

**University Of Namibia Jose Do Santos Campus | Faculty of  
Engineering and IT**

**Department of Electronics and Computer Engineering**

**Software Engineering Assingnment 1**

**Research Paper on Programing Languages | Agile Methods | System Modeling  
(UML)**

**Name: Mwandila Victor**

**Student No:**

**Lecturer: Mr. P Mulunga**

## Table of Contents

|                                                           |   |
|-----------------------------------------------------------|---|
| <b>Introduction</b> .....                                 | 2 |
| <b>1. Programing language</b> .....                       | 2 |
| a. Low-level Language .....                               | 2 |
| b. High-level Languages .....                             | 2 |
| c. Object-Oriented and System Programming Languages ..... | 3 |
| d. Natural Language .....                                 | 3 |
| <b>2. Extreme programming XP</b> .....                    | 3 |
| Pair Programing approach .....                            | 3 |
| Test first development approach .....                     | 4 |
| Pros .....                                                | 4 |
| Cons .....                                                | 4 |
| <b>3. Agile methods</b> .....                             | 4 |
| Table DSDM and Lean Software Development Principles ..... | 4 |
| <b>4. Unified Modeling Language UML</b> .....             | 6 |
| Diagrams .....                                            | 6 |
| <b>Conclusion</b> .....                                   | 8 |
| <b>References</b> .....                                   | 9 |

## Introduction

The purpose of this paper is to carry out a research on several key topics that are found in software engineering. Firstly, it will elaborate the evolution of programming languages from machine language, low-level language through to high-level languages. This however, will focus mostly on the programming languages of the electronic computer. This paper will also discuss Extreme programming XP, an agile software development method, and will outline the main principle upon which it is based. This paper will continue to discuss agile methods, specifically the Dynamic System Development Methodology and Lean Software Development. It will compare and contrast the principle upon which the two agile methods are based and review any similarity. Finally the paper discusses system modeling, specifically the Unified Modeling Language UML and will elaborate and illustrate its functions.

### 1. Programming language

This is a logical notation of setting up of a set of steps which a computer can follow during its operation in carrying out a task [1]. Programming language is used to describe the solution to a problem that a computer must understand, follow and execute. Because the tasks that a computer must carry out nowadays have become diverse, its basic hard-coded abilities seem too simple and primitive. When a computer is manufactured it may only be able to perform arithmetic and logic operations, input and output and some limited control functions. These initial capabilities are implemented by the use of machine language. Machine language is often too long and complex, it does not give a full opportunity for people to describe clearly the solutions to problems that a computer must carry out. Thus, the high-level languages and eventually NL programming languages were invented. These languages use less primitive language as compared to machine language and are human readable. However, they require a program that can interpret to the computer back into machine language (interpreters and compilers) [2].

#### a. Low-level Language

When the first modern computer was created, Programmers first relied on the low-level language, assembly language. Assembly language can be described as a second generation low-level language from machine code [1]. It is a mapping of human readable symbols which hold symbolic addresses to machine op-code, string and numeric constraints. One line of machine code is typically represented by one line of assembly code. E.g. SUB R16, R17, this instruction would subtract the contents of the register R16 from the contents of R17. Assembly required a lot of intellectual effort and is bug prone.

#### b. High-level Languages

In 1949 *short code* was designed. This was one of the first high-level programming languages for an electronics computer. It represented mathematical expressions in a human readable form. *Short code* statements, however needed to be converted back into machine language every time they were run. Nevertheless, a year later, a new language was designed at university of Manchester, *Auto Code*. This language used a compiler to automatically convert the statements into machine language. This

programming language became the first compiled high-level programming language and was used largely for apple's mark 1[2].

### **c. Object-Oriented and System Programming Languages**

After 1960, simula was invented to support object oriented programming, C also eventually came as the first system programming language, developed by Denis Ritchie and Ken Thompson. This language was very key because it's a language for system programming, designed for writing system software and distinct from application software. System software is computer software that operates and controls the computer hardware and provides a platform for running application software. An example of such is an operating system. Later on, languages like BLISS, C++, GO, D came about [2].

### **d. Natural Language**

Now, there are programming languages that use our ordinary language i.e. English. These are the most human readable languages, they are simply known as Natural Languages NL. They use very fast and sophisticated compilers and libraries to be understood by the computer. Inform 7 NL and Wolfram Alpha are some examples.

## **2. Extreme programming XP**

This is a software development process as well as a methodology. It describes who is doing what, when and how. It describes principles, techniques and practice that ensure an efficient and repeatable process of software system production. Consequently, it serves as a template for creation of a project software products. . Some software that are built on such a method would be web applications and small and medium scale software [3].

