Use Cases in Software Development: An Investigation in its Roles and Values

Grace L. Samson
Department of Computer Science
Faculty of Science, University of Abuja,
Nigeria
Email: gracedyk [AT] yahoo.com

Aminat A. Showole
Department of Computer Science
Faculty of Science, University of Abuja,
Nigeria

Abstract— **This research work identifies the roles and values of USE CASES in software development, and analysed the concept of USE CASES critically in other to ascertain their usefulness in achieving a user's requirement when applied to a system development methodology. We examined the major considerations for deploying advanced UML modelling by considering the best approach for a programmer to get to the source code by putting the right aspect of the UML to work at the right stage of a system life cycle in an object (OO) oriented analysis and design practice. research has found out that to achieve a successful object oriented programming design and implementation, an analyst should strive to drive an OO software design from USE CASES**

Keywords--- Object Oriented Design; Use –Cases; Software Development; UML Modelling; Software methodology; Classes and Objects

I. INTRODUCTION

A. System Development Stages

This research has helped us to appreciate what [15] meant in the statement below.

"In theory everything in the UML is useful, but in practice a whole lot of people and projects need to know how to drive an OO software design from USE CASES. And they also need to know which diagram from the UML directly helps to accomplish this" ([5], pp. xxvii).

Some researchers feel that "the roles and values of use cases are unclear and debatable" and argue that it is more useful to develop a class model before a use case model, as such in this research work. A system can only be successful if it meets the user's needs. Every object oriented (OO) software development requires a specialized technique that can be used for making good interpretations of real world phenomena as such; designing an effective system is always the main aim of software developers. However the framework for structuring the process of this development must go through the stages known as software development life cycle (SDLC). According to [9], the software development life cycle specifies the stages through which a software project must go through in its life time before it finally gets to the user. These stages may typically include: (1) System Analysis (2) System Design (3) System Implementation (4) System Testing (5) Documentation and (6) Maintenance.

Regardless of the general view of system development life cycle, several life-cycle models also exist and all these models have their stipulations and preferences as we shall examine below. Generally, a software development life cycle model is a framework that describes the activities performed at each stage of a software development project and there are various models that exist. Some of these models (methods) include: the waterfall model, prototyping, v-model, incremental model, spiral model and the general agile methods (which includes, FDD – Feature driven development, crystal clear, DSDM – Dynamic Software Development, RAD – Rapid Application Development, SACRUM, XP – Extreme Programming, RUP – Rational Unified Process and USDP)

In all the different models, the basic step/processes involved in every software development life cycle include:

- The existing system is evaluated/assessed
- The new system requirements are defined/analysed
- The proposed system is designed
- The proposed system is developed
- The system is put into use/Implemented
- The new system is tested
- The new system is maintained

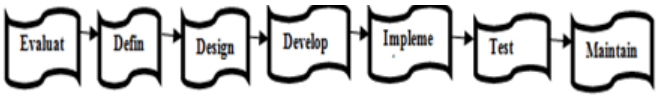

Figure 1: stages in a system development life cycle

In other to develop good computer systems and projects, these stages mentioned above are essential for good success, although smaller systems may afford to ignore some of these stages, large complex systems depends on them to completely analyse their requirements, design and their implementation using both an appropriate method (- an appropriate tool-) and an appropriate notations (- an appropriate modelling language-) that are suitable for modelling these large, complex systems. A modelling language can be used to express information, knowledge or a system in a procedure that is defined by consistent set of rules that are used for understanding of the meaning of the components of a given system. Some typical examples of a modelling language include: (a) SDL – Specification and description language, (b) ADL – Architectural Description language, (d) SOM – Service Oriented Modelling, (e) UML – Universal Modelling Language etc.

In general, all modelling languages can be used to specify; system requirements, behaviours and structures, but we shall be looking at

the UML because it is most often referred to as the de-facto standard formalism for software design and analysis [4].

The second stage of the development cycle (as we have seen in the review above) has to do with defining the specific problems to be solved which of course is the essence of the development process hence, in other to understand this specific problem system analyst gather information from the intended user of the system so as to determine the appropriate requirement specification. One useful tool for capturing what the proposed system should do is the UML modeling tools.

