2018

# Selecting a Software Development Methodology Based on Project Characteristics

Semir Music
*University of Northern Iowa*

# Selecting a Software Development Methodology Based on Project Characteristics


## Abstract

Methodologies used for developing software include procedures, techniques, tools, and documentation, to help every stage of the development process. A number of software development methodologies exist, all with their own characteristics, features, advantages, and disadvantages. This paper presents information on some of the more popular methodologies, and the ones most frequently used in today's software development world. The reviewed methodologies in this paper include waterfall, spiral, agile and its subsets, and hybrid. Research and information gathering phase in this study was conducted by performing a thorough literature review of a variety of different sources. Following the introduction of each methodology, a selection chart is presented that can help with selecting the most appropriate one to use, based on the project characteristics.

When going through the process of implementing a new software methodology, most companies will run into management and organizational issues, people-related issues, process-related issues, or technological issues. A number of different solutions to these challenges is presented including training, job rotation, additional classes, proper planning efforts, and use of a coach or a mentor. A variety of tools and techniques exists that make the software development process more streamlined, efficient, and easier to manage. The methodologies evaluated here include use of source code management tools, objected oriented programming, documentation tools, and automated testing including unit tests. Research shows that using a software development methodology, including ones that were presented in this paper, will increase the chances of the project being successful. The use of an adequate methodology plays an important role in developing software to assure that it is delivered within schedule, cost, and meets users' requirements.

# SELECTING A SOFTWARE DEVELOPMENT METHODOLOGY BASED ON PROJECT CHARACTERISTICS

**A Graduate Research/Project Paper**
**Presented to the Graduate Faculty**
**of the**
**Department of Technology**
**University of Northern Iowa**

**In Partial Fulfillment of the Requirements**
**for the**
**Non-Thesis Master of Science in Technology Degree**

**By**
**Semir Music**
**10/15/2018**

**Approved by:**

Lisa Riedle

4/26/19
**Signature of Advisor** **Date**

4/26/2019
**Signature of Second Reviewer** **Date**

## Abstract

Methodologies used for developing software include procedures, techniques, tools, and documentation, to help every stage of the development process. A number of software development methodologies exist, all with their own characteristics, features, advantages, and disadvantages. This paper presents information on some of the more popular methodologies, and the ones most frequently used in today's software development world. The reviewed methodologies in this paper include waterfall, spiral, agile and its subsets, and hybrid. Research and information gathering phase in this study was conducted by performing a thorough literature review of a variety of different sources. Following the introduction of each methodology, a selection chart is presented that can help with selecting the most appropriate one to use, based on the project characteristics.

When going through the process of implementing a new software methodology, most companies will run into management and organizational issues, people-related issues, process-related issues, or technological issues. A number of different solutions to these challenges is presented including training, job rotation, additional classes, proper planning efforts, and use of a coach or a mentor. A variety of tools and techniques exists that make the software development process more streamlined, efficient, and easier to manage. The methodologies evaluated here include use of source code management tools, objected oriented programming, documentation tools, and automated testing including unit tests. Research shows that using a software development methodology, including ones that were presented in this paper, will increase the chances of the project being successful. The use of an adequate methodology plays an important role in developing software to assure that it is delivered within schedule, cost, and meets users' requirements.

# Table of Contents

# Table of Figures

# 1 INTRODUCTION

Methodologies used for developing software can be defined as a framework, which includes procedures, techniques, tools and documentation, designed to help every stage of the development process. Software development revolves around a life cycle called software development life cycle (SDLC) (Mahanti & Jiju, 2005). SDLC is a framework that defines all the tasks and steps performed in software development. The goal of SDLC is to produce quality software that is released within the time and budget requirements, and that meets or exceeds the customer expectations. Generally, it consists of five different phases (Kaur, 2015).

I. Requirement Gathering and Analysis

II. Designing

III. Coding

IV. Testing

V. Maintenance and Support

Each phase produces deliverables that are required in the next phase, and are to be followed in sequential order. The requirements are gathered and analyzed in order to understand the problem at hand. During the design phase a plan is developed that will help solve this problem, and during the coding phase the plan is implemented. Testing puts the product through a series of tests that will make sure it performs all the functionality required. Deployment and maintenance are the last steps of the cycle.

This study reviews several different software development methodologies (SDM) that are available and most commonly in use today. The review starts with the traditional or the waterfall methodology then moves on to spiral, agile, and covers all the different subsets of agile development. After the initial review of the given methodologies, a general guide is presented

that can be used to assist with selecting the most appropriate methodology based on the project characteristics. Some additional information presented includes challenges that companies may face when trying to adopt a new methodology, ways to overcome those challenges, and some of the best practices used for developing software.

## 1.1 Purpose of the Study

The purpose of this study is to present and introduce some background information on different software development methodologies in use, including traditional or waterfall and agile. Then present information that can serve as a guide and a good starting point for individuals and organizations in determining which methodology to use and when to use it. The study consists of a thorough literature review and includes information on the different methodologies, their advantages and disadvantages, characteristics, and working principles of each.

The emphasis is on the project characteristics including schedule, project and team size, complexity, budget, risk, specifications and functionality, customer involvement, project documentation, and how these methodologies affect the methodology that's used. This is a very important thing to consider before the development starts, since there is not a one solution that fits best for every scenario or project. Since most organizations deal with a variety of different projects, the goal is to come up with a guide that makes this decision easier, especially in the beginning stages of the project. Software project failures usually result in projects that are delivered late, over budgeted, non-functional at the end of the software development life cycle, having bugs, missing features and requirements etc. The research presented answers the following questions:

1. *What are the different software development methodologies available?*
2. *Which one should be used based on the given project characteristics?*
3. *What are some of the challenges that an individual or an organization may face when implementing a new software development methodology?*
4. *What are some ways to overcome these challenges?*
5. *What are some of the best practices to use when developing software?*

Use of a methodology can serve as a road map, that makes the management of the project easier, and helps to avoid some common development problems. Adhering to a properly-defined methodology that enables a project to provide better estimates, delivers stable systems, keeps the customer informed, creates a clear understanding of the task ahead, and identifies pitfalls earlier, allowing for ample time to make adjustments (Faridani, 2011). Research also suggests that using a structured process, as in one of the suggested methodologies helps in reducing risk, and increases the chances of the project being a success.

## 1.2 Statement of Need and Justification

The topic selected for this research is based on the identification of gap in knowledge when it comes to the selection and adoption of software development methodologies, and some of the challenges encountered when going through the process. This knowledge gap was discovered by performing a thorough literature review, which did not turn up significant results, especially when it comes to the application of different methodologies based on the project characteristics. There are advantages and disadvantages to every methodology and a proper application for each. The research presented can serve as a guide for individuals and organizations to help make decisions when picking an SDM. This is especially important early

on in a process. Making the right call when selecting a methodology to use on a project is one of the first steps that will help organizations implement projects in a time and cost effective way, will reduce risk, and lead to a successful project execution.

## 1.3 Limitations and or Delimitations

Numerous software development methodologies exist, and because only a subset was selected, not all of them will be studied. Inclusion of additional software development methodologies would provide a more comprehensive guide, and give the best and most accurate results. Research presented here considers the most used ones in today's software development environment. Presented in this evaluation is a high level overview of selecting a methodology and this does not cover specific software coding practices, programming languages, different management strategies, and practices. Another limitation is that not all project characteristics are studied, and items here concentrate on the extreme values. For example, in case of project complexity, only simple and highly complex projects are considered.

## 1.4 Definition of Terms

**Agile Manifesto:** Formal publication of four core values and 12 principles to guide the iterative agile approach to software development. The four core values include:

1. Valuing individuals and interactions over processes and tools
2. Working software over comprehensive documentation
3. Customer collaboration over contract negotiation

4. Responding to change over following a plan.

**Agile Methodology:** Instead of sequential process flow, agile uses incremental process design. A very basic or simple project design is the starting point; work is broken up into modules, and then the developers work on these in weekly or monthly increments. At the end of an increment, project evaluations are performed, and the work for the next increment is planned out.

**Class-Responsibility Collaboration (CRC):** A tool that is used as part of Object Oriented Programming (OOP) to help identify classes and objects, identify their responsibilities, and list collaborators or other objects that are associated with them.

**Coach:** An individual who is familiar with agile and scrum practices; is there to help the team adopt and improve their agile process; and leads the efforts as the rest of the development process unfolds.

