Software Engineering

Lecture 4

Agile Software Development

Dr. Obuhuma James

Description

This topic gives a detailed coverage of the agile methodology as applied to software development. Examples of software process models that embrace the concept of agile development are also covered. The benefits and downsides of the agile methodology are also covered. Finally, the process of managing agile projects has also been covered.

Learning Outcomes

By the end of this topic, you will be able to:

- Broadly describe the agile methodology as applied in Software Engineering.
- Describe examples of software process models that embrace the agile methodology from those that do not.
- Outline benefits and downsides of software process models that use agile methodology.
- Describe the process of managing agile projects.

Overview of Agile Methodology

The previous topic explored the plan-driven software development methodology which is one of the two software development methodologies. This topic focuses on the agile methodology as the most commonly used approach in the current world of rapid software development. Most of the existing software process models align to either of the two methodologies with a few instances of process models that are hybrid models embracing features of both methodologies.

The agile methodology interleaves software development phases giving room for backtracking. This is an exact inverse of the plan-driven methodology which is a traditional approach to software development that follows a set of predefined steps with no flexibility for backtracking. Deciding on the appropriate methodology to use for a given software development project depends on a number of factors as covered in the previous topic. However, just as a form of recap, the agile methodology is appropriate if:

- Incremental delivery strategy is used, where you deliver the software to customers and get rapid feedback from them before starting the next increment.

- The system can be developed with a small co-located team who can communicate informally.

This topic focusses on exploring the agile methodology by detailing its features, the software process models that embrace the agile approach, and finally, the benefits and downsides of the agile methodology.

Features of Agile Methodology

According to Sommerville [1], businesses in the current world operate in a global, rapidly changing environment. This makes it practically impossible for software developers to derive a complete set of stable software requirements. Thus, the initial requirements inevitably change because customers find it impossible to predict how a system will affect working practices, how it will interact with other systems, and what user operations should be automated. Probably after a system has been delivered and users gain experience with it that the real requirements become clear. Even then, the requirements are likely to change quickly and unpredictably due to external factors. Unfortunately, a software is likely to be out of date when it is delivered. This makes the agile methodology to come in handy as a software development approach best suited for rapid development. Figure 1 shows the process of agile development where the requirements engineering phase is fully interleaved with the design and implementation phase. This means that both in-phase and cross-phase iterations are a norm in the agile methodology.

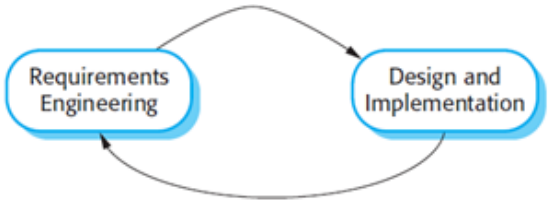

Figure 1. Agile Methodology [1]

Such an approach is geared towards production of useful software in an accelerated fashion where a software is developed and delivered as a series of increments. Each increment includes a new system functionality. This means that the requirements engineering phase does not have to result into a detailed system specification document. The user requirements document hence only defines the most important characteristics of the system. Similarly, the

design documentation is also minimized to a great extent. At some point automatic design documents generated by the programming environments are considered during system implementation. Thus, the system is developed in a series of versions with end users and other stakeholders being involved in specification and evaluation of each version. New requirements and changes to earlier versions are accepted and considered in later versions of the system.

The main principles driving the success of agile methodologies include: customer involvement, incremental development and delivery, focus on people and not processes, accommodation of changing user needs, and simplicity for both the software under development and the development process in use.

Agile Process Models

A number of process models can be categorized under the agile methodology depending on the nature of their system development process. This include extreme programming, feature driven development, scrum, lean software development, crystal, adaptive software development, among others. The following subsections will try to explore three of these models as a way of helping the learner to comprehend the agile nature of the models. These are the eXtreme Programming (XP), Feature Driven Development (FDD) and the Lean Software Development process models.

Extreme Programming (XP)

Extreme Programming, commonly abbreviated as XP, is an agile methodology suitable in software development where customers are constantly changing demands or requirements. It may also apply in cases where system's functionalities are not clear. XP promotes the involvement of frequent product release in short development cycles. This technique in return improves the productivity of the system and also introduces checkpoints that give room for easy improvement of system requirements. The customer is always the target in XP.

