Journal of Theoretical and Applied Information Technology
31st August 2011. Vol. 30 No.2

ISSN: **1992-8645** www.jatit.org E-ISSN: **1817-3195**

SOFTWARE ENGINEERING PROCESS: YAAM DEPLOYMENT IN E-BOOKSHOP USE CASE SCENARIO

[1] Eke B. O., MACM, MCPN* and [2]Nwachukwu E. O., Ph.D. FCPN, FNCS

[1] Department of Computer Science, University of Port Harcourt, Nigeria

[2]Prof., Department of Computer Science, University of Port Harcourt, Nigeria

*E-mail: bathoyol@gmail.com

ABSTRACT

The development of Yet Another Agile Methodology (YAAM) has added to the list of Agile software processes which is a novel software development process introduced for use in modern software development. In this paper, YAAM is deployed in E-Bookshop use case scenario to substantiate the development benefit claims made during the introduction of the software process. E-Bookshop Scenario was selected due to its varying complexity concerns which are suitable in testing various areas that were perceived to be difficult to handle using other software processes. The deployment is implemented using open source tools and languages.

Keywords: *YAAM, Agile, E-Bookshop, Use Case, SE Process*

1. INTRODUCTION

Software process is a framework for the tasks that are required to build high-quality software [3]. A software process defines the approach that is taken as software is engineered. These also encompass technical methods and automated tools. Software engineering methods provide the technical "how to" for building software. Methods encompass a broad array of tasks that include communication, requirements analysis, design, modeling, program construction, testing, and support. It relies on a set of basic principles that govern each area of the technology and include modeling activities and other descriptive techniques.

In this paper, a Yet Another Agile Methodology (YAAM) software engineering method [1] is deployed in E-Bookshop use case scenario to substantiate the development benefit claims made by the new software process model. E-Bookshop Scenario was selected due to its varying complexity concerns which are suitable in testing various software engineering concerns such as development time, working software, efficiency, reuse of process components, process consistency and applicability of process to modern development paradigms.

These concerns have been perceived to have some short-comings when handled using other software processes [2]. The deployment is implemented using Linux Server (Apache), PHP and MySQL which are open source languages and database management system.

2. YAAM METHODS AND DEPLOYMENT

YAAM Process Theory

YAAM as an agile Software process is characterized in a manner that addresses three key assumptions about the majority of software projects [9]:

(i) It is difficult to predict in advance which software requirements will persist and which will change. It is equally difficult to predict how customer priorities will change as a project proceeds.

(ii) For many type of software, design and construction are interleaved. That is, both activities should be performed in tandem so that design models are proven as they are created. It is difficult to predict how much design is necessary before construction is used to prove the design.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

(iii) Analysis, design, construction, and testing are not as predictable (from a planning point of view) as we might like.

With the three assumptions in mind, an important question arises, how do YAAM process manage unpredictability? YAAM uses process adaptability (to rapidly changing project and technical conditions). YAAM process is therefore, highly adaptable and forward progressive in use case implementation.

Continual adaptation without forward progress accomplishes little. There must be the ability to provide upfront reuse capability to make a software process forward progressive [4]. This reuse is not only provided by YAAM at implementation and refactoring of code, but also analysis and design reuse is equally provided in the YAAM software engineering process. Therefore, YAAM software process seem to adapt incrementally. To accomplish incremental adaptation, YAAM process model provides for an agile team customer feedback (so that the appropriate adaptations can be made). An effective catalyst for customer feedback provided is an operational prototype or a portion of an operational system. Hence, an incremental development strategy should be instituted. Software increments (operational system) must be delivered in short time period so that adaptation keeps pace with change (unpredictability). This operative approach enables the customer to evaluate the software increment regularly, provide necessary feedback to the software team, and influence the process adaptations that are made to accommodate the feedback.