**Development Team:** The development team works on the backlog items assigned by the product owner. They are in charge of developing, testing, running, and executing the entire project's code.

**Empirical Process Control Theory:** Theory which states that knowledge comes from experience, and that decisions are made based on experimentation and observation, and not on upfront planning. This concept relies on three main ideas, which include transparency, inspection and adaptation.

**Extreme Programming (XP):** Subset of agile software development. The focus with XP is to get a limited working version of software developed as fast as possible, after this is done then additional features can be added.

**Heavy-weight or waterfall methodology:** Is a sequential software development design process in which steps of the software development life cycle are strictly followed in a specific order.

**Hybrid:** Combination of elements from different methodologies. Some organizations may choose this approach if none of the standard methodologies apply or work well in their case.

**Increment:** The sum of all the backlog items completed during a Sprint. This must be equal to a working product at the end of the Sprint cycle.

**Object Oriented Programming (OOP):** Programming model that is organized around objects and data as opposed to the traditional ways of programming that are based on actions and logic.

**Pair Programming:** Technique that is used in agile development where two programmers work together on the same project. One programmer will write code while the other reviews the code as it is written. The two programmers will switch roles throughout the process.

**Peer Reviews:** Software assessment/analysis in which the program written is usually reviewed by a team of developers. The review team usually consists of a few different individuals in addition to the main designer.

**Product Backlog:** Prioritized features list that contains a short description of all the features and functionality that are desired in a product. This list is a good starting point for the first sprint iteration and it will grow as more is learned about the product or the customers' needs.

**Product Owner:** Usually the project's key stakeholder. The product owner is responsible for the project success and is responsible to the team, stakeholders and the company.

**Rapid Application Development (RAD):** Software development methodology with a focus on rapid prototyping and development in order to ensure faster delivery of products.

**Scrum:** Subset of agile or in other words, one of the most popular agile methodologies.

**Scrum Artifacts:** Represent the work and value in a multitude of ways that are used to reach transparency between team members and the work that they do. In addition, they provide opportunities for inspection and adaptation. There are three scrum artifacts including product backlog, sprint backlog, and increment.

**Scrum Master:** One of the members of the agile team. Scrum Master is responsible for making sure that the rest of the team abides by the scrum values and practices. The Scrum Master is in charge of doing tasks are needed in order for the team to perform at their highest level.

**Software Development Methodology (SDM):** Set of rules and guidelines that are used in the process of researching, planning, designing, developing, testing, setup and maintaining a software product

**Spiral:** Software development methodology that is a variation of the heavyweight methodologies, and is risk driven.

**Sprint:** Basic unit of development in Scrum. Sprints are of a limited duration and typically last anywhere from a week to a month. Another word for sprint is an iteration. Sprints begin with a planning exercise or a meeting where Sprint backlog work items and work schedules are determined. Sprints end with reviews and retrospectives. The emphasis is placed on a working product at the end of the sprint.

**Sprint Backlog:** List of tasks and items that the Scrum team identifies, to work on, and complete during a Scrum sprint. The team selects the items from the product backlog during the sprint planning meeting. It is very important to have the team pick the size and select the items for the sprint because they will ultimately be working on them and are responsible for their completion.

**Team Foundation Server (TFS):** Source code management product used to manage an entire process of the software development life cycle.

**Unified Modeling Language (UML):** Tool that can be used to specify, visualize, modify, construct, and document a software development process from the beginning stages to a finished product.

# 2 SOFTWARE DEVELOPMENT METHODOLOGIES

## 2.1 Definition

According to Mahapatra & Goswami (2009), software development methodology is a formalized approach for the development of software. A software development methodology describes an environment that is used to organize, plan, and direct the process of developing a software system (Verma, Bansal & Pandey, 2014). Another definition by Liviu (2014) states that a software development methodology is a set of rules and guidelines that are used in the process of researching, planning, designing, developing, testing, setup, and maintaining a software product.

A number of different methodologies exist. While each contains its own characteristics and features, they all contain some basic stages of the software development life cycle. Each stage has its own deliverables and is bound by a specific time frame (Liviu, 2014). These stages are planning, analysis, design, implementation, and maintenance. Methodologies discussed in this evaluation include waterfall, agile, scrum, extreme programming, rapid application development, spiral, and hybrid.

## 2.2 Waterfall or heavy weight methodologies

Traditional ways of developing software are often referred to as the heavy weight methodologies or sometimes as plan-driven. As the name suggests the waterfall methodology is a sequential process in which the steps are followed in a specific order. Using this methodology requires a lot of planning and project managers are very observant and stay in control of the whole process by using better planning strategies, constant reports and updates, frequent checkpoints, and meetings. The waterfall methodology is predictable and values rigorous software planning and architecture (Liviu, 2014). As shown in Figure 2-1, the development process consists of five different phases: requirements, design, implementation, testing, and maintenance.

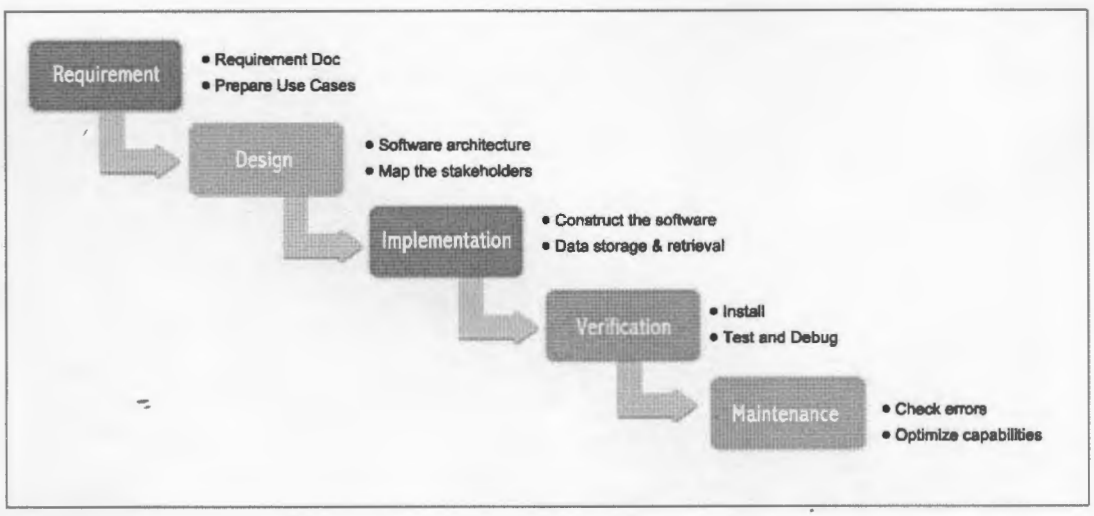


*Figure 2-1 Waterfall Development Methodology.* Reprinted from *DeveloperSourceCode*, by Admin, 2018, Retrieved from www.developersourcecode.net/2018/09/20/waterfall-software-development-methodology/.

Waterfall methodology works well until there are changes that happen. For this reason, teams that use them will try and resist any changes in the specification, design, or requirements. The steps of the SDLC are followed in strict order. Once a step is completed, only then can the process advance on to the next step, and at no point is there going back to the previous step. Another important characteristic is that a functioning version of software is not produced until

the life cycle is complete. This level or rigidity is a limiting factor and in today's fast changing world it is, in many cases, a deciding factor for companies working on figuring out what methodology to use.

Waterfall has been in use for years and many successful applications of its use show that it's a viable option. One of the main advantages of this methodology is that it's very rigid in nature, which makes it easy to use and understand. Because the steps are well documented, it is easy to visualize and understand the key concepts and points of every step of the process. Some additional advantages include: ease of making estimates concerning project budget and deadlines, well suited for mid-size to large projects, and frequent progress reviews. Some of the drawbacks or disadvantages of using it include: costly changes to the process once it is started, strict definition of requirements that must be defined before the project starts, and changes are discouraged. Because Waterfall methodology is sequential, it is not possible to go back to the previous step of the process. Software is complete and ready to use only at the end of the cycle. Project integration is done at the end which does not provide enough time if issues or problems are encountered.

Waterfall methodology is well suited for critical projects where safety and security are very important. Some examples include: automation projects, manufacturing equipment, flight controls, antivirus software etc. The strict nature of the process ensures that the requirements are well defined from the beginning, everything is followed in the specific order, and finally well tested before releasing.

