International Journal of Software Engineering and Its Application
Vol.12, No.1 (2018), pp.57-66
http://dx.doi.org/10.21742/ijseia.2018.12.1.05

The Integration of MVC Framework in Rapid Application Development (RAD) Process Model

Carlos Ramos[1], Subramaniam Ganesan[2] and Ronnie Caytiles[3*]

[1]ISEP-IPP, Portugal
[2]Oakland University Rochester, MI, USA
[3]Department of Multimedia, Hannam University, Korea
rdcaytiles@gmail.com

Abstract

This paper presents an innovative methodology for software development optimizing the features of a model-view-controller (MVC) framework. The MVC framework is integrated with the Rapid Application Development (RAD) process model in order to map the specified user requirements into the architecture making the individual components of the software application to be simultaneously developed. A software prototype is rapidly designed, implemented, tested in an iterative process. Quality of Service (QoS) software application development is guaranteed because users can be directly in the testing process.

Keywords: *Rapid Application Development (RAD,) MVC, Software Development*

1. Introduction

Over the past decades, software development process models have evolved starting with the primitive linear sequential process model. By then, innovative software development solutions has been introduced in order to meet customer demands and the fast growing market in software products. The early software development process models are considered to have numerous setbacks that causes the delays on the delivery of software products as well as its performance sometimes suffer from faults and failures [1].

This paper outlines the integration of the MVC framework into RAD process model in order to alleviate the delivery of software products and guarantee the QoS of the performance of such application. A review of some popular software development process models have been presented and RAD process model is chosen because of its modular-based approach on software development. The components of the software system can be developed simultaneously and reuse of its model components becomes available. The emergence of MVC architectural framework has provided and insight for delivering timely QoS guaranteed software systems.

The rest of this paper is organized as follows: Section 2 discusses about the different software development stages and introduces some popular software development process models; Section 3 outlines the concepts of MVC for software development; the integration of the MVC framework into RAD process model is outlined in Section 4; and the concluding remarks in Section 5.

2. Software Development Process Models

A software or application development refers to a planned and structured software development which includes development stages such as problem analysis, requirements gathering, design, implementation, testing, deployment, and maintenance and bug fixing.

Received (October 15, 2017), Review Result (January 15, 2018), Accepted (January 25, 2018)
* Corresponding Author

Other process involves research, prototyping, documentation, reuse and re-engineering, coding, and all activities that can be conducted for the creation of software application products [1].

2.1. Features of Well-structured Software

The following features define well-structured software (Figure 1):

- Usability. The software is easy to use wherein the users can communicate with the system effectively.
- Portability. The software can be executed in different platforms and architectures.
- Reusability. The software can be transferred from one system to another.

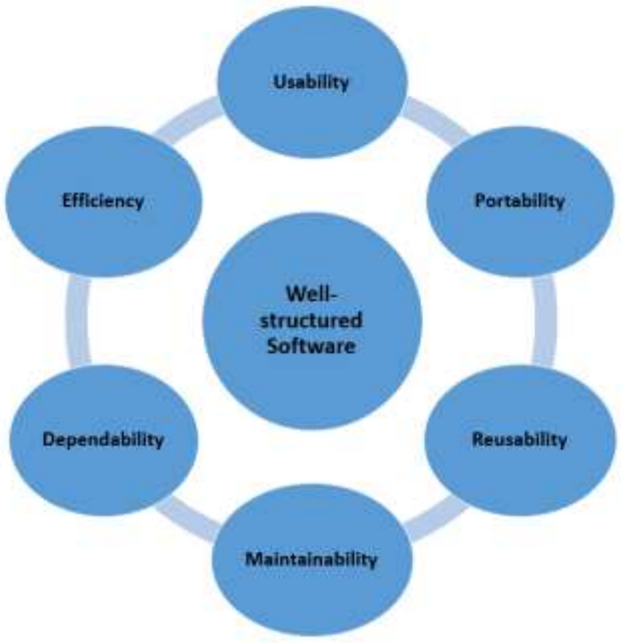

Figure 1. Well-Structured Software

- Maintainability. The software can adapt to changes over time, that is, upgrading and maintaining could be easy.
- Dependability. The software must be reliable, secure, and safe.
- Efficiency. The software is capable of using the resources efficiently.

2.2. Stages of Software Development

Software development process models ranges from conventional to innovative methodologies such as linear sequential, prototyping, iterative and incremental development, spiral development model, rapid application development (RAD), extreme programming, rational unified process (RUP), and agile development. These development process models generally share some of the following stages of development:

- Problem Analysis Phase. This phase identifies the problems that a user wanted to address. The results of this phase will be the basis for identifying the user requirements of the software application project.