The history of software engineering is littered with dozens of obsolete processes, descriptions and methodologies, modeling methods and notations, tools, and technology. The methods started well and was then over shadowed by something new and (purportedly) better. With the introduction of a wide array of agile process models –each contending for acceptance within the software development community –the agile movement is following the same historical path. There are number of agile process models used in different projects. There are many similarities (in philosophy and practice) among these approaches. However there are some characteristics of each method that makes it unique. It is important to note that all agile models conform (to a greater or lesser degree) to the manifesto for Agile software Development and their principle [9]. YAAM as an agile process model does conform to the agile manifesto, however the model offers room for upfront planning where systems that require high level accuracy in model development is to be developed. It also offers a unique process model that is agile in nature once the first release have been developed. This paper is a systemic deployment of the YAAM process.

YAAM Scenario Description

Software engineering from YAAM perspective is more than concepts, tools, techniques, methods and processes. It is about people working and interacting well with people. In our scenario, Ogboke Ekebong, who is the Chief Architect software engineering team and director Osoft Nigeria Limited (Osoft) receives a letter from Tina Jonah the Content developer and project secretary Osoft. Chukwudi Ordu the System analyst/programmer at Osoft is seated. Musa Abdullahi Programmer I at Osoft is busy on the keyboard, while Gbenga Onoyiga the Quality assurance manager at Osoft just works into his desk all in the open office. Other members of staff in the adjacent room are busy with their work.

In the YAAM deployment, scenario similar to the one described above is used to describe how interaction and communication ensure in the process. Where necessary use case diagrams using Universal Modeling Language (UML) were deployed to represent some concrete issues. Flow directions were used to indicate the communication flow and the activity process and paths in accomplishing any given task within the project implementation lifecycle [6].

Project Initiation Use Case

Scenario: Worlu worked into the open office and announced that Ubookshop has indicated interest in building an online store to boost its sales to a wider market after a proposal is submitted to them. A project team is immediately selected to meet Ubookshop for more clarification. A meeting holds between Osoft and Ubookshop and an agreement is reached to build an online store for Ubookshop. Figure 1 illustrates the UML use case of the project initiation state.

Journal of Theoretical and Applied Information Technology
31st August 2011. Vol. 30 No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

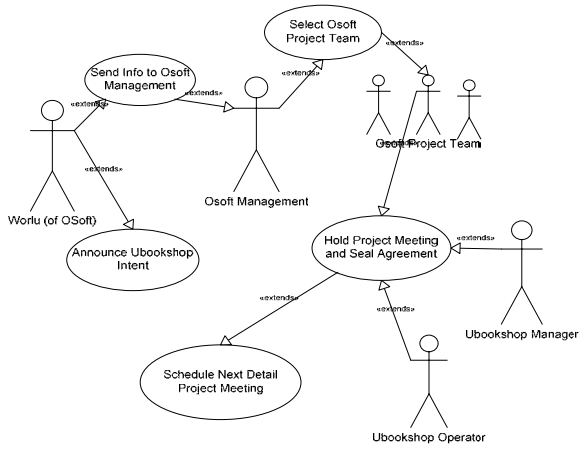

Figure 1: UML Use Case of Project Initiation.

Project Planning Use Case Scenario

The chief architect and director Osoft Ogboke Ekebong calls the project team so that a YAAM plan for the online bookshop project will be developed. The team agreed that the YAAM methodology is to be used in the development and four days mile stone is given to the planning pair to submit its work as shown in UML use case of figure 2.

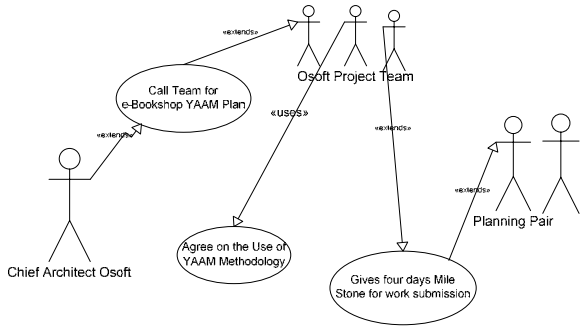

Figure 2: UML Use Case of Project PIanning.

YAAM Plan

The team meets and outlines the following functionality for the application:

- A database of the books Ubookshop wants to sell online
- Online catalog of all the books by category
- Shopping cart to track the books a user wants to buy
- Checkout section that processes payment and delivery details
- An Administrative interface used for adding books, removing books, fixing and updating prices and other general administrative activity.