## 2.3 Spiral

The main focus of the spiral model is on the identification and reduction of risk associated with a project. As a starting point the project is explored on a small scale, risks are examined along the line, a plan is generated to handle the risks, and then based on the risk level it is decided if the next step of the process should be taken. In this case, taking the next step would mean performing another iteration of the spiral. A software project passes through the steps repeatedly in iterations corresponding to various spirals in the model (Kaur, 2015). The constant reduction of risk has a positive effect on the project life cycle, and the overall delivery.

Spiral methodology is a modification of Waterfall which includes the concept of prototyping. It is best suited for complex, large, and expensive projects. The spiral process starts with a detailed definition of the system requirements. The requirement gathering step includes data from external and internal users and any additional system specifications. Following the requirements gathering step, a preliminary or starting design is created. Next is developing a prototype using the preliminary design. A prototype is typically a scaled down version of the finalized product that includes most of the functionality. The second prototype is created by evaluating the first prototype using the following criteria:

1. Examining the first prototype and its risks, strengths, and weaknesses
2. Defining the new requirements for the second prototype
3. Planning and designing the second prototype
4. Constructing the second prototype

The project can be scrapped at any point if it is determined that the risk is too great to continue. Some of the reasons for scrapping the project could include exceeding the budget limits,

miscalculations in effort required, time requirements, etc. Another prototype is created using the previously mentioned evaluation criteria. The process is repeated until the model becomes a fully functional application that meets all the project owner's requirements (Liviu, 2014). Only after all steps are completed the final product is developed. Complete system evaluation and testing followed by maintaining the product are the last steps of the process. These steps are shown in Figure 2-2.

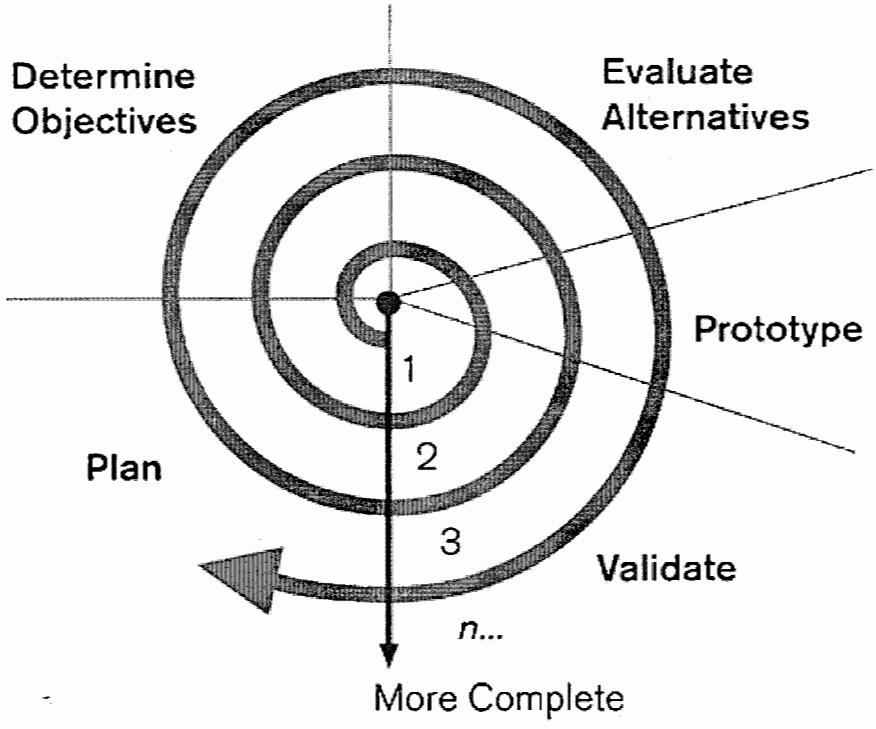


*Figure 2-2 Spiral Software Development Methodology.* Reprinted from *WXEC*, by N.A., 2018, Retrieved from www.wxec.info/spiral-software-development-process.html.

Spiral methodology is best applied in large scale products that are used in critical applications. One of the main advantages is that functioning version of software exists early on in the process. In addition to this, a significant amount of risk analysis is performed throughout the process. This process makes it easier to evaluate if the project is worth continuing or if it's almost guaranteed to fail and should be abandoned early on before too much time and money is

wasted. Some of the disadvantages of the spiral methodology include a higher cost to implement, and the fact that it's not meant for smaller projects. Moreover, the success of the project is highly dependent on the risk analysis phase. Being familiar with risk analysis is a requirement in Spiral analysis.

## 2.4 Agile

Contrary to the heavy weight methodologies, which like to plan everything in advance, are agile or lightweight methodologies, which focus on incremental and iterative development in order to enable changes to the requirements at any stage during the software development. Agile methodologies generally promote a project management process that encourages stakeholder involvement, feedback, objective metrics, and effective controls (Dyba & Dingsoyr, 2009). Agile methods can be described by a formula of deliver today and adapt tomorrow (Highsmith, 2010). Agile Manifesto was published in 2001 as the first official presentation of the Agile Software Development. Agile manifesto consists of the following statements:

*We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:*

*Individuals and interactions over processes and tools*

*Working software over comprehensive documentation*

*Customer collaboration over contract negotiation*

*Responding to change over following a plan*

*That is, while there is value in the items on the right, we value the items on the left more.*

Using agile, software is developed in small iterations. After every iteration customer is able to see the progress and the results, and can provide instant feedback. The iterations typically

last anywhere from one to two weeks and generally should not exceed four weeks. Every step of the SDLC is contained within each iteration as shown in Figure 2-3. This means that during every iteration a set of requirements is selected, completed, implemented, tested, and if necessary the product could be released. After an iteration is complete, if there were any problems discovered, they are reviewed by the entire team and the appropriate changes to the functionality are made. Agile software development is largely dependent upon the agile team, which is a cross-functional group of individuals responsible for defining, building, and testing the solution software (Kaur, 2015). Once an iteration is complete, a new set of requirements are selected and the next iteration can begin.

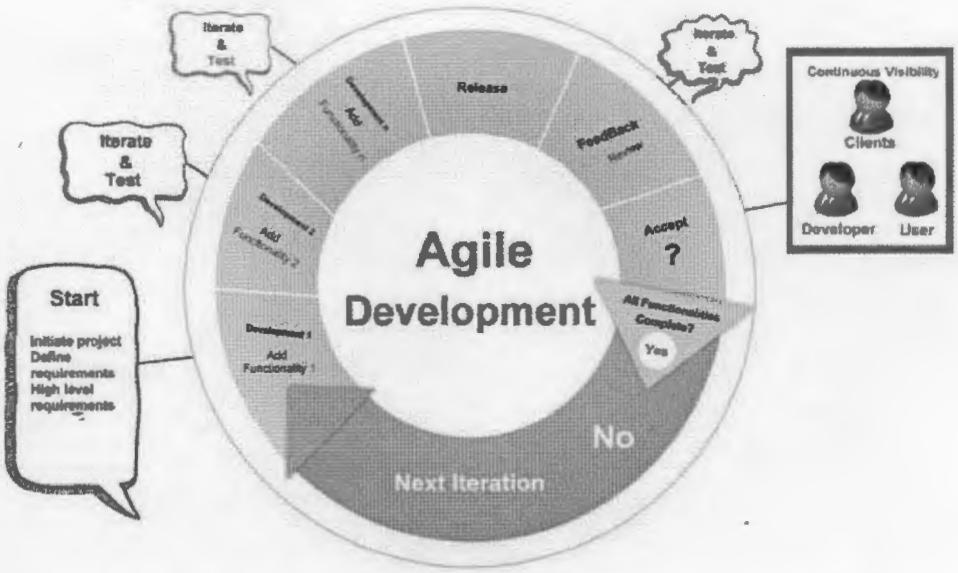


*Figure 2-3 Agile Development Methodology.* Reprinted from *Home Security*, by N.A., 2018, Retrieved from www.homesecurity.press/quotes/building-in-parallel-lines.html.

