# Methodologies: When Are They Useful?

By

**Abdulrahman Alaryani**

Department of Mathematics and Computer Since

Submitted in partial fulfillment of the requirements for the degree

Computer Science – Business System Management

Algoma University

Sault Ste. Marie, Ontario

March 2014

## Abstract

There are many methodologies are being used in the industry for software development. With the improvements in technology, more methodologies are introduced. Most methodologies follow the same phases: requirement, analysis, design, coding, testing and implementation. The use of the traditional approaches remains higher since managers and developers are not willing to take any risk. These traditional approaches include Waterfall and Agile Methodologies. Agile Methodologies is being practiced more often due to the low risks involved with it and its adaptability to changes. WaterSluice is not widely known methodology that suits project of clear and unambiguous requirements. Software methodology for teaching and learning is a new developed methodology to help teacher delivers properly and students understand theoretically and practically. However, it is limited to software engineering courses. I did research on several modern development methodologies and comparison among the selected methodologies was an objective. A survey was also designed and conducted in order to collect real data.

## Acknowledgment

I would like to thank my advisor Dr. Simon Xu for the continuous support of my bachelor study and my thesis for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis.

I would like to also thank Dr. Miguel Garcia for all the advice he gave to motivate me in writing my thesis and in how to research better.

My sincere thanks to Dr. Yi Feng for giving the time to read my thesis and providing feedback. I thank you for the insight comments and the encouragement.

# Table of Contents

**Chapter 1: Introduction** ... **1**
- **1.1. Software Development** ... **1**
- **1.2. Factors** ... **2**
  - 1.2.1. Requirements ... 2
  - 1.2.2. Type of End User ... 2
  - 1.2.3. Timeline of the Project ... 3
  - 1.2.4. Type of Project ... 3
  - 1.2.5. Size of Project ... 3
  - 1.2.6. Geographical Location of Team ... 4
  - 1.2.7. Resources ... 4
- **1.3. Major Stages in Software Development Life Cycle** ... **4**
  - 1.3.1. Requirement Gathering ... 4
  - 1.3.2. Analysis ... 5
  - 1.3.3. Design ... 5
  - 1.3.4. Coding & Testing ... 5
  - 1.3.5. Acceptance ... 6
- **1.4. Conclusion** ... **6**

**Chapter 2: WaterSluice** ... **7**
- **2.1. Introduction** ... **7**
- **2.2. Hybrid Approach** ... **7**
- **2.3. Factors** ... **9**
  - 2.3.1. Requirements ... 9
  - 2.3.2. Size of Project ... 9
  - 2.3.3. Timeline ... 9
  - 2.3.4. Development Team ... 10
- **2.4. Advantages** ... **10**
- **2.5. Disadvantages** ... **11**

**Chapter 3: Agile Methodology** ... **12**
- **3.1. Overview** ... **12**
- **3.2. Factors** ... **13**
  - 3.2.1. Nature of Project ... 13
  - 3.2.2. Size of Project ... 13
  - 3.2.3. Type of End User ... 14
  - 3.2.4. Time of the Project ... 14
  - 3.2.5. Resources ... 15
  - 3.2.6. Geographical Location ... 15
- **3.3. Categories** ... **15**
  - 3.3.1. Extreme Programming ... 15
  - 3.3.2. Crystal Methodology ... 17

3.3.3. Scrum .......... 19
3.3.4. Feature driven development .......... 20
**3.4. Advantages .......... 22**
**3.5. Disadvantages .......... 22**
**3.6. Scenario .......... 23**

**Chapter 4: Software Methodology for Teaching and Learning for Software Development .......... 24**
**4.1. Overview .......... 24**
**4.2. Advantages .......... 25**
**4.3. Disadvantages .......... 26**
**4.4. Scenario .......... 26**

**Chapter 5: Model Driven Development and Service Oriented Modeling .......... 29**
**5.1. Model Driven Development .......... 29**
5.1.1 Factors to be considered for model driven development. .......... 30
5.1.2 MDA implementation apprehensions: .......... 30
5.1.3 Advantages .......... 31
5.1.4 Disadvantages .......... 32
5.1.5 Reasons For Failure .......... 32
**5.2 Service Oriented Modeling .......... 33**
5.2.1 Service Oriented Modeling Disciplines .......... 33
5.2.2 Working .......... 35
5.2.3 Advantages .......... 35
5.2.4 Disadvantages .......... 36

**Chapter 6: Survey Results .......... 37**
**6.1. Overview .......... 37**
**6.2. Conclusion .......... 52**

**Chapter 7: Conclusion .......... 53**
**7.1. WaterSluice .......... 53**
**7.2. Agile .......... 53**
**7.3. Strategy for Teaching and Learning Software Development .......... 54**
**7.4. Service Oriented Modeling .......... 54**
**7.5. Model driven development .......... 54**

**Bibliography .......... 56**

# List of Figures

Figure 1 WaterSluice ........ 7

Figure 2 WaterSluice Methodology ........ 8

Figure 3 Extreme Programming ........ 17

Figure 4 Data Values ........ 18

Figure 5 Scrum ........ 20

Figure 6 Feature Driven Development ........ 21

Figure 7 Methodology for Teaching and Learning outcome ........ 27

Figure 8 Model-Driven ........ 29

Figure 9 Service-oriented modeling discipline ........ 34

Figure 10 Conceptual model of a service-oriented style ........ 35

Figure 11 Survey result: question 1 ........ 38

Figure 12 Survey result: question 2 ........ 39

Figure 13 Survey result: question 3 ........ 40

Figure 14 Survey result: question 4 ........ 41

Figure 15 Survey result: question 5 ........ 42

Figure 16 Survey result: question 6 ........ 43

Figure 17 Survey result: question 7 ........ 44

Figure 18 Survey result: question 8 ........ 45

Figure 19 Survey result: question 9 ........ 46

Figure 20 Survey result: question 10 ........ 47

Figure 21 Survey result: question 11 ........ 48

Figure 22 Survey result: question 12 ........ 49

Figure 23 Survey result: question 13 ........ 50

Figure 24 Survey result: question 14 ........ 51

Figure 25 Survey result: question 15 ........ 52

# Chapter 1: Introduction

Technology has drastically changed the world's approach towards work. It has changed the human life by providing ease of working and communicating. Software has a significant share when the word technology is defined. Software development is a very difficult and complicated process, which involves requirement gathering, analysis, design, coding, testing and finally deployment. There are many problems, which are confronted by the people making the software and people using software. This research study will elaborate on the use of different methodologies for developing software through proper software development life cycle. This is just to make sure that a product being delivered is of the best quality and in time. As far as the software development is concerned, there are three dimensions. These three dimensions are technology, people and requirements.

## 1.1.Software Development

Software development is a field that requires a thorough understanding of many concepts, and without understanding, it is not possible to deliver high quality products. There are many factors, which affect the development of software. Any company, which makes use of any software methodology, understands and analyzes these factors thoroughly. An analysis of the requirements is very necessary for successful projects. There are many methodologies, which are being used in the industry to deliver successful projects. WaterSluice, scrum, spiral, agile, RUP, RAD, and Extreme programming are the renowned methods being used in the world for software development. More concentration has to be put on the success of a project rather than selecting a methodology. However, stressing so much on the selection of the methodology and ignoring the actual design is not an efficient approach. Sometimes one methodology is suitable and other times, another methodology is suitable. Sometime one has to go with a hybrid approach also.

## 1.2.Factors

### 1.2.1. Requirements

Large organizations conduct analysis on the nature of the product before they take them or start working on them. They do so to make an exact choice of the accurate methodology that will be the best for developing software. One of the critical factors for choosing a methodology is the requirement. Some customers are knowledgeable of the industry and related software, and know what they exactly need, whereas other customers are not well educated on technology and they are not clear about the requirements. Having different type of customers lead to diverse projects and to use different type of methodologies. Depending on whether requirements are clear or ambiguous, the choice of which methodology to be implement is made. Therefore, the nature of requirement is essential in choosing the methodology for a software development.

### 1.2.2. Type of End User

There are advantages and disadvantages of each methodology of software development. A successful manager’s responsibility is to analyze the factors affecting the choice of methodology and choose the best one. The end user of the system being developed is a crucial part, and it affects the project implementation and its successful delivery. Controlled or expert group of end user will have an effect on the end product and quality of the software. If the end user of the product being developed are dispersed and widely distributed, then requirements will be changed and can have an effect on the end product. The awareness of the technology in the end user is also imperative. Moreover, feedback of customers is also significantly important. If the end users are positive, they will give encouraging feedback, which in return will add to the success of the software. If the end users have negative criticism, then this part will derail the project implementation.

### 1.2.3. Timeline of the Project

The time line or duration of the project is an essential factor and this factor has decisive power as far as the choice of methodology is concerned. If the project has a tight or aggressive deadline then the method of working would be different, and methodologies will be different. Otherwise, if the deadline is not strict and client is not so much concerned to deliver the project on a specific date, then different methodologies should be involved. So the time frame in which project has to be completed is very important during the process of selecting a methodology.

### 1.2.4. Type of Project

The nature of the project is also a paramount determinant of the methodology. Sometimes there are projects, in which teams have hands-on-experience and knowledge of technology. For these types of projects, the selection of strategy for development would be different as compared to projects whose nature is unclear. There are projects where the technology is not fully understandable to the team of the developers. Therefore, the selection of a methodology is complicated and may lead to failure of new projects.

### 1.2.5. Size of Project

The most important factor for the selection of methodology is the size of the project. Large projects need large teams and extensive technologies. Small sized projects need less effort and can be handled by small number of developers and designers; on the other hand, large projects need different types of methodologies to be implemented. A great project means many functionalities and complex modules. Software, which are distributed and have to be used by many people are also known as large projects.

### 1.2.6. Geographical Location of Team

The development and design team's location plays an important role in the selection of methodology. If the teams are geographically located together, then it is very easy to coordinate and collaborate in terms of project's discussion. This will lead to selection of a methodology where requirements are clear and not ambiguous. If the teams are geographically apart, then there will be problems of coordination and in return, the problems of requirements gathering will arise which will lead to unclear and ever changing requirements and ultimately different methodology would have to be chosen.

### 1.2.7. Resources

There are many types of resources, which are needed during the completion of a software project. Resource can be designer, developer, time they are consuming to complete the project, technology infrastructure, etc.  Resource availability also plays a great role in the selection process of methodology. It is imperative to analyze the resources, which are critical before implementation. If the resources are less and not enough, then different strategy would be selected as compared to the situation when the resources are available and are enough.

## 1.3.Major Stages in Software Development Life Cycle

Major phases, which are involved in any of the methodology, are following

### 1.3.1. Requirement Gathering

Requirement gathering is the most important phase in SDLC. Requirements can be gathered through different methods like interviewing the clients, distributing surveys and questionnaires, and conducting meetings with management and technical employees of the company. The more the requirements are clear, the more it will count towards the success of the project.

### 1.3.2. Analysis

Analysis is about studying the existing system of an organization and then comparing it with a solution, which is to be made. Analysis is a crucial step, and it counts towards the success of the project very much. Analysis clears the image of the solution for the developers as well as the designers of the project. They can confidently work toward the goal of their project.

### 1.3.3. Design

Design is the logical picture of the solution. An architect designs a map before he constructs the building. Similarly, it is very necessary to design the product when developer is going to code the software. It is just like logically devising the solution and representing in pictorial of a model form. Design is also the foundation phase and needs a high level of commitment and concentration. The model of design models can be class diagrams, data flow diagrams, entity relationship diagrams, etc.

### 1.3.4. Coding & Testing

The actual phase is the coding. Coding is done in programming language, and choice of the language depends on many factors. Client's choice is the biggest factor. Sometimes, it depends on the functionality of the software. The functionality required in the software can be dependent on specific language. For example, Java is renowned for its networking library, and Fortran is famous for its mathematical capabilities. This is the reason that coding is a critical phase. After coding the product, testing of the product is made. This is to check the accuracy of the solution and whether the solution is accurate for every user input and to find if all of the conditions are met before time and in an accurate sense.

### 1.3.5. Acceptance

Acceptance is testing performed by the client. The clients test the system solution and based on the results, it checks the validity of the system. The clients see if all the requirements for the system are met. If that is the case, it provides an OK signal to the developers. Otherwise it provides feedback so that developers and designers together find out the solution for the problems encountered during the acceptance testing. In short, acceptance testing is about testing the solution by the customers to check if the result is acceptable to them or not.

## 1.4.Conclusion

Having all the factors discussed, it is now very clear that these factors will ultimately impact on the selection of methodology. There are many methodologies, which have been being used for decades. With the passage of time, software industry, through proper research and feedback discover different methods to complete the software project efficiently. Thus there are dozens of methodologies, which are proven to help in getting projects done in time, and based on schedule.

# Chapter 2: WaterSluice

Below is the pictorial representation of the WaterSluice model

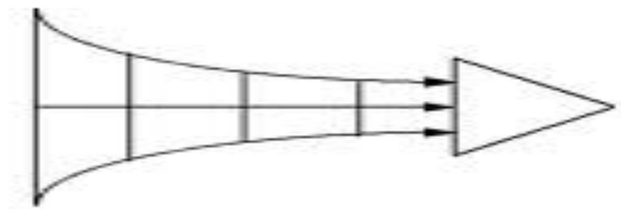


**Figure 1 WaterSluice**

## 2.1. Introduction

WaterSluice strategy is a hybrid technology with a mix of sequential and iterative strategy. Figure 1 shows the steps involved in the WaterSluice model. WaterSluice is a sequential and iterative approach. There are certain conditions under which WaterSluice model will be implemented. WaterSluice model is the most well known methodology of the present. Since traditional methodologies were not enough to fulfill requirements of technology and software, Software engineers discovered some other ways to solve their problems in efficient time.

## 2.2. Hybrid Approach

In WaterSluice model, the software is divided into small modules. These modules are small functionalities, which can be termed as activities. These modules and activities when integrated together can make fully functioning software. For every module, the above-mentioned steps are done. These steps are repeated for each and every module in the same sequence. When the small modules are complete, they are iterated again and again to make sure that software is of good quality.  Doing so, the whole software is developed, or the project is completed. If there is any problem in a specific module, then all the steps are repeated again for that specific module. If the client is not satisfied with a specific module or the whole project, then all the above steps are repeated again sequentially and iteratively.