XP is an agile method which advocates for frequent releases of the product in short development cycles. These intend to improve the productivity of the process and introduces check points where customer requirements can be reviewed and adopted for the successive releases.

XP also includes the phenomenon of pair programming, extensive code review and unit testing. It takes only necessary features for the software under development and incorporates others only until they are needed or important.

In XP, it is expected that customer requirement will change inevitably (possibly after every release), therefore it follows a principle of simplicity and clarity in the code to allow flexibility. Frequent communication between the customers and programmers make it easier for the problems and adjustments to be better understood by the two entities. Code review is taken to extreme levels and it is done continuously throughout the development process via pair programming and test first programming approach.

### **Pair Programming approach**

Pair programming is when two programmers work together at a workstation. One of the two writing the code and the other reviewing each line of the code. From study, this technique has been proven to be 15% more efficiency than individual work [4].

## **Test first development approach**

XP also follows the principle that if a small test can eliminate few bugs, then more tests can eliminate many bugs. Because of this, programmers in this approach write as many test programs as they can to catch out as many bugs in their code.

### **Pros**

XP allows programmers to focus more time on programming since there is less needless paperwork and meetings. It brings a good social atmosphere for programmers to learn new skills from each other. It also gives a frequent sense of shared achievement.

XP creates working software products faster which tends to have very few defects. It gives the customer an option to change requirements at any time.

The delivery process is not expensive because of less formalities and the software is most likely to pass the acceptance test because of the test first approach that is used. . Teamwork improves employee satisfaction, this makes it easier to work with them.

### **Cons**

Even though the pros outweigh the cons, it is good to mention that XP is not easy to implement. It is difficult to find willing and capable programmers to take up the approach. Customers may not like the idea of being involved in the process. It also threatens the jobs of software testers and architects.

## **3. Agile methods**

There are several agile methods that are used today, some example of these methods are XP, Scrum, Lean software Development and Dynamic System Development Methodology. In this discussion I will focus on Dynamic System Development Methodology and lean software Development.

DSDM expresses much of the current knowledge known about project management. It came about in the year 1994 [5]. This method is suited for developing complex software projects and is based on the overall principle of rapid delivery. However it can be described extensively by its nine principles.

Lean development method came from lean manufacturing also known as just in time manufacturing. It was first used by the Toyota Company for their manufacturing processes. Lean software development was formulated from lean manufacturing in a popular book by Tom and Mary Poppenieck [6] [7].

Both lean Software development and Dynamic System Development have become widely accepted and popular in the world of agile. This discussion will define the principles upon which the two methods are based while making comparisons or/and contrasts between the two. The table below outlines the principle of the two software development methods.