From the requirement gotten from the client, users are supposed to:

- browse books based on category for instance by field of study, publisher or author.
- Users should also be able to select item from the catalog and the system should be able to track which items are selected.
- The system should be able to total up users order, take their delivery details, and process their payment when they finish shopping.
- An administrator interface to Ubookshop site should also be built so that the administrator can add and edit books and categories on the site.

In figure 3 the UML Use case indicates two main functionality, admin and general function performed by admin and users (Ubookshop) respectively.

From the users perspective these requirements are just simple requirement but we know that from software engineering perspective they are up-hill tasks that require serious implementation efforts. For instance it is much more easier to say prepare book catalog or check out from the shopping cart as a user functionality or requirement. However the process of developing the book catalog or developing the check-out of any given shopping cart is much more tasking.

Irrespective of the complexity involved, the first step remains the identification and outlining of the requirement and then identifying how the entire system is supposed to fit-in and interact with the actors and the entire system. This interaction is clear from figure 3. where each of the actors are clearly mapped out and the functionality boxed using a rectangle.

Journal of Theoretical and Applied Information Technology
31st August 2011. Vol. 30 No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

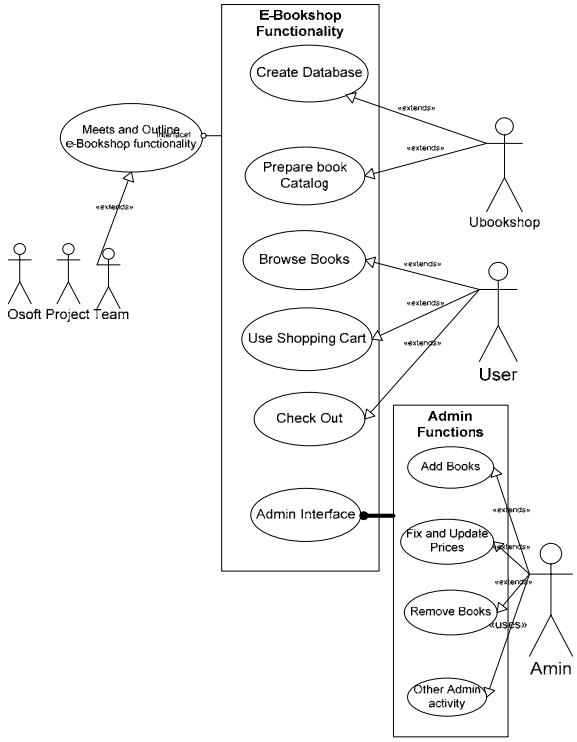

Figure 3: UML Use Case of Project Functionality

Once the plan has been made and the use case clearly defined the system can then move to analysis stage as specified by the YAAM model.

3. YAAM ANALYSIS OF THE E-BOOKSHOP USE CASE

The YAAM analysis requires the use of the agile list and the YAAM Card for drafting the major items that will be carried out. Some of this items or activities include**:**

-Building the Database backend
The actors the online catalog hence there is a need to develop a database using MySQL that will handle all the data and the categories of books, shipping addresses, payment details and so on.

-Building the tracking of user purchases
The tracking section need to be built, which tracks a user's purchase when the shopping is going on, either by putting book selections into the database or by using a session variable. In YAAM a balance of usability and efficiency is considered so session control need to be selected. The system therefore will need session variables to store a user's selection to save the overhead of constant querying of the database for the information. When the user finishes shopping and pays for the items the information can then be put in the database as a record of the transaction. The stored information can also be used to give sales report.

-Build payment processing
Payment involves lots of other issues which in an open source system call for outright reuse of payment systems of a reputable online merchant. In the system there is need only for the building of an interface that will connect, instantiate and link to the payment gateway. In the payment system one needs to organize a merchant account with a bank for the cards to be accepted. When done the session variables can be destroyed.

-Building administrative interface
This interface will allow Ubookshop to add, delete, and edit books and categories from the database. This will allow Ubookshop staff to change book prices as the publishing cost changes or to remove a book that has been out of stock for a very long time.

