Industrial Web Application Customization Mechanism to Improve Software Quality and Productivity

Azham Hussain, Mohammad Nuruzzaman and Hatim Mohamad Tahir

***Abstract*—**Competition in the software market for industrial use is very challenging. Quality and productivity of software is very important to the software industry to remain competitive. Most of the commercial and industrial web applications are complex, hard to implement, risky to maintain and customization requires deep understanding of the requirements. Research showed that customization and reusability may increase the productivity and quality of the software and also decrease the development time. Unfortunately, implementing systematic reuse and customize existing system has proven to be a difficult process. While software engineers continue to struggle with cost and time, reuse has emerged as a good engineering principles and practice in various fields. However, technology to completely integrate user interface, reuse design, customization and implementation is still immature. The aim of this study is to provide a novel visual object sharing technique for designing, customizing, reusing and visualizing web elements to provide a breakthrough solution for the given problems. This technique support and provide rapid development of web-based business application where all of these underlying data and application codes are defined by meta-data, tag library and XSLT schema. This study contributes mainly in the field of reusability and customization for the web application. This study also demonstrated empirical data from two commercial projects and the results indicated that proposed object-oriented application framework (OOAF) is consistently better than traditional methods. By using OOAF, software industries are able to reduce development time, increase the quality and productivity of web application..

***Keywords*—**Reusability, Software Customization, Framework Toolkit, Web Engineering, Software Productivity, Software Metrics.

I. Introduction

Object-oriented (OO) based web application development is not easy, mapping user requirements into a function is complex, customization requires deep understanding and risky to maintain. Technology for completely integrated user interface, reuse design, customization environment and implementation is still immature in the area of web engineering. It is different from traditional web development as it focuses on visual elements [1]. OO software development method includes requirements analysis, system design, development, testing and documentation that enable web engineers to repeat Software Development Life Cycle (SDLC) phases and avoid possible failure of the current ubiquitous web. This revolution makes easier for the web engineers to develop software packages and also made a significant impact to working on it.

Previously, most of the developed applications were procedure-oriented. It is an ever-growing complexity due to an exponential increase in software size. It also makes it unsuitable to reuse and customize based on user preferences. Considering this effort has pushed legacy applications into the new OO-based web application development. There are numerous recurring efforts, particularly in the user interface design and coding phase [2] and [3]. An approach is needed to accomplish the web application customization and reuse design for improving development productivity as well as software quality. The necessity of an OO framework designer becomes obvious due to complexities in object relationship, requires deep understanding of requirements for customization and difficulties in deployment. The idea behind this approach is the integration of design reusability and customization through a software visualization tool. It will increase greater consistency of operation, reduce complexity, easier maintenance, reduced development time, increase productivity and quality of the application [4]. Reuse has a computable and significant impact on reducing development efforts and improving software quality [5].

Web engineers often proclaim *"Do not reinvent, the inverse of reinvention is reuse"*. Many projects fail due to poor quality of application frameworks, poor design, lack of clarification of requirements, developer's skill and administration issues. Therefore, it will be significantly important to examine current problems in Object-Oriented Application Framework (OOAF) development and looking for alternate solutions. There are several factors need to take into consideration when designing OOAF such as requirements simplicity, complexity in object relationship, classes collaboration and complication in using an application framework. A new customization technique is essential to support more effective and rapid development of applications with lower efforts. By integrating component-based software engineering (CBSE), object-oriented technology (OOT) and customization techniques could achieve

This work was funded by Ministry of Education Malaysia, under the Research Acculturation Grant Scheme (RAGS).

A. Hussain, M. Nuruzzaman & H.M Tahir are now with School of Computing, Universiti Utara Malaysia, 06010 UUM Sintok, Kedah, Malaysia. (e-mail: azham.h@uum.edu.my, nayan@egudang.com & hatim@uum.edu.my)

web application productivity improvement as well as quality. This research proposed visual object sharing technique to support customizable and reusable OOAF architecture. A big picture behind the approach to reduce development efforts, complexities and increase productivity through visual object sharing technique in OOAF has been demonstrated.

II. Framework Concept

In an OO software environment, a framework is a "*reusable software component, including reuse of analysis and design*" [6]. According to Wallace and Bruce [7] and Fayad, Hamza and Yi Chen [8] - "*a framework is more than a class hierarchy*". It depends on Hollywood Principle: - "*Don't call us, we'll call you*". It says that web developers handle it by applying inheritance, polymorphism or generalization methods so that developers spend fewer efforts in coding and spend more efforts on the business specific problems. An application could be implemented flexibly and within the shortest time - frame through the framework.