II. UML MODELLING – REVIEW

A. *What is Unified Modeling Language (UML) Model?*

The UML is the most widely used graphical representation scheme for modeling object oriented systems. It allows people who design software systems to use an industry standard notation to represent their systems [9]. The UML is a very useful tool in the analysis of user requirement and in the design of the system; it uses notations to produce models of computer systems. [17] described the UML as a graphical language designed to capture the artefacts of an OOAD (Object Oriented Analysis and Design – which is a programming technique that models the real world as a group of related objects -) process, this is useful because these programmatic objects maps naturally to real-world objects [17].

B. *UML Notations*

[12] described UML notations as graphical symbols used to represent the elements in a UML diagrams, these are also referred to as model elements. According to them, an element can exist in several different types of diagrams following a given set of rules. Some examples of these model elements are described below:
Class, Object, State, Node, Package, Component, Comment, Initial state
Others could include: Activity, Use-case, Interface, Actors, Final state Etc. (figure 2)

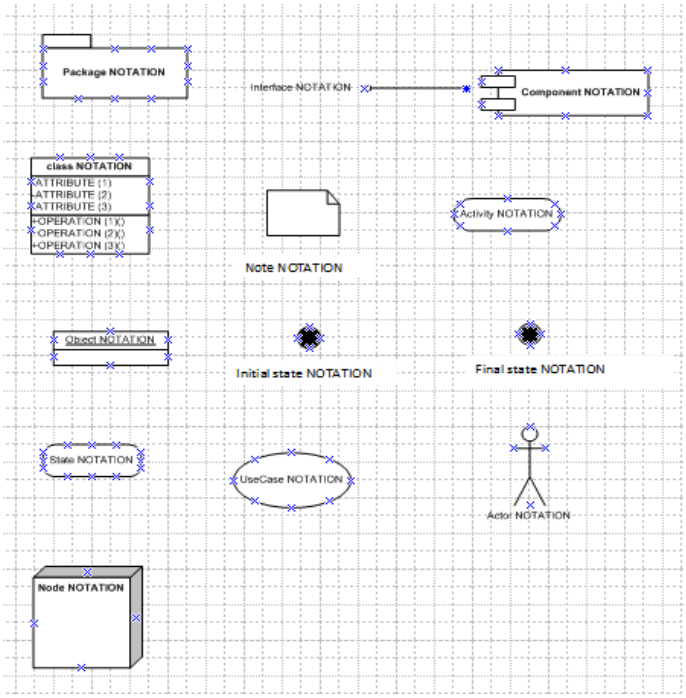

Figure 2: showing UML notations

However, it is important to note that these notations relate to each other with yet other forms of notations knows as the relationship components which are the elements that connects one notation or model element to the other , some of these relationship notations include: association, generalization e.t.c. (see figure 3)

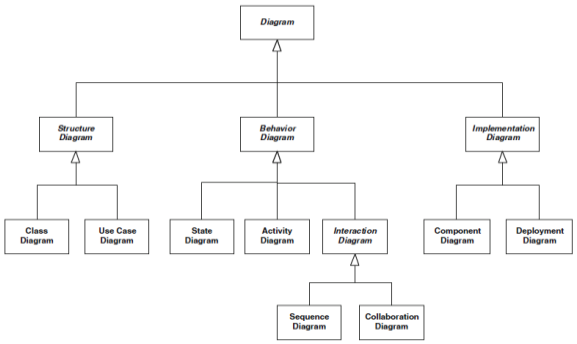

Figure 3: UML relationship components [7]

III. USE CASE MODEL AND THE CLASS DIAGRAMS – REVIEW

A. *UML Diagram*

A UML diagram is a combination of UML notations to produce a meaningful interpretation of an intended model of a software system. In [14], UML diagrams are divided into 4 major categories: USE-CASE diagram (UCD), Class Diagram (CD), Behavior Diagram (BD – Activity, Collaboration, Sequence, and State) and Implementation Diagram (ID - Component Diagram

and deployment). USE-CASE diagrams according to them are used to identify actors, USE-CASEs, and their relationship while CLASS diagrams identify the classes and their relationship. In [11], modeling a complex system, requires describing the system in a number of views, where each view represents a projection of the complete system that shows a particular aspect, hence in modeling a system with the UML, four basic views has been adopted namely: USE-CASE, Logical, Implementation, Process and Deployment views respectively, where USE CASE view shows the functionality of the system as perceived by external actors. Similarly, from the four system analysis phases of object oriented (OO) system developments as described, in [8], the USE-CASE model was specifically adopted for modeling the functional requirements of the system while the CLASS diagram is for structural modeling of an intended system[13] adopted the facts that modeling with UML can simply be described as a USE-CASE based approach.