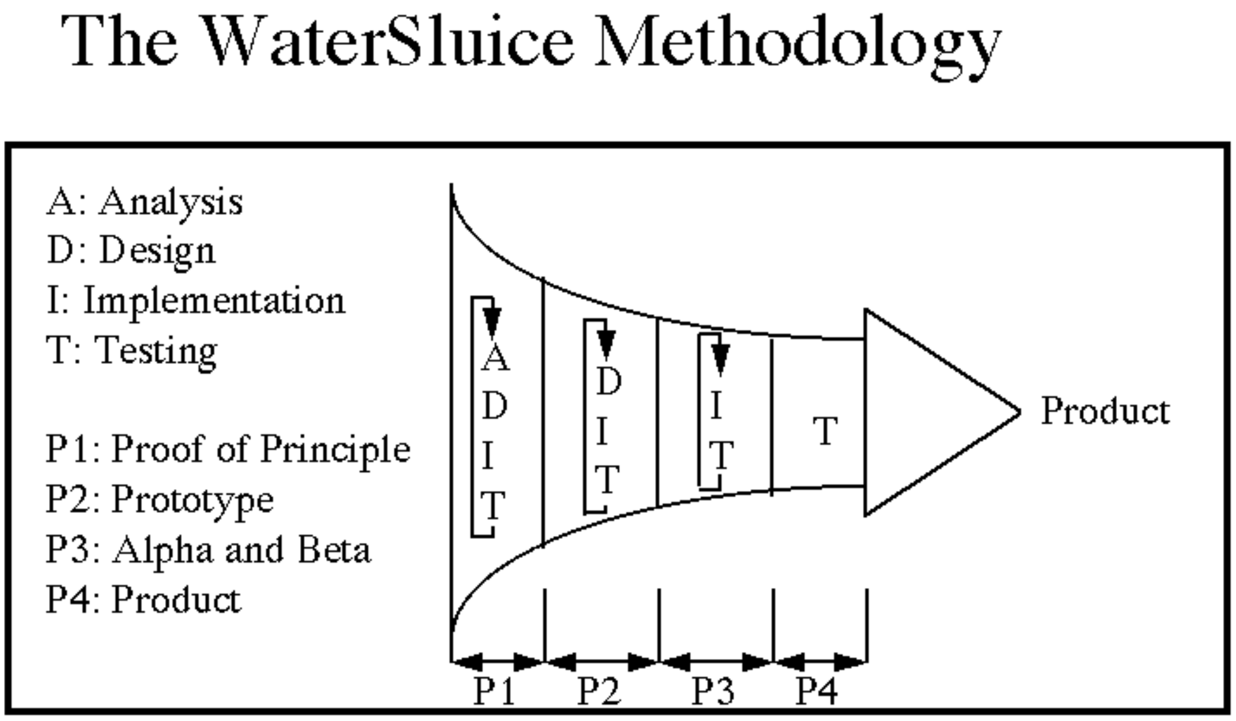


**Figure 2 WaterSluice Methodology**

Figure 2 shows the elaboration of the WaterSluice methodology. Product development is iterated through four phases, which are Proof of Principal, Prototype, Alpha and Beta, and Product.

These are just the iterations and increments to make the product of good quality. In every phase, there are couples of steps, which are followed. These steps are general software development steps as analysis, design, coding, and testing. The above paragraph explains all the methodology very clearly, and it is self evident that an intermediate and hybrid technology will help the developers to come up with good quality product.

## 2.3. Factors

### 2.3.1. Requirements

WaterSluice is a linear as well as an iterative method. It is selected as the methodology when the requirements are fully clear and unambiguous. There is no need to further elaborate the requirements. Designers are fully satisfied with their approach and developers are satisfied with the functionalities they have to develop. This is only possible when the customer has been able to totally deliver their requirements and the people on the client side have some-how knowledge of the technology.

### 2.3.2. Size of Project

As far as the size of the project is concerned, WaterSluice model is best for large sized projects. Large size projects hire large teams for their implementation. In such projects, objectives are clearly defined along with responsibilities of the team; therefore, it is very suitable to choose WaterSluice methodology.

### 2.3.3. Timeline

If the time span specified by the client is tight or less, it will not be good to adapt WaterSluice models. WaterSluice model is good for projects where the client has given a loose or open-ended deadline. There is no problem in coping with the deadline of the project. So, the WaterSluice model is best for projects having a non-strict project deadline.

### 2.3.4. Development Team

It is significantly important to choose WaterSluice when development teams are geographically located at one location and are not dispersed. If they are close to each other, collaboration among them is strong. Strong collaboration among team members and developers is needed if the WaterSluice model is selected. If this is not the case, then the selection of WaterSluice as methodology will be a very wrong choice. Wrapping up the discussion of the factors, WaterSluice model will be the best choice when requirements are clear and project size is large. The people working on the project are completely trained about the solution and the technology being used for the project solution. It will also be good to opt for WaterSluice model when the team members are not dispersed and are at the same place. It is also important to take care of project's time line also. If the project's deadline is strict, then it is best to not opt for the WaterSluice model.

## 2.4. Advantages

WaterSluice takes the benefits of both approaches as iterative as well as linear. When chosen after clear attention, it is said that WaterSluice has many benefits. The most important advantage is the simplicity of the WaterSluice. WaterSluice is very simple and very easy to follow. It gives the control of the project to the manager and departments. Testing is done on a modular level, so there is a reliability involved. When the project is reliable, then your customers are happy and satisfied with your work. If customers are satisfied, then it will count for the overall success of your business and team. The projects, which are selected after careful consideration and are made through strictly following WaterSluice model, are always on time. To be on time, is one of the qualities of the project or business, which counts towards the success of the business.

## 2.5. Disadvantages

While there are some advantages, there are some disadvantages also. The WaterSluice model is a very new model and managers are very reluctant to follow this approach. If the project is very simple then, it is wastage of time to adopt the WaterSluice methodology. But if the project is complicated and new and needs to be improved over the span of time, then WaterSluice methodology will be the best. So, wastage of time is one of the disadvantages of this methodology. It also makes the simple things complicated, thus fewer managers go for this approach. It is time consuming. The project's overview or the end product's shape is visible by the end of the project. If you decide to go back and repeat all the steps, in case of any problem, then you will lag behind the schedule. In addition, there are very less helping material available for this methodology. So, if some manager decides to follow these methodologies, then he/she has to dig the information.

## 2.6. Scenario

It would be best to choose WaterSluice if somebody is developing a dynamic website and with no doubt knows which language will be used. In this situation, the developer should be able to have a clear estimate that, in how much time, the project will be completed. However, some of the functionalities need to be prioritized. The time given for the project is possibly to be flexible. As a result, the level of risk involved is low. WaterSluice methodology must be adopted for such system where quality and prioritize functionalities exist.

# Chapter 3: Agile Methodology

## 3.1. Overview

Agile software development is a group of software development methods based on iterative and incremental development, where requirements and solutions evolve through collaboration between self-organizing, cross-functional teams. It promotes adaptive planning, evolutionary development and delivery, a time-boxed iterative approach, and encourages rapid and flexible response to change.

Agile development is chosen when the requirements are not clear which means that risk is high for most project will be developed using Agile. Agile software development is best for these projects can be divided into mini projects, and passing through a phase of design, coding and testing completes these mini projects. The results of these iterations and customers' feedback clear the requirements gradually. Companies choose Agile development methodology due to its capability to adapt to the difficult situation and chase strict deadlines. Agile method is being used all over the software industry. Most developers are aware of the fact that Agile methodology is best for them.

Most of the software development projects change just while the development is in the process. Therefore, Agile methodology is best to be chosen for such projects since it provides the developer the ability to modify the project according to changes in requirements.

## 3.2. Factors

### 3.2.1. Nature of Project

As indicated previously, Agile methodology is used where the requirements are ambiguous. Developers, sometimes, find difficulties in taking requirement from those customers who are unaware of technology. In such case, it is necessary to build multiple features and the show them to customers to clarify if they match their expectations or not. Doing so along with the passage of time, requirements become clearer to developers and enable them to move toward the project completion.

The development of a mini product enables developers to ensure that they are working toward the client's requirements. Hence iterations in Agile help developer, in case of misunderstanding, to revert back and clear any mistakes before going into the development. If any other sequential methodology were adapted, then it would delay the project because so much time is wasted in developing the wrong product. Thus Agile development alert the developer in advance that the requirements of the project are being misinterpreted.

### 3.2.2. Size of Project

The size of the project is also one of the main factors and it is very necessary to have the idea of size of product before choosing any software methodology. Software size is very easy to measure and it can also be measured through the use of function point analysis. The projects of large size are not suitable to be developed by Agile methods. Agile development is suitable for projects of small sizes where projects are completed in months, not in years.  It is easy to identify small mini projects when the overall size of the project is small. Short iterations could lead to a more composed and comprehensive project. Therefore, when the project size is small, then agile method is the best method to work with.

### 3.2.3. Type of End User

Sometimes end users are fully prepared for the product, and they have thought about what is being developed. For these types of projects, sequential models can be the best approach. However, as technology is unpredictable or customers less knowing of technology are concern at, it is good to adopt agile method. In Agile methods, small feature is delivered to the customer and then they give feedback whether the project is going in the right direction. There are many customers in the industry who have still not coped with the changes in the software industry. This is the method to adapt when working with such customers.

### 3.2.4. Time of the Project

Agile development is best when the time of the project is short. Iterations are done in short time and then they can be modified and improved over the passage of time. So, if the duration or the time limit for the project is short, developers should adapt agile development. Software projects nowadays have the condition of time and software developers have the challenge of completing the work before time so many companies are following the agile methodology. If time is critical for the project, this is very important to follow such a method, in which results can be predicted in early completion of project. This will result in accomplished commitment and happy customers. Happy customers will increase developers rating and will result in success for the business.

### 3.2.5. Resources

Resources are important factors, which help the success of the project. Sometimes, there are adequate resources for the project and sometimes, the reverse are the case. When resources are not adequate with respect to the project, it is a challenge to use these resources in such a way that maximum efficiency can be achieved. Sequential approach will be good if resources allocated to the projects are enough otherwise iterative and incremental approach is good. Insufficient resources are the major problem confronted by the software industry.

### 3.2.6. Geographical Location

Geographical team has been one of the main factors in the project completion. This is because of the fact that team collaboration and coordination is very important as far as the selection of the methodology is concerned. Teams are better in collaboration when they are geographically close to each other. Otherwise there would be a gap of communication between the team. If team is not fully integrated and close to each other then it is good to choose agile methodology because it gives a space where risk is involved. Sequential models are not good for projects where risk is involved.

## 3.3. Categories

### 3.3.1. Extreme Programming

As the name suggests, Extreme programming involves the extensive programming efforts. There are trial and test scenarios also for the extreme programming. There are some unique features developed in the Extreme Programming, which make it very adaptable and success friendly. Thus mature companies, for most of their projects, choose extreme programming. The success rate of the projects, which were made by using Extreme programming, is very high. The features are communication, simplicity, feedback, courage and respect. Communication in extreme programming methodology is very straightforward. Oral communication is in the culture for normal projects. When the project is very critical, and audit of the employees is one of the issues, then communication is made orally as well as through story cards, class responsibility and collaboration. Customer acceptance tests are made in order to make sure about the requirements and responsibility distribution of the projects. Therefore, requirement gathering and team collaboration is one of the most important phases of any project especially software projects. Extreme Programming suggests that the design of the solution should be the simplest one. This means that it should fulfill maximum requirements of the project but should be kept simple. This would lead to less effort and more understanding of the solution. The popularity of Extreme Programming is due to that developer can deliver more when using it.

The value of feedback is worth mentioning when it comes to the development of the software. Extreme Programming stresses upon the quality of feedback because customer's feedback can increase the quality of the product, and it can result in a successful project that satisfy customers. This also leads to early identification of problems in design, if there are any.  Courage is one of the most important features of the Extreme Programming. Courage means to take decisions as a team and then work on them together. Extreme programming environment provides the developers this opportunity to choose and have some power over development tasks. Last feature of the extreme programming is the respect. Every member in the team is encouraged to take part in the discussions and collaboration so respect feature among the team is developed. This gives developers a sense of accomplishment and association with the company.

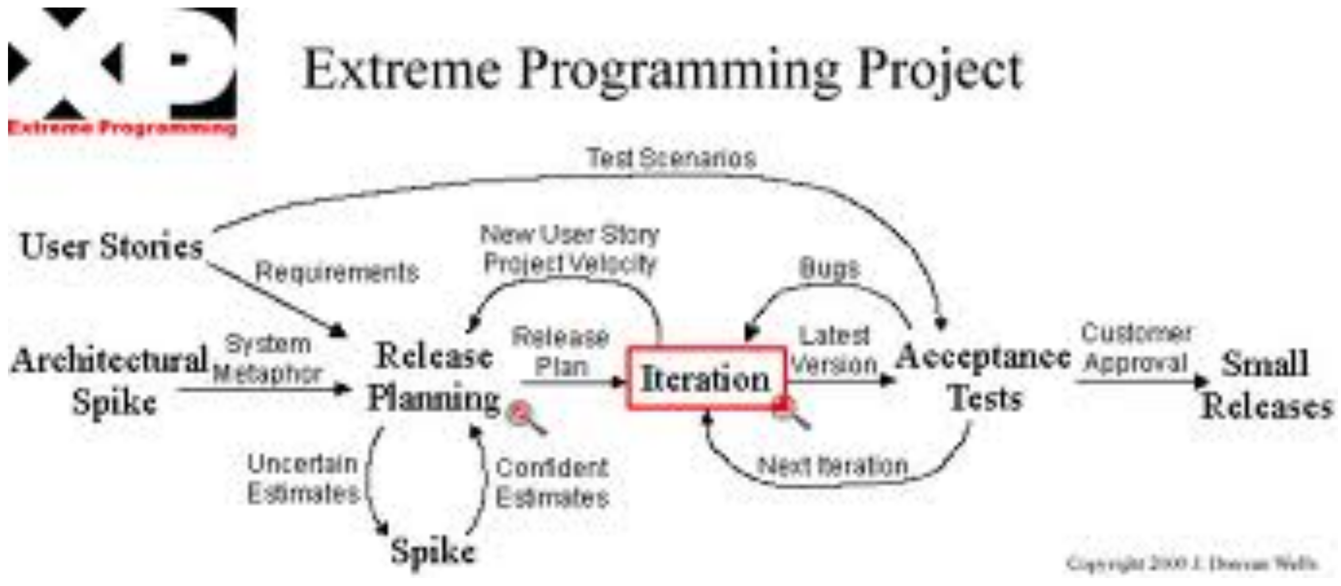
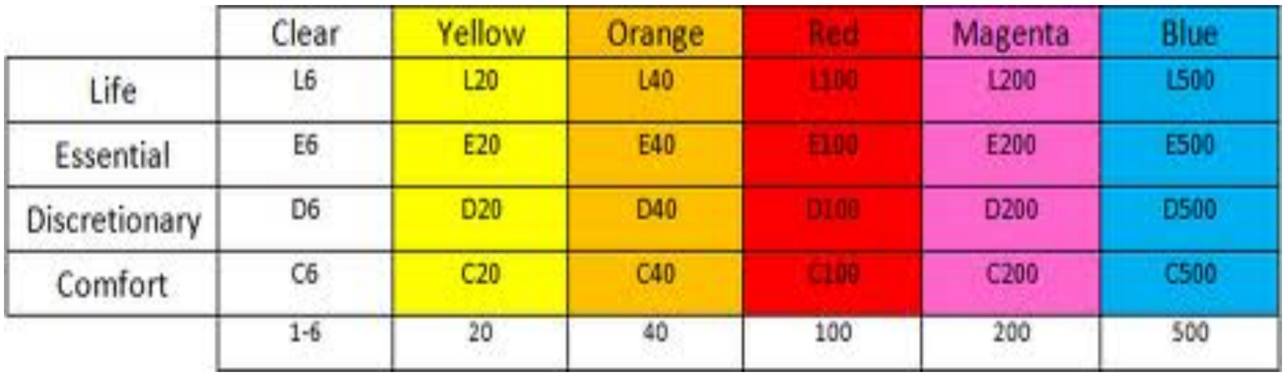


**Figure 3 Extreme Programming**

### 3.3.2. Crystal Methodology

Crystal methodology is the part of Agile methodologies and it has capability to except change. Change can be of different types, but the specialization of crystal method is to deal with the number of people in the project and then the system potential for creating changes. System potential for causing damage can be of different types. For example, the minimal problem is the loss of comfort and loss of money. There are some other possible problems that are tremendous, and an example would be the loss of life.