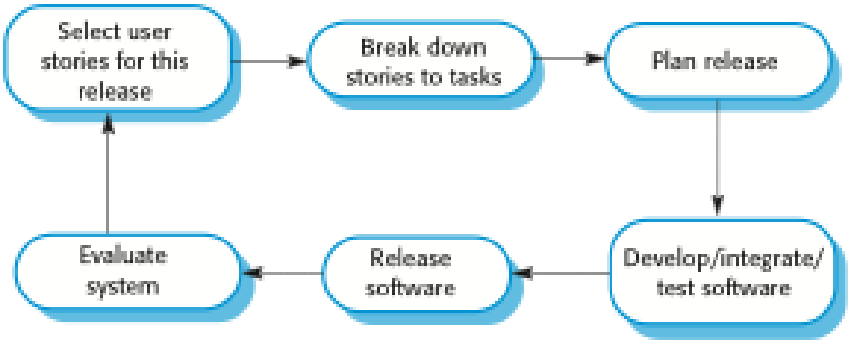

Figure 2. eXtreme Programming Process [1]

One unique attribute about XP is the fact that the customer is normally part of the XP team. He/she is responsible for decision-making on requirements. Based on Figure 2, the first step involves determination of user requirements that are normally expressed as scenarios or user stories. Each story is based on a release to be worked on. The stories are written on cards. This is followed by the development team breaking the stories into implementation tasks. These tasks end up being the basis of schedule and cost estimates during the third phase on planning of releases. Thus, the customer chooses the stories for inclusion in the next release based on their priorities and the schedule estimates. Once a task is selected for release, it transitions to the fourth step that involves development, integration and testing. The fifth and sixth phases involve release of the software and system evaluation respectively. The cycle is repeated for other user stories until the full system is developed and released.

Usually, software engineering aims at designing for change. Thus, spending time and effort towards anticipation of changes is worth in reduction of costs later in the life cycle. However, based on XP, since changes cannot be reliably anticipated, maintaining this is not worthwhile. Instead, constant code improvement (refactoring) is proposed as a way that could make changes easier if they must be implemented.

The eXtreme Programming approach relies on a number of principles, including,

1. Incremental planning
2. Small releases
3. Simple designs
4. Test-first development
5. Refactoring

6. Pair programming
7. Collective ownership
8. Continuous integration
9. Sustainable pace
10. On-site customer

Feature Driven Development (FDD)

Feature Driven Development focuses on "Designing and Building" of features where a feature in this case refers to a client-valued function of the desired system. FDD approach describes small steps of work to be done separately per feature (function). These have to be very specific and short phases of tasks to be accomplished within two weeks or less. Each phase includes elements of domain walkthrough, design inspection, promotion to building, code inspection and design with the following aspects in target:

1. Domain object Modeling
2. Development by feature
3. Component/Class Ownership
4. Feature Teams
5. Inspections
6. Configuration Management
7. Regular Builds
8. Visibility of progress and results

Feature Driven Development follows the following philosophy:

1. Underscores strong collaboration among team members
2. Handles problems and projects' complexities through feature-based decomposition followed by integration of software increments
3. Emphasizes on technical communication achieved through verbal, graphical, and textual means
4. Stresses on software quality inspired through incremental development, design and code inspections, quality assurance audits, metric collection, and use of patterns (analysis, design, construction)

Lean Software Development

The process model is anchored on the principle of "Just in time production" whose aim is to increase the speed of software development as you decrease the cost. The following seven steps summarizes the Lean Software Development process:

1. Elimination of waste
2. Amplification of learning
3. Deferment of commitment (deciding as late as possible)
4. Early delivery
5. Empowerment of the team
6. Building Integrity
7. Optimization of the whole process

Benefits and Downsides of the Agile Methodology

Agile methods embrace incremental development and delivery strategies in which the increments are small and, typically, new releases of the system created and made available to customers every two or three weeks. Customers are fully involved in the development process as a way of soliciting for rapid feedback on changing requirements. Furthermore, agile approaches minimize documentation using informal communications rather than formal meetings with written documents. In summary, agile methodologies work well under the following conditions:

1. Frequent changes are required.
2. A highly qualified and experienced software development team is available.
3. The customer is ready for full involvement including meeting with the software development team all the time.
4. Project size is small.

In the long runs, agile methodologies guarantee on frequent delivery of software increments, face-to-face communication with customers, efficient design and fulfilment of business requirements, flexible accommodation and incorporation of changes and finally reduction of total development time.