A study carried out by [10], revealed from a detailed survey that in terms of usage, the frequency of use of UML components varies considerably but while the CLASS Diagrams were the most frequently used component, the USE-CASE Diagram benefit from the advantage of being very useful when it comes to identifying the key purpose of system elements as we can see in table 1a & b

Table1 a: showing rate of use of UML Components

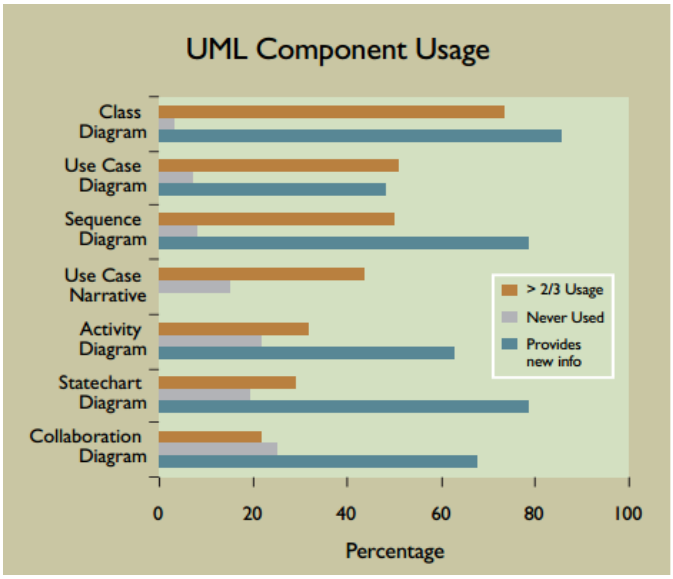

Table 1b: Percentage of clients using UML component for major purposes [10]

UML Component	Client Verification	Clarifying Tech Understanding	Programmer Specifications	Maintenance Documentation
Use Case Narrative	87	74	79	68
Activity Diagram	77	80	81	73
Use Case Diagram	74	66	62	61
Class Diagram	57	93	89	92
Sequence Diagram	62	91	84	85
Statechart Diagram	50	82	79	71
Collaboration Diagram	50	74	70	62

The results of their analysis show that clients are most likely to be involved in developing, reviewing, and approving the USE CASE Narratives and associated USE CASE Diagrams and the study also admitted that USE CASE diagrams are very essential when working with clients.

A more detailed description of the UML diagram was seen in [[9], here the USE-CASE diagram was seen as a model of the interactions (different from the sequence diagram which only models interactions but does not highlight when the interaction will occur) between a system and its external entities (known as the "actors"). The Class diagram on the other hand models the entities (which forms the building block used in a system), representing each entity as an object to be analyzed or present in the system.

[14], explained further by adding that a USE-CASE diagram models the activities of a system from the actors perspective and that USE-CASEs are always initiated by the actors, while CLASS diagrams defines the attributes and operations of both the internal and the external entities. In [16], the USE-CASE diagram is seen as the primary means by which you can use the UML to capture functional requirements expressing these requirements in terms of the specific actions that external entities and the system perform in executing required and optional behavior. This is explained in figure 4

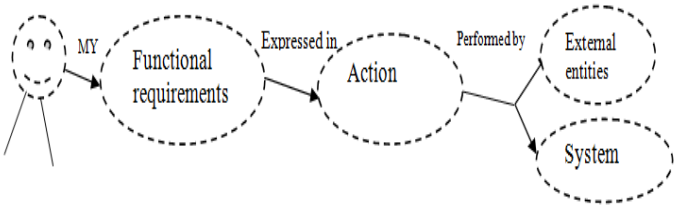

Figure 4: USE-CASE structure

In addition, from the description in [7], UML diagram is sub divided into three major categories namely: structural, behavioural and implementation diagrams. This classification puts the USE-CASE diagram and the CLASS diagram in the same level irrespective of extent. By structural, we are looking at the components of the UML diagram that must be present in the system being modelled as such, it would be right to say

that structural diagrams determine the architecture of the system being defined.