EE-Path Analysis State Use Case

At this stage YAAM performs lots of outside requirement abstractions to unearth requirements from a virtual dimension. This abstraction represents unbounded group of possible requirements and behaviors. For instance Ubookshop staff may not know that online robot may come to buy products. Such possibilities need to be considered so that such robots may not come in and place some wonderful order that will distract management attention. Security of payment processing may not be fully clearified by Ubookshop actors but it is a very serious requirement for the success of the system. There is a need to make provision for security in all information on transit across the client and server of the E-Bookshop system. In other to achieve this the system may have to make member variables to be private and avoid global variables in client server-communication. The EE-path in addition to considering the ***security unknown*** also takes into account the fact that the system may change during its life cycle [7]. The stability requirement is also unknown if the online bookshop will last above the first version. The database -MySQL version used in developing the system may change, the programming language-PHP version also used in the development of the system may also change. Similarly the server (Apache) version

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

where the system runs may also change hence we must also have development ***tools version stability unknown*** as built-in component of the system.

4. YAAM DESIGN USE CASE

Before we look into the detail design YAAM encourages minimal documentation or reuse of documentation done in previous project. We shall reuse theories of Bertrand Meyer [11] to try to compare the EE-Path stability unknowns [7]. According to Meyer, Software entities (classes, modules, functions, etc) should be open for extension, but closed for modification. This means that it is undesirable for a single change to a program to result in a cascade of changes to dependent modules. The open-closed principle resolves this by recommending the design of modules that never change. If requirement changes, the behavior of such modules should be extended by adding new codes, not by changing old codes that already work as recommended by open-closed principle. But if there is tools version change refactoring can set in as part of redesign of the system after the first deployment.

YAAM believes that if requirement changes a module can be refactored to accommodate the change without affecting the external modules. Even if the design has virtual or abstract class implementation, conditions may warrant the addition of a virtual member to the class. We believe that extending the abstract class does not make a good design sense. We prefer to refactor the abstract class to reflect the new changes which will not affect other classes or their object implementations.

Preliminary YAAM Design

In the preliminary design, the system flow design is provided to guide the process. This will help the team on the components that are required. It will also guide the team in selecting components that can be reused and those that must be coded and the once that can be refactored and used in the project [8]. We can now look at the views of the system and from the requirement we have the user view at the front-end and the administrator view at the back-end. Figure 4 illustrates the user view system flow design of the Ubookshop system which allows the users browse books by category, view book details, add books to their cart, and purchase them.

The main links between the scripts in the user part of the online site is shown in figure 4. A customer (User)
gets to the main page, which lists all the categories of books in the site.
The user can proceed to a particular category of books, and then gets the individual book details. The user is also given a link to add a particular book to the cart from where he can check out of the online bookshop. The information collected from the user after payment is recorded in the database.

Journal of Theoretical and Applied Information Technology
31st August 2011. Vol. 30 No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