The focus of agile is to address the customer needs in the best way possible. Transparency exists throughout the product development cycle and the customer or end user involvement is encouraged during every phase. According to Edeki (2015), in addition to the customer, agile focuses on including all of the key stakeholders throughout the entire software

lifecycle. Agile methodologies provide a number of benefits and advantages to the organizations using them, some of which are mentioned below. Because the process is iterative, incremental changes are made, and the customer feedback is always available, it is easier to make changes to requirements or include new ones. Some additional advantages of agile methodology include rapid response to change and higher product quality. Testing is integrated into every iteration and issues can be fixed sooner rather than later. Because work is broken up into smaller chunks, automated tests can be used. Collaboration and working directly with the customer lead to a higher customer satisfaction, increased product control and transparency, greatly improved software delivery times, and reduced risk.

One major drawback of using agile is that there is less predictability, which makes it more difficult to estimate the amount of effort and time required. Since there is uncertainty and unknowns, teams that are new to agile methods are more likely to make poor decisions, and use incorrect practices, all of which contribute to a less successful project. The process can be time consuming because the customers are always involved and a constant communication is kept up resulting in more demands being placed on both the developers and end users. Lack of documentation makes it difficult to bring new people up to speed. Not enough planning and working under the assumption that customer requirements are likely to change creates a less structured process. If the communication channels with the customer are not clear, the developers could be focusing or working on incorrect or incomplete requirements.

Agile is well suited for projects in which the scope is not well defined, where requirements can change at any point, and where levels of uncertainty are high. The nature of agile allows projects using it to quickly adapt and overcome these challenges. There are about a

dozen different agile methodologies used. The most popular ones including scrum, extreme programming (XP), and Rapid Application Development (RAD). Each are reviewed below.

### 2.4.1 Scrum

Scrum encourages iterative as well as incremental work and is the most popular way to implement agile methods. Here the focus is on delivering the highest value to the customer in the shortest time possible. Empirical process control theory is a core Scrum principle and is supported by three pillars. First is transparency, meaning that the process must be visible to all of those who are responsible for the outcome, inspection, and the approval of the deliverables. The adaptation which is the process of applying what was learned during the transparency and inspection, is the goal of improving the process or the work done.

Scrum teams consist of the product owner, the development team, and lastly the scrum master. The product owner is in charge of increasing the productivity and maximizing the work done by the development team. Another role that the product owner is in charge of is managing and keeping track of the backlog, which is the list of all items that need to be worked on and completed. The backlog list consists of requirements, issues, bugs, tasks etc. The development team works on the backlog items assigned by the product owner. Scrum development teams are self-organized and task assignment is a process where every team member is involved (Liviu, 2014). The division of work and the organization is up to the team and managed without any external intervention. A Scrum Master is in charge of ensuring that the team is functioning properly, keeps the team aligned with the scrum principles, and makes sure that the whole process goes smoothly.

Scrum artifacts are a set of tools and a source of information that everyone involved including the scrum team, and the stakeholders need to be familiar with. This will help better understand the product that is being developed, and the activities that are performed and planned during the development process. There are three scrum artifacts: product backlog, sprint backlog, and increment. Product backlog includes all the changes, features, functions, requirements, and issues that need to be made to the product in future releases. Product backlog items have attributes that describe each work item in terms of importance, description, value, effort required etc. Product backlog items can be updated at any time, and more details can be added that will enhance the quality and help others better understand what needs to be done. For each task, the developer should be required to make an estimation of how long they believe they will need to complete the task. After the task is completed, they should enter in the actual time spent on the task (Edeki, 2015). This will help the developer keep track of time, track performance, and will eventually lead to more accurate estimates.

Sprint backlog is a sub-set of product backlog items that are selected for any given Sprint. The items selected for the Sprint will vary depending on what the goal is, and what features and functionality is needed in the next increment of code. Increment is the sum of all the backlog items completed during a Sprint that must equal a working product at the end of the Sprint cycle. The main scrum event is the Sprint. It is a time duration of a month or less in some cases, in-which a working version of the product is released. Right after one Sprint finishes, another one begins, and this way the product that is under development is advanced every iteration. The Sprint consists of the Sprint Planning, Daily Scrums, development work, Sprint Review and Retrospective (Schwaber & Sutherland, 2013). The scrum process is shown in Figure 2-4.

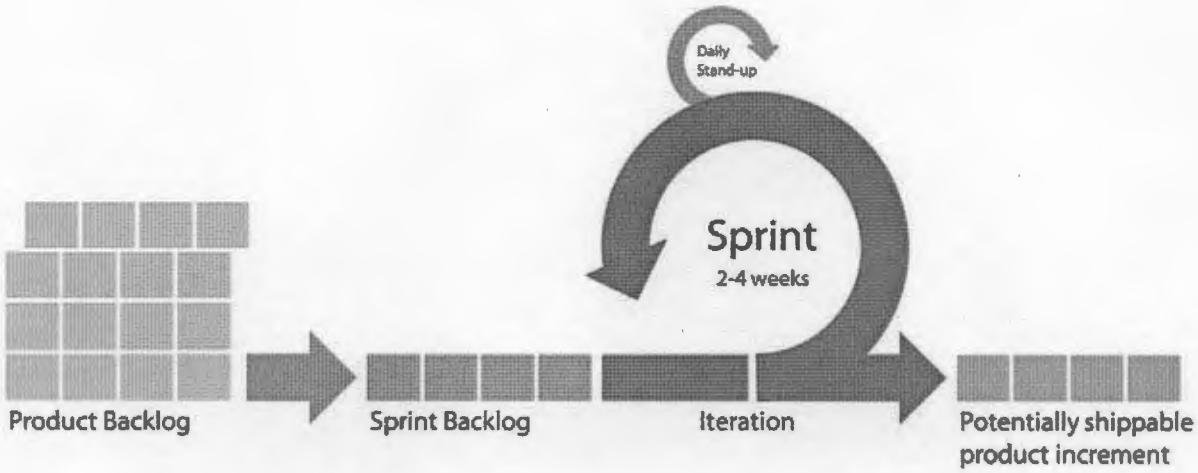


*Figure 2-4 Scrum Development Methodology.* Reprinted from *Aperigae*, by N.A., 2018, Retrieved from www.aperigae.com/tr/urun/30/software-and-system-development-services.

Scrum is best suited for new projects that have no maintenance issues so the team can focus on working and adding new features during every sprint, small teams are available, software needs to be delivered on a regular basis, and projects that need constant feedback. One of the main advantages of scrum is shorter delivery times. Testing is conducted during and throughout the process leading to a higher quality product. Scrum also allows for change of scope and pace at any point and increases process visibility. Daily meetings help identify and resolve any problems early on. A significant disadvantage of using Scrum is that it can be difficult to bring new people in, and to get them up to speed. There is a potential for issues with scope creep because there are no defined project deadlines. A few other ones include: lack of documentation, difficult to estimate costs, requires strong commitment, and experienced team members.

### 2.4.2 Extreme Programming (XP)

Extreme programming (XP) is an agile software development methodology with the main goal of producing high quality software and delivering it within a very short timeline. Using XP,

significantly reduces the SDLC process time. One of the characteristics of XP is that it allows frequent releases in a relatively short amount of time and allows the developers to respond to changes in requirements. Some of these characteristics may sound familiar because they're part of other agile methodologies, but as the name suggests XP takes these to another level.

In XP, the team only works on the requirements assigned, and in most cases these are the most crucial ones. Team collaborates and works together on every step of the process; starting with the requirements and coding all the way to implementation. Good interpersonal skills and trust are very important characteristics for a successful XP team (Dyba & Dingsoyr, 2009). According to Fojtik (2011), there are five values that are part of XP including communication, simplicity, feedback, courage, and respect. Communication is an essential component and means that everyone is part of the team and communicates and works on all activities daily. Simplicity refers to working on only the assigned tasks and nothing else. Feedback consists of discussing the project and demonstrating the work completed during every iteration of the process. Every team member is valued and respected and their input is encouraged. Finally, courage means that the progress is reported honestly without hiding anything during the process. Changes are likely to happen and proper ways to adapt to them need to be implemented.

Just as was the case with agile and waterfall methodologies, XP has steps that are followed that lead to a completed product. The XP process consists of planning, managing, designing, coding, and testing. The basic steps of the XP cycle are shown in Figure 2-5. During the planning stage, user stories are created, which are definitions of a requirement that are used to estimate the effort required to complete the said requirement. At this same time, the release schedule is created and the project is split up into iterations. Managing phase consists of creating

a productive work space for the team, selecting a team by moving people around and assigning tasks, setting a good working pace, and holding short stand up meetings every day.

