering practices that indicate that done means coded. Or maybe it means coded, unit tested, checked in, built, and acceptance tested. Either way, there must be a common understanding; otherwise mistakes may be made as various team members inappropriately adapt based on incorrect assumptions of what they inspected. Similarly, the increment created every iteration must be well-defined and similar every time. In Scrum, each increment is an increment of potentially shippable product functionality. If a stakeholder inspects demonstrated functionality, they can presume that it is fully developed, clean, refactored, fully tested, and with only a small amount of packaging can be implemented.

The last aspect of agile processes is the heart. This occurs within the skeleton’s iteration. At the OOPSLA’02 workshop on commonality of agile processes, we decided that creativity, the ability to figure out the right thing to do and then do it, is common and the heart of all agile processes. Rather than someone else telling the team what to do and how to do it, the team is free to devise the most appropriate engineering practices and approaches to turning requirements into functionality.

The skeleton queues up the most important work for the business every iteration, maximizing the return on investment of the project. Sashimi ensures that progress is inspectable. The heart maximizes the productivity of turning the requirements into functionality.

I’ve been conducting training sessions over the last six months during which I help people who understand Scrum, Extreme Programming and other agile processes to understand the nuances and ins-and-outs of Scrum. I have been struck that the people who know XP through years of practice see how complimentary Scrum and Extreme Programming are. Both are iterative and incremental, both aim at improving the lives of the team members, and both focus on improving the quality of the product. Extreme Programming has even adopted the daily standup from Scrum’s Daily Scrum meeting. People have told me that they see Scrum as the management approach to agile development and XP as the engineering practices that make it effective, both bonded together by complimentary practices and goals.

The differences that I’ve noted between the two as I’ve explained Scrum to XP users are:

1. Scrum is more coarse grained. The Product Backlog items are coarser than stories, and the iterations are coarser at thirty days than more XP iterations. The reason for Scrum’s coarse grained practices is to keep everything in easy terms for the user, stakeholders, and customers to understand. XP stories are more specification and feature oriented, Scrum Product Backlog is more user functionality oriented.

2. Scrum requires the team members to learn how to talk to the customer without involving technology. XP seems to require the customer to learn how to talk to the developers in terms of specifications.
3. Scrum’s teams are more self-organizing and self-managing. The ScrumMaster is not allowed to tell the team how to do its work, whereas the Coach is much more active in directing the team’s work.
4. Scrum is much more the art of the possible, where customers, stakeholders and users are expected to inspect what a team has been able to do and make the appropriate adaptations. XP focuses more on the precision of estimating with “yesterday’s weather” and velocity, looking to show the customer that they can be trusted to know what can be delivered. Scrum delivers what it cans and places the responsibility on management to then figure out what to do.
5. Scrum can be implemented in days, and XP takes longer.
6. The name Scrum is just kind of gross, but the name “Extreme Programming” is frightful.

I’m sure that the workshop will uncover thousands of other variations and similarities, but overall I find their consonance to be greater than the discord. Accordingly, whenever I have a situation in which software is being developed, Scrum is being used, and the engineering practices are substandard, I also call for XP to be implemented.