It is worth noting that agile methodologies that use the rapid development strategy are not suited for software development processes that plan on completely specifying the

requirements and then designing, building, and testing the system at the end. Hence, some application types like safety-critical control systems, where a complete analysis of the system is essential, a plan-driven approach is the best. In contrary, in a fast-moving business environment, this can be problematic in some way since in an ideal case, by the time the software is available for use, the original reason for its procurement may have changed so radically that the software is effectively useless. Thus, business systems are better suited for the agile methodologies.

Unfortunately, due to the minimization of documentation, confusions are inevitable. Furthermore, critical decisions taken throughout various phases can be misinterpreted at any time by different team members. Moreover, upon project completion with the development team transitioning to another project, system maintenance sometimes becomes a challenge.

It is worth noting that most software development strategies employed in the current world are hybrid approaches. This means that elements of both plan-driven and agile methodologies are involved. This helps in complementing deficiencies of either approach with the aim of bringing a balance between meeting the ever-changing customer requirements and sticking to predefined process steps.

Agile Project Management

Agile projects are normally managed using the scrum process model. Scrum is one of agile software process models whose focus is on managing the iterative development approach rather than specific agile practices [1, 2]. Figure 3 summarises the scrum process.

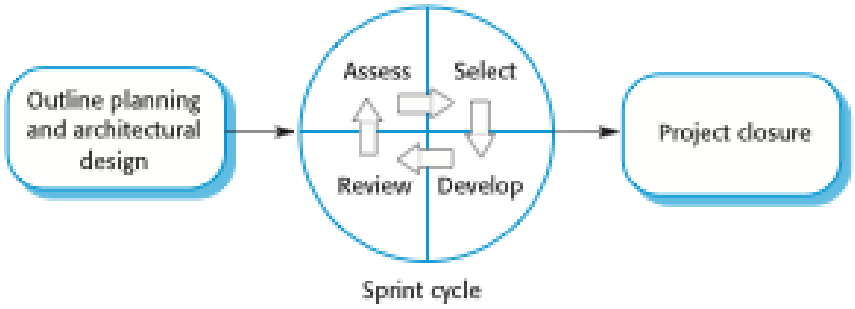

Figure 3. The Scrum Process [1]

The scrum process is anchored on three main phases as shown in Figure 3:

1. The outline planning phase where the general objectives for the project and design the software architecture are defined.
2. A series of sprint cycles with each cycle geared at developing an increment of the system. These is accomplished through increment assessment, selection, development, and review activities.
3. The project closure phase that wraps up the entire project, completes required documentation such as system help frames and user manuals and assesses the lessons learned from the project.

According to Pries and Quigley [2], both the customer and developers can use the scrum approach to reduce the risk caused by midstream changes in the specification of the product. The scrum is not only iterative but has actions that require team learning and allows the rest of the project to be built on such learnings. Schwaber [3] summarises the entire scrum process using Figure 4.