Figure 4 is the chart of the explanation for the crystal methodology. Horizontally it is showing the number of people who are working in the project and vertically is showing the potential loss due to the damages in the software development.



| | Clear | Yellow | Orange | Red | Magenta | Blue |
|---|---|---|---|---|---|---|
| Life | L6 | L20 | L40 | L100 | L200 | L500 |
| Essential | E6 | E20 | E40 | E100 | E200 | E500 |
| Discretionary | D6 | D20 | D40 | D100 | D200 | D500 |
| Comfort | C6 | C20 | C40 | C100 | C200 | C500 |
| | 1-6 | 20 | 40 | 100 | 200 | 500 |

**Figure 4 Data Values**

Based on the different values, the category is further divided into different types. When the developers are 1 to 6 then the methodology to be chosen is Crystal Clear. When the number of members increased from 6 to 20, then recommended methodology would be Yellow crystal. So there are different methodologies such as Orange Crystal, Red Crystal, Magenta Crystal and Blue Crystal.

There are different phase, which are developed in different types of crystal methodologies. In Crystal Clear method, it is documents, roles of the people and the process, which is being developed. It is best for companies where all the employees are sitting in the same room. This stresses upon the importance of collaboration and coordination among the team members. It is very obvious that if the people were discussing project design, priorities, problems, possible solutions in the same room then it would affect the overall performance of the team. It will enhance their performance and their understanding about the project techniques. This is the reason that companies doing small projects which are following the Crystal Clear methodology achieve maximum with the help of this project related communication and collaboration.

### 3.3.3. Scrum

Scrum is also one of the most important categories of the agile methodology. Scrum is composed of some iteration and these iterations and increments are very important. Meeting among the developers and team members are held everyday and these meetings are for 15 minutes. Sprint meeting and sprint review follow 30 days scrum meetings.  In daily 15 minutes meeting, following three things are discussed.

- What is today’s goal?
- What obstacles where confronted day before the meeting?
- What can be the solution of the problem?

In this way project progress can be tracked on daily basis which is very necessary and can lead to early detection of problems in case if there are any. As a result, each and every day is a day of accomplishments. Moreover, this increases the efficiency of the team members because each and every day is accountable for what they have done in the last day. Overall, this enhances the performance of the people in the company.

The major feature of the Scrum is its management wrapped around the methodology of development. There are many companies who are following scrum and they have over-achieved in the development. Their employees are very trained and efficient. The result is they do work on commitment and failure risk is minimized.

There are three roles in the scrum meeting: Product owner, Scrum master and the Developer. The person who is responsible for the product and product backlogs is the product owner. The person who is responsible for conducting scrum and sprint meetings is known as scrum master. He maintains all the records and he is responsible for keeping track of the performance of each and every employee. Finally, the developer is the person who codes and then tests the product for a given set of requirements. In such a way, roles and responsibilities are clearly divided and identified. Moreover, conflicts between the roles and responsibilities of the persons in the company are reduced.

It is very obvious that the size of the team of scrum is very small because it is almost impossible to conduct 15-minute meetings with many people and all the issues are discussed. 15 minutes are enough for the scrum meetings because the team size is small and it is very easy to discuss project related issues in a short time. Scrum methodology is inappropriate for the companies who have large number of people working on one project.

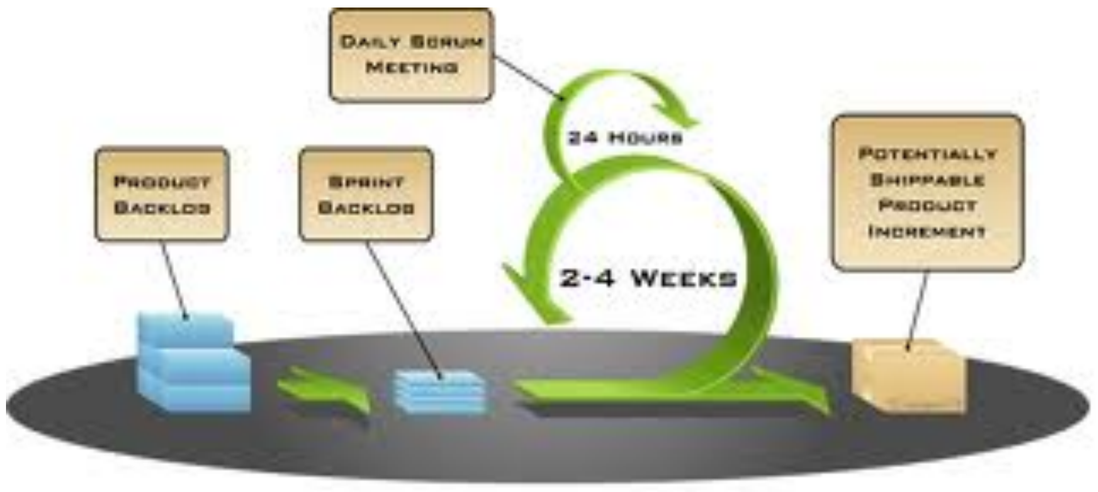


**Figure 5 Scrum**

### 3.3.4. Feature driven development

Feature driven development is the best where the projects can be clearly cut down in small features and at the end of the project; the features are integrated in such a way that they give the whole product. There are some processes in the feature driven development and they are described briefly below.

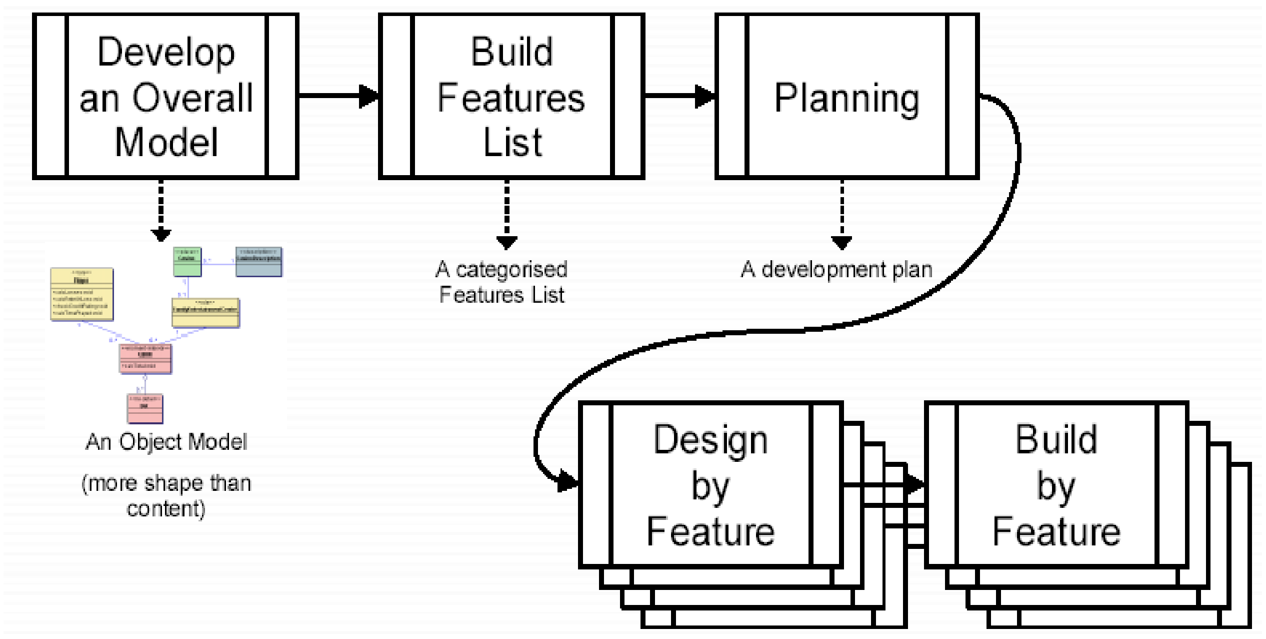


**Figure 6 Feature Driven Development**

First, the team members have to develop overall class/object model. This model will be helpful in understanding about the features of the product. There are many things, which are to be considered when the overall model is being developed. After the design of the overall model, the features list is developed from the overall model. These two processes will take about one to two weeks. After the feature list is developed, each feature is assigned to a particular developer or team of developers. The developer then plan about the design solution of the feature assigned to him/her. Final design is done in the phase of Design by Feature whereas the developer develops the product in the phase of Build by Feature. The last three processes takes about two to three weeks which means that it is more lengthy process as compared to the first two phases. The main work of the developer is in the three last phases of feature driven development. However, they are also involved in the meetings for the developing of the overall model.

There are many roles in the feature driven development. These roles are of Development Managers, Project Manager, Domain Experts, Class Owners and Chief Architects. These roles are to facilitate and break down the work of roles and responsibilities. There are different types of formats in which the overall output of the phases of the project is shown. This is done due to the reason that records about the progress of the project completion must be kept. In short, Feature driven development is best for the projects where there is a defining line between the functionalities of the project being developed.

## 3.4. Advantages

There are many categories of Agile due to the fact that each category is to deal with different types of changes, which occur during software development. In sequential methods you deliver the product at the end and there is possibility that unidentified problems could occur. On the other hand, iterative and incremental approach makes it possible to identify the problems in design and solution earlier as a result of feedback. This leads to time saving as well as money saving affect. Agile methodologies are in practice everywhere due to their vast adaptability feature. Further differentiation in agile methodologies leads to more flexibility and adaptability.

## 3.5. Disadvantages

As always there are two sides of the mirror, Agile methodologies have some cons also. The main disadvantage is the risk factor. This creates uncertainty among the development team. Customers Sometimes demand complicated or impossible requirements. Agile methodologies are diverse and sometimes much work is needed to decide which agile methodology would be the best for the project. Therefore, much time of the project could be wasted only on deciding which Agile methodology to implement. As a result, project can be behind the schedule, which leads to unsatisfied customers who are consider being the indication of unsuccessful business.

## 3.6. Scenario

There are many projects, which are good to be followed through agile methodologies. As we know that the projects with unclear requirements are the cause of selecting agile methodologies. Agile methodologies with their increments clear all the requirement ambiguities through time. For example, research projects always are uncertain. When a new product is being developed, there is always a chance of uncertainty that it will not be successful. Research projects are mostly about finding a solution to a problem, which has not been discovered before. Research projects are not always successful due to hit and trial method where researchers do not know which hit will lead to the successful results.

FORTRAN is a language, which is very rich in mathematical capabilities. Java is specialized for networking. If the FORTRAN Company is hunting for a solution or API that will help to cover a networking aspect, it will be a research project and there is a possibility that they require much time on it because they do not know the exact technique to do such a project. Therefore, it is usually recommended that any new research project or any new project to be done by using agile methodology.

# Chapter 4: Software Methodology for Teaching and Learning for Software Development

## 4.1. Overview

As the software methodology field is advancing, software engineers are specifying or developing different types of methodologies for particular scenarios of the software development. With so much advancement, the world has not stopped here. It is being advanced day by day. As a result, more and more youngsters are trying to fit in this field. This is very difficult to find a university, which does not have computer science discipline. That is why there are so many computer science graduates out there in the market. The problem is that software engineering is not an easy field. Programming and software development is very difficult.

Due to the complexity level of programming and software development, it is very difficult for students to learn programming. On the other hand, it is also very difficult for teachers also to teach because the ideas are very vast and complicated, and there is no method to ease understanding. When there is a problem, there must be a solution. This need or problem compelled some people in the research industry to come up with a methodology to solve the problem. The proposed problem is for the students learning the software development programs and the teachers teaching these courses. Through the collaboration of both parties, it is possible to come up with a technique that is convenient and helps in learning the concepts and complexities easily with much less effort on both sides. The main crux of the methodology is to expand the practical work as compared to theoretical work.  This will help students understand the development in an easier way.

This methodology is made up of some phases, some milestones and iterations, which must be accurately followed by the students and teachers. Otherwise, it will not be possible to take maximum throughput from this methodology.