B. *Identifying the place of USE CASE in UML classification*

From the work of [2] on "developing agile methods with UML", we have used the diagram below to structure the thirteen main diagrams of UML according to their prospective values; this will give us a better clarification of the Characteristics, category and values of each diagram as we shall apply in our analysis in section 4.

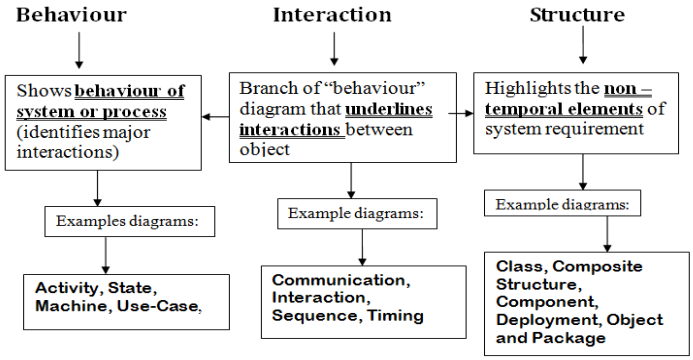

Figure 5: classifying UML diagram according to characteristics

C. *Components Description – USE-CASE modelling*

[15] acknowledged that "USE-CASEs are used to describe the way the user will interact with the system and how the system will respond". In their own view [5] recognized that USE-CASEs helps a system analyst or modeller (in organizing thoughts about an intended systems) thus making it easier for him to understand the purpose of a system or its components. Likewise, for every system development process, the first step of the proposed technique consists in building a USE-CASE diagram, in which actors and USE-CASEs are identified [1].

In analysing system requirements, according to [15], USE-CASEs are used to;

(i) Identify end users and stake holders' interest
(ii) Design the context of the user interface
(iii) Organise objects into packages
(iv) Identify the concept of the object model
(v) Describe courses of action in (both basic and alternative)

IV. EVALUATING USE-CASES AGAINST CLASS DIAGRAMS

A. *Comparism*

We have used the table below to critically evaluate the usefulness of the USE CASE diagram in contrast to the CLASS diagram.

Table 2: been used to identify the major values, characteristics and roles of the USE-CASE diagram

USE-CASE	**Class diagram**
USE-CASEs are used to identify the Interactions and responses between users and systems [15]	Class diagrams according are used to identify the objects in a system [5]
USE-CASE diagrams are used to identify actors, USE-CASEs, and their relationship [14]	Class diagrams identify the classes and their relationship. [14]),
Use-case view shows the functionality of the system as perceived by external actors. [11]	No specific view identified [11]
Basic framework for modeling with UML as described in [13]	Not applicable
Explains behaviour (Ambler, 2004)	Explains structure [2]
With a medium …. Priority, USE-CASE diagram shows actors, USE-CASEs, and their interrelationships. [2]	The class diagram enjoys high …. Priority, in terms of usage…, it also highlights collections of the static model elements such as classes and types, their contents, and their relationships. [2]
USE-CASEs helps a system analyst or modeller in organizing thoughts about an intended systems - Value [5])	Class diagrams also helps a system analyst or modeller in organizing thoughts about an intended systems: but in its case it considers majorly elements which do not depend on time [2]

The table above has been used to identify the major values, characteristics and roles of the USE-CASE diagram and we have used this as a means of comparism between the USE CASE and the CLASS diagram

B. *Reflection and Analysis*

A simple case study that we can use at this point is purpose of systems an online transaction system. Every system is designed to meet a user' requirement as such there must be a way of graphical representing how this system will behave in terms of meeting the users requirements and how the user will interact with the system; this is the gap that USE-CASEs are bridging and this is what defines its value. There are basically no competition between the USE-CASE diagram and the class diagram or any other diagram in the UML family as the case may be (as we have seen from this study), the basic issue of contention is which is better to come first; again the answer to this question is quite simple, if what determines a system structure (which the CLASS diagram is an element of) is the requirement of that system and the behaviour of its elements as we have seen, then we would be right to say the USE-CASE diagram must be developed first before the Class diagram. However, looking at [7] description of in section 2.2, then there would not be any gainsaying that any of these diagrams is more important than the other and as such will not require any measurement to determine significance of each.