- Requirements Gathering Phase. It also refers to as requirements engineering which is the process of identifying the desired expectations for a software application project. The concept of the software application project is explored and refined, and the user requirements based on the results of the problem analysis phase are elicited. It answers the question "what the product is supposed to do". This phase includes analysis, specification, and validation.
- Design Phase. The architecture of the software application project is established in this phase. The architectural design could be first broken into components called modules and each of the modules will be designed. The design of the software application will be based on the specified expectations in the requirements gathering phase. While the requirements gathering phase represents that "what" phase, the design phase is representing the "how" phase.
- Implementation (coding) Phase. The architecture and requirement documents give the guidance for the software developers for the implementation. The software application is built exactly on what has been specified on the previous phases.
- Testing. This is to validate the quality of the developed software application project. Testing can be done while the implementation is ongoing. It includes component, integration, and user testing. Then, the software application project is tested as a whole when components are combined.
- Deployment. The software application project as a complete product is delivered to the users for use.
- Maintenance and bug fixing. This refers to the correction of faults of the software product in order to improve its performance.

The results on the problem analysis lead to the selection of the best software development process solution. The next sections will provide a discussion some of the notable software development process models. Its advantages and disadvantages are also identified in order to visualize its effectiveness whenever applied to the results of the problem analysis.

2.3. Linear Sequential Process Model

The Linear Sequential Process Model or the waterfall process model development is a linear sequence of the software development phases from the requirements analysis, design, implementation, testing or validation, deployment, and maintenance [2, 3]. Every phase needs to be done in order to start the next phase. The linear sequential process model is outlined in Figure 2.

The advantages of Linear Sequential Process model include the following [1, 2, 3]:

- Simple. This model is easy to understand and use.
- Rigidity. Management is easy for each phase contains specific deliverables and review process.
- Measurable. It is very easy to measure the progress for each phase in the development.
- It provides a basis for other software process models.
- Quality. Ensured quality through the orderly sequence of development phases.

- Reliability. It ensures reliability since every phase must be completed before going through the next phases.
- The development phases do not overlap; every phase needs to be completed at a time.
- It works well with smaller projects wherein the requirements can be well understood.

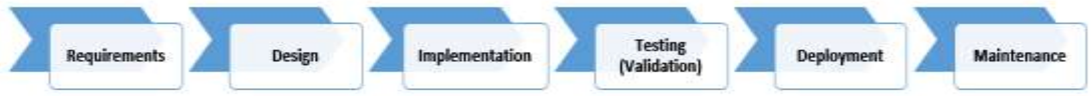

Figure 2. Linear Sequential Process Model

The disadvantages of the Linear Sequential Process model include the following [1, 2, 3]:

- Inflexible. Changes are not allowed because it strictly follows a linear sequential flow of development phases. It cannot address circumstances when the specified requirements may change during the course of the software development because the end-users are only involved in the requirements phase and in the operations and maintenance phase.
- Slow. It is very difficult on deciding when to end or start a developmental phase.
- It is very difficult to address the missing specified requirements. It is not suitable for software projects that the requirements are subject for moderate or high risk of changing.
- It is not suitable for long and ongoing projects or systems.
- Problems or unexpected errors can only be determined during the testing phase. It is also very difficult to go back to the requirements stage if developers have to change something.
- It is not suitable for complex and object-oriented software projects.

2.4. Prototyping

The partially development of software and applications in prototypes allows the end-users and software developers to examine and understand the concepts of the proposed developed software system [1, 3]. The prototyping software development model is outlined in Figure 3.

The advantages of prototyping process model include the following:

- Since the performance of the functionalities of the software system is tested early, the errors can also be detected much earlier.
- It is very easy to address the missing requirements and functionalities.
- It encourages innovative and flexible software designs.
- The users are actively and have a direct involvement in the development and in defining the human-computer interaction (HCI) requirements of the software system.

- The users are provided with a better understanding of the software system as they are directly involved in the development.
- Better solutions are enabled due to quick user feedback.

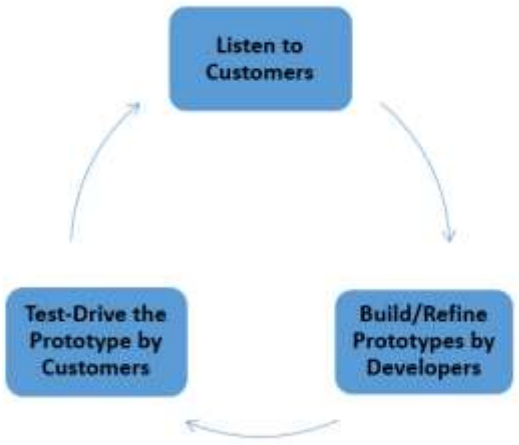
Listen to Customers
Build/Refine Prototypes by Developers
Test-Drive the Prototype by Customers

Figure 3. Prototyping

The disadvantages of the prototyping process model include the following:

- It is expected in having frequent significant changes of requirements.
- The documentation for future developments can be neglected.
- Costly. The iterations can add significant numbers to project budgets and schedules.
- Complexity of software systems will be increased since the scope of the project can expand due to significant changes in the requirements.