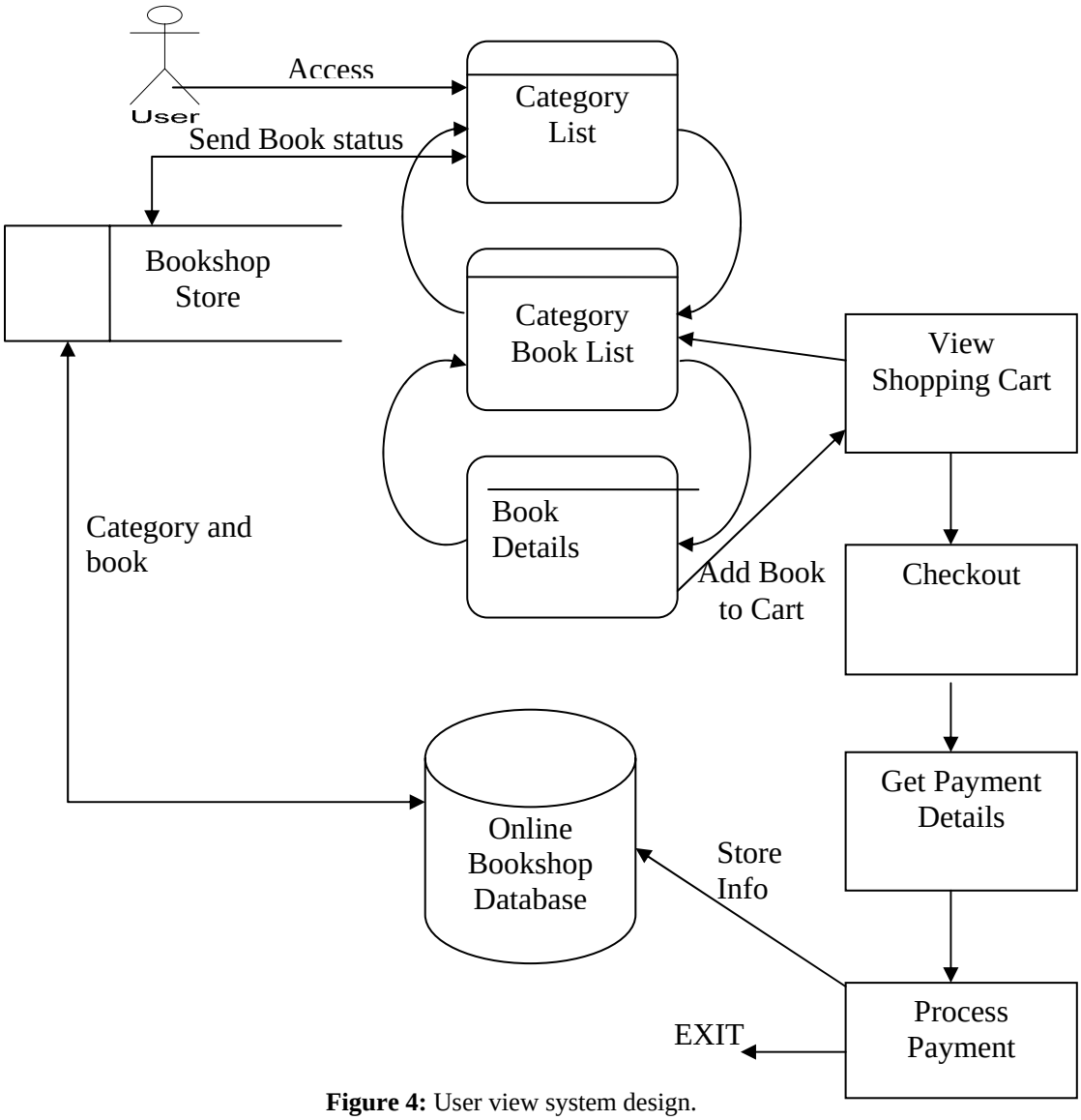

Figure 4: User view system design.

The second view in the online bookshop is the administrators' interface that will allow Ubookshop staff to add, delete and edit books and categories from the database. Figure 5. shows the administrative view of the system flow diagram which is a slightly different version of the user view. The administrator will still be able to browse categories and books, but instead of having access to shopping cart the admin will be able to go to a particular book or category and edit or delete that book or category. Hence the system design and possible code that is developed for users for similar areas can still be reused for the admin without much alteration. The three main modules for the admin will be the Catalog, shopping cart and order processing and administration.

Journal of Theoretical and Applied Information Technology
31st August 2011. Vol. 30 No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