### **Table DSDM and Lean Software Development Principles**

| *  | <b>Dynamic System Development Methodology<br/>DSDM</b><br><small>Mwandila   @Unam</small>                                                                                                                                                      | <b>Lean Software Development</b>                                                                                                                                                                                                                                                                                                                                                                                                                          |
|----|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1. | <b>Active user involvement</b>                                                                                                                                                                                                                 | <b>Elimination waste</b>                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|    | This is one of the most important principle throughout all agile methods. The involvement of the user reduces error during the development process. The user's perspective is important when it comes to error detection.                      | In a lean software development, anything that does not add value to the process or product is eliminated. Any unnecessary code or functionality, unclear requirements and slow internal communications.                                                                                                                                                                                                                                                   |
| 2. | <b>Team Empowerment</b>                                                                                                                                                                                                                        | <b>Amplify learning</b>                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|    | The DSDM participants are given authority to make decisions that may slow down the project sufficiently if the normal process was to be carried out. E.g. prioritizing requirement or what functionality to increment.                         | The best way to improve software development is to amplify learning. Increasing feedback via short feedback sessions with customers helps determine current state of development and adjusting work effort for future improvement. Both the customer representatives and developer tend to learn more about the problem domain during these short sessions.                                                                                               |
| 3. | <b>Frequent Delivery</b>                                                                                                                                                                                                                       | <b>Decide as late as possible</b>                                                                                                                                                                                                                                                                                                                                                                                                                         |
|    | Frequent deliver of the software may ensure that defects are quickly identified and fixed in the next delivery.                                                                                                                                | Decisions are not made in haste, teams wait until the decision is based on facts and not uncertain assumption. This prevents later cost of reversing a wrong decision.                                                                                                                                                                                                                                                                                    |
| 4. | <b>Fitness for business</b>                                                                                                                                                                                                                    | <b>Deliver as fast as possible</b>                                                                                                                                                                                                                                                                                                                                                                                                                        |
|    | DSDM requires that the delivered software is good enough to solve business needs even after any enhancement in a later iteration. It requires that the business needs are the priorities. Any alteration to this is unwanted and is discarded. | Technology is all about speed and time today, the faster the better. In lean software development it is important to deliver the end product faster and with little defects. This makes it possible for quick feedback which is incorporated in the next iteration. Beside, customers like fast delivery.                                                                                                                                                 |
| 5. | <b>Iteration and incremental Development</b>                                                                                                                                                                                                   | <b>Empower the team</b>                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|    | Complex software projects are not easily manageable, therefore there is need to decompose the software into small feature packages which can be added to the mother component until the business requirements are fulfilled.                   | To encourage progress, developers are allowed to do their jobs without waiting for orders and make their own decisions. Management needs to respect, motivate and give the teams purpose to work hard.                                                                                                                                                                                                                                                    |
| 6. | <b>Reversible changes</b>                                                                                                                                                                                                                      | <b>Build integrity within</b>                                                                                                                                                                                                                                                                                                                                                                                                                             |
|    | All changes during the development process must be reversible. For this principle, there are modern software tools that are implemented.                                                                                                       | Customers need to have an overall experience of the system, how it will be delivered, deployed, its price etc. This is known as perceived integrity. The system's separate components must work as a whole with balance between flexibility maintenance and efficiency. This is conceptual integrity. For this to be possible, the system's domain problem must be understood deeply so that it can be solved at the same time instead of a sequentially. |
| 7. | <b>Requirements must be base lined at higher-level</b>                                                                                                                                                                                         | <b>See the Whole</b>                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|    | This principle implies that there is a limitation to the degree of freedom to which requirements can be changed during the development process of the software.                                                                                | The best parts do not necessarily make the best system; the ability of a system to achieve its purpose depends on how well the parts work together, not just how well they perform individually. Therefore lean takes an                                                                                                                                                                                                                                  |

|    |                                                                                                                                                                                                                                                                             |                                                                                |
|----|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------|
|    |                                                                                                                                                                                                                                                                             | approach of optimizing the whole system and how different parts work together. |
| 8. | <b>Testing is integrated throughout the process</b>                                                                                                                                                                                                                         |                                                                                |
|    | DSDM requires that testing be done from early stages of the development process till the end of the process.                                                                                                                                                                |                                                                                |
| 9. | <b>Collaborative and co-operative approach</b>                                                                                                                                                                                                                              |                                                                                |
|    | It is important in DSDM to give technical staff and business an atmosphere of trust and honest, while avoiding separation and encouraging collaboration. This is mandatory throughout this process. Without this principle, it would be hard to gather honest requirements. |                                                                                |

Table 1. The principles of DSDM and Lean Software Development

### Review

Both DSDM and Lean Software development share common principles like customer involvement, fast delivery and team empowerment. They focus on delivering high quality products to the customers meaning that they try to minimize the defects in the product. Lean and DSDM may differ in some way but the overall aim is the same. They do not show any contradictory principle as far as there principles are concerned.

## 4. Unified Modeling Language UML

It is a system modeling notation or language that represents, graphically the design of a software system. It provides a standard way of visualizing the design of a system's architecture. UML represents two views of a system model; static and dynamic views. The static view or structural view is concerned with modeling the system's attributes, objects and operations however, the dynamic or behavioral view is concerned with the relationship between objects of the system and their internal states.

When complex system designs cannot be described by text only, UML becomes very handy. It has three fundamental benefits i.e. Visualization, validation and clear communication [8].

UML for these reasons has been adopted by the object management group and also accepted by the international organization for standards.

### Diagrams

As mentioned earlier, UML represents two views of a system design, namely structural and behavioral views. This is done by the use of diagrams.

#### a. Structural diagrams

This depicts the elements of specification that are irrespective of time, static. They include classes, composite structures, components, objects and packaged diagrams.

## b. Behavior Diagram

This depicts the behavioral features or interaction of the static features in the system. It includes activities, state machines, use case diagrams and the four interaction diagrams

## c. Interaction Diagrams

This is a subset of the behavior diagram. It emphasizes object interactions and includes communication, interaction over view, timing diagram and sequence diagrams.