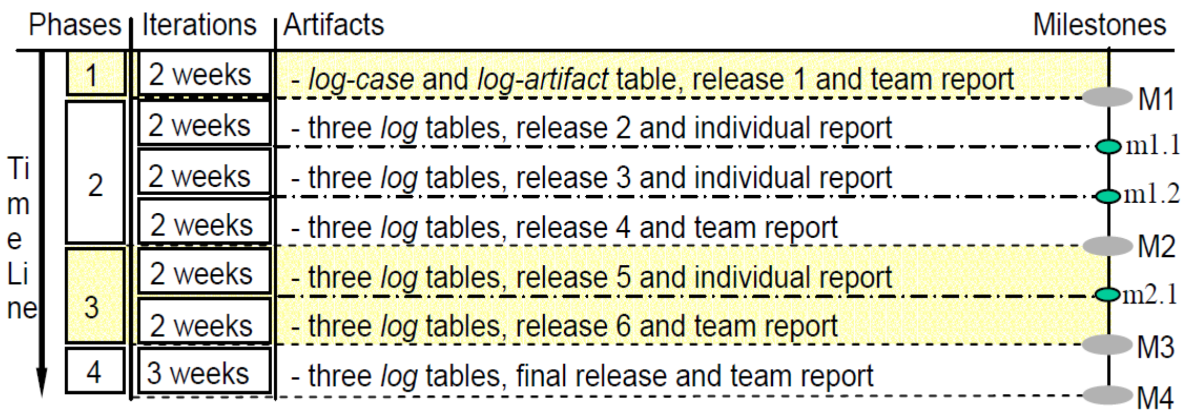

Phases
Iterations
Artifacts
Milestones
Time Line
1
2
3
4
2 weeks
2 weeks
2 weeks
2 weeks
2 weeks
2 weeks
3 weeks
- log-case and log-artifact table, release 1 and team report
- three log tables, release 2 and individual report
- three log tables, release 3 and individual report
- three log tables, release 4 and team report
- three log tables, release 5 and individual report
- three log tables, release 6 and team report
- three log tables, final release and team report
M1
m1.1
m1.2
M2
m2.1
M3
M4

**Figure 1. Schedule of process model proposed**

The proposed methodology is very clear from the above picture. The output or results of the iterations are in the form of log cases; log tables, individual report, team report, a final release, and so on. Iterations are divided into two weeks except the last iteration. The last iteration consists of three weeks. In each week student has to spend four hours and these four hours are to be spent at the lab. Extensive lab work helps the students to understand the difficulties and complexities of the systems, and this is the reason that when these methodologies were applied to the universities, then the output was much better than the results taken without any methodologies.

### 4.2. Advantages

There is a need to develop the system for taking software development course since the needs of the world for technology are drastically increasing. Therefore, institutes should come up with some ideas to boost the learning process. This methodology was applied to multiple institutions and the conclusions were that the application of the methodology has changed the results, and students were willing in to learn by theory as well as practically. Moreover, practical means more understanding of concept so in return improvement in the system of learning and teaching can be made.

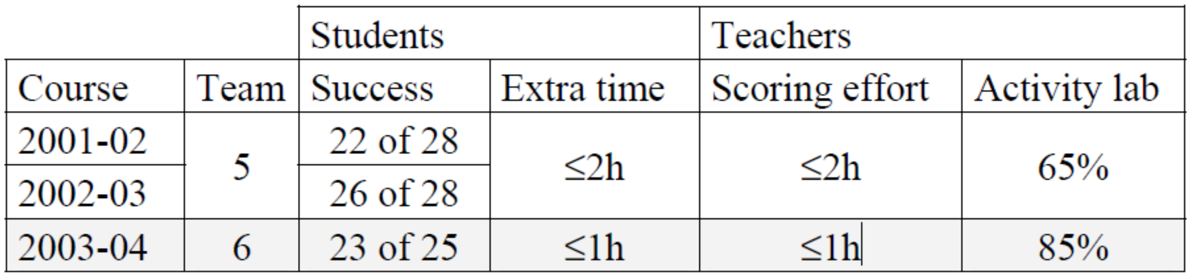
## 4.3. Disadvantages

This result applies to the courses of software development only. This is their limitation, and they cannot be used to taking care of any software development project or any other subject related teaching. So they have no direct contact with the achievement of the software project. It works on the base of the idea that the team of the project and their skills are imperative for the overall achievement of the project. Therefore, it works on the fundamental needs and team improvement.

## 4.4. Scenario

During the academic year 2003-04, the number of students was 153. They were divided into 22 groups of six, and three groups of seven. In one laboratory session (two hours per week), there were a maximum of three or four teams. There were five teachers, and they worked in team software development since 1995. In past years, they proposed several process models for the effective learning of software engineering. More specifically, they used a weighted UP-based process model driven by functional services instead of use cases and worked with groups of five students. The new approach they proposed emphasizes the role of the process model as a pedagogical tool for the teacher. As a result, they obtained a greater level of satisfaction in both teachers and students.

Figure 7 illustrates the result obtained by the application of methodology



| | | Students | | Teachers | |
|---|---|---|---|---|---|
| Course | Team | Success | Extra time | Scoring effort | Activity lab |
| 2001-02 | 5 | 22 of 28 | ≤2h | ≤2h | 65% |
| 2002-03 | | 26 of 28 | | | |
| 2003-04 | 6 | 23 of 25 | ≤1h | ≤1h | 85% |

**Figure 7 Methodology for Teaching and Learning outcome**

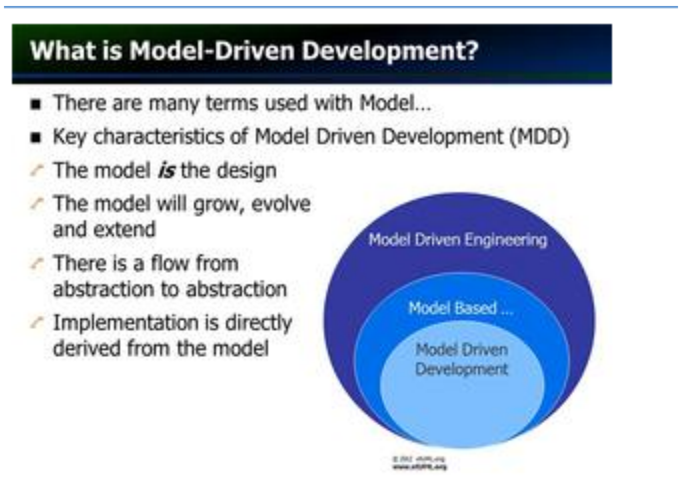
In 2003-04, 23 groups from a total of 25 developed a successful project; this means that only two groups have not completed the project. They noticed that a similar result was obtained in 2002-03. The improvement of 2002-03 results in comparison with 2001-02 is primarily the result of the teachers' greater experience; in that they also learned from the difficulties encountered by students and can consequently guide the students' work more efficiently. They also expected this percentage to increase for 2004-05. As for the time spent by students on the proposed project, the groups needed to work at home for a maximum of two extra hours per week in order to complete the project during 2001-03. In 2003-04, the time spent by students was reduced by 50%. This is a consequence of the agile character of the new proposed process model, which reduces the overload of documenting the project. Teachers reduced the volume of documents to be delivered to a set of simple, standard calc sheets, and a set of automatically generated UML diagrams. This means that students can document the project efficiently in a short time with all the information required for the future maintenance of the system produced. Teachers also benefit from the proposed process model as a 50% reduction in the efforts for scoring students. This effort was reduced in many ways. Teachers standardized and simplified the set of artifacts and the repository students had to deliver in each iteration, providing the corresponding templates in such a way that the teacher knows that all the documents (including the code) will have the same structure and format, and the same items of information, the teacher reviewed the deliveries with the students during laboratory hours and receives a verbal report at each phase with three points that summarized the work carried out by the groups. This verbal report is very important and improves students' capacity for expression, concision and clarity in that they had a time limited to a maximum of five minutes to summaries several weeks' group work; and (c) the teacher can execute the partially implemented system at each iteration (every two weeks), and can compare the code with the documents delivered with ease.

Finally, teachers benefited from short iterations, development that is driven by use cases, and the development of high-risk use cases in early iterations. These risks are addressed up front and the teacher can help solve problems actively during laboratory hours. This increased the teacher's work in laboratory hours and reduced his/her total work in answering students' questions. During the academic year 2003-04 the activity carried out by teachers during laboratory hours represented 85% of the total time spent, reducing the extra hours of the total time spent during 2001-03 by 20%. So this means that this methodology is very beneficial for the people having software development programs. They take the advantage of this methodology and increase the performance of the students taking software development programs.

# Chapter 5: Model Driven Development and Service Oriented Modeling

## 5.1. Model Driven Development

Going by the literal meaning of a 'model', it is a simplified representation used to explain the workings of a real world system or event. Traditional software development focused more on computing concepts. However, model-driven development is different in the sense; it focuses on creating models and incorporating domain knowledge in them. Here domain models mean high level depiction of the facts and data that govern a particular application domain. The model can be considered as the visualization of the program's structure. It is also known as domain engineering or MDA and was introduced by the Object Management Group (OMG) in 2001. The model driven development defines functionality of a system using models that are platform independent. Then depending on the given platform, the model is converted to the platform specific models.



What is Model-Driven Development?

- There are many terms used with Model...
- Key characteristics of Model Driven Development (MDD)
  - The model *is* the design
  - The model will grow, evolve and extend
  - There is a flow from abstraction to abstraction
  - Implementation is directly derived from the model

Model Driven Engineering
Model Based ...
Model Driven Development

**Figure 8 Model-Driven**

### 5.1.1 Factors to be considered for model driven development.

Model driven development focuses on forward engineering i.e. producing code from abstract, human-elaborated modeling diagrams. It separates the design from the architecture.

An MDA (Model driven architecture) tool is a tool used to develop, interpret, compare, align, measure, verify, transform, etc. models or met models. It can be one of the following types

- Creation tool - used to create an initial set of models.
- Analysis tool - used to analyze models for the completion or any inconsistencies with the real world on which the model has been created.
- Transformation tool - used to generate code or documentation from the created models.
- Test tool - used to test the created models for quality.
- Simulation tool - used to simulate the execution of the system represented by the model.

Various other tools also exist as composition tool, metadata management tool, reverse engineering tool, etc.

### 5.1.2 MDA implementation apprehensions:

- Since MDA is based on development using models the expertise required for it is limited in the current world. Due to majority of developers being trained in traditional software development, people skilled to implement MDA are hard to find.
- The MDA was originally proposed by the OMG group, which has failed to maintain a good track record given the way CORBA, another technology of OMG has fared.
- MDA approach is based on incomplete standards.

### 5.1.3 Advantages

Model Driven Development gives architects the ability to define and communicate a solution while creating artifacts that become part of the overall solution.

1. Model driven development is faster. In MDD, as compared to normal programming languages, the model is created on a higher generalization level, which is automatically transformed into a working software application. Thus more functionality can be built in the same amount of time.
2. Using MDD reduces cost of developing software. This is because it takes shorter time to market.
3. MDD results in improved quality. This is due to the fact that, in developing applications using MDD instead of a bunch of people doing the development, an engine is used to generate code from the high level models. So the quality of the software depends on how excellent the engine used to generate it is. The software therefore is devoid of common human errors, which would be evident in applications not built out of MDD.
4. MDD gives power to the domain specialists. It imposes a clear bifurcation of concerns and skills. People specializing in the field knowledge can focus on creating the models while the technical experts can put their energy in building tools that would be required for MDD.
5. Less sensitive - Changes in the needs of the business or constantly evolving technology won't have much impact on software development on the MDD.
6. Using MDD results not only in creation of software but also documenting domain knowledge in models. Also the documentation is never incomplete or outdated since the model itself is the documentation.

7. The people with the business knowledge can create the software models so you're less likely to lose anything in translation.
8. Since models are graphical they are easier to understand by all those concerned in the software project.
9. The final view of the product can be obtained much earlier in the software development lifecycle if developed on the basis of models. Thus any discrepancies can be resolved much earlier.

### 5.1.4 Disadvantages

1. MDD introduces a lot of rigidity. Since it requires programming at a higher level of abstraction, it becomes difficult for people used to programming everything by hand.
2. Models are lesser flexible in terms of change. Flexibility is limited by the kind of tool being used.
3. The roles of project members are quite different. The real technical or programming part is mostly moved to the team building the model driven software factory.
4. The model developers need to consider all the possible real world aspects before freeing onto a particular model.

### 5.1.5 Reasons For Failure

1. Due to limited skill set of people engaged in modeling, the models turn out to be incomplete or incorrect or turn out to be rigid thereby posing difficulties in scalability.
2. The tools used to develop the models can themselves be buggy resulting in defect-prone models.
3. Freezing on the right number of models for a given software project can be very subjective and may vary as per an individual's understanding of the domain.

4. Since the models can't be verified easily, ensuring quality models becomes a difficult task.

### 5.2 Service Oriented Modeling

Service oriented model is a practice of software development that provides solutions to problems based on modeling disciplines and languages. This approach takes into account the holistic view of the business while performing its analysis and design. It is mainly about simulation of the real world. It includes mimicking the actual product on a smaller scale to initially only include its key features, and then scale it to meet the actual requirements.

The three driving principles of service-oriented modeling are:

- **Virtualization** - it includes creating visual models of the software elements by considering them as real physical objects.
- **Metamorphosis** - all the software that is being created needs to be scalable and flexible so that it could be adapted to the ever-changing business requirements.
- **Literate modeling** - programs written to realize the model should be clear, concise and readable so as to fulfill the need of documentation for that software.

#### 5.2.1 Service Oriented Modeling Disciplines

1. Service oriented conceptualization - Creation of a common language or vocabulary to be used throughout the project lifecycle.
2. Service oriented discovery and analysis - This involves identifying the potential candidates for services based on the above-mentioned principles.
3. Service oriented business integration - It involves identifying as to how the services can be associated with the business processes.

4. Service oriented design - Establish how the services would communicate with each other.
5. Service oriented conceptual architecture - Identify the technological stack that the service would be implemented on.
6. Service oriented logical architecture. - Integrate the service with existing business assets and also determine the ways in which the service could be reused.