A. Application Framework

An application framework also known as "Toolkit" that allows for the creation of application. It consists of a framework used by software engineers that provides a fundamental structure to support the development of an application for a specific environment. An application framework or toolkit acts as a tool to supply the structure and templates for constructing an application. It becomes popular with the rise of GUI. Web engineers and developers found that to create a user interface (UI) with less effort application framework proved to be a good solution [9]. It provides a standard framework with underlying pre-defined code structure. The intention of designing an application framework is to lessen the issues faced entire software development life cycle (SDLC). Web engineers usually use OOP techniques to implement framework, whereby unique parts of application framework can simply inherit from pre-existing classes in the framework. The advantages of using an application framework include extensibility, simplicity, easier customization, code and design reuse. These advantages lead to lower cost of development, reduction of errors, reduce web development effort, increase quality and rapid application development.

B. Object-oriented Application Framework

Object-oriented Application Framework (OOAF) sets of libraries. It is designed to help web engineers to solve problems and build applications. The aim is to improve the overhead related to common activities in SDLC. According to Williams, Szyperski, and Wittenberg [10], an OO application framework organizes and interconnect software components, generates runtime structure and manages execution of the application.

Object-oriented application framework or OO toolkit acts as a tool to supply the structure and templates for constructing an application. By using OO techniques while implementing the framework, pre-existing classes can be used to build the application easily. The primary benefits of OOAF are componentization, *extensibility for customization, reusability, modularity* and *inversion of control*.

According to Carlos and Pedro [11] - "*OO application framework is a reusable, semi-complete application, which produces custom applications and collaborates to carry out a set of responsibilities*". OOAF provides reusability technique by utilizing the domain knowledge and experiences. It supports to avoid recreating common solutions to the application, decrease development time, cost and improve web application quality such as a design pattern. Design pattern recurring solution for design and development problems.

Object-oriented application framework typically provides core functionality and underlying pre-define the code structure to most of the application like session management, security, caching, interface template and data persistence. By using an appropriate framework, web engineers can save countless of hours. This is achieved through reuse design and code that share across different modules of an application [12].

From the current literature, numerous research works have done in the area of software customization, reusability and productivity improvement of web application development. Every single of them is dealing with dissimilar concern, concept and point of view [13]. It is agreed that universal framework cannot exist due to different domains.

C. Factors Affecting Productivity

Software productivity measure or define is a complex process. Most of the software productivity studies are inadequate and misleading. Productivity development should focus not only on the efficient development, but also should emphasize the quality and value of application developed [14]. There are several factors that impact the clarification of productivity. For example, if the product output has defects and need to rework or bugs fix, it will decrease the productivity.

Most of the researchers related to web application development productivity focused on the study of an individual developer's efficiency. In addition, for large project development team often works with new tools, client participation, requirements and faced developer turnover, unclear goals, complexity, communication between team members that affect software productivity [15] and [16]. Thus, team size and team activities also important factor in project success. Research shows those strong skills team has higher productivity whereby weak skills team has lower productivity overall assigned tasks [17]. Additionally, complexity raises the team size and it reduces the productivity [18]. Their research also showed that productivity varies from C2C, B2B, working environment and software development process. A company with different business sector could accomplish different level of productivity.

III. Research Approach

The approach of this study was divided into four steps which are theoretical study, framework design, development and evaluation.

In the theoretical study, the board range of study was required in many different aspects of current application

frameworks and architectures. This research analyzed concepts of web application design, common practices and assessment benchmarks. In addition, thoughtful study on the existing OOAF, issues interrelated to application framework design strategies; customization and reusability techniques that simplify to implement an OOAF has carried out. This research found that the current OOAF do not provide design reuse with underlying structure generation. Hence, it is vital to realize obstacles and current practices by web developers. Thus, the aim is to undertake all of the mentioned complications by presenting visual object sharing technique in OOAF. To get a good foundation of knowledge, this study review previous literature, issues and articles related to OOAF.

The design for application framework begun after complete requirements obtained. The proposed application framework design is Model-View-Controller (MVC) pattern. Design stage divided into two sections: MVC layer-based design and architectural detail design. Layer-based design outlined high-level strategy and road map for the solution of the problem. MVC Layer-based design proposed for modularity, reusability and extensibility.

This study applied design pattern to solve design problems and blueprint of the proposed application framework for developing web applications. Design of user interface, object interaction, relationship and visual object sharing technique applied to produce a robust structure or desire behavior. In the design phase, efforts directed to visual object sharing technique which generates underlying structure, event handler, and codes so that web engineers will able to focus on implementing value-added services.