- **Our case study: an online transaction system**

The case study stated above of an online transaction system, will depict the relationship that may exist between a buyer (customer) and a seller (management). We would like to see how these objects interact with each other and with the "system" to be able to meet the goal of an online business. Since the business is majorly to meet customers need as well as increase organizational profit, we would like to bring out the steps below as a way of analysing the system:

- ❖ Actors and USE-CASEs are identified (where each actors has a set of USE-CASEs that describe the task due to the actor)

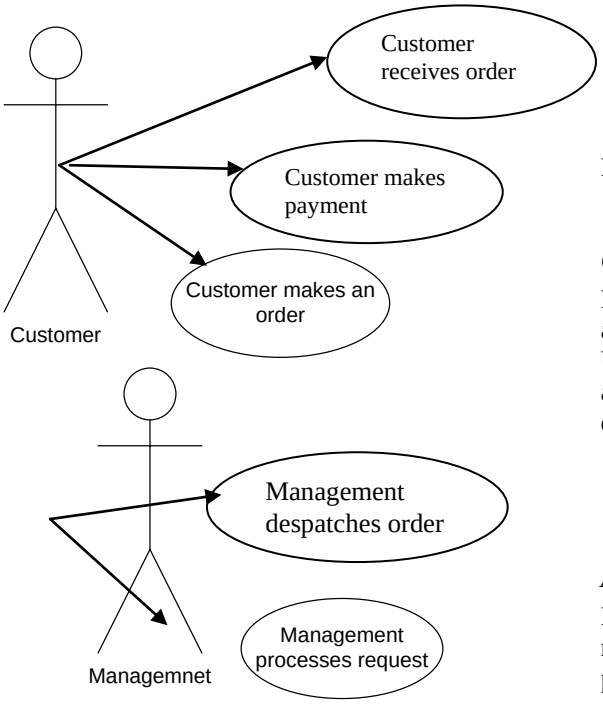

Figure 6: Actors and USE-CASEs (i.e. tasks) are connected as shown in this diagram

- ❖ Task due the actors are identified as we can see above

Actor 1 → Customer
Tasks:
Make order request
Make payment
Receive order

Actor 2 → Management
Task:
Process order
Dispatch order

With the identification of the use cases, the analyst would easily develop the class diagram based on those cases where the actors are the entities (object) and the use cases are the operations as we can see below:

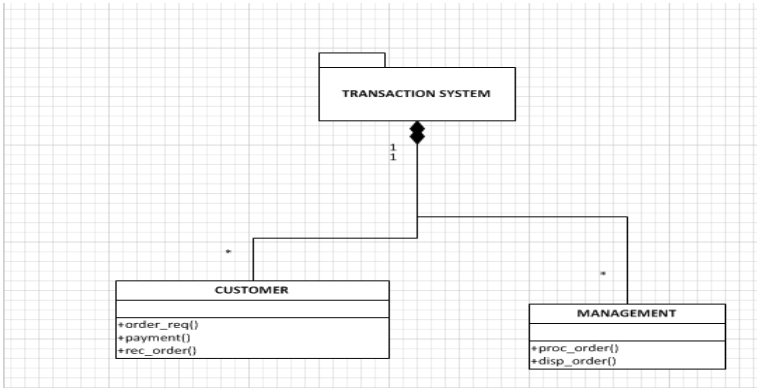

Figure 7: Class Diagram for the transaction system

C. UML Ranking

From the case study above and from all the review and analyses, we have seen that it always very useful to design the USE CASE diagram before the CLASS diagram, so as to be able to gather all the necessary user requirements before designing the system using the CLASS model.

V. RESULT AND DISCUSSION

A. USE-CASEs Roles and values

From the above analyses, and from a thorough and reflective research on USE-CASEs; their roles characteristics and values, we present in acknowledgement to the propositions of [3] the major findings we have made on the significance of USE-CASEs and when to use them in a system design process, in essence we have listed below some of these findings about USE CASES.