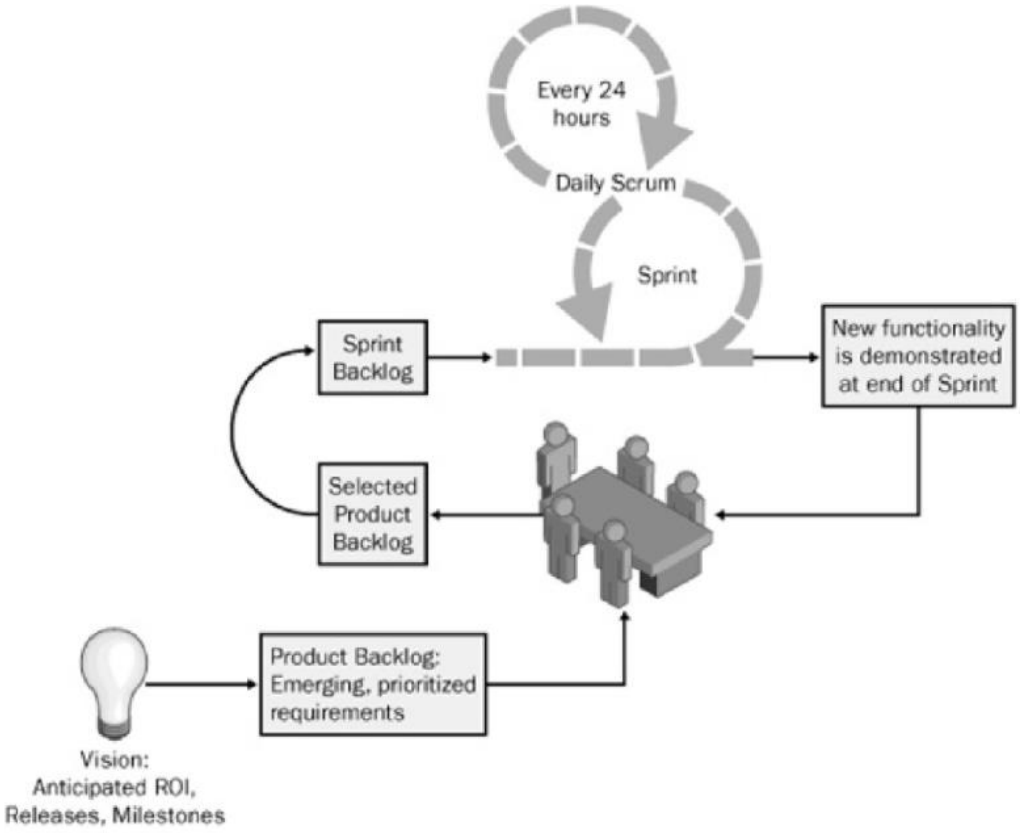

Figure 4. A Summary of the Scrum Process [3]

According to Figure 4:

- A backlog is a prioritized list of requirements or features that provide business value to customer. Items can be added to the backlog at any time with the selection phase involving the entire project team including the customer.
- Sprints are fixed length work units, normally 2 – 4 weeks, required to achieve one of the backlog items.
    - The development team is isolated from the customer and the organization, with all communications channeled through the scrum master, whose role is to protect the development team from external distractions.
    - At the end of the sprint, the work done is reviewed and presented to stakeholders by the scrum master. The next sprint cycle then begins.
- 15 minutes daily scrum meetings are organized by the scrum master as a way of tracking the backlog of work to be done, making decisions, measuring progress against the backlog. For better synchronization of operations, the entire development team attends the daily meetings with a focus on trying to answer the following questions:
    - What was done since last meeting?
    - What obstacles were encountered?
    - What will be done by the next meeting?
- The final stage delivers software increment to customer for evaluation in form of demos.

In the long run, the scrum approach brings forth the following benefits [1]:

1. The product is broken down into a set of manageable and understandable chunks.
2. Unstable requirements do not hold up progress.
3. The whole team have visibility of everything and consequently team communication is improved.
4. Customers see on-time delivery of increments and gain feedback on how the product works.
5. Trust between customers and developers is established and a positive culture is created in which everyone expects the project to succeed.

Summary

The topic has expounded on the agile methodology for software development. This was achieved by exploring the concept of agile development that is more flexible compared to the plan-driven approach. The eXtreme Programming (XP), Feature Driven Development (FDD) and the Lean Software Development have been covered as examples of process models that embrace the agile approach of software development. The topic has also clearly outlined the pros and cons of the agile approach as a way of determining the suitability of the approach in software development. Finally, the management of agile projects using the scrum process has been extensively discussed.

Check Points

1. Discuss the salient features of the agile methodology.
2. State and explain what makes the eXtreme Programming approach to be categorized as an agile process model.
3. State and explain what makes the Feature Driven Development (FDD) to be categorized as an agile process model.
4. State and explain what makes the Lead Software Development to be categorized as an agile process model.
5. Discuss any two benefits and downsides of the agile methodology.
6. Discuss the scrum approach as used in agile project management.

Learning Resources

Core Textbooks

1. Sommerville, I., Software Engineering, 10th Edition, Addison Wesley, 2016.

Other Resources

2. Pries, K. H., & Quigley, J. M. (2010). Scrum project management. CRC press.
3. Schwaber, K. (2004). Agile project management with Scrum. Microsoft press.

References

[1] Sommerville, I., Software Engineering, 10th Edition, Addison Wesley, 2016.

[2] Pries, K. H., & Quigley, J. M. (2010). Scrum project management. CRC press.

[3] Schwaber, K. (2004). Agile project management with Scrum. Microsoft press.

[4] Gotterbarn, D., Miller, K., & Rogerson, S. (1997). Software engineering code of ethics. Communications of the ACM, 40(11), 110-118.

[5] Pressman, R. S. (2005). Software engineering: a practitioner's approach. Palgrave macmillan.

[6] Kendall, K. E. and Kendall, J. E., Systems Analysis and Design, 8th Edition, Pearson, 2011.