2.5. Spiral Process Model

The combination of the iterative nature of prototyping and the systematic aspects of linear sequential process model have led to an innovative spiral process model. The rapid development of incremental prototypes or versions of the proposed system are provided [1].

The advantages of the spiral process model include the following:

- The risk avoidance is enhanced.
- Other process models can be incorporated.
- It has a strong approval and documentation control.
- It is possible to add additional functionalities at the later date.

The disadvantages of the spiral process model include the following:

- Control in moving from one cycle to another is not established.
- Reusability is limited since it is quite complex.
- The cycles can affect the schedule if the exact deadlines of cycle terminations will not be determined.

- Costly.
- The success of the software project is highly dependent on the risk analysis phase.
- It is not suitable for smaller software projects.

2.6. RAD Process Model

Rapid application development (RAD) consists of iterative development and construction of project prototypes. It has an extremely short development cycle and integrates modular-based construction approach [1, 2]. The RAD process model is outlined in Figure 4.

The advantages of the RAD software development process model include the following:

- Finished software can be delivered faster.
- Faster delivery minimizes cost.
- System design can be changed rapidly as the user demands.
- It encourages customer feedback.
- The reusability of components is increased.

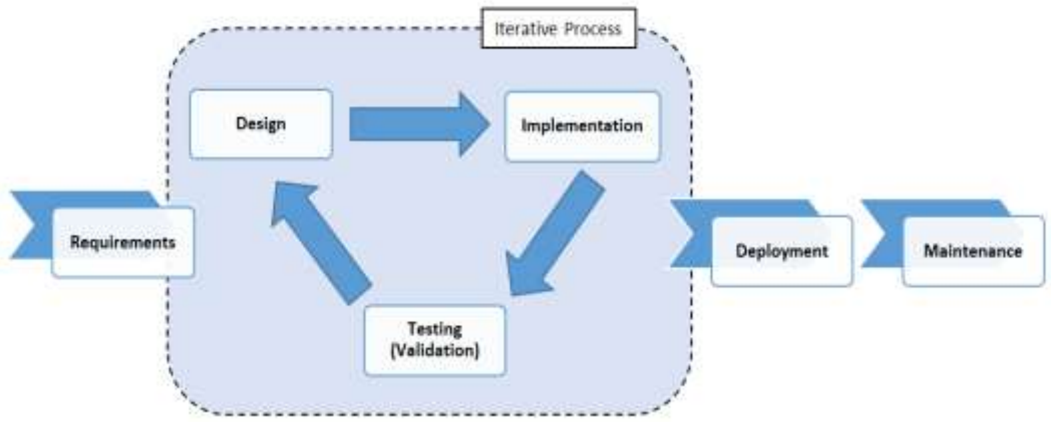

Figure 4. RAD Process Model

The disadvantages of the RAD software development process model include the following:

- It requires a number of developers, thus, it lacks on scalability. The requirements are dependent on the strong software development team and their individual performances.
- Interoperability with existing systems can be complex.
- The faster delivery and lower cost can affect the quality of the delivered product.
- It is only suitable for modularized systems.

2.7. Other Process Models

Other software development process models that are popular include incremental software development model, agile software development, concurrent development model, evolutionary process models, unified process model, Scrum, extreme programming, and many more [1, 2].

3. The MVC Framework

The Model–View–Controller (MVC) is a software architectural platform used in a software application development. It involves the division of a software application into three organized components (i.e., model, view, and controller) wherein the computer representation of information is visualized by the user [4, 7]. The MVC framework tends to provide simultaneous development of the different components. It also allows the developers to reuse the components quickly and easily for other application since each component can be independent from each other.

- Model. The model is independent of the graphical user interface that adapts the behavior of the software application. It signifies the information or data of the application and defines the logic for manipulating them. It directly manages the data, logic, and rules of the software application [7].
- View. The view corresponds to any output representation of information or elements of the user interface (graphical and/or textual output) such as text, checkbox items, a chart or a diagram, and so on [5].
- Controller. The controller interprets user inputs from the mouse and keyboard converting it to commands for the model or view components [4, 5].

The MVC framework interactions are outlined in Figure 5.