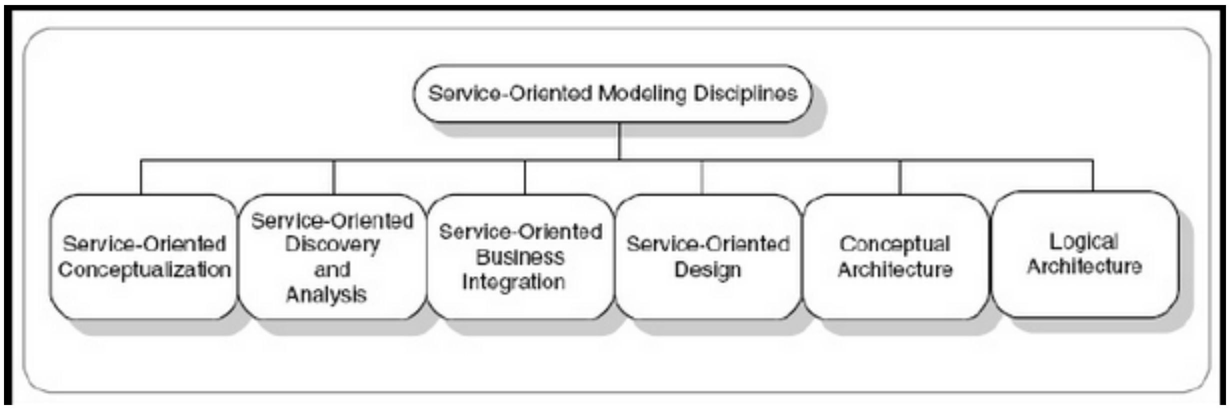

Service-Oriented Modeling Disciplines
Service-Oriented Conceptualization
Service-Oriented Discovery and Analysis
Service-Oriented Business Integration
Service-Oriented Design
Conceptual Architecture
Logical Architecture

**Figure 9 Service-oriented modeling discipline**

Based on its various attributes, a service can be categorized into the following:

- Atomic - Services that cannot be further subdivided into smaller units are atomic services. These services can exist independently.
- Composite - multiple atomic services are integrated to form composite services. This integration is based logically so that related services get grouped to provide one big unified service.
- Service cluster - Distributed services are congregated together based on the common functionality they expose.
- Cloud services - It is similar to a cluster of services; but these are provided using cloud infrastructure. These include PaaS, IaaS, Saas etc.

### 5.2.2 Working

Service oriented modeling makes a clear distinction between the business logic and the platform on which the software is built. The success of service-oriented modeling depends on how well the business analysts gel with the technical experts to create a feasible software solution. The business analysts are closer to the customer and hence find it easy to develop the models. The technical experts are more acquainted with the technology that would be used to transform the domain models into working software. The collaboration between these two groups goes a long way in the success of the service-oriented model itself. In service oriented modeling the models are theorized prior to the actual software development tasks. Throughout the modeling process the underlying key concepts that are followed are reusability and service discoverability. Getting the service model right will ensure that the organization succeeds in its service oriented architecture or SOA.

Conceptual model of a Service oriented style.

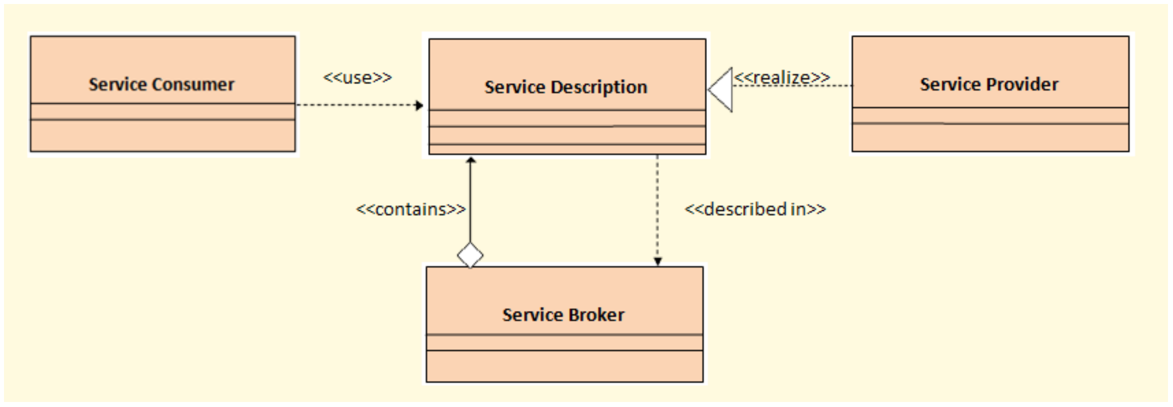


**Figure 10 Conceptual model of a service-oriented style**

### 5.2.3 Advantages

- Service oriented modeling contributes to developing loosely coupled software systems.

- It enhances the reusability of software assets since the components are loosely dependent on each and hence can easily be replaced.
- Also reusability results in less time to market these products and thereby leads to reduced software development cost i.e. reduced expenditure.
- Since the modeling is done on the basis of real world assets, there is a simplification in the terminology, which makes it easy to understand.
- Since service oriented modeling is based on the high level view of the organization's assets, it can be understood much clearly by people with varied levels of understanding.
- Using service oriented models enables to provide software solutions in an incremental agile manner.

### 5.2.4 Disadvantages

- If the services are not identified appropriately in the conceptualization and identification phase, the whole model may become difficult to scale.
- Like each technology, even service oriented modeling would take time to show its effects. It's essential therefore that the managers don't get impatient and expect overnight success.
- A thorough analysis of the business activities and a well-analyzed construction of solution are essential for service modeling to be of benefit.
- A well defines set of tools, framework and resources is required to realize service-oriented model in the best possible manner. Often this can prove to be quite costly for an organization.

# Chapter 6: Survey Results

## 6.1. Overview

To get more real facts and figures about anything, it is always good to have suggestions from the domain experts. Domain experts not only rely on the theoretical knowledge but they do have practical knowledge. It is more important to have practical knowledge because software development is purely practical where you have to convert the designs of the solution into full fledge working examples. Due to this reason I made a questionnaire and I contacted some professionals in the industry working on software development. I sent them the questionnaire online and they filled it and returned back to me. I compiled the results and come to some conclusion which is discussed in the conclusion chapter. The questionnaire was distributed to 100 people and this is the result was obtained from the answers of the questionnaires.

1. What is your designation in your company?
    - o Developer
    - o Designer
    - o Project Manager
    - o Analyzer

This was asked in order to have an idea about the people participated in the survey. A layman person in terms of software methodologies will be of no use as far as this questionnaire is concerned. The view software methodology best, feedback can be given by the developers or project managers. The result for this question is presented in the form of following graph.

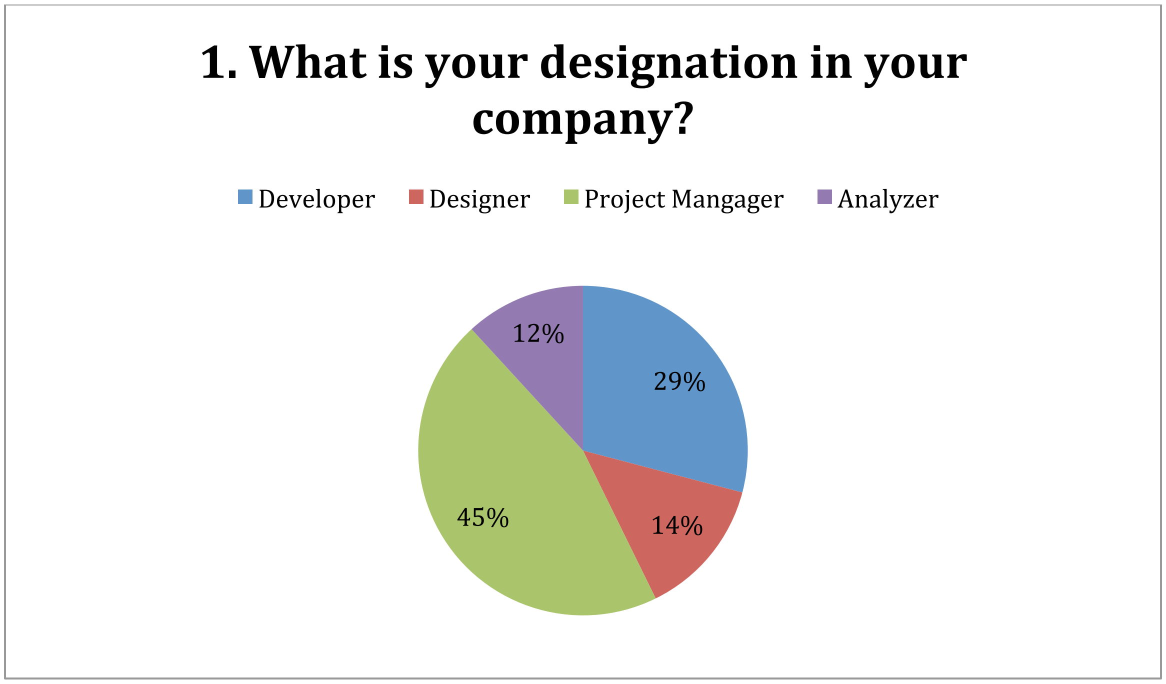


**Figure 11 Survey result: question 1**

2. Do you think software methodologies play important part in success of the project?

    - Strongly believe
    - Strongly disbelieve
    - Believe
    - Disbelieve

This question was asked to know the perception of a person about the software methodology. Some people concentrate on work and are not concerned about the methodology. This is the reason that this question was asked. This question is checking the opinion of experts about methodologies. There are different types of views people are having about this question. The chart is explaining the domain expert's opinion about this question.

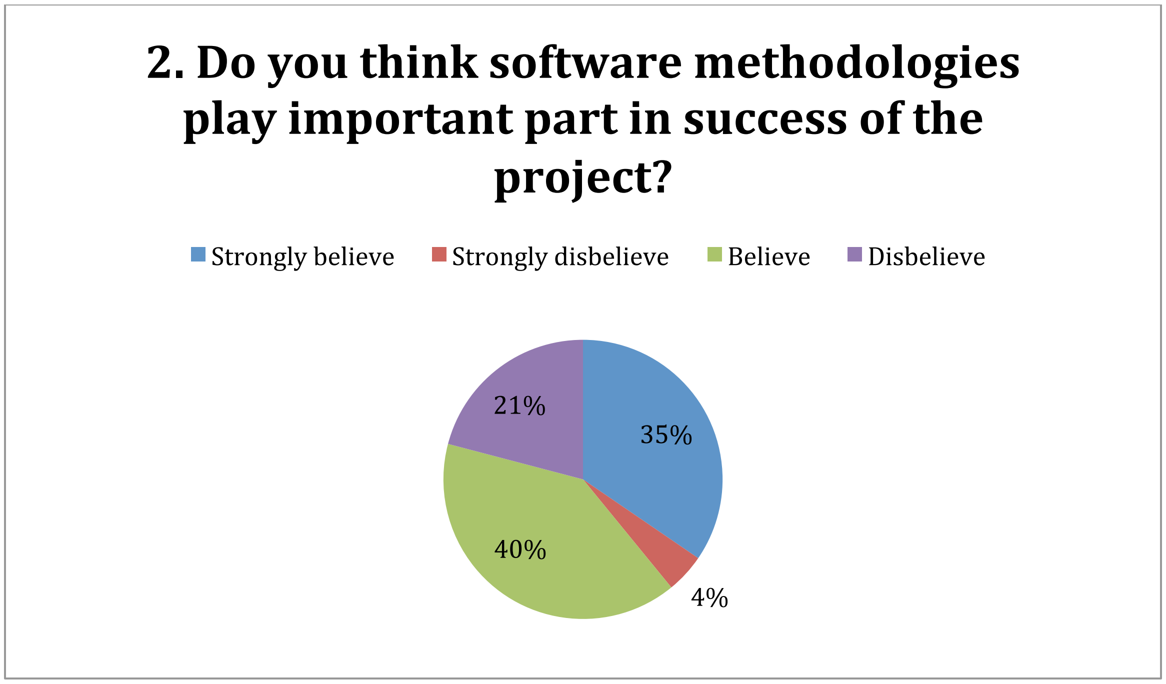


**Figure 12 Survey result: question 2**

3. Which software methodology do mostly you have used or have seen to be used by others?
    - Waterfall
    - WaterSluice
    - Agile
    - Rapid Application Development

This question is asked in order to know the opinion of the people filling survey forms about methodologies. The given options are about different methodologies. Some of the given methodologies are also discussed in this thesis also. Results have shown that people are very much fond of the agile methodologies and waterfall methodologies. The purpose behind this can be the fact that they are traditional approaches and people are used to these methodologies. WaterSluice is the perfect blend of sequential and iterative approach but still this approach is not so popular in the professional network. This might be because of the fact that companies are reluctant to take risks about the projects. They are reluctant to try out new things. This is not a healthy thing about the software industry. Following is the result about the result of the asked question.

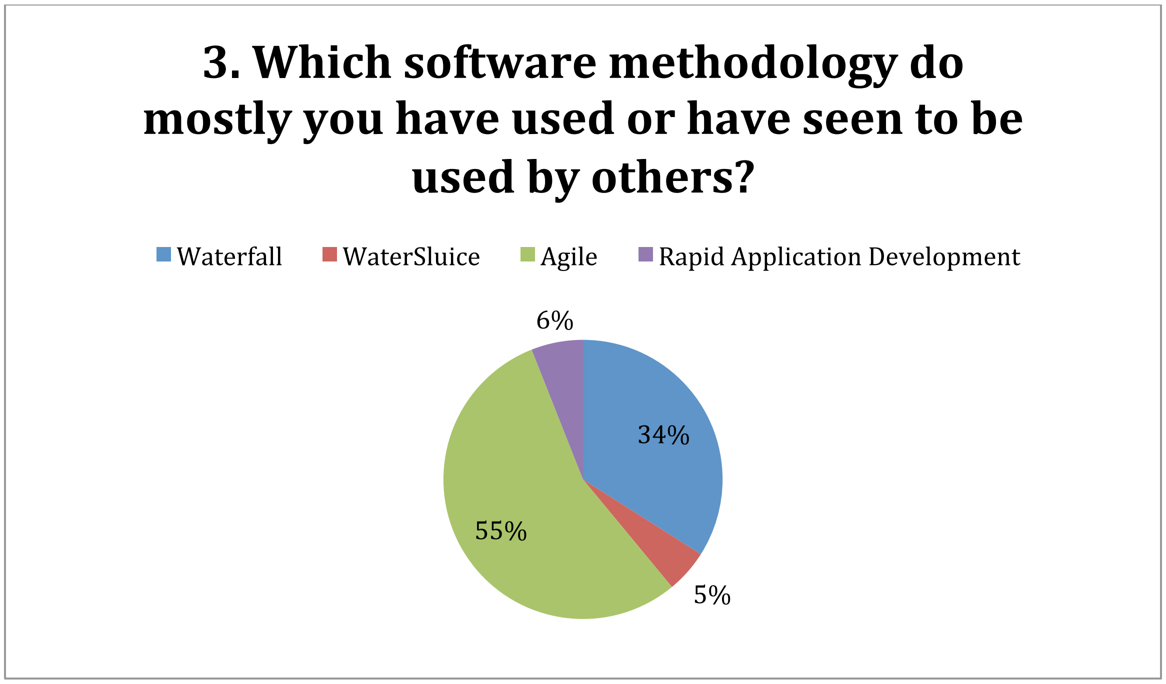


**Figure 13 Survey result: question 3**

4. Which methodology in your opinion, gives the most throughput?
    - WaterSluice
    - Agile
    - Extreme Programming
    - Scrum

This question was asked to the experienced project managers to know about their opinion that which is the best methodology to use. They use the methodologies practically so they know the limitations and problems they encounter due to use of particular methodology. The opinion about the answer of the question is given below. Most of the professionals were in the favor of agile methodology. This is because of the fact that it has the maximum adaptability capacity as compared to the other methodologies.