Development of an OO application framework is even harder than design. New software components, objects, classes followed the architectural strategy of the OOAF. This stage prepared inventive OOAF interface design concept as shown in Fig. 1 below:

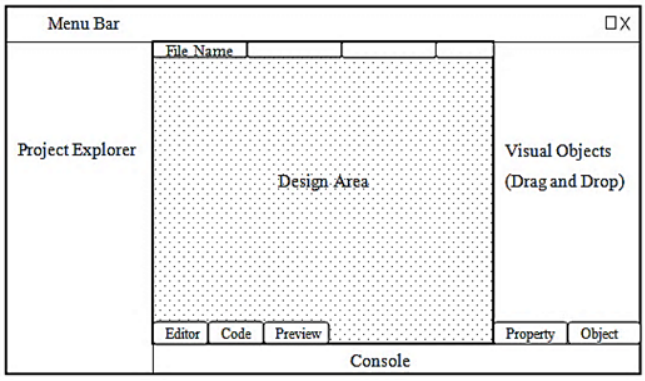

Fig. 1: Preliminary Interface Design of OOAF

This study adopted agile development methodology (XP Programming) for faster requirements gathering, design and development. The combination of Java and .NET used to develop the proposed OOAF. "Eclipse" as Java IDE, JDK-1.7, VS2012 as .NET IDE and .NET 4.0 as GUI designer used in a development environment. We focused on using OOP because it has essential features for developing OOAF.

The final stage is to evaluate OOAF framework. The aim is to estimate productivity of the OOAF toolkit in the actual working environment. A key element for any estimation or measurement is to know what is needed to be measured because without this element, it is impossible to establish a measurement [19] and [20]. We considered software size, function complexity, effort required and process to build the software.

Knowledge gained from the development phase merged into a case study and requested web engineers to develop modules and observed its performance on an individual. In this stage, the study focused on novice and professional web engineers who asked to design and develop "Automatic Jobsheet Processing and Invoice Management System" through the proposed OOAF and traditional method. Then questionnaire was provided to collect data from two different projects. One project was developed by proposed OOAF, and another project developed by structure method. Both of the projects were similar in terms of domain, product size, programming language, development environment and system requirements.

IV. The Application Design

This section presented a layer-based architecture to describe proposed OOAF. This study also indicates that layer-based architecture and separation of concerns approach reduces application design complexity to develop domain-oriented OO system. The design stage divided into two sections: architectural design and detail application design. Architectural design outlined high-level strategy and road map for the solution of the problem. Detail design identified objects, described in the target implementation language and constructed core functionalities of the OOAF.

A. MVC Layers Design

The design goal of the OOAF is layer-based (MVC) with reusable, interactive and robust UI elements. The architecture of the OOAF divided into five (5) layers as a presentation layer, resource layer, business logic layer, controller layer and database layer. Each layer has a specific function and set of modeling activities. The following section presented the high-level design of the each of the layers.

B. Presentation Layer

Proposed OOAF provides a set of UI elements and design templates that encapsulated visual presentation, event handling, and code generation. Visual objects defined in presentation layer with the associate component model, server-site events and object stateful data. They form the foundation of custom web application development. Fig. 2 below show the presentation layer module that used Java Server Pages (JSP) API from Sun Microsystems.

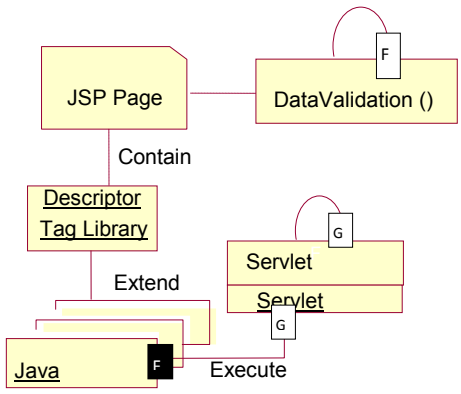

Fig. 2: Presentation Layer Module

C. Resources Layer

On top of the presentation layer, resources layer is responsible for both sharing of resources and the enforcement of visual object access based on the available resources. Since access to resources is most often highly security-critical, the resources layer has to provide important visual object properties. The resource layer also contains a template engine or service agent which is liable for communicating with other services to retrieve resources as shown in Fig. 3 below:

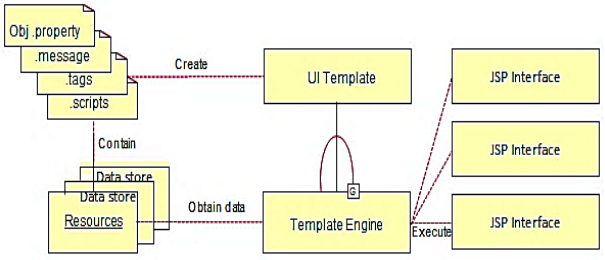

Fig. 3: Resources Layer Module

D. Business Logic Layer

Model-View-Controller (MVC) based system develops and implement only in Java/Servlet. If UI change, then OOAF generates relevant HTML code, meta-data, XML files, event handling and underlying structure. Similarly, if the business logic changes, then only visual object's event codes need to change. An application develops with MVC methodology should easier to customize web application. Fig. 4 shows the business logic layer of proposed OOAF. Logic layer is concerned with business logic only. A web designer who knows nothing about Java can concentrate on the look and feel of the UI layout. Whereby, web developers can focus on the core logic and Java Beans. A change to the user interface only concerns changes to the relevant JSP only.

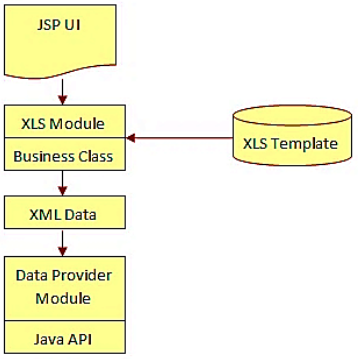

Fig 4: Business Logic Layer Module

E. Controller Layer

In OOAF architectural design, system flow is intermediated by controller layer. It delegates the request to the controller technique or handler. A controller receives input from the presentation layer, initiates a visual object's instance and executes action as shown in Fig. 5. If an invalid input is sent to the handler, the object instance notifies to handler to forward an error and notifies to re-input.

The controller layer consists of Java classes and interfaces that provide the runtime environment for the components used within an OOAF. It is responsible for handling application level events such as switching, dispatching events to UI modules, authentication, state management, error processing, log on and log off.

The proposed OOAF controller technique has a specialized view since it is integrated with visual object sharing technique. It is actually one or more visual UI element in JSP and therefore model can inject what it should display. The controller could add necessary parameterization, so that the JSP event controller can observe the input.

As shown in Fig. 5, controller adopts the request from client browser or presentation layer and dispatches UI elements or event handlers. UI element represents object state or business logic. It notifies any observer when data changes. Every request passes via controller who retrieves visual object values. The visual object sends the result back to the controller. The controller will take the result and forward to JSP.

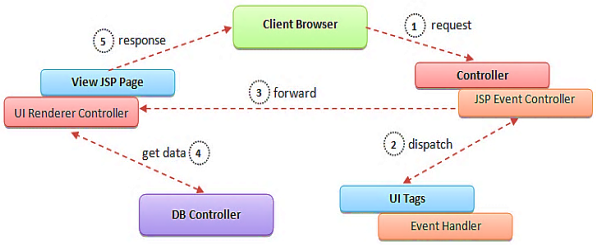

Fig. 5: Controller Layer Module

F. Database Layer

The database layer provides and stores information. It is very crucial and internal layer that is protected from user's view. There are no directly accesses to the database from the upper layers. Its access is routed through the database layer as shown in Fig. 6 below:

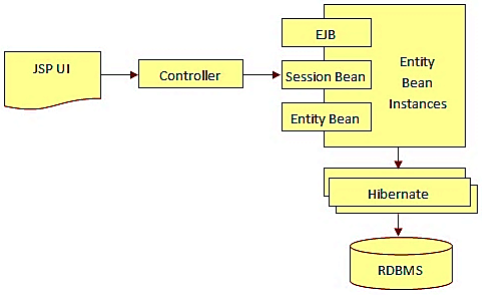

Fig. 6: Database Layer Module

All query subjects in this layer are imported from the data source. Because these query subjects point directly to the database, actions such as join, relationship, or renaming of query items cannot be done. Future model changes caused by schema changes are made in this layer. All other layers are unaffected by the schema changes. The ability to leverage code generation tools is one of the keys to a flexible architecture. Proposed OOAF comes with this feature. It produced Java sources, object-relational mapping and configuration XML files.

V. Detail Architectural Design

This section focused on detail design which is the process of defining the lower-level components, modules and interfaces. We verified detailed designs in design reviews and level-by-level. We used OO design method to define module processing and divided into four (4) categories- conceptual architecture, module diagram, class diagram and presentation layer. Fig. 7 shows the relationships among these architectural layers.