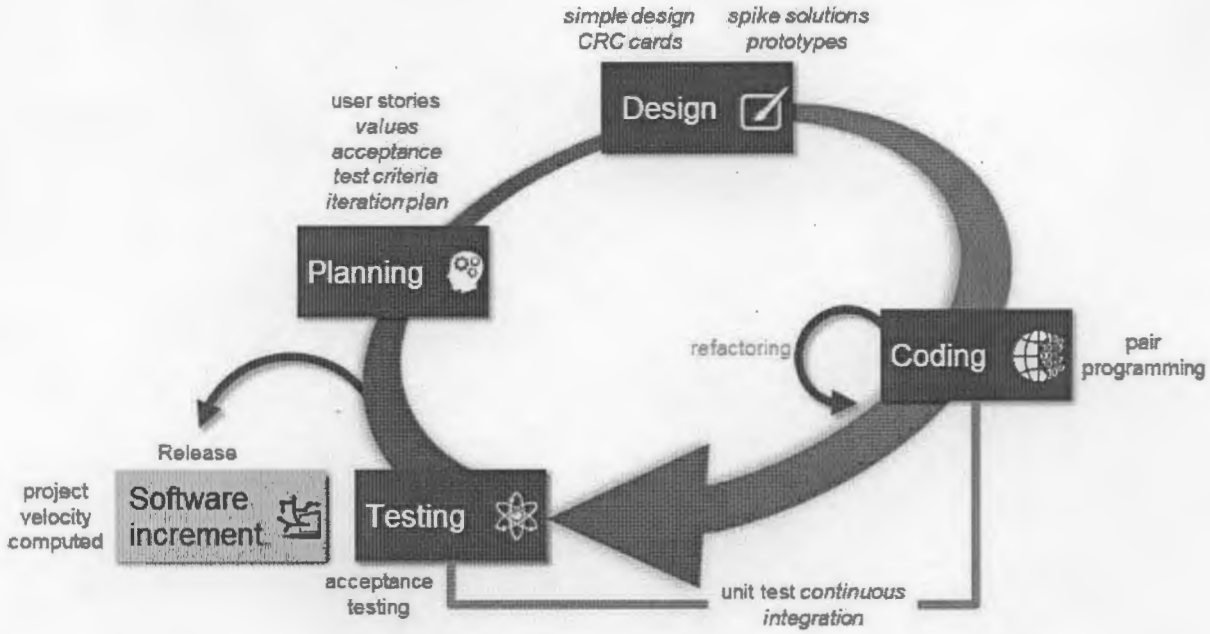


*Figure 2-5 Extreme Programming (XP) Development Methodology.* Reprinted from *SlideModel*, by N.A., 2018, Retrieved from www.slidemodel.com/templates/tag/extreme-programming/.

During the design phase, simplicity is one of the most important factors. Simplicity means that the team only works on the assigned items and everything else is left for another iteration. Class-responsibility-collaboration (CRC) cards are used as a brainstorming tool to help in writing the code. For example, a CRC card contains a portion of the name of the object in code, its responsibilities or functions, and finally a list of objects that it interacts with to fulfill its responsibilities. Experiments are conducted that help in reducing risk, refactoring of code or restructuring without affecting functionality is done whenever needed. Refactoring helps with readability and other individuals to better understand the code.

Completion of the design leads to coding which is the next phase of the development cycle. During this phase the customer is always available and can provide quick input on status

of the product. Code is written per company standards, and unit tests are usually written first. All of the coding is pair programmed, meaning that two people work on the same piece of code together, one writing and one reviewing. Integration of code is done often, so no code is lost, and integration is easier to manage. The final state of the process consists of testing the code. Every bit of code has unit tests associated with it which test the complete functionality of the code by itself and then as a completed product.

One of the main advantages of using XP is that it removes unneeded activities from the process such as long meetings, paperwork, and the need for separate testing. This makes the process a lot more efficient and results in significant cost savings. Division of work into tasks and iterations, spread of responsibility, and use of pairs of programmers serve as a way to lower risk. XP is a very resilient methodology and can accommodate changes very quickly. Development is test driven which leads to a robust end product. XP can also be used on projects with no clear initial definition, where the customers do not have a clear idea about the output product (Fojtik, 2011). Disadvantages include: difficulties with implementation, requires constant involvement from the customer, and commitment and discipline on the developer side. Because the process is code centric it can be difficult to manage when it is used on large projects. There is a possibility of having to do significant code refactoring if XP is to be used on an existing project. XP does not work well for teams that are not in the same geographic location.

A use of a Coach is highly recommended for a team just starting out with using XP. The Coach will have experience using the methodology, can help the team transition, and avoid any mistakes. XP is best suited for applications where the functionality is expected to change every few weeks to months, the team is working with customers who are unsure of what they want,

single projects that are worked on by one team, and teams that want to minimize the risk and are able to work closely with the customers.

### 2.4.3 Rapid Application Development (RAD)

Rapid Application Development (RAD) uses a condensed process cycle to produce a high quality finished product in the shortest amount of time. The software is divided into smaller components, which facilitates making changes throughout the development process (Geambasu, Jianu & Gavrila, 2011). The four phases of RAD, as shown in Figure 2-6, are requirements planning, user design, construction, and cutover. During the requirements planning phase workshops and focus groups are used that help gather and best define all the needed requirements. Workshops and focus groups are important tools and help in speeding up the process. Once the requirements are known, the user design and construction phases can occur. These two phases are repeated as many times as needed until the end user or customer accepts that all the functionality is present. During cutover, final steps of the cycle are performed, which include: implementation, final stages of testing, and user training.

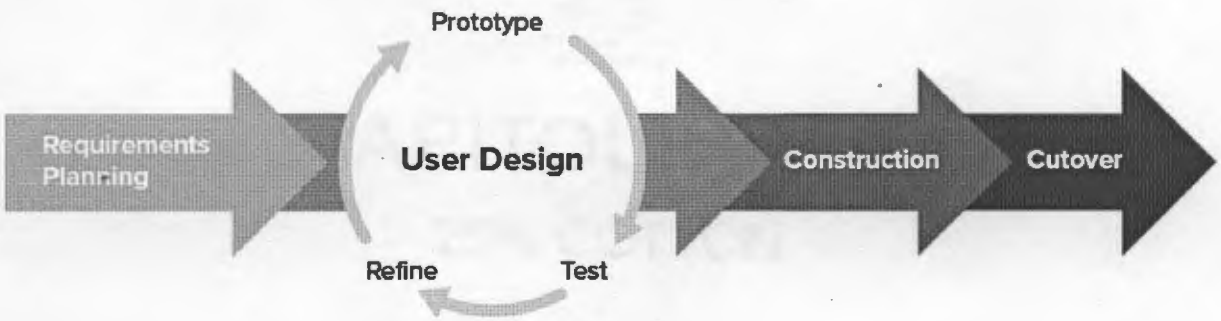


*Figure 2-6 Rapid Application Development (RAD) Methodology* Reprinted from *Thewilcoxgroup*, by L.Harding, 2018, Retrieved from https://www.thewilcoxgroup.com/rad-project-management/4-phases-of-rapid-application-development-methodology-lucidchart-blog/

One of the ways that RAD makes the process of development more efficient is by reusing software components (Geambasu, et al., 2011). During the development, programmers make use of object oriented programming (OOP) in which code is broken up into small logical components or objects. Each of these objects has data associated with it and some action or functionality that it performs. By making these objects generic, it is possible to re-use them in future projects that require the same or similar functionality. Another method that RAD uses in order to increase efficiency involves the use of prototypes and having users test out some of the code prior to delivery. To keep the schedule timeline short, any project improvements are delayed until later versions. Lastly, review meetings and communication that occurs between team members is informal.

If a project is not very complex, objectives and the user base are clearly defined, then RAD is a good methodology to use. It is best suited for small to medium projects that need to be completed in a short amount of time. One significant advantage of using RAD is that it breaks up the project into smaller manageable tasks. Efficiency is increased by assigning tasks based on user experience and areas that they're most familiar with. Some additional advantages include: short delivery times for a working product and increased efficiency of the design process due to constant communication with the customer. Disadvantages that teams may encounter include dependence on the team for performance, should not be used on a project with a small budget, and reduced scalability. Short schedules result in removing features and pushing them out to later versions. The developed system will have less features than in the case of using structured methodologies, because of delivery deadlines (Geambasu, et al., 2011).