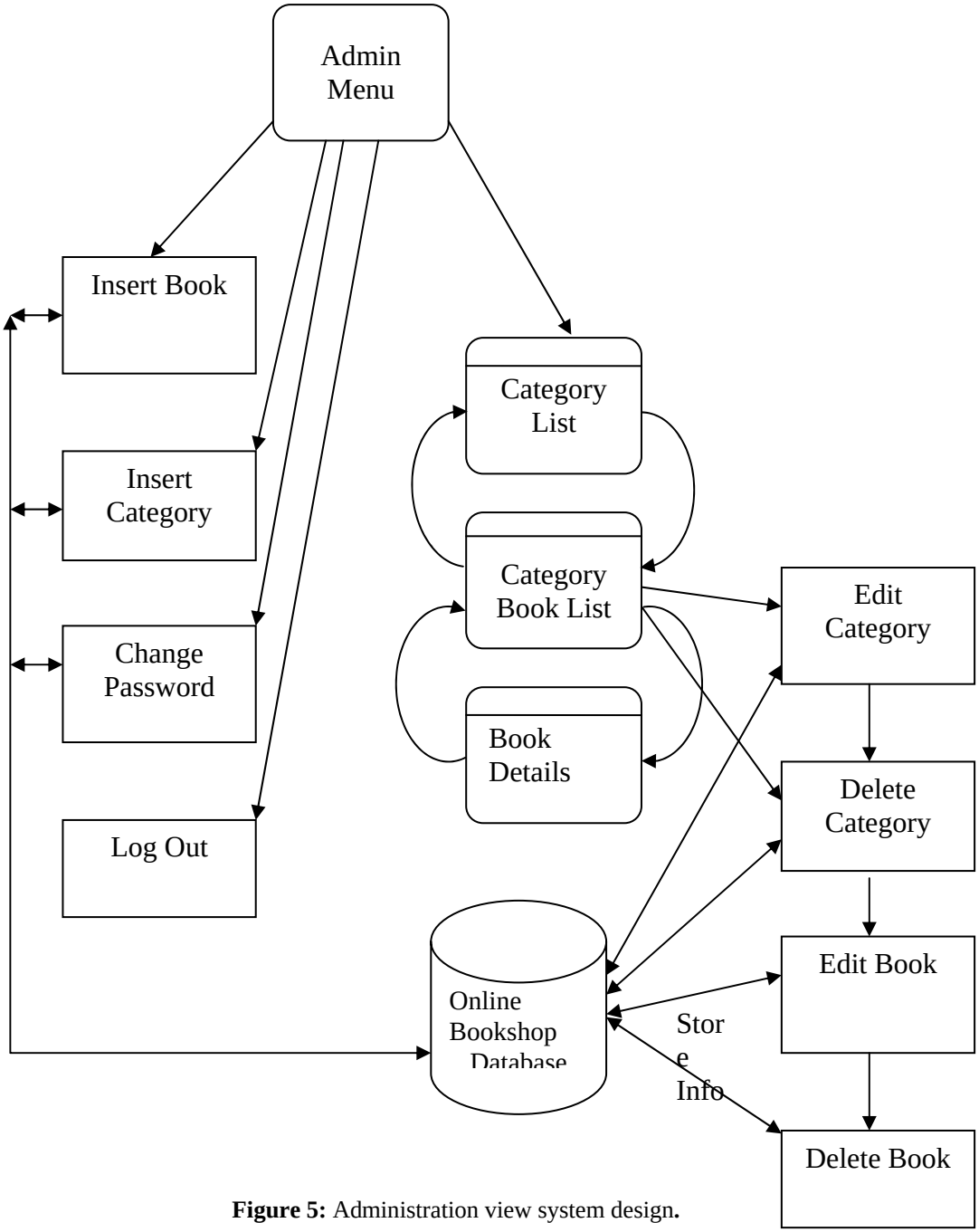

Figure 5: Administration view system design.

We can implement the system using the modules identified in the design diagrams of figure 4 and figure 5. Each of the modules will require a corresponding code for its implementation, and the availability will be searched to ascertain whether there is components that can be reused. Where there are reusable components the code will be refactored to accommodate the new design. Once the design is completed the process will proceed to the YAAM Implementation primitives.

5. YAAM IMPLEMENTATION USE CASE

In our implementation use case we will discuss details of how the YAAM team codes the Ubookshop online application using PHP and MySQL. In a complex or advanced project codes

Journal of Theoretical and Applied Information Technology
31st August 2011. Vol. 30 No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

are not just written, there must be coding theoretical or documentation primitives that illustrate how the coding is executed. This coding-documentation primitive is often referred to as YAAM documentation of the system. It is important to be reused when the code need to be refactored [10]. However the documentation referred to in YAAM is quit different from the over blotted documentation used in other classical methodologies. In YAAM the documentation is done in such a way that the core coding theory is preserved in language that is independent of any computer programming language. Since YAAM uses object-oriented paradigm in its core system development the Unified Modelling Language (UML) is preferred for use in its coding-documentation.

