Agile Development

Lucas Vandermaarel

Software Engineering Technology, Centennial College

COMP120: Software Engineering Fundamentals

Ehsan Ullah

October 10, 2024

**Agile Development**

Agile development is a commonly used flexible method to develop software. Agile development has a focus on collaboration, quick delivery of components, and adaptation. The key aspects of agile development are iterative development, continuous feedback, cross-functional teams, customer centricity and development flexibility.

Iterative development is the idea that projects are split up into small chunks called sprints. These sprints have action items assigned that a team would be tasked to complete. Sprints typically last one to four weeks and can be thought of as a self-contained mini project that fits into the overarching project. (Brush, K., & Silverthorne, V. 2022, November 15)

Continuous feedback is the part of agile development that involves communication with stakeholders and end-users. The stakeholders and end-users provide feedback during and after each sprint to allow for modifications throughout the development process. These modifications can be small, large, or anywhere in between, effectively changing the original idea into a similar product than first thought or creating a product embodies a complete pivot in another direction. (Hoek, J. van der. 2024, September 10)

Cross-functional teams is the idea that agile development teams are small, between five and ten members, that can work on the current objectives without management intervention. The members would be diverse in the skills that they bring to the team to tackle any challenge that they come across during any given sprint. (Brush, K., & Silverthorne, V. 2022, November 15)   
For example, a team member could be a strong developing back-end developer of programs, another member could be strong at developing the front-end, a dev-ops engineer could be used to build the continuous delivery and continuous integration workflow to speed up the overall speed at which new features are implemented and pushed to production, and a database engineer could make sure the database has security features implemented along side creating the tables necessary to store important data to the functions of a piece of software. Since each member would have experience in each of the other members’ scope of work, they would be able to do their main task, but help another member troubleshoot issues when they arise.

Having customer centricity means prioritizing the distribution of a given feature quickly and on a short schedule. This schedule tends to be a sprint length as that is when a feature would be developed. Presenting a new feature to end-users quickly creates a focal point on the development of software which forces the documentation process to be concise. By focusing on the developing of software over extensive documentation, the team can provide more value to end-users quicker. This can be important in retaining customers as they will notice the value being added with frequent software updates and notice documentation less as it may not pertain to their use cases.

In agile development, flexibility is very important. The idea is that agile teams are adaptable to any changes presented to them. Changes can arise from the continuous feedback they receive from stakeholders and end-users as well as from technical obstacles they come across during development. (Cio. 2023, June 2) Some technical obstacles could be unforeseen latency issues by which the code goes from one component to the next, the underlying hardware running the software is not powerful enough to support the computations, and team members could come and go. The team has to reevaluate their position as these changes arise and adapt to overcome and complete their sprints.

There are many different frameworks that could be used to implement these aspects. Some of the most notable are Scrum, Kanban, Extreme Programming, and Lean Software Development.

**The five framework activities**

**Communication**

The communication activity in agile development involves the team and stakeholders to converse about any new or existing requirements as well as create project goals and constraints. The communication activity also entails consistent dialogue between the team, any clients involved, and end-users. The format of an agile communication activity is commonly seen during daily standups, sprint planning meetings, and stakeholder reviews.

**Planning**Planning is done at every governance level and ranges from high-level strategic planning to highly detailed sprint planning and can even go down to as specific as one team member’s exact steps to complete a piece of a given sprint. The planning activity includes defining the project’s scope, the objectives and/or milestones of the project, estimating the time needed to complete the project, and identifying risks alongside mitigation measures to deal with issues. Often, the objectives and milestones would be combined with timelines to create a roadmap that keep end-users in the loop of features to come and provide a quick overview to the stakeholders about what to expect.

**Modeling**The modeling activity is a much more visual activity than the other activities as there are user interface designs, data models and database schemas prototyped, as well as system architecture diagrams created. User interface models entail the layout and interaction of a piece of software. Data models will be comprised of specifications on how data should be held within a program and database schemas will specify how data is stored. System architecture diagrams will display the entire software’s functionalities and features that are originally thought of. All these models may change as the software’s requirements change.

**Construction**Construction is where code is being written and workflows are being developed. This entails writing code, creating unit and integration tests, reviewing code, refactoring code to optimize the performance. Due to continuous integration processes, over time the construction activity will become more streamlined which in turn may increase efficiency when building features.

**Deployment**

Deployment is when the continuous deployment practices would likely be used to frequently release updates to production instances. Some actions that would be taken are the preparation of software, the configuration of the production environments, migrating data, creating documentation, and possibly training users. The preparation of the software might entail last minute testing and formatting the code to be more readable. The configuration of production environments might mean setting up SSL certificates and private keys to secure a web application or it could mean setting up the project on different virtual machines that point to a separate, secure IP outside of the organization’s LAN. The migration of data could mean transferring a database or other resources the piece of software needs to function into a production database or storage container. Creating documentation would be the team providing specifications on how the software should be used and training users would be similar in the information that is presented to provide users with the ability to use the software as it was intended.

**Software Engineering Practice and General Principles**

**Modularity**

Modularity is the idea of splitting software components into management components that can be tested and maintained independently from one another. Reusability of code and troubleshooting capabilities are improved as a result of implementing this principle.