## 2.5 Hybrid

Once familiar with the different methodologies available and their characteristics, teams may choose to use a hybrid methodology. Some organizations go this route because the given methodologies don't apply to their process. In some cases it is best to consider the advantages of all the methodologies and combine them into one. This does require more work and forces an organization to come up with their own standards and practices that have to be managed and documented. Use of hybrid is certainly an option that allows a lot of freedom and flexibility. A prerequisite to using a hybrid methodology is an understanding of the methodologies available and when each is most applicable. A drawback here is that if a hybrid model is created that works well for a given project, it does not mean that it's automatically a good fit for a different project. Modifications may be needed, and another evaluation may be required that could suggest that another methodology is a better fit. According to a study conducted by Kuhrmann (2018), hybrid development approaches have become a reality for nearly all companies. They are applied to specific projects even in the presence of company-wide policies for process usage. Use of hybrid methodologies is neither planned nor designed but emerges from the evolution of different work practices. Finally, they are consistently used regardless of company size or industry sector.

# 3 PROJECT CHARACTERISTICS

Sotware projects have certain characteristics that should be evaluated before any work is started. The following list consists of characteristics that have the greatest impact on project performance. Depending on the characteristics, a proper methodology is selected that will

produce the best results. Project characteristics should be reviewed during the project proposal phase, and proper metrics need to be evaluated that will determine the value of each. Most of the characteristics are self-explanatory and with the intent of keeping the research evaluation simple, only the extreme values for each category are examined. For example, when looking at the project size, small and large projects are considered.

**Schedule** – Consists of the project milestones, activities involved, and finally the deliverables associated with the project. Every activity will have a start and finish date. Accumulation of these activities will produce the finish date of the project. Two categories considered in this paper are: short duration and long duration projects.

**Project Size** – Size of the project which combines the project duration, team size, and budget. Small and large projects will be considered.

**Team Size** – Size of the team assigned to work on the given project or in some cases available to work on a project. Small teams usually consists of up to ten individuals, and larger are ten and up.

**Complexity** – Used to determine the project complexity. Does the project require research and components that have not been used before? Some of the other considerations include: skills and experience of current workers, possibility of hiring outside help, collaboration with another organization in order to accomplish the goal. Highly complex and simple are the two categories.

**Budget** – An estimate of all the costs associated with running and completing a project and this includes cost to complete the tasks, labor costs, operation costs, management, travel, deliveries etc. Small and large budgets are evaluated.

**Risk** – Level of risk associated with taking on the project. If a project is similar to one that was done in the past, the risk level associated with it will typically be low. If proper documentation exists, the completed project is reviewed to see if any of the areas could be improved, and those changes are applied to the new project. If a project is completely different from anything that was done in the past the risk will be high.

**Specifications and Functionality** – Are the specifications and functionality well defined? Does the customer know what they want, or will they change their mind half-way through the process? Two categories looked at here will be: well defined and likely to change.

**Customer involvement** – Is the customer available and involved during the process? Or do they send a specification document and stand by until the project is completed? Categories: involved and available to provide constant feedback, not involved.

**Project Documentation** – Used to document the steps of the software development process. It is used for training purposes, makes it easier to bring new individuals on a project, can help the customers get familiar with the product and how it works, and aids in troubleshooting. Process is either well documented or lacks documentation.

# 4 SELECTING A METHODOLOGY

Each model has certain conditions of use, advantages and disadvantages depending on the project characteristics and circumstances; thereby, each model's effectiveness varies with project characteristics and other factors affecting the project, such as enterprise environmental factors. Because of this, the most effective methodology needs to be selected for each project that an organization takes on (Medvedska & Berzisa, 2015). Selection of the correct methodology is one

of the first steps that will lead to a successful project. According to Boyd, 2009, use of an incorrect or ineffective SDM can adversely affect how the project is managed and have a negative impact on project quality. Furthermore, adopting an incorrect methodology could result in slippages, lack of communication and administrative overheads, leading to customer dissatisfaction (Rajagopalan & Mathew, 2016).

Because there are many different methodologies available and in many cases with similar or just slightly different characteristics, the selection process can be a daunting task. The role of the project manager is to help the organization in implementing the given strategy, while working on minimizing the risk, and ensuring that the most effective and efficient methods are used. To make the right selection, the project manager has to have a deep understanding of a wide range of methodologies and processes and know when each can have the greatest positive outcome.

The selection process begins and is usually completed before any actual work on the project is done. This consists of both external meetings with the customer, or the end user, and internal meetings including the team members and other members of the organization. During the internal meetings, the information and specifications gathered from the customer are reviewed, and appropriate project characteristics are developed. The table shown in Figure 4-1, can be used to select a methodology based on the previously identified project characteristics. The table was developed based on the research information just presented, by performing a thorough literature review, and researching the methodologies, their characteristics, and advantages and disadvantages of each.

| Project Characteristics | Waterfall or Heavy Weight | Spiral | Agile | Scrum | Extreme Programming (XP) | Rapid Application Development (RAD) | Hybrid |
|---|---|---|---|---|---|---|---|
| **Schedule** | Longer project | Longer | Short | Longer | Short | Short | Variable |
| **Project Size** | Large projects | Larger projects | Smaller projects | All sizes of projects | Smaller projects | Smaller projects | Variable |
| **Team Size** | Large teams, because members are not interchangeable | Large team consisting of experts in the field | Smaller, well coordinated teams | Good fit for all sizes. Scrum of scrum concept implemented | Smaller teams, usually less than 20 team members | Smaller teams to medium | Variable |
| **Complexity** | Not for complex projects | Appropriate for complex projects | Appropriate for complex projects | Appropriate for complex projects | Appropriate for complex projects | Not appropriate for complex projects | Variable |
| **Budget** | Not very costly | Not costly | Costly, usually hard to estimate | Costly, usually hard to estimate | Costly, usually hard to estimate | Costly, usually hard to estimate | Variable |
| **Risk** | Inappropriate for high risk projects | Appropriate for high risk projects | Appropriate for high risk projects | Appropriate for high risk projects | Appropriate for high risk projects | Appropriate for high risk projects | Variable |
| **Specifications and Functionality** | All requirements are specified at the beginning, and not changing | Not all requirements specified and frequently changing | Not all requirements specified and frequently changing | Not all requirements specified and frequently changing | Not all requirements specified and frequently changing | Not all requirements specified and frequently changing | Variable |
| **Customer Involvement** | Not involved, only at milestones | Customer is involved | Involved, after each iteration | Customer is involved through the role of product owner | Customer is involved | Customer is involved | Variable |
| **Project Documentation** | Well documented | Lack of documentation, only basic | Lack of documentation, only basic | Lack of documentation, only basic | Lack of documentation, only basic | Lack of documentation, only basic | Variable |

*Figure 4-1 Software Development Methodology Selection Chart*

Figure 4-2 shows additional information on the given methodologies, presents some of the special features that each methodology possesses, and finally summarizes advantages and disadvantages of each. Implementation and success of a hybrid methodology will depend on the application and the decisions that an organization makes during the design process. The special features, advantages and disadvantages will also vary depending on the choices that an organization makes.

| Project Characteristics | Waterfall or Heavy Weight | Spiral | Agile | Scrum | Extreme Programming (XP) | Rapid Application Development (RAD) | Hybrid |
|---|---|---|---|---|---|---|---|
| **Specialties** | Rigid, well documented process | Prototyping, risk management techniques | Iterative and incremental development | Sprint, and product backlog, scrum master. Iterative and incremental, testing | User stories, refactoring, Class-responsibility-collaboration, unit testing | Object oriented programming (OOP), prototyping | Variable |
| **Advantages** | Easy to use and understand, easy to estimate time and budget requirements, process well documented | Can be used in larger projects, critical applications, risk management | Feedback is always available, testing is integrated into every iteration, increased project control and transparency | High level of collaboration, effective communication | Collaborative workplace, customer is part of the team, feedback available instantly, well developed and defined practices | Breaks up project into smaller manageable tasks, increased efficiency, short delivery times, | Variable |
| **Disadvantages** | Costly changes once started, software only ready at the end of cycle | Higher cost to implement, not for smaller projects | Less predictability, hard to estimate effort required, lack of documentation | Control over project, and lack of documentation | Customer presence is required, lack of documentation | Short schedules result in missing features, project success depends on teams performance, costly | Variable |

*Figure 4-2 Software Development Methodology Specialties, Advantages, and Disadvantages*

Using the table in Figure 4-1 can help make the selection of an SDM an easier task. As mentioned earlier, the requirements and project characteristics have to be well defined before the process can continue. Following this, the entire team examines the characteristics and compares them to the two tables above. Methodologies that are not a good fit or do not apply are eliminated, that leaves the team with a more manageable list to select from. When there are only a few methodologies remaining, each is looked at and examined in greater detail, and either one is selected or different elements of the remaining methodologies are combined. This combination would result in a hybrid methodology, which is a preferred solution for some projects. In this case, it is important to examine the elements from each methodology and understand how they may work when combined. This examination helps to avoid any possible conflicts down the line.