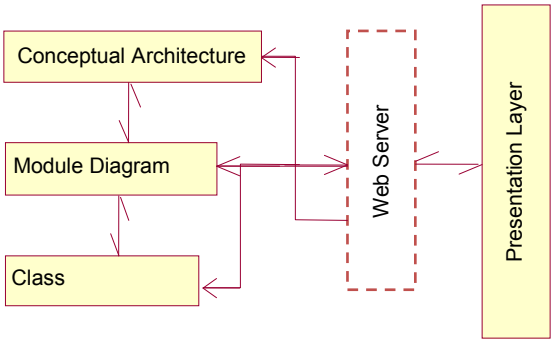

Fig.7: Relationship among the Architecture Layers

The following sub-topics described detail on conceptual architecture, module diagram, class diagram and presentation layer.

A. Conceptual Architecture Design

The conceptual architecture is very high-level structure of an application. It describes the major design elements and relationships among them. Fig. 8 demonstrated what developed OOAF can generate, how the data acquisition is connected, underlying structure and Java classes and mapping files.

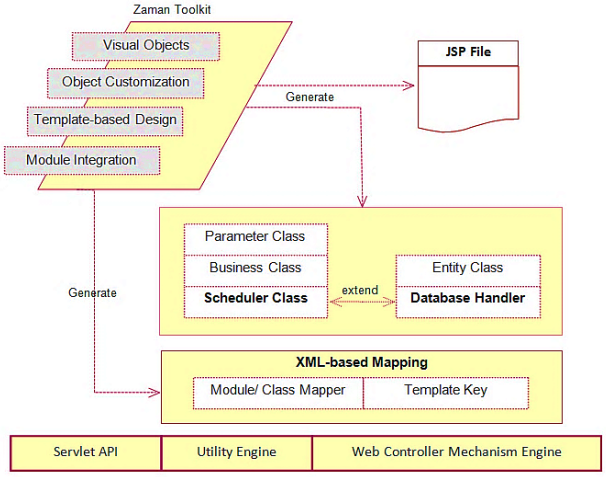

Fig. 8: Conceptual Architecture Design

Fig. 8 showed that developed OOAF contains UI elements, visual objects, template design and object customization technique. Upon interface design completion, OOAF automatically generates a JSP file, Java classes with event handlers, business class, entity class, database handler, relevant data elements and XML mapping files.

B. Module Diagram Design

The OOAF's module architecture encompasses into two structures- functional decomposition and UI layer. Functional decomposition captures the way system is logically decomposed into subsystem, modules, file management, handler, controller, parameter, resources and abstract program units as shown in Fig 9. It captures visual object interrelationships in terms of exported and imported interface.

C. Class Diagram Design

The class architecture is used to organize the source code into packages, directories and libraries. This facilitates system building, installation, configuration management and minimizes dependencies among sub-projects to enforce import/export constraints specified in the module architecture. Fig. 10 shows the sample class diagram of visual element (FreeTextBox) and JSP Page.

