St.Xavier’s College

Maitighar, Kathmandu



**SOFTWARE ENGINEERING**

Lab Assignment #6

|  |  |
| --- | --- |
| Submitted By: | Submitted To: |
| Aishwarya Rai | Mr. Balkrishna Subedi |
| BIM 6th Semester | Lecturer |
| Sec: A Roll:02 | Department of Computer Science |

**AGILE METHOD**

In software application development, agile software development (ASD) is a methodology for the creative process that anticipates the need for flexibility and applies a level of pragmatism into the delivery of the finished product. Agile software development focuses on keeping code simple, testing often, and delivering functional bits of the application as soon as they're ready. The goal of ASD is to build upon small client-approved parts as the project progresses, as opposed to delivering one large application at the end of the project.

**Agile methods are lightweight software development processes**

Although Agile methods rely on well-defined processes (change, risk, analysis, testing, etc.) and plans, they highlight that everything can be easily changed if the needs arise. In other words, the underlying processes are not edged in stone and can be adapted to the reality at any time.

**The downsides of Agile methods**

* Even more than with non-Agile methods, the success of the project heavily depends on the motivation, know-how, team abilities, and common sense of team members. This is a direct result of empowering team members to make decisions that can deeply affect the project without going through “committees” and “boards”.
* The success of Agile methods depends on its fit with the organizational culture.
* Agile methods are often perceived as synonyms to cowboy-style, opening the door to excess of creativity. Although this is (generally) a misconception, it means that team members have to be (re-)educated concerning the Agile methods before starting the project.
* Agile methods lack credibility for senior management. As a result, it may require strong negotiation power and increased risks for the project manager to use Agile methods for strategic projects. Most companies will only allow Agile methods for small exploratory projects at first. CIO magazine has a [good real-world story](http://www.cio.com/article/112151/How_One_CIO_Was_Converted_to_Agile_)about a CIO being converted to Agile.
* Agile methods may not be appropriate for large-scale projects (this is an open topic).

**Rapid application development** (**RAD**)

**Rapid application development** (**RAD**) is both a general term used to refer to alternatives to the conventional [waterfall model](https://en.wikipedia.org/wiki/Waterfall_model) of software development as well as the name for [James Martin](https://en.wikipedia.org/wiki/James_Martin_(author))'s approach to rapid development. In general, RAD approaches to software development put less emphasis on planning and more emphasis on process. In contrast to the waterfall model, which calls for rigorously defined specification to be established prior to entering the development phase, RAD approaches emphasize adaptability and the necessity of adjusting requirements in response to knowledge gained as the project progresses. Prototypes are often used in addition to or sometimes even in place of design specifications.

## SCRUM

Scrum is a subset of Agile. It is a lightweight process framework for agile development, and the most widely-used one.

* A “process framework” is a particular set of practices that must be followed in order for a process to be consistent with the framework. (For example, the Scrum process framework requires the use of development cycles called Sprints, the XP framework requires pair programming, and so forth.)
* “Lightweight” means that the overhead of the process is kept as small as possible, to maximize the amount of productive time available for getting useful work done.

A [Scrum process](https://www.cprime.com/2015/03/5-tips-to-manage-scrum-processes-in-the-real-world/) is distinguished from other agile processes by specific concepts and practices, divided into the three categories of Roles, Artifacts, and Time Boxes. These and other terms used in Scrum are defined below. Scrum is most often used to manage complex software and product development, using iterative and incremental practices. Scrum significantly increases productivity and reduces time to benefits relative to classic “[waterfall](https://www.cprime.com/2011/01/integrating-waterfall-and-agile-development-hybrid-model/)” processes. Scrum processes enable organizations to adjust smoothly to rapidly-changing requirements, and produce a product that meets evolving business goals. An agile Scrum process benefits the organization by helping it to

* Increase the quality of the deliverables
* Cope better with change (and expect the changes)
* Provide better estimates while spending less time creating them
* Be more in control of the project schedule and state

### Scrum Roles

Scrum has three roles: Product Owner, Scrum Master, and Team.

The Example Scrum Master’s Checklist

* **Product Owner:** The Product Owner should be a person with vision, authority, and availability. The Product Owner is responsible for continuously communicating the vision and priorities to the development team.

It’s sometimes hard for Product Owners to strike the right balance of involvement. Because Scrum values self-organization among teams, a Product Owner must fight the urge to micro-manage. At the same time, Product Owners must be available to answer questions from the team.

* **Scrum Master:** The Scrum Master acts as a facilitator for the Product Owner and the team. The Scrum Master does not manage the team. The Scrum Master works to remove any impediments that are obstructing the team from achieving its sprint goals. This helps the team remain creative and productive while making sure its successes are visible to the Product Owner. The Scrum Master also works to advise the Product Owner about how to maximize ROI for the team.
* **Team:** According to Scrum’s founder, “the team is utterly self managing.” The development team is responsible for self organizing to complete work. A Scrum development team contains about seven fully dedicated members (officially 3-9), ideally in one team room protected from outside distractions. For software projects, a typical team includes a mix of software engineers, architects, programmers, analysts, QA experts, testers, and UI designers. Each sprint, the team is responsible for determining how it will accomplish the work to be completed. The team has autonomy and responsibility to meet the goals of the sprint.

**Extreme Programming (XP)**

Extreme Programming (XP) is a [software engineering methodology](http://www.selectbs.com/analysis-and-design/what-is-a-software-development-process), the most prominent of several [agile software development](http://www.selectbs.com/process-maturity/what-is-agile-modeling) methodologies. Like other agile methodologies, [Extreme Programming](http://www.selectbs.com/process-improvement/extreme-programming) differs from traditional methodologies primarily in placing a higher value on adaptability than on predictability. Proponents of XP regard ongoing changes to requirements as an often natural and often inescapable aspect of software development projects; they believe that being able to adapt to changing requirements at any point during the project life is a more realistic and better approach than attempting to define all requirements at the beginning of a project and then expending effort to control changes to the requirements.

XP prescribes a set of day-to-day practices for managers and developers; the practices are meant to embody and encourage particular values. Proponents believe that the exercise of these practices—which are traditional software engineering practices taken to so-called "extreme" levels—leads to a development process that is more responsive to customer needs ("agile") than traditional methods, while creating software of similar or better quality.

**XP values**

Extreme Programming initially recognized four values in 1999. A new value was added in the second edition of *Extreme Programming Explained*. The five values are:

* Communication
* Simplicity
* Feedback
* Courage
* Respect