# 5 CHALLENGES ASSOCIATED WITH APPLYING A SOFTWARE DEVELOPMENT METHODOLOGY

Software development is a complex activity consisting of tasks and requirements that may exhibit a high amount of variability. Changes to the development process or adoption of a new one are usually significant and the tools and techniques used in each are not directly replaceable or necessarily compatible. According to Nerur, Mahapatra & Mangalara (2005) when migrating to a new methodology, most companies will run into management and organizational issues, people-related issues, process-related issues, and technological issues.

One of the biggest contributors to the organizational issues is the company culture. The values, norms, and assumptions of an organization are stabilized and reinforced over time and are reflected in the policies embodied in organizational routines (Nerur et al., 2005). Without the right company culture, any new initiatives or methodology implementation is not likely to succeed. Training will initiate this process of changing the company's attitude towards trying out and implementing new systems and methodologies. Resistance to change is a very important challenge that a company is faced with, and only through proper training and explanation of benefits, processes and better communication can this be overcome. Proper planning and execution are crucial as they will lead to a positive outcome, boost the team morale, and increase productivity. Having upper and senior management's support will help smooth over the transition process.

Depending on the methodology selected, the role of individuals involved will change. For example, while some of the more traditional methodologies encourage individual work, agile is geared toward team work and collaboration. The role of the project manager also changes

depending on the methodology used. In the traditional sense, a project manager is responsible for controlling and planning all of the project's activities but in agile, that role changes to the facilitator who is there to coordinate team efforts and to make sure everyone has input in the final decision. In Scrum, the manager needs to listen to the team and help them remove impediments (Friis, Ostergaard & Sutherland, 2011). One of the ways to overcome challenges related to roles of individuals is to get everyone involved familiar with different types of methodologies and activities involved in them. This could be accomplished by job rotation, switching tasks between individuals on a team, additional training, and classes. Some other things to consider include: number of projects happening at the same time, skills and experiences of employees, available resources, etc.

Proper planning efforts will help reduce the process related issues. Whether adopting a structured process as in one of the outlined methodologies, or changing to a different methodology, it is important to understand the steps involved and the amount of effort required. Once an initial assessment of suitability is completed, the methodology could be applied to a test project and results observed. Depending on the outcome of the test project, the results could be used to prove that the given methodology is a viable option, and at the same time used to gain company wide support for its adoption. Another way to overcome process related challenges when applying a methodology is to have at least one person who is very knowledgeable about the practices, serve as a coach and mentor to the rest of the group as it transitions. According to Hajjdiab & Taleb (2012), this is an essential role during the adoption process of agile, scrum, or any other process that's new to the organization. If this is not the case, and the team attempts to work on their own, there is a high chance that the transition or the implementation will not be successful.

It is important to remember that some of the methodologies mentioned do not place great emphasis on documentation and are flexible in nature. At the beginning stages, and especially when just starting a new project, documentation is crucial and is needed for tracking progress, reviewing results, and any future training efforts. Every methodology has its own tools and techniques that it uses. Tools play a critical role in successful implementation of a software development methodology (Nerur et al., 2005). In order to support the new methodology, an organization will need to upgrade some of the existing technologies and train their personnel on how to use them.

# 6 BEST PRACTICES FOR DEVELOPING SOFTWARE

A variety of tools and techniques exists that make the software development process more streamlined, efficient, and easier to manage. The ones evaluated here include use of source code management tools, objected oriented programming, documentation tools, and automated testing including unit tests. Team Foundation Server (TFS) from Microsoft is one example of a source code management product used to manage an entire process of the software development life cycle. At the beginning of the project, a type of development methodology is selected, and TFS provides a template that can be used to implement the methodology and create a more structured process. The template allows users to create and manage different types of tasks. These tasks are referred to as work items.

A work item can be a task, bug, a change request, feature, an issue, requirement, review, or risk. At the beginning of a project all the tasks, requirements, and features are listed out, and initially are marked "Proposed". The work item form also includes the size, priority, hours or

effort required, schedule, description of an item, comments etc. Work items can be assigned to different team members and iterations. When an individual is ready to start working on an item, they will mark it "Active" and then the work will start. When the item is completed, it will first be marked "Resolved", and then after it's been tested it will be "Closed". If during the testing process or at any other point in the development process problems are discovered, then additional work items such as issues or bugs will be added. Weekly meetings exist to the status of the work items, and it is determined if additional help is needed to get the work done on time. TFS provides a way to manage the entire process of software development, an easy to use interface, tracks changes to the source code and allows for easy status reporting.

The second technique that most organizations should adopt when developing software products is the use of Object Oriented Programming (OOP), which is supported by most of the modern languages in use today. OOP is a programming model consisting of objects, which contain data or attributes, and actions or methods that they perform. These objects can be integrated and work together to perform a task or to solve some problem. Breaking up code into more manageable objects not only makes it easier to understand buy also makes it easier to design, debug, test, and explain to others. One of the first steps conducted in OOP is data modeling in which all the objects and their interactions are defined. Making the objects generic allows them to be re-used on future projects that require similar functionality. Concept of inheritance allows objects to inherit attributes and functionality from other objects, and extensibility allows for modifying existing or adding new attributes and functionality. One of the main advantages of OOP is that it improves software development productivity by employing the concepts of reusability, modularity, and extensibility. In addition, OOP can help improve software maintenance, speed up development, decreases cost, and produces a higher quality

product overall. OOP is already employed, and its use is encouraged by some of the software development methodologies mentioned in this paper.

Software documentation is an essential feature of both software projects and software engineering (Chomal & Saini, 2015). Documenting every step of the process allows for later review and possible improvements by figuring out which steps need to be changed. In software development, documentation is one of the main tools used to communicate not only among the team members but also with the end user and others involved. Another benefit of having well documented processes is that it makes it easier to bring new members onto a team and simplifies the training process. Documents should be easily accessible by all parties and always kept up to date. Unified Modeling Language or UML is defined as a standardized general-purpose modeling language in the field of object-oriented software engineering (Sunguk, 2012). UML is a tool that can be used to specify, visualize, modify, construct, and document a software development process from the beginning stages to a finished product.

Last of the recommended practices that should be employed is the use of testing and specifically automated tests. Testing can help speed up the development process by reducing the testing time, from days and weeks to minutes or seconds in some cases. A number of tests and testing techniques exist that can be used to test out anything from an individual object and its functionality to testing out entire systems by using unit tests. Some of the different types of tests include: integration, system, interface, performance, reliability, acceptance, black box testing etc. One of very few downsides of automated tests is that they have to be designed ahead of time, and can be time consuming. Once written, they can be re-used and run without user interference.

# 7 SUMMARY AND CONCLUSION

Selecting a development methodology depends on different project characteristics and no one methodology is ideal or always the best. (Mahapatra & Goswami, 2015). A number of software development methodologies exist. The ones evaluated here include some of the more popular ones including waterfall, agile, scrum, extreme programming, rapid application development, spiral, and hybrid. Generally, waterfall and the spiral methodologies require a clearly defined process, and planning and documentation in order to make the development more efficient and predictable. "Traditional" methodologies can be used with success only for developing systems for which the requirements are clearly defined from the beginning of the project (Geambasu, et al., 2011). Contrary to the traditional approach are lightweight methodologies, including agile, and its subsets. Agile methodologies don't require clear or complete requirements at the beginning of the project and are more flexible. Selection of the methodology will vary depending on a variety of factors, including project characteristics, organization, skills, and experience of the development team, management's input, customer involvement, etc.