**Abstraction**Abstraction is the principle where complex processes are hidden behind easy to use methods. Abstracting complexities allows developers to use code without understanding how it works at the lowest level. This will save time when there is a need to use a similar complex process in multiple areas of a code base.

**Encapsulation**Encapsulation puts data together into all-encompassing units, typically an object. For example, a student object might have data regarding the student’s grades, homeroom teacher, sex, and name. Calling a student object will provide quick and easy access to the attributes assigned to it while allowing the developers to understand what they are working with based on the name. Encapsulation will help reduce the complexity of a program and increase data integrity by giving contextual meaning to data.

**Information Hiding**Information hiding is to restrict data or how a method operates from the end-user. This increases the security of the software and reinforces the modular design of an application.

**Don’t Repeat Yourself (DRY)**Don’t Repeat Yourself, also known as the acronym DRY, is about avoiding duplication of code when possible. This reinforces modularity again as methods that follow the DRY principle are typically created with a focus on being general enough to be used in multiple locations and processes within an application.

**Test-Driven Development**Test-driven development is the idea that tests are written before the functional code is. This will help with keeping the functional code correct in regard to expectations.

**Code Reviews**Code reviews are about examining prospective code changes before they are included in the main code base. Code quality is maintained by having multiple developers use their various levels of knowledge to provide feedback and share knowledge.

**Version Control**Version Control is when software is tracked with each change to parts or the entirety of a code base. Developers can look back through old versions of their code base and revert to past versions if needed easily through the usage of tools such as Git. Code bases can then be tracked remotely when connecting a code repository using Git to a platform such as GitHub or GitLab.

**Process Models and the difference between them**

**Waterfall**The waterfall model is a linear sequential approach to developing software. There are unique stages that a waterfall method is comprised of.

The stages are:

* Gathering of requirements and analysis
* Design
* Implementation
* Testing
* Deployment
* Maintenance

Being a linear sequential model, each stage must be completed before the next stage can be started. This requires a lot of planning and documentation initially.

The advantages of the waterfall model are that it is straightforward, can be effective for smaller projects that can be defined extensively, and there are clear milestones that can be strived for.

The disadvantages of this model are that there is not much room for change after the requirements are set and most stages will have to be fully completed before a working piece of software is available for stakeholders to see and test. These disadvantages bring higher risk than alternative methods.

**Incremental**The incremental model effectively miniaturizes the waterfall model and replicates it for each stage in the model. This results in a functional product after each stage that will have improvements or modifications since the end of the last stage. A notable difference between the waterfall model and incremental model is feedback can be incorporated after each stage to influence the decisions of subsequent stages.

The advantages of the incremental model are that it allows developers to push out new features quickly, it is more flexible to requirement changes and can be efficiently tested due to each feature being smaller than a full project.  
The disadvantages of the incremental model include a large amount of planning and designing, the costs could be higher than the waterfall model, and efficiency could depreciate if the code base is not well maintained.

**Evolutionary**Evolutionary models have a focus on iterative development and improvement based on feedback from end-users. There are specific models such as the Spiral model and Prototyping that reside under this overarching model type. The evolutionary model has cycles of development and evaluation of the features and direction of the software. Risk analysis and disaster mitigation are priorities during the frequent development of prototypes and/or production versions of the project.

The advantages are that this model is very flexible to any changes necessary, involves the user early and often, and reduces risk through regular evaluations of the features.

The disadvantages are that the model can be difficult to manage as the features required expand beyond the capabilities of the team and a higher cost could be incurred due to the multiple iterations involved.

**Concurrent**

The concurrent model is a model that allows multiple components to be worked on simultaneously. Multiple teams would work on different components and be encouraged to have often communication between themselves to ensure that functionalities work smoothly with one another. This model is quite different from the rest as action items can be in sequential order or they can overlap.

The advantages of the concurrent model are that it can reduce the overall development time on a project, create more specialized teams, and is flexible in having different development approaches for features that any given team is responsible for.

The disadvantages are that there are a lot of moving parts at any given time which will require a strong project manager to coordinate everything and can become complex to implement features if teams are not on the same page. Integrations are likely to have many issues that need to be troubleshooted.

Overall, they complete the same steps, just in different orders and configurations. Some are slower to complete but have lower costs and lower complexity. Other models focus on the speed at which features are sent out and will incur higher costs and complexity for this higher speed of development. The Waterfall model might be better to use with small teams while incremental and evolutionary would fit well with small, medium, and large teams. The Concurrent model would be best if a large company has many people to develop a product and splitting up the development of features would more effectively use their team member’s time.

References:  
Hoek, J. van der. (2024, September 10). *The 5 stages of the Agile Software Development Lifecycle*. Mendix. https://www.mendix.com/blog/agile-software-development-lifecycle-stages/

Cio. (2023, June 2). *Optimal balance between agility and stability in software development*. CioPages. https://www.ciopages.com/agility-and-stability-in-software-development/

Brush, K., & Silverthorne, V. (2022, November 15). *What is Agile Software Development (agile methodologies)?*. Software Quality. https://www.techtarget.com/searchsoftwarequality/definition/agile-software-development