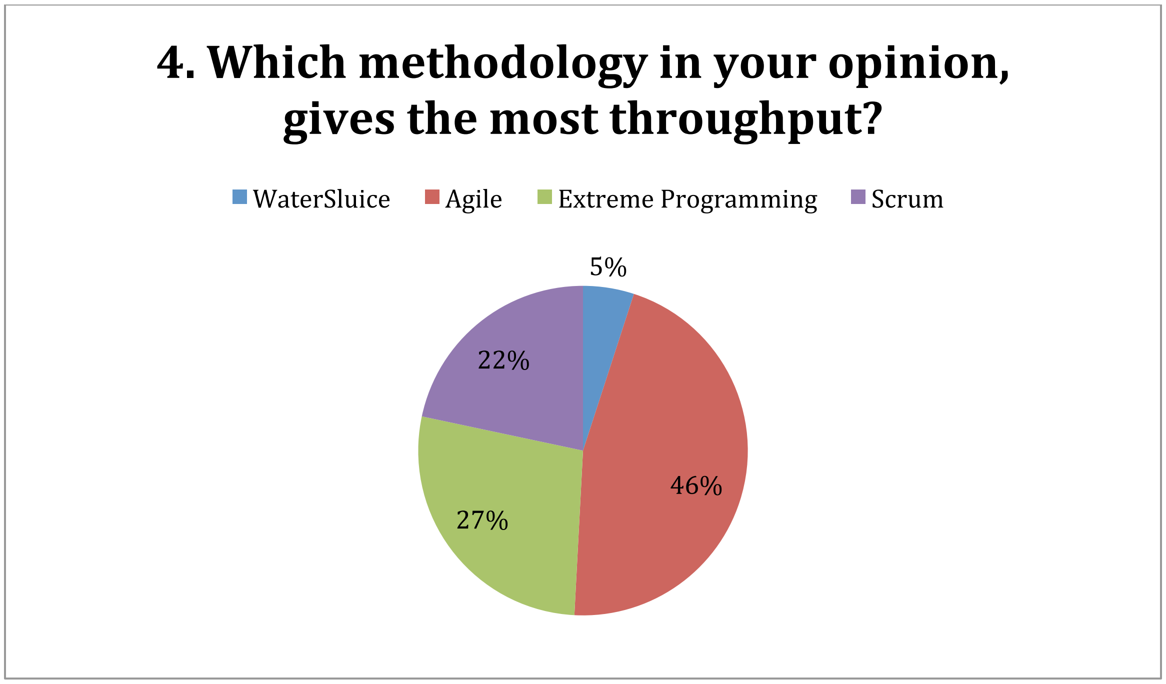


**Figure 14 Survey result: question 4**

5. Which factor must be evaluated first in order to decide about the methodology?
    - Project Size
    - Project Time Limit
    - Development Team
    - Requirements

This question is to evaluate about the importance of any factor towards the selection of the methodology. There are many factors discussed in this regard but the most important factor is the requirements. The clarity or ambiguity of the requirements decides which methodology will be adopted. This means that it has much influence on the end result of the software projects. The percentage given by the professionals about the requirements is also very high.

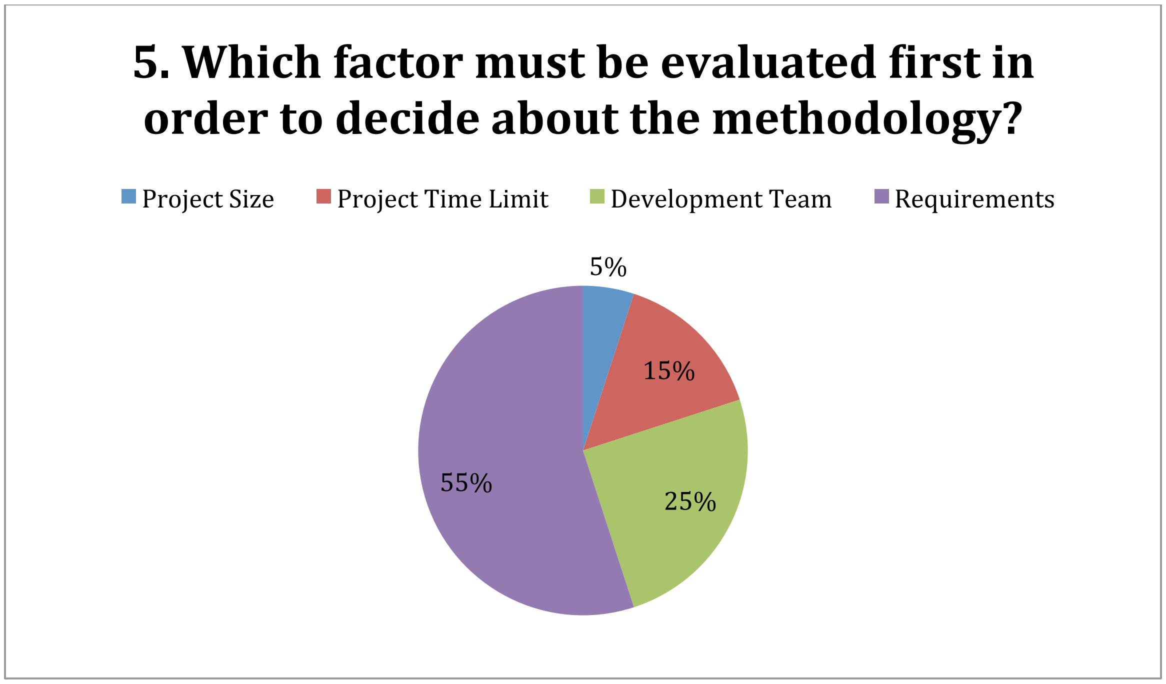


**Figure 15 Survey result: question 5**

6. Do you think that present traditional methodologies are enough for software development?
    - No
    - Yes, they are so much advanced
    - Some research would be enough
    - I, don’t follow methodologies, I follow my own rules

This question inquires about the developers and manager wishes or personal opinions. Are these traditional methodologies enough for the projects or they need some new methods, which are more adaptable for their problems and reduce risks. Most developers are in the view or opinion that there is a severe need of new methodologies to be developed. This would be because they might be feeling a need of new methodologies.

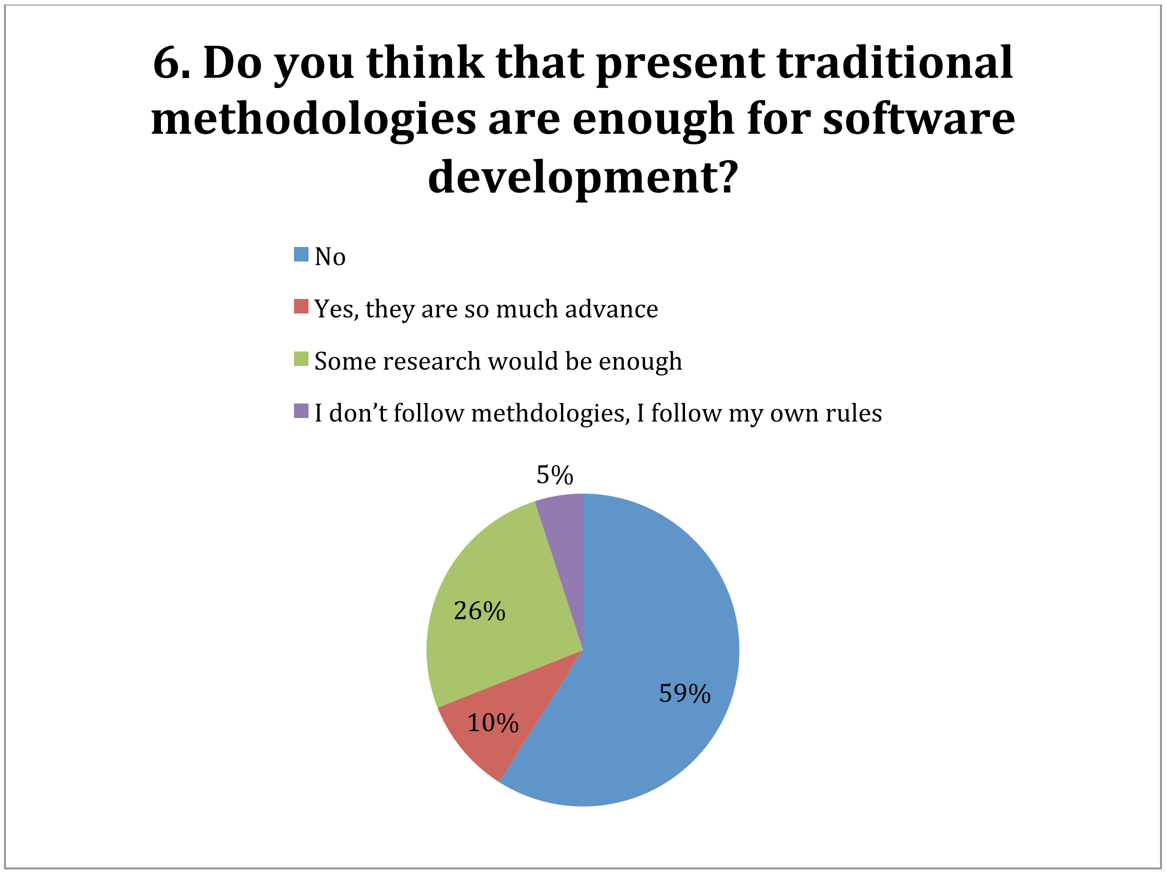


**Figure 16 Survey result: question 6**

7. Which methodology should be used when the team is inexperienced and you want to increase the collaboration with them on daily basis?
    - Extreme Programming
    - Scrum
    - Agile
    - Waterfall

This question now is specifically targeting the agile methodologies. There are many categories of agile methodologies and they are selected in certain conditions. This question is talking about the condition of new companies. Most beginners have inexperience people and the managers need to collaborate on daily basis to have success in the projects. In such conditions most of the employees are in the favor of scrum methodology because in this methodology, everyday 15 minutes meeting with the staff is recommended.

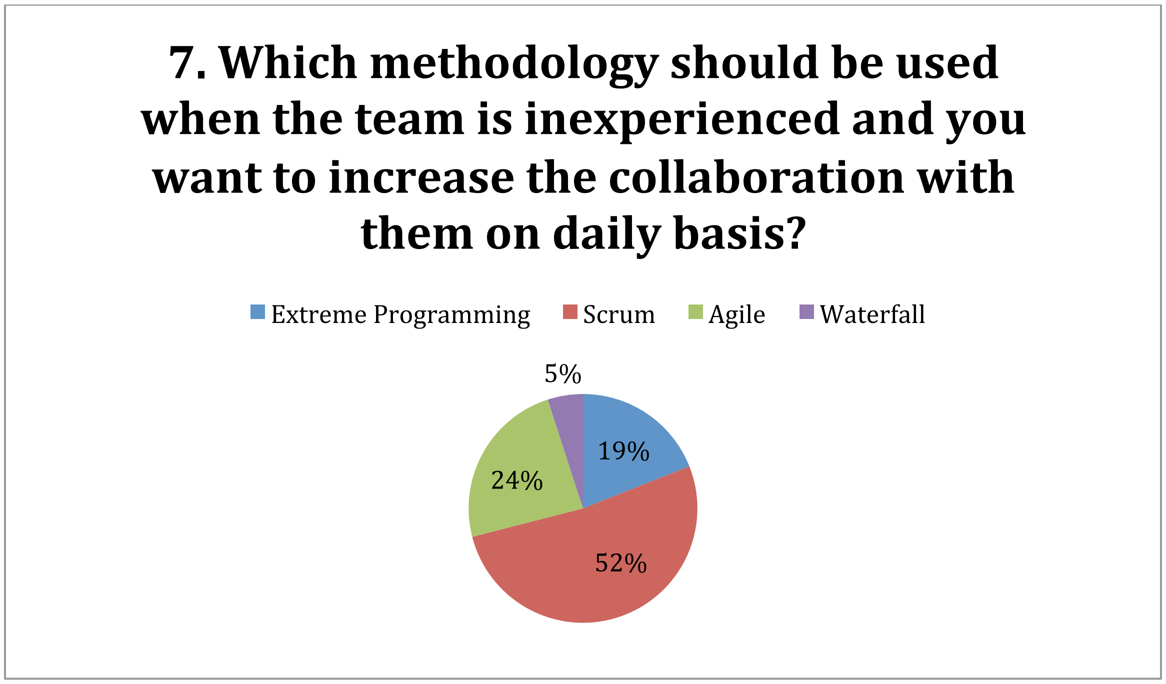


**Figure 17 Survey result: question 7**

8. Which methodology is best when the project has a lot of development and time limit is strict?

    - Extreme Programming
    - Scrum
    - Lean Development
    - Feature Driven Development

This question is again about the conditions particular to selecting any specific category of agile methodology. This time condition is different. The condition is when overall the time limit of deadline of the project is tough and development is intensive. Most of the developers are in favor of rapid application development. This is because it allows the developers to concentrate on the business logic instead of making the overall structure of the solution. It can automatically convert the classes and attributes from user case diagram to source code.