When working on adopting a new methodology, a company may run into organizational, people-related, process-related, or technological issues. To deal with these a full commitment from senior management and other important stakeholders is needed. Their buy-in will help remove any barriers encountered and will speed up the implementation process. To increase the chances of a successful implementation, proper training will need to be conducted, along with an investment in new tools and technologies.

Best practices for developing software presented in this study include the use of source tracking tools such as Team Foundation Server (TFS), object oriented programming, documentation tools, and finally use of automated testing. Implementing these tools will lead to a

more efficient process, easier to understand software, better documentation, higher quality, and finally a product that will meet or exceed customer expectations.

Various methodologies were presented in this analysis, along with their strengths and weaknesses. Neither one is ideal or best suited for every application. Proper research and evaluation can lead to the selection of the one that's most appropriate in a given situation. The chart presented in Figure 4-1 can be used to help make the selection process easier. One important thing to consider is that the project characteristics must be well documented, detailed, and understood before the selection can occur. According to the report published by Standish Group International (2009) regarding software project success rates:

- 32% of all projects succeeded (Completed on-time and on-budget, with all features and functionality as initially specified).
- 44% were challenged (Completed and operational, but over-budget, over the time estimate, and offers fewer features and functions than originally specified).
- 24% failed (Cancelled at some point during the development).

These numbers are significant and show that there is still a lot of room for improvement. Research shows that using a software development methodology, including ones that were presented in this paper, will increase the chances of the project being successful. The use of an adequate methodology plays an important role in developing software to assure that it is delivered within schedule, within cost and meets users' requirements (Geambasu, et al., 2011).

# 8 RECOMMENDATIONS FOR FURTHER RESEARCH

This paper reviews some of the most popular software development methodologies in use today, and the selection process in regards to the project characteristics. In addition to this the challenges of adopting a new software development methodology and the ways to overcome them are outlined, and finally the best practices for developing software products are presented. Review and the selection process does not include all available methodologies, and the project characteristics that are the basis of the paper only account for the extreme values. Having all the available software development methodologies would provide better results, would provide the developers with more options, and would make the selection process more accurate. Recommendation for further research would be to review and include additional methodologies in the study, and provide a more detailed chart to aid in the selection process.

# 9 REFERENCES

Beck, K., Beedle, M., Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R., Mellor, S., Schwaber, K., Sutherland, J. and Thomas, D. (2001). *Manifesto for Agile Software Development*. Retrieved from http://www.agilemanifesto.org/ [Accessed Oct. 03 2018].

Boyd, R. W. (2009) Software Lifecycle Model Selection Criteria for Safety-critical Software.

Chomal, Vikas & Saini, Jatinderkumar. (2015). Software Project Documentation – An Essence of Software Development. *International Journal of Advanced Networking and Applications*. 6(1), 2563-2572. Retrieved from http://www.ijana.in/papers/V6I6-9.pdf

Dyba,T., & Dingsoyr, T. (2009) What do we know about agile software development?. *Software, IEEE*, *26*(5), 6-9. Retrieved from https://ieeexplore.ieee.org/document/5222784/.

Edeki, C., (2015). Agile Software Development Methodology. *European Journal of Mathematics and Computer Science,* 2(1), 22-27. Retrieved from https://www.idpublications.org/wp.../Agile-Software-Development-Methodology.pdf

Faridani, H. (2011). A Guide to Selecting Software Development Methodologies. [e-book] Newtown Square: PMI SOC Greater Toronto Information Systems Branch. Retrieved from http://www.gtislig.org/HamidFaridani_GuideToSelectingSWMethodologies_SOC_PDD_20110305.pdf

Fojtik, R., (2011). Extreme Programming in Development of Specific Software. *Procedia Computer Science,* 3(1), 1464-1468. Retrieved from

https://www.researchgate.net/publication/220307727_Extreme_Programming_in_development_of_specific_software.

Friis, D., Ostergaard, J. & Sutherland, J. (2011). Virtual Reality Meets Scrum: How a Senior Team Moved from Management to Leadership. *HICSS*, 1-7. Retrieved from https://www.researchgate.net/publication/224221505_Virtual_Reality_Meets_Scrum_How_a_Senior_Team_Moved_from_Management_to_Leadership

Geambasu, C.V., Jianu, I., Jianu, I., & Gavrila, A. (2011). Influence Factors for the Choice of a Software Development Methodology. Retrieved from https://www.semanticscholar.org/paper/Influence-Factors-for-the-Choice-of-a-Software-GEAMBA%C5%9EU-JIANU/aeb046761b1fe1cbcce57153f2fc2c4e936989cb

Hajjdiab, H., Taleb, A. S. & Ali, J. (2012) An Industrial Case Study for Scrum Adoption. *Journal of Software 7(1)*, 237-242. Retrieved from http://www.jsoftware.us/index.php?m=content&c=index&a=show&catid=55&id=1080.

Highsmith, J. (2010). *Agile Project Management: Creating Innovative Products*. New York: Addison Wesley.

Kaur, S. (2015). A Review of Software Development Life Cycle Models. *International Journal of Advanced Research in Computer Science and Software Engineering,* 5(11), 354-360. Retrieved from http://ijarcsse.com/Before_August_2017/docs/papers/Volume_5/11_November2015/V5I11-0234.pdf.

Kuhrmann, Marco & Diebold, Philipp & Münch, Jürgen & Tell, Paolo & Trektere, Kitija & Mccaffery, Fergal & Garousi, Vahid & Felderer, Michael & Linssen, Oliver & Hanser, Eckhart & Prause, Christian. (2018). Hybrid Software Development Approaches in Practice: A European Perspective. *IEEE Software*. Retrieved from https://ieeexplore.ieee.org/document/8254323.

Liviu, M. (2014). Comparative Study of Software Development Methodologies. *Database Systems Journal*, 5(3), 37-56. Retrieved from http://www.academia.edu/22091353/Comparative_study_on_software_development_methodologies.

Mahanti, R., & Antony, J. (2005). Confluence of Six Sigma Simulation and Software Development. *Managerial Auditing Journal*, 20(7), 739-762. Retrieved from https://www.emeraldinsight.com/doi/abs/10.1108/02686900510611267.

Mahapatra, H., & Goswami, B. (2015). Selection of Software Development Methodology (SDM): A Comparative Approach. *International Journal of Advanced Research in Computer Science and Software Engineering*, 5(3), 58-61. Retrieved from https://pdfs.semanticscholar.org/cdb3/29ffaa401b2c878e5874583c2e48bfff9d5d.pdf.

Medvedska Oksana, & Berzisa Solvita. (2015). Selection of Software Development Project Lifecycle Model in Government Institution. *Information Technology and Management Science, 18*(1), 5-11.

P. Nerur, Sridhar & Mahapatra, RadhaKanta & Mangalaraj, George. (2005). Challenges of Migrating to Agile Methodologies. *Commun. ACM.* 48. 72-78. Retrieved from http://www.academia.edu/14859277/Challenges_of_migrating_to_agile_methodologies.

Rajagopalan, S., & Mathew, S., (2016). Choice of Agile Methodologies in Software Development: A Vendor Perspective. *Journal of International Technology and Information Management,* 25(1),

39-54. Retrieved from https://pdfs.semanticscholar.org/d161/50e945f5f07fd5352b035946e207361022a8.pdf.

Schwaber, K. and Sutherland, J. (2013). The Scrum Guide. Retrieved from https://www.scrum.org/Portals/0/Documents/Scrum%20 Guides/2013/Scrum-Guide.pdf.

Sunguk, L. (2012). Unified Modeling Language (UML) for Database Systems and Computer Applications. *International Journal of Database Theory and Application,* 5(1), 157-164. Retrieved from https://www.semanticscholar.org/paper/Unified-Modeling-Language-(-UML-)-for-Database-and-Lee/371d7b51f389f3ea78408d5386a91dfc49fc0d07.

Verma, J., Bansal, S., & Pandey, H. (2014). Develop Framework for Selecting Best Software Development Methodology. *International Journal of Scientific & Engineering Research,* 5(4), 1067-1070. Retrieved from https://pdfs.semanticscholar.org/34ae/7330122b4d2f3030f86f0bf8653771124e32.pdf.