YAAM Documentation of E-Bookshop Installer

The E-Bookshop developed for the implementation of our system requires an installer that will make the users free from the difficult job of Server setup, Server Configuration, Database Creation in MySQL, SQL query implementation for table creation and table population. Our installer takes care of all this making the users to get started with a click of few buttons. In figure 6 the UML diagram developed using the visual paradigm tool [5] shows the installer documentation in a UML diagram.

The UML diagram show the installer code aggregation based on the PHP pages, the [[..]] symbol indicate that the rectangular boxes are not classes but code pages while the […] show embedded code aggregately identifiable such as a JavaScript code or PHP code.

In figure 6 it is clear that the [[install.php]] page requires the [[install.php_HTML]] for the display of the form needed in collecting information for configuration of the server and databases in the system and the execution of the Reset_connect_configuration_setting() operation specified in the [[install.php]] page. The [[install.php_HTML]] on the other hand need the CSS in [[Main.css]] page to format its presentation and the embedded JavaScript [Validate.js] to validate the form before it can be submitted. The two will aggregate with the HTML to produce the do_header() for the head of the page, do_body() for the body of the page and do_footer() for the page footer.

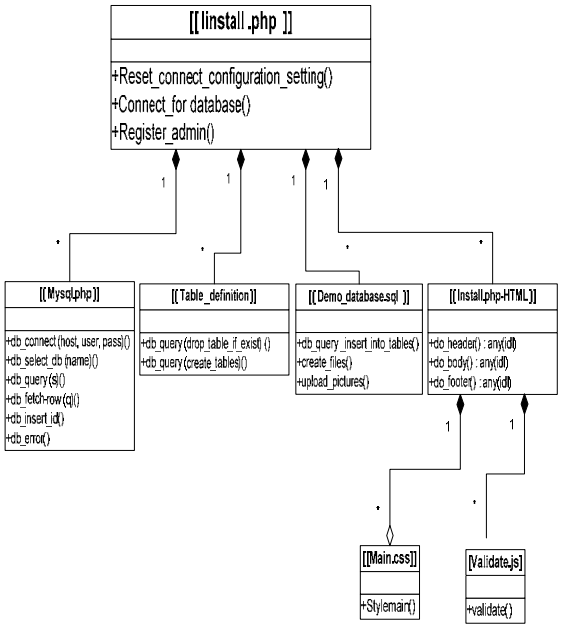

Figure 6: Installer PHP Code Aggregation UML diagram

The [[MySQL.php]] page equally requires the setting to connect, select and query the database which will be created by aggregating with [[Table_definition]]. The [[Table_definition]] checks if a similar table already exists in which case it will execute drop_table_if_exists() query which will remove the old tables from the database. If the table does not already exist it will then execute create table which will create new but empty tables in the database. The created but empty tables will need the [[Demo_database.sql]] to populate the databases using its insert_into_table() query making the installation of the program complete and ready for use. This process is clearly illustrated in figure 6 and the implementation logic can be easily derived from the UML documentation even without going through the code that will be developed to implement the diagram. However on seeing the code the process gets much clearer and easy to maintain. This also offers developers freedom to develop the system using any language without changing the original implementation documentation for the system. Migration is equally guaranteed, since a program developed in one language can be easily ported or translated without compromising the implementation documentation and without the need for new documentation.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Documentation of E-Bookshop Major Class Component-Smarty

In other to implement the other functionalities expected from the e-bookshop system we needed to bundle certain functions and activities that are common across different operational platforms within the system together in classes. In the full implementation major class like Smarty where reused. Smarty class operations needed to be compiled before they can be deployed in any other operation within the system so the Smarty class is extended in the newly created class to derive certain operations that were used in the code implementation of the operations.

6. CONCLUSION

While some software engineering group believes that a set of common purposes and principles will benefit the users of agile methodologies, we believe that variety and diversity of practices are necessary because each project is different and each project team is different—there's no one-size-fits-all solution.