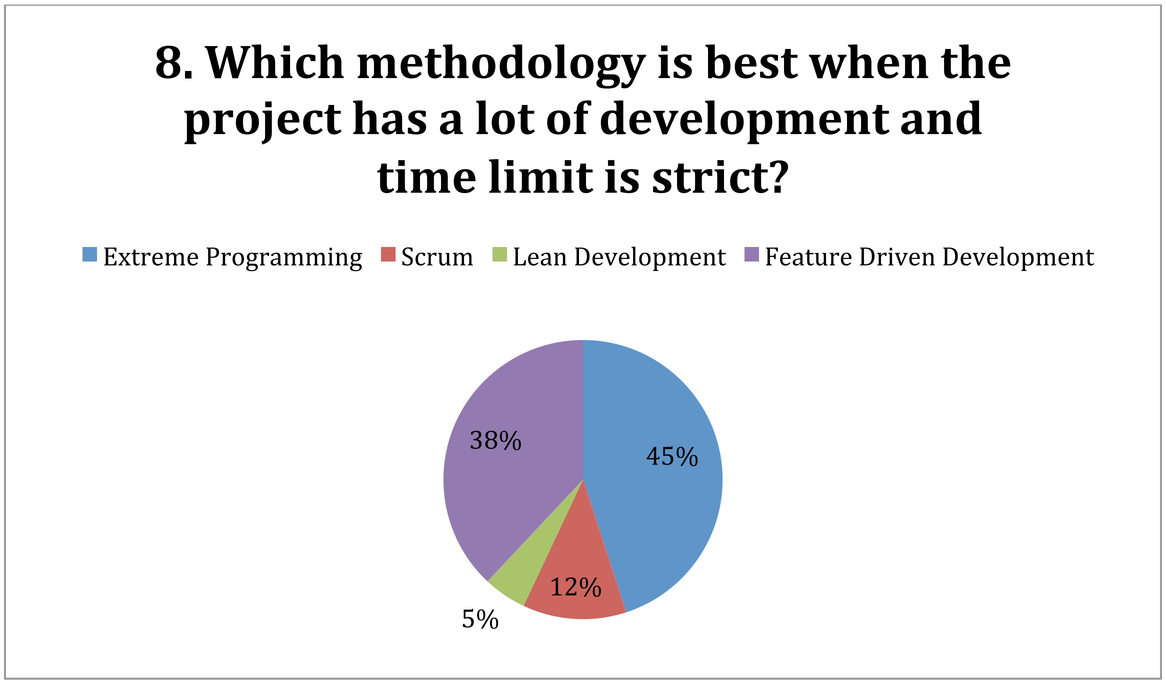


**Figure 18 Survey result: question 8**

9. Which methodology is best when modules or functionalities should be prioritizing in development?
    - Waterfall
    - Agile
    - WaterSluice
    - Extreme Programming

This question is again about a particular condition, which is about prioritizing. This means that if features have to be developed on the basis of priority, then what methodology should be followed. Most of the professionals selected about agile. However, I personally believe it should be WaterSluice. Since WaterSluice methodology is not in practice, professionals cannot know that it is a perfect blend of linear and iterative approach besides concentrating on the priority factor.

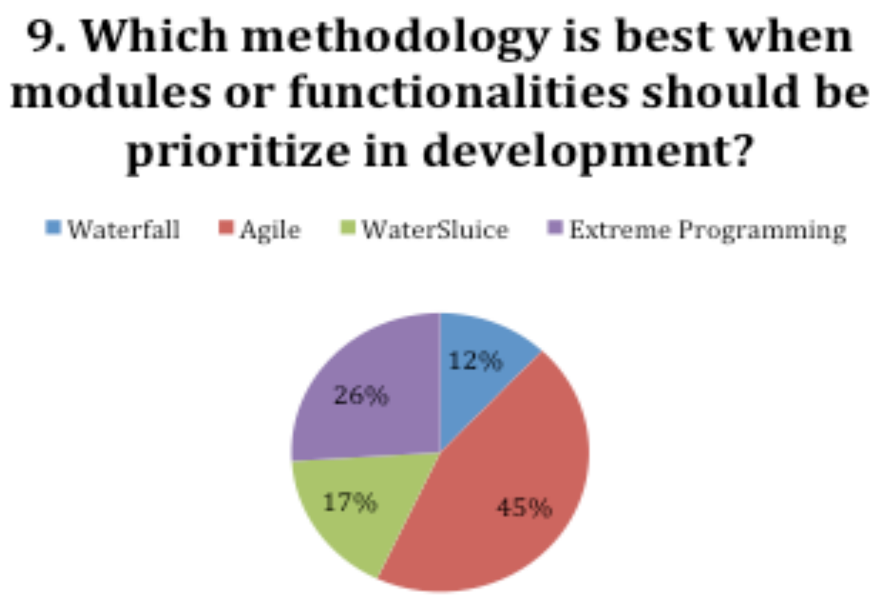


**Figure 19 Survey result: question 9**

10. Have you ever heard about a methodology, which improves the teaching and learning throughput of the software development courses?
    - No
    - Seriously, Are you joking?
    - Yes, I have read a thesis
    - No idea, might be there is something like this

This question was included to know that how many people know and are curious to know about the new methodologies. Survey result have shown that majority of the people are unaware that such a methodology exists. It clearly shows lack of reading technology related thesis especially the new methodologies.

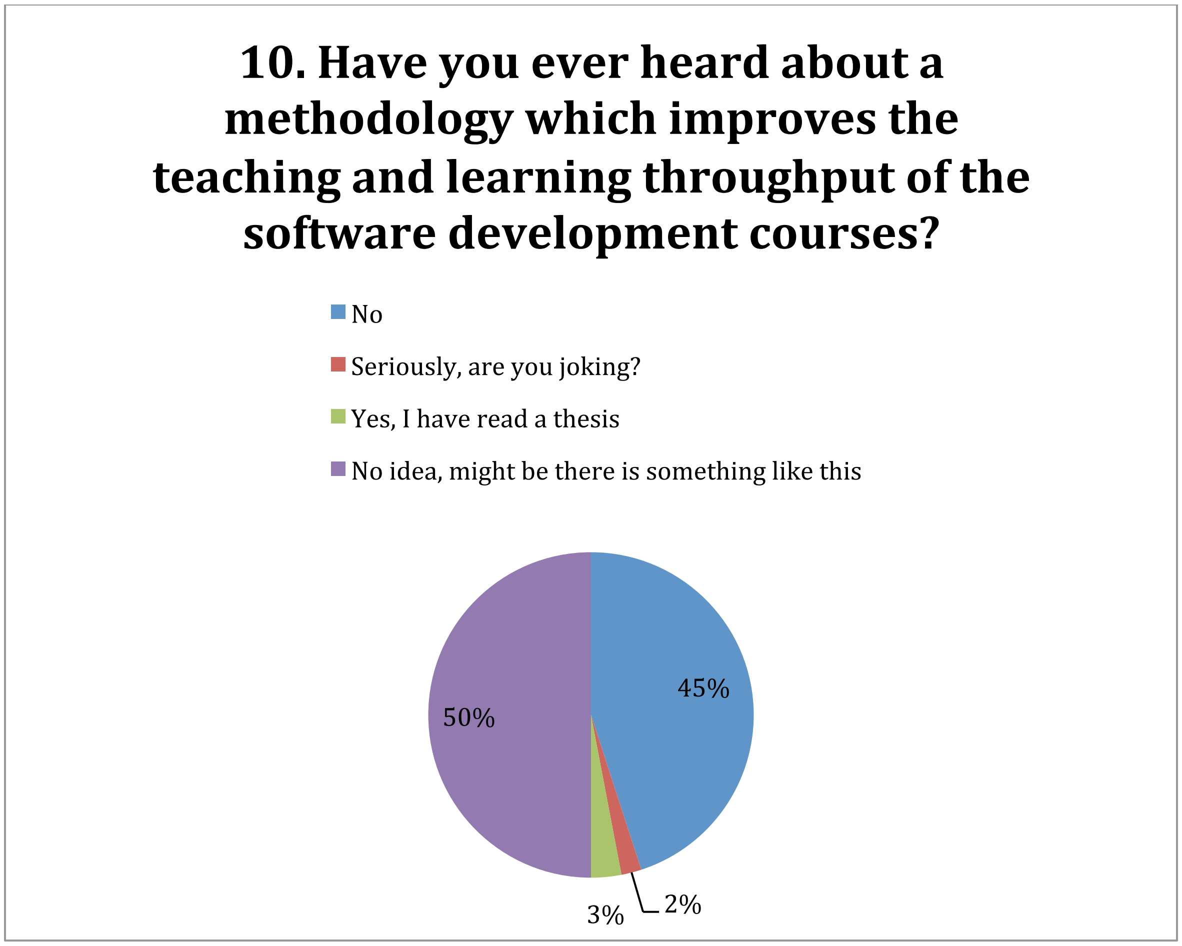


**Figure 20 Survey result: question 10**

11. Which of these software development methodologies have you personally used?
    - Flowcharting
    - Waterfall
    - Agile Unified Process

- Test Driven Development
- Agile Methodologies
- Structured Programming
- Object-oriented Programming

This question was added to check the popularity and usage percentage of overall renowned methodologies. Some of these methodologies have not been discussed in the thesis because thesis is specifically concentrating on the newest and latest technologies.

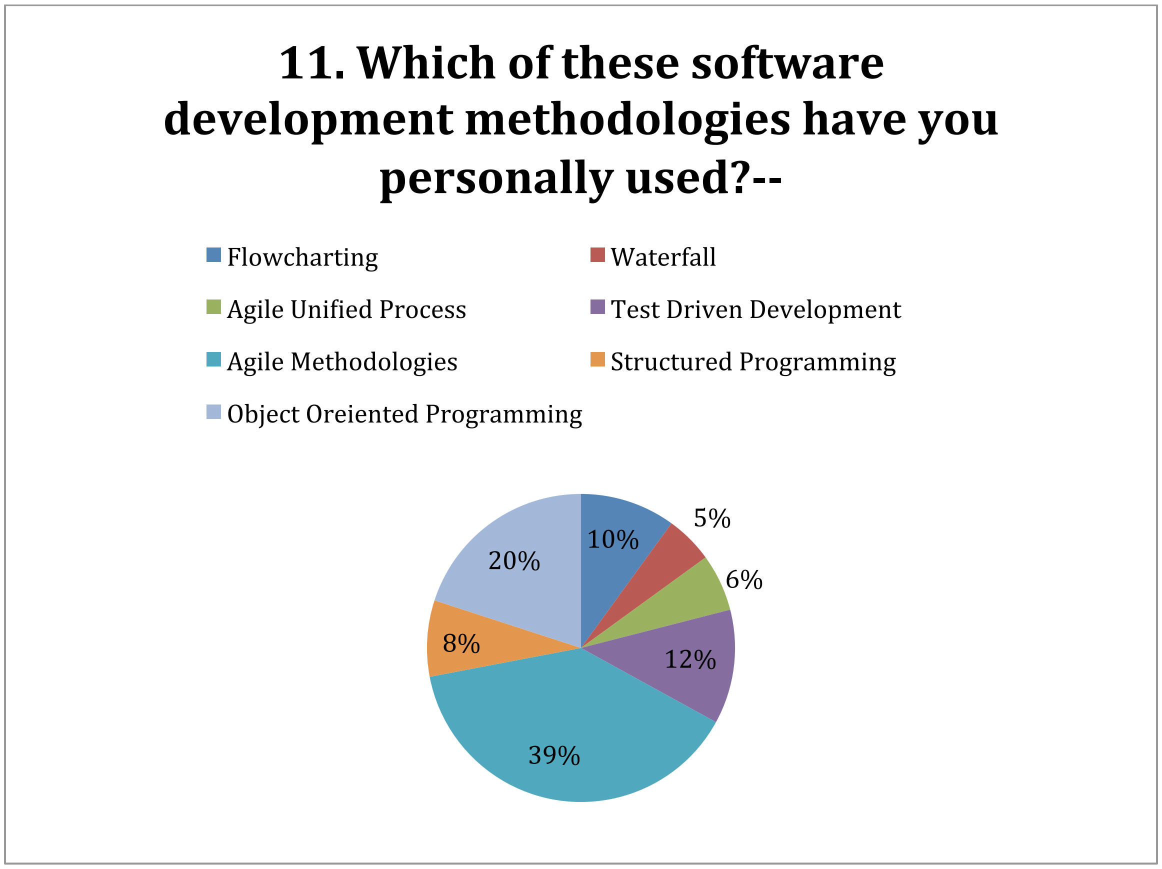


**Figure 21 Survey result: question 11**

12. Management should provide flexibility for employees to form project teams.

- Strongly believe
- Strongly disbelieve
- Believe
- Disbelieve

This question is not particularly attached with the methodologies. Rather it is concerned with the overall attitude of management towards the employees. This is because it is the employees who make the project successful, not the methodologies.

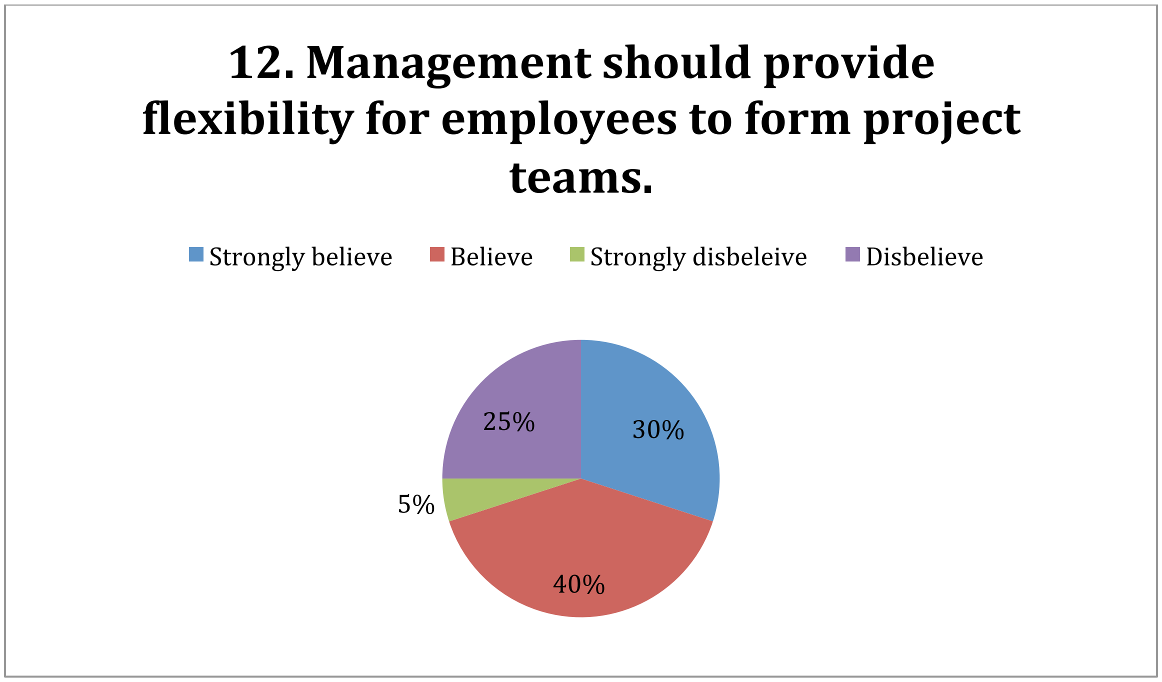


**Figure 22 Survey result: question 12**

13. Which of these software metrics have you personally used?

- Order of Growth ("Big O")
- Source Lines of Code (SLOC)

- Cyclomatic Complexity (“McCabe”)
- Bugs per line of Code

There are many metrics in the market to analyze the development quantity. Sometimes it is estimated for payment purposes, other times for the performance measures. This question asks about the ways in which is the most popular.