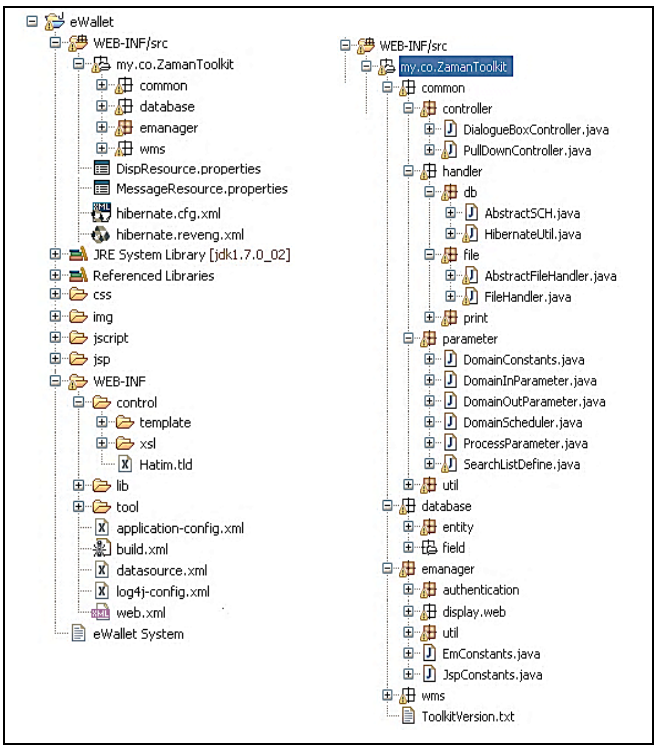

Fig. 9: Module Architecture of OOAF Application

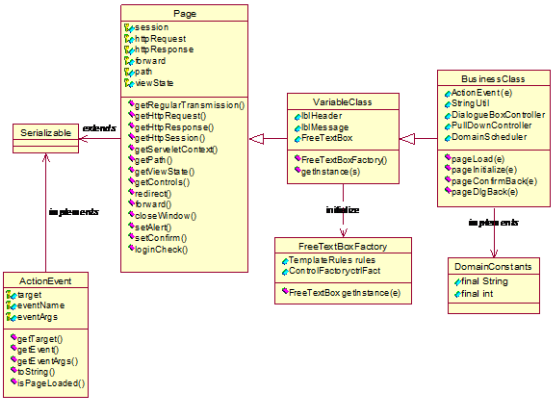

Fig. 10: Class Diagram of a Visual Object

As we can see that "Page" class is superclass which extends Serializable class and implements the ActionEvent class. Page class is parent and controls HttpRequest, HttpResponse, HttpSession, viewstate, forward, redirect, alert and many more. Visual object initializes "Factory" class to display the object into JSP through XML.

D. Presentation Layer Design

The presentation layer is responsible for the binding object behaviours to page elements, display visual objects into JSP and destroys objects binding. This allows a dynamic and loose association of bindings on a particular page. This approach enables resource reusability.

Fig. 11 shows the steps which occur when a JSP page request and loaded. Initially OOAF will define page header <h: Head>, the body of the page to create object binding definition and servlet execution to load the JSP page. When the page loaded into a browser, object binds with OOAF where binding declaration used XML and creates a binding instance. Bindings can be declared as a group or individually. Initial bindings on a page typically established using a binding element. In this instance, the CSS, JavaScript class, events bound to the HTML element on the web page and display visual objects.

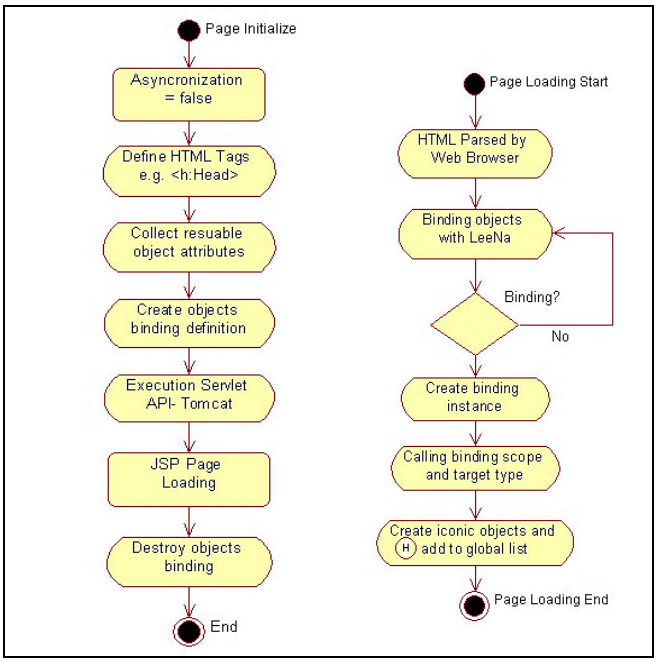

Fig. 11: Visual Object Presentation Layer

VI. Interface Design

Fig. 12 shows the proposed OOAF toolkit prototype. The prototype divided into five parts-(1) Menu and Toolbar- where contains execution commands and other features, (2) Project explorer- where shows project's JSP files, (3) Working area- where visual object can be drug to design user interface/ create JSP, (4) Toolbox- visual objects or user interface elements and view object properties (5) Console area- where shows relevant error and guided message.

Web developers require to drag visual elements from no. (4) to working area at no. (3). OOAF will visualize the drag object and developer need to name it. Upon completion UI design, developers require to click "Generate" button from the toolbar which will produce all related codes and JSP file. Fig. 13 shows complete and final user interface of a module generated from proposed OOAF.