The following figure show the hierarchy of UML

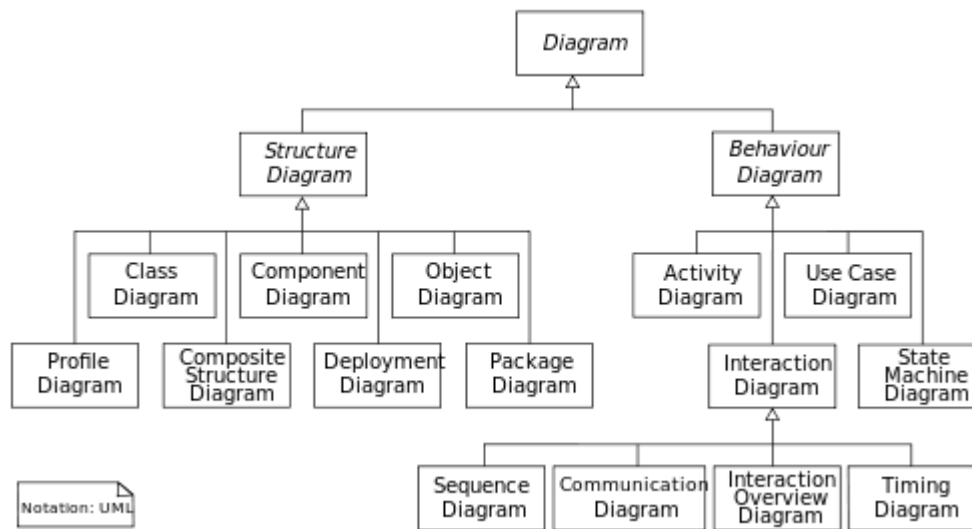


Figure 1 UML hierarchy [9]

UML is used to model extremely large software systems that maybe very difficult to describe using text or speech.

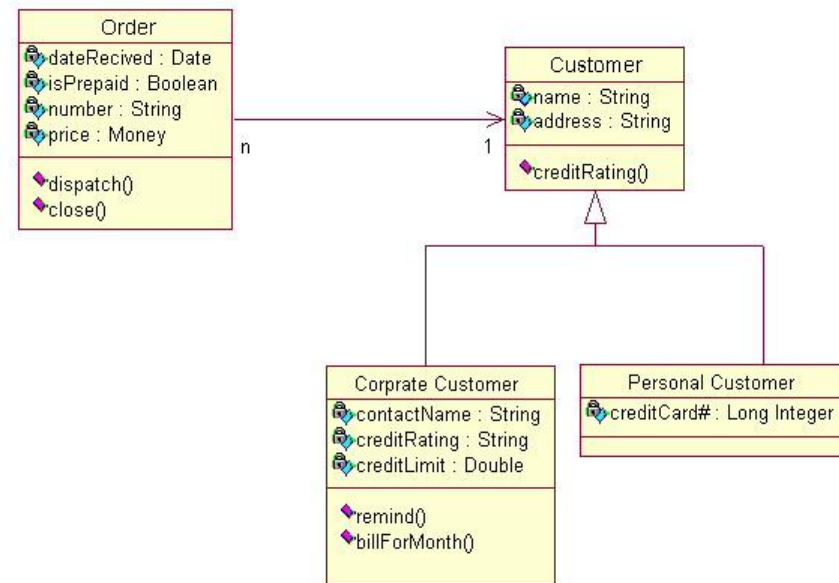


Figure 2 An Example of a UML Diagram of a system that records customer orders [10]

## Conclusion

The purpose of this paper was to research on several topics that are found in software engineering. It has elaborated the evolution of programming languages from machine language to high-level languages. This however, focusing mostly on the electronic computer. This paper also discussed the agile method, Extreme programming XP and outlined the main principle upon which it is based. It discussed Dynamic System Development Methodology and Lean Software Development, comparing and contrasting the principle upon which the two agile methods are based. Finally the paper discussed the Unified Modeling Language and elaborated and illustrated its functions.

## References

1. Rojas, Raúl, et al. (2000). "Plankalkül: The First High-Level Programming Language and its Implementation". Institut für Informatik, Freie Universität Berlin, Technical Report B-3/2000. (full text)
2. Knuth, Donald E.; Pardo, Luis Trabb. "Early development of programming languages". *Encyclopedia of Computer Science and Technology* (Marcel Dekker) 7: 419–493.
3. "Extreme Programming", *Computerworld* (online), December 2001, webpage: Computerworld-appdev-92
4. "Extreme Programming" USFCA-edu-601-lecture
5. The DSDM Agile Project Framework manual, 2014 pages 4, 16
6. Ward, Allen. *Lean Product and Process Development*
7. Sato, Masaaki (December 16, 2008). *The Toyota Leaders:*
8. OMG Unified Modeling Language (OMG UML), Superstructure. Version 2.4.1
9. Unified Modeling Language User Guide, The (2 Ed.). Addison-Wesley. 2005. p. 496.
10. <http://www.cs.pomona.edu/classes/cs121/supp/UML%20tutorial/class.htm>