YAAM has been developed as a unique methodology different in process model and conceptualization from other methodologies presently in existence. YAAM uses object abstraction and process flexibility to overcome these problems. YAAM has all the benefits of Object orientation due to its object abstraction property. In addition, it also has far more better process model that eliminates the rigidity and lack of quick response to changes using agility philosophy in its process strategy. The benefits of reuse which is a great benefit always cast shadow on the pit falls of the OO Development as a software engineering development methodology. YAAM builds-in reuse as not just a factor of coding but as part of all the process activities in its software development life cycle.

YAAM delivers all the agility benefit of XP without been extreme. YAAM refined analysis to what we call Agile List (AL) analysis, and made it to be concurrent with design so that the benefit of analysis-design-coding- refactoring will be retained giving chance for expert analyst to share their domain experience with coding expert during group programming. We believe that design is not dead but in slumber and has been reawakened by YAAM. This was done by making design to be consciously concurrent with analysis and coding so that wrong coding can be detected via the analysis and wrong design can be corrected by code refactoring; each process serving as an agile checker for the other. YAAM also has the benefits of adaptation to change, concurrency of development operations and steady improvement in the entire software over a relatively shorter period. This is particularly true for entirely non-existent systems since it has EE-Path inbuilt in all the activities in its process model.

YAAM takes care of personal relationships in development teams by emphasizing people instead of process and document. In this paper the Yet Another Agile Methodology (YAAM) that is new, novel and versatile have been deployed and verified in an open source platform. An e-bookshop use case has equally being used to show how YAAM process can be used in project development. The deployment is meant to verify the authenticity of YAAM use in real life projects.

REFERENCES:

[1] B. O. Eke . Agile Development In An Open Source World Of Software Engineering. PhD Thesis SGS, University of Port Harcourt 2011 Unpublished.

[2] E. O. Nwachukwu, E. O and B. O. Eke . *Critical Analysis of Software Development Strategies,* Proceeding of Second International Conference on Scientific and Industrial Studies, Vol. 2 No 3, pp.30-36.2008

[3] S. R. Pressman . Software Engineering A Practitioner's Approach. Sixth edition. Mc-GrawHill, USA, 2005.

[4] W. Tracz, Software Reuse: Emerging Technology, IEEE Computer Society Press, USA, 1988.

[5] VP Visual Paradigm, Computer Aided Software Engineering (CASE) Tool Developer Manual, Feb, 2008

[6] T. Roy Software Architecture in an Open Source World, 27th International Conference on Software Engineering, St Louis Missouri, USA Proceeding on Doctoral Symposium Vol ICSE-17
, 2005

[7] B.O Eke and Z. P. Piah , *Deploying EE-Path Software Strategy in Enterprise Application Development,* Proceedings of Fourth Annual

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Conference of IRDI, Science and Technology Forum Vol. 4, No. 2 PP10-15 , 2008.

[8] T. Elrad, R. Filman and A. Bader, Aspect-Oriented Programming, Comm. ACM, Vol. 44, no. 10, October 2001, special issue.

[9] M. Fowler, The New Methodology, Accessed Sep, 2010, http://www.martinfowler.com/articles/new Methodology.html N8B 2002

[10] M. Fowler , Refactoring: Improving the Design of Existing Code, Addison–Wesley, 2000.

[11] B. Meyer, Reusable Software: The Base Object-Oriented Component Libraries, prentice-Hall, USA, 1995.

AUTHOR PROFILES:

Eke Bartholomew Ogboke. is a Lecturer at the Department of Computer Science, University of Port Harcourt. He is a Practicing Software Engineer and a member of the Nigerian Computer Professional (MCPN) as well as Association of Computing Mercenary (ACM). He holds a Master of Science (M.Sc.) in Computer Science from the University of Port Harcourt. His is currently into a research leading to a Ph.D in Computer Science. His research interests are in Software Engineering and Model Development. He has over 30 publications in both local and international Journals and conferences.

Prof. Nwachukwu Enoch O. is a Professor of Computer Science in the University of Port Harcourt. He has an M.Sc and Ph.D from Manchester University in United Kingdom. He is a fellow of the Nigerian Computer Society and Computer Professional of Nigeria. His areas of research interest include Artificial Intelligence, System Analysis and Software Engineering. He had supervised many MSc and Ph.D research work in his fields and serve as external examiner in different universities in Nigeria and beyond.