USE CASE MODELS

- ❖ Are means of capturing the requirement (functional and non-functional) of a system
- ❖ Help to envision application structure
- ❖ Are used to express communication link between organisation and end user
- ❖ Act as a base for deriving objects (objects in programming are noted as CLASSES) – this can also be seen from our case study in section 4.3 and likewise [3] specifically stated that "objects naturally fall out of USE-CASEs".
- ❖ Are the framework for the design of user interfaces
- ❖ Help the analyst assign functionalities to objects (CLASSES)
- ❖ Incremental system methodology has its background on USE-CASES

- ❖ Most importantly, USE-CASES helps to design test cases

The greatest value of USE-CASES is based on the fact that they have become the standard for representing business processes [6]; [3].
However, it would be very expedient at this point, to state clearly some of the limitations of USE-CASEs in other to avoid doubt and clarify the basic concept of USE-CASES, in other words we have stated some important points to note below according to [6];

USE-CASEs are basically interested in highlighting system processes and not the design

VI. CONCLUSION

We have had a critical evaluation of the UML diagrams, notations, and classification – type and we have also seen the different methods of developing a system using the SDLC and the appropriate UML modeling tool for a specific development methodologies. We have explored the existing literature to see the place of USE-CASES in the development of a system and we have come to the conclusion that there is basically no conflict between USE-CASES and CLASSES (OBJECT) or even any other UML modeling tool as they all have their distinct place in the development of a system. We have also seen that it is indeed more useful to design the USE CASE diagram before the CLASS diagram. At the end of this research project, we have arrived at the fact that Use cases are very helpful In terms of

- Determining features or requirements
- Communicating with clients and
- Generating test cases

REFERENCES

[1]. Almendros-Jime´ Nez, J. M. and Iribarne, L. (2009) "UML Modeling of User and Database Interaction" THE COMPUTER JOURNAL. 52 (3) pp. 348 - 367

[2]. Ambler, S. W. (2004) The Object Primer: Agile Model-Driven Development with UML 2.0. Cambridge University Press.

[3]. Amour, A. and Miller, G. (2001) Advanced USE-CASE modeling: software systems. Boston (Mass): Addison-Wesley,

[4]. Berardi, D., Calvanese, D. and De Giacomo, G. (2005), "Reasoning on UML class diagrams", Artificial Intelligence, 168 (1), pp. 70-118.

[5]. Chonoles, J. M. and James, A.S. (2003) UML 2 for dummies. New York: Wiley

[6]. Cockburn, A. (2001) Writing effective USE-CASEs. Harlow: Addison-Wesley

[7]. Cook, S. (2012), "Looking back at UML", Software and Systems Modelling. 11, (4) pp. 471

[8]. Dennis, A., Wixom, B. and Tegarden, D. (2005) Systems Analysis and Design with UML Version 2.0 - An Object Oriented Approach. 2nd Edition. Hoboken, N.J: John Wiley & Sons. [Online] Available at:http://www.knovel.com/web/portal/browse/display?_EXT_KNOVEL_DISPLAY_bookid=1420&VerticalID=0 [Accessed 9th December 2012]

[9]. Dietel, P.J. and Dietel H. M. (2010) Java: how to program. Upper Saddle River, N.J: Pearson Prentice Hall

[10]. Dobing, B. and Parsons, J. (2006), "How UML is used", Communications of the ACM, 49 (5), pp. 109-113

[11]. Eriksson, H. (2004), UML 2 toolkit. Indianapolis: John Wiley & Sons (US),

[12]. Eriksson, H., Penker, M., Lyons, B.and Fado, D. (2011), UML 2 Toolkit: CafeScribe. Hoboke: John Wiley & Sons, Inc,

[13]. Jacobson, I., Booch, G. and Rumbaugh, J. (1999) "The unified process", IEEE Software. 16 (3), pp. 96.

[14]. Oestereich, B. (1999) Developing software with UML: object-oriented analysis and design in practice. Harlow, England: Addison-Wesley.

[15]. Rosenberg, D. and Stephens, M. (2007) USE-CASE Driven Object Modelling with UML: Theory and Practice. NY: Apress

[16]. Scott, K. (2004) Fast track UML 2.0. Berkeley, Calif:

[17]. Shoemaker, M. L. (2004), UML Applied: A .NET Perspective. Apress.