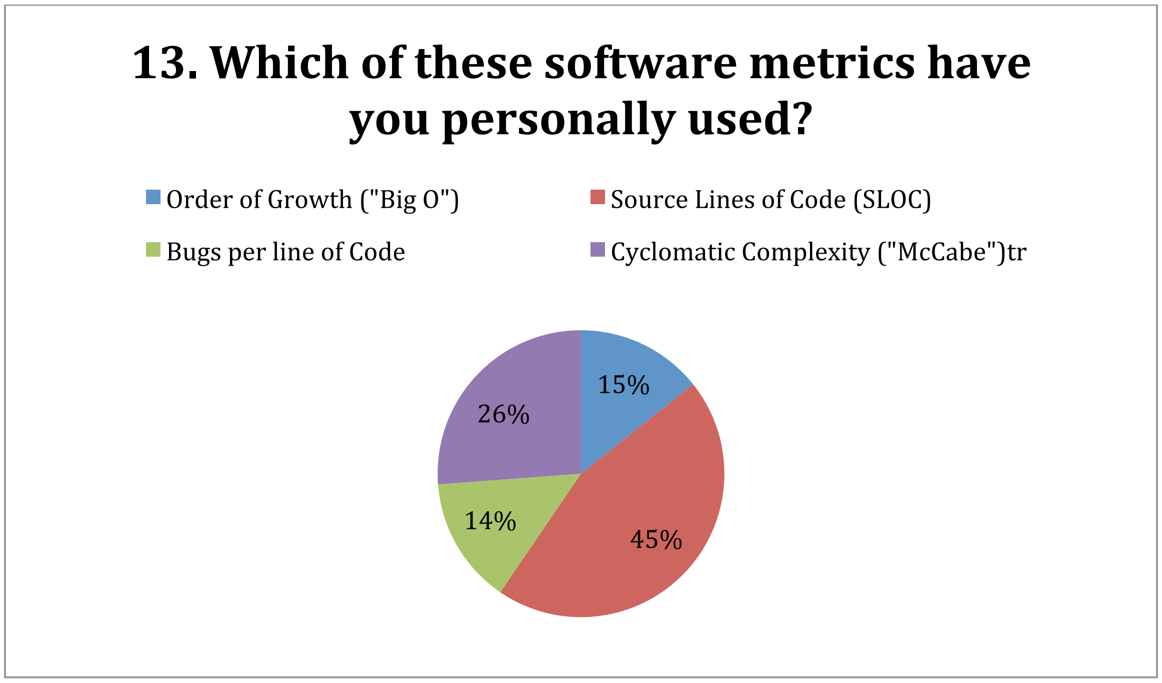


**Figure 23 Survey result: question 13**

14. There should be general guidelines and principles for software development but not detailed rules?
    - Strongly believe
    - Strongly disbelieve
    - Believe
    - Disbelieve

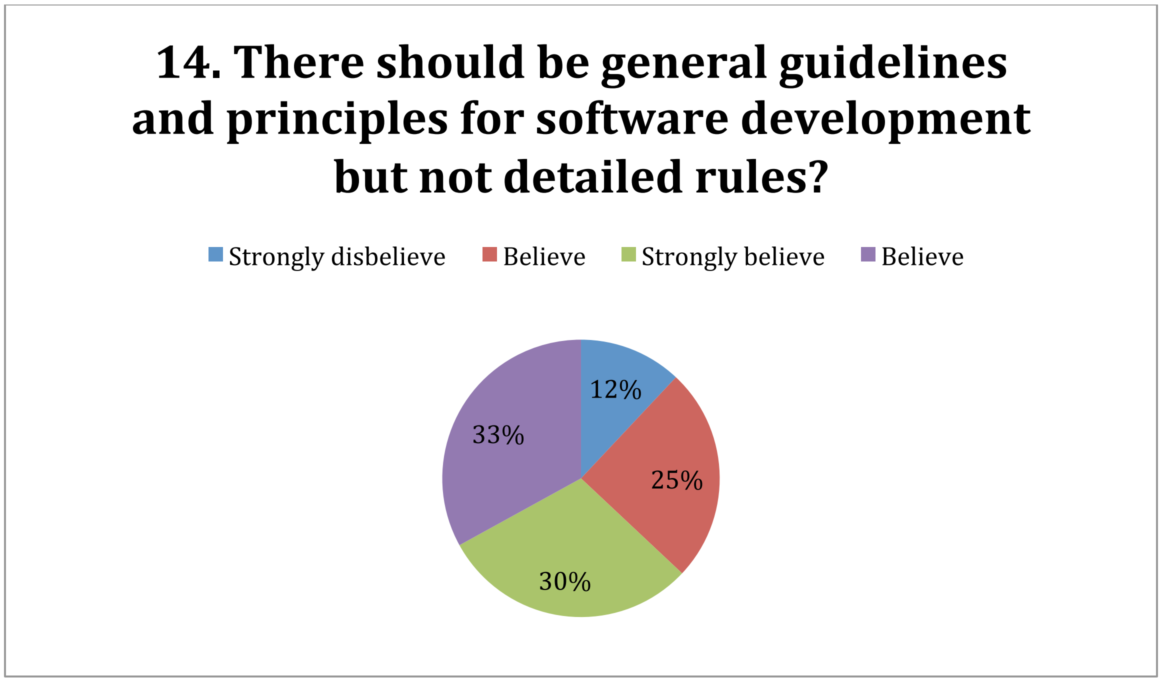


**Figure 24 Survey result: question 14**

15. Project members should frequently meet to update each other on progress of the project?

- Strongly believe
- Strongly disbelieve
- Believe
- Disbelieve

This question stresses upon the importance of meeting of the developers on a routine basis. This can help them exchange their problems and difficulties. This will increase their collaboration, which in return will make them better stakeholders of the project.

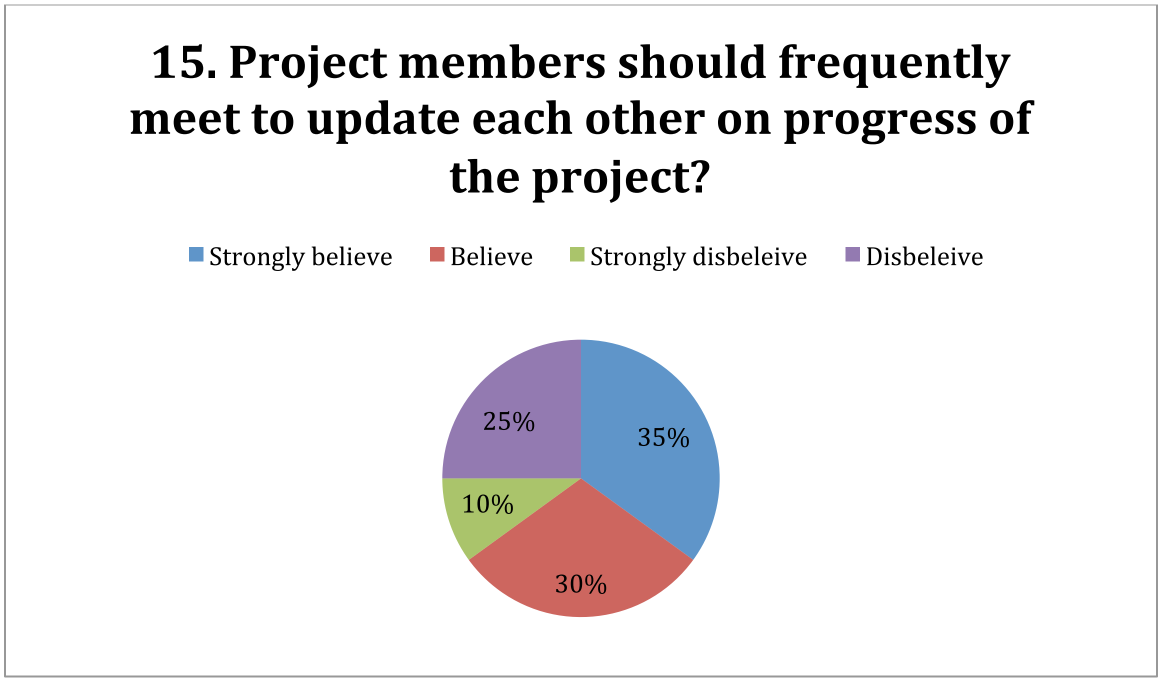


**Figure 25 Survey result: question 15**

## 6.2. Conclusion

All the questions were very carefully selected and there was a significant consideration after each question. The answers of participants were in favor that present methodologies are not enough to fulfill the problems of software development and there is a need of research for this. All the people who filled the survey were professional and domain experts in the software development industries. They were contact through LinkedIn and other professional networks, which means that their opinion is very accurate and right. There are many limitations of the software industries and this is the reason that there is a lot of space and room for research about the software methodologies. Researchers are carrying out research in order to cross the bridge of limitations.

# Chapter 7: Conclusion

All these methodologies, which are covered in the thesis, are not in extensive practice as far as the software industry is concerned. This is because there is a risk involved in each and every new thing comes into the market. Technology is evolving so much, therefore, there is so much to develop and more room to develop. As software developers are doing new projects, the need for new and adaptable methodologies is increasing.

## 7.1. WaterSluice

Taking a look at the survey results, we can conclude that there are so many less developers, designers, project managers who are aware of this methodology. This methodology is the perfect blend of sequential and iterative approach. It takes advantages of both approaches and eliminates the limitations of both methodologies. Apart from this, it has another critical feature, which is development on priority. It is essential to prioritize the features, which are developed with more concentration. In short, this methodology must be followed more often and developers should start taking risks and reduce following traditional approaches.

## 7.2. Agile

Agile methodology is having many categories and with each passing day, people are coming up with more and more different strategies. As a result, it has maximum capability to adapt with the new environment and cope with the risk factor, which is considered to be the best feature of Agile.

## 7.3. Strategy for Teaching and Learning Software Development

This methodology is about improving and taking maximum throughput of learning and teaching mechanism in software development courses. This strategy is very new and it is under discussion of panels of software engineers. To conclude, attaching milestones with the iterations of this strategy will help in maximum learning in the software development.

## 7.4. Service Oriented Modeling

Service oriented model has also made its name in the world of software methodologies. This methodology takes the software elements as simulations or real world objects. It focuses on business needs and business integration. This is the reason that this modeling technique is best for the business applications of different types. I failed to examine this methodology in depth due to limited resources and time. Therefore, I am willing to explore it more in the future to find why it is being used and when it is best to use it.

## 7.5. Model driven development

Model driven development concentrates on the design and the implementation or the design evaluates from the design. It seems that implementation will take too much time but it is not the case and the implementation and code is derived from the design itself. Nowadays, this approach is very much followed because of the fact that development is technology has made all the things easy. It travels from abstraction to abstraction. If the design is accurate, there is no need to worry about the whole product.

Overall, it is very important to note that there is no hard and fast rule to select the methodology. The scenario for the project should be analyzed at first. Then, it is good to study the methodologies and choose the best one which fits the scenario and requirements of the project. Moreover, it is very good to strictly follow all the standards and guidelines of that methodology because every methodology has clearly defined these standards according to the suited scenario.

## Bibliography

1. (n.d.). From Scrum: http://www.mountaingoatsoftware.com/agile/scrum

2. *Agile Methodologies for Software Development*. (n.d.). From http://www.versionone.com/Agile101/Agile-Development-Methodologies-Scrum-Kanban-Lean-XP/

3. *Agile Methodology*. (n.d.). From http://www.codeproject.com/Articles/616070/Agile-Methodology

4. Alfonso. (n.d.). Learning Software Engineering with Group Work.

5. *An Introduction to Feature Driven Development*. (n.d.). From http://agile.dzone.com/articles/introduction-feature-driven

6. Burback, R. (1998). SOFTWARE ENGINEERING METHODOLOGY:. 263.

7. *Feature Driven Development*. (n.d.). From http://www.featuredrivendevelopment.com/

8. *Feature Driven Development (FDD) and Agile Modeling*. (n.d.). From http://www.agilemodeling.com/essays/fdd.htm

9. Kancherla, K. (n.d.). WaterSluice Methodology. 15.

10. Larman, C. (2002). Applying UML and Patterns: An introduction to Object-Oriented analysis and design and.

11. *Manifesto of Agile Software Development*. (n.d.). From http://agilemanifesto.org/

12. Maria Isabel Alfonso, A. B. (n.d.). An Iterative and Agile Process Model for Teaching Software. 8.

13. McDonald, K. J. (n.d.). *Agile Method Brief – Crystal*. From http://www.projectconnections.com/templates/detail/agile-techniques-crystal.html

14. PERIYASAMY, J. (n.d.). *What is Waterfall? list its advatages and disadvantages*. From http://www.jobsnewstoday.com/2013/04/what-is-waterfall-model.html

15. *Software Engineering Methodology: The WateraSluice*. (n.d.). From http://infolab.stanford.edu/~burback/water_sluice/sluice6.25.97/ws/watersluice.html

16. *Software Engineering Methodology: The WaterSluice* . (n.d.). From http://www.freetechbooks.com/software-engineering-methodology-the-watersluice-t573.html

17. Zulkefli Mansor, S. Y. (n.d.). Success Determinants in Agile Software Development Methodology.

18. Bieberstein, N., G. Laird, R., Jones, K., & Mitra, T. (2008, 05 05). *A methodology for service modeling and design*. Retrieved from http://media.techtarget.com/searchSOA/downloads/ExecutingSOA_ch04.pdf
19. Hofstader, J. (2006, 11). *Model-driven development*. Retrieved from http://msdn.microsoft.com/en-us/library/aa964145.aspx
20. (n.d.). Retrieved from http://www.theenterprisearchitect.eu/blog/2009/11/25/15-reasons-why-you-should-start-using-model-driven-development/
21. (n.d.). Retrieved from http://www.slideshare.net/awahid/introduction-to-service-oriented-modeling
22. Bell, M. (2008). *Service oriented modeling: Service analysis, design and architecture*.
23. (n.d.). Retrieved from http://www.ibm.com/developerworks/library/ws-soa-design1/
24. (n.d.). Retrieved from http://www.slideshare.net/eelcovisser/modeldriven-software-development-introduction-overview-3061258
25. Stephen , W. L. (n.d.). *Model-driven software development*. Retrieved from http://www.deg.byu.edu/papers/LiddleMDD.pdf