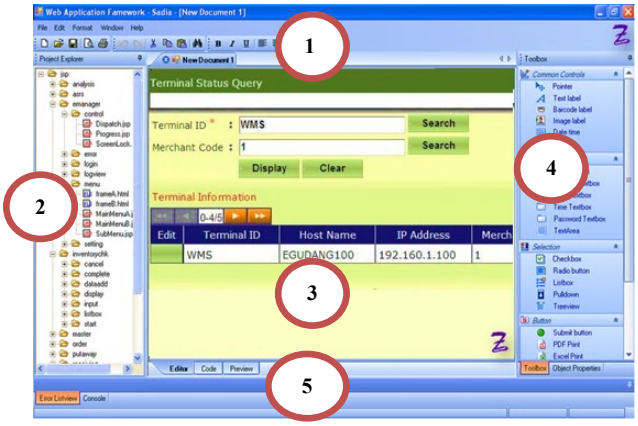

Fig 12: Interface Design of OOAF

Fig. 13: A Complete UI Design form

VII. OOAF Evaluation

There are several factors affect case studies for instance; management issues, personal capabilities, experiences, level of skills and communication. In order to estimate accurate software productivity, we ensure that both of the projects were supervised by a similar set of skills, experiences and personnel capability. Both projects were set to non-embedded, similar domain, requirements, level of complexity, design, functionality, development environment, programming language, team size and similar capabilities in team members.

This study compared code size and effort to measure productivity from two projects. The company is "*MSC Status*" certified and hires about 50 staffs, where nearly 85% directly involved in software development. The data were documented by members of each development team leader and used to controlling and managing projects. The type of application has developed is "Automatic Jobsheet Processing and Invoice Management System" using 3GL. The core programming language was Java. Project 1 (SWT-P1) was developed by proposing OOAF, while project 2 (ENT-P2) was developed in traditional method.

Measuring productivity in OOAF is complex because OOAF use codes reuse techniques. Sometimes developers reuse whole program without modification and often modify a module to some extent. However, we accounted reused code, distinguished a module with one modified line from a module with 100 modified lines. Thus, this research considered the notion of reuse on an ordinal scale as shown Tab. 1 below:

Type	Description
New Code	None of the code comes from previously constructed class.
Reuse	Code reused without any changes or less than 25% of lines of code in a class were modified.
Modified	More than 30% of lines of code in a class were modified.

Tab. 1: Code Reuse Classification

The total software size comprised of all new, reuse and modified codes added together as shown in Tab. 2. It is required to measure the modification of existing classes. The change size metrics used to count effective SLOC modified or adapted from existing class.

Project	Method	Code Size [KSLOC]			
		New	Reuse	Modified	Total KSLOC
SWT-P1	OOAF	75	275	15	365
ENT-P2	Traditional	107	202	20	329

Tab. 2: Actual Project Size Data

The duration of the development was documented in man-day (MD) from project started date to the deployment date. The total efforts consist of the sum of the man days comprising analysis, design, development, test and all others days as shown in Tab. 3 below. All other days comprise time spent in discovering defects, configuration, learning new tools and supervision of the project

Project	Effort [MD]					
	Design	Develop	Test	Rework	All Others	Total Effort
SWT-P1	15	45	7	8	10	85
ENT-P2	20	55	10	12	10	107

Tab. 3: Actual Efforts Data

Both of the projects SWT-P1 and ENT-P2 were both non-embedded, partial real-time with same functionalities. They were implemented on commercial servers. The complexity was medium on their formal QA level. The teams were formed based on experiences and level of skills. The team that developed using OOAF was well-trained in this discipline and tools used.

The result showed that OOAF method has a significant factor affecting productivity. OOAF can dramatically improve higher productivity over traditional methods. Even though the result is promising, it cannot be generalized as it is only being implemented at two projects.

VIII. Conclusion

The purpose of this study is to reduce development effort and increase productivity. As markets more competitive, continues productivity improvement becomes a major concern in the software industry. OOAF and reuse is an important aspect of software productivity when size is provided as input to effort and productivity model. This study demonstrated that it provides a breakthrough solution for software customization, design reuse and productivity improvement. It is fair to claim that the goal of this study is achieved.

This study discovers that poor design, static interface, lack of clarification of requirements, complexity of the domain, complex relationship among objects, size of classes and proper documentation are the root cause of software complexity. We also discover that productivity varies on individual developer's efficiency, understanding requirements, complexity, communication between team members, team size, working environment, process, development tools, and methods. This study overcomes problems above by proposing OOAF.

We proposed a novel OOAF which generally increased web development productivity and decrease development efforts for web engineers. It supports developer easier customize, reuse and flexible module integration without any code modification.

However, this study does not cover more detail design of OOAF. A more detail design is necessary to make the concept of framework widely accepted. The detailed analysis also required in different domains to explore for better requirements, reusable and customization technique.

We used visual object sharing technique to visualize an object. It could be great if model done from Unified Modeling Language (UML) diagrams. Determining the feasibility of using requirements to generate business logic code remains a future research topic.

Acknowledgment

The authors thank the Ministry of Education Malaysia and Universiti Utara Malaysia for support under the Research Acculturation Grant Scheme (RAGS).

References

[1] K. Parminder, and S. Hardeep, "Certification process of software components," *SIGSOFT Softw. Eng. Notes*, vol. 33, no. 4, 2008, pp. 1-6

[2] O. Pastor, Y. Yu, L. Liu, E.S.K. Yu, and J. Mylopoulos, "Quality-Based Software Reuse," *Advanced Information Systems Engineering*, Lecture Notes in Computer Science 3520, Springer Berlin Heidelberg, 2005, pp. 535-550.

[3] O. David, J.C. Ascough Ii, W. Lloyd, T.R. Green, K.W. Rojas, G.H. Leavesley, and L.R. Ahuja, "A software engineering perspective on environmental modeling framework design: The Object Modeling System," *Environmental Modelling & Software*, vol. 39, no. 0, 2013, pp. 201-213

[4] W.B. Frakes, and K. Kyo, "Software reuse research: status and future," *Software Engineering, IEEE Transactions on*, vol. 31, no. 7, 2005, pp. 529-536.

[5] L. Kung-Kiu, and W. Zheng, "Software Component Models," *Software Engineering, IEEE Transactions on*, vol. 33, no. 10, 2007, pp. 709-724.

[6] S. Aleksandar, and D. Aleksandar, *"Architectural framework for dynamic web-applications,"* in Proceedings of the 9th International Conference on Computer Systems and Technologies and Workshop for PhD Students in Computing, Gabrovo, Bulgaria, 2008.

[7] Wallace & Bruce (2011). *"A Hole for every Component and every Component in its Hole", Existential Programming-2011, Retrieved on 9th April, 2011 from* http://existentialprogramming.blogspot.com/2010/05/hole-for-every-component-and-every.html

[8] Fayad, M.E., Hamza, S.H. & Yi Chen, (2005). *A Framework for Developing Design Models with Analysis and Design Patterns"*, Communication of IEEE, 0-7803-9093-8/05.

[9] Z. Bosikashvili, (2014)*."Extension of the Architecture of Software Systems with Elements Artificial Intelligence,"* presented at 14th International Conference on Applied Computer Science, Cambridge, MA, USA.

[10] Williams, A.S, Szyperski, C.A., and Wittenberg, C. (2012), *XML Application Framework*, Patent No. US 8132148B2, pp. 37, Date of Patent- Mar 6, 2012.

[11] Carlos, J. & A. Pedro (2002). *Domain Analysis of Object-oriented Frameworks in FrameDoc*. SEKE'02, Ischia, Italy. Journal of ACM, 1-58-113-556-4/02/0700, vol. 4, no. 2, pp. 27-33.

[12] J. Kuchta, (2014)*."Framework Reuse – Heaven or Hell,"* presented at 13th International Conference on Software Engineering, Parallel and Distributed Systems (SEPADS '14), Gdansk, Poland.

[13] M. Nuruzzaman, A. Hussain, and H. M. Tahir, (2012). "Towards Increasing Web Application Development Productivity through Object-Oriented Framework," International Journal of Future Computer and Communication, vol. 2, pp. 220.

[14] A. Trendowicz, and J. Münch, " Chapter 6 Factors Influencing Software Development Productivity-State-of-the-Art and Industrial Experiences," *Advances in Computers*, pp. 185-241: Elsevier, 2009.

[15] R. Premraj, M. Shepperd, B. Kitchenham *et al.*, "An empirical analysis of software productivity over time." pp. 10 pp.-37, 2005.

[16] Sudhakar, P. G., Farooq, A., & Patnaik, S. (2012). Measuring productivity of software development teams. Serbian Journal of Management, 7(1), 65-75.

[17] E. Nwelih and I.F. Amadin , 2008. Modeling Software Reuse in Traditional Productivity Model. Asian Journal of Information Technology, 7: 484-488.

[18] Hernández-López, A., Colomo-Palacios, R., García-Crespo, Á. and Cabezas-Isla, F. (2011). *Software Engineering Productivity: Concepts, Issues and Challenges*‖, International Journal of Information Technology Project Management, vol. 2, no. 1, pp. 37-47.

[19] N. H. Hassan, S. R. Selamat, S. Sahib, and B. Hussin, (2014). *"Incorporating Evaluation Criteria in Meta-process of Classification to Increase the Acceptance Level"* presented at 13th International Conference on Applied Computer and Applied Computational Science (ACACOS '14), Kuala Lumpur.

[20] Tangen, S. (2005). Demystifying productivity and performance, International Journal of Productivity and Performance Management, vol. 54, no. 1, pp. 34-46

.