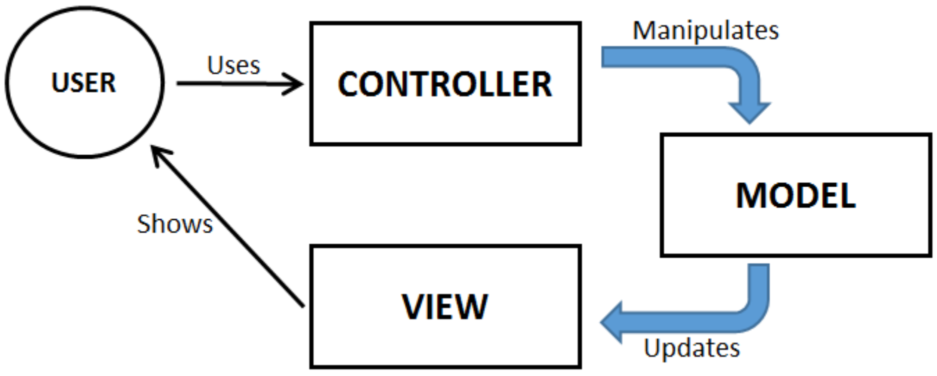

Figure 5. MVC Components

The interactions between the three components of the MVC framework as shown in Figure 6 are summarized as follows [6]:

- First, the user provides for the input information for the modifications of the model components of the software application through the controller.

- Then, the controller interprets these inputs into commands to the model to update or modify the component. The reuse of controllers cannot be done since controllers are application specific.
- After a particular component in the model has been modified, the model notifies the associated view component to update the output representation of information or elements. The model also gives feedback to the controller of the updates in order that some controller commands can be changed. The model components are reusable.
- The view generates output representations based on the model update and display it to the user.

4. Integration of an MVC Framework for Software Development

Innovative software development process models can be chosen by software developers based on the results of the problem analysis and the requirements that were specified for the software application project. The integration of MVC framework for the software development could make the work of developers easier and convenient. In this paper, the RAD process model was chosen since the development of software systems can be much faster and MVC framework can fit very well on its implementation.

During the design phase of the development, the specified requirements can be mapped into the software application architecture using an MVC framework. The architecture defines the components, their interface, and behaviors. A quick design of the software application prototype is built. The software process includes an iterative process of design, implementation, and testing as shown in Figure 6.

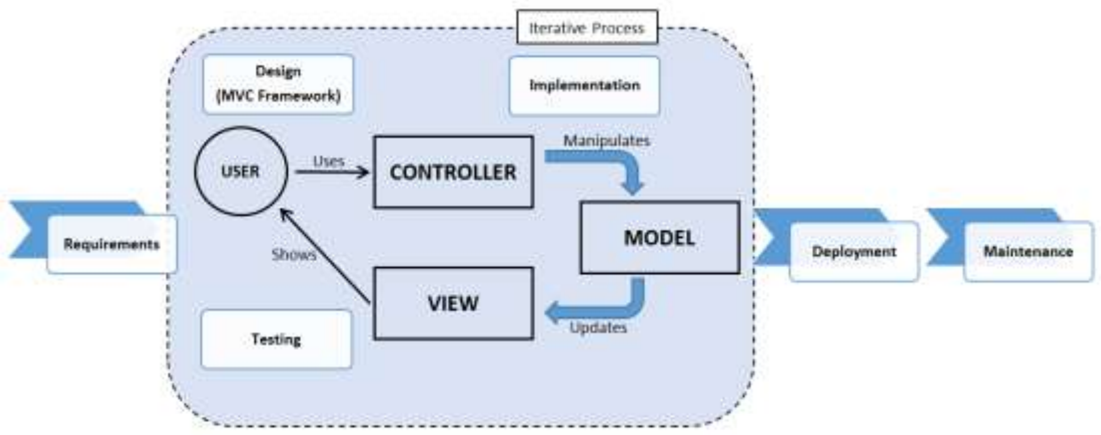

Figure 6. MVC Framework based Software Development Process

The following is the summary of the flow of the development phases for the MVC based software development process.

- First, the requirements are elicited based on the analysis of the problem, user expectations, and necessary software functionalities.
- The specified requirements are then mapped into the MVC architecture. The software project as a whole is broken down into components called modules. Each module can represent a single functionality (user requirement) of the software project.
- Based on the designed architecture, a quick implementation of the prototype is built.

- The prototype is then tested that includes component testing, integration testing, and, system (combined components) testing.
- With the user feedback, the architecture is revisited, and another prototype is built.
- The iterative process will continue until the software system validity, performance, could be well satisfying.
- Finally, the software application project will be deployed to the customer for use and a continuous monitoring will be conducted in order to ensure the quality of service the software product can provide.

5. Conclusions

This paper deals with the integration of MVC framework in the rapid application development (RAD) software development process model in order to address significant delays on the life cycle of software development. The MVC framework alleviates the software development by allowing simultaneous development of several modules or components of a software system. It also allows reusability of model components when developing other software applications. Moreover, the outlined modification of the software development process allows for the delivery of a quality of service (QoS) performance of a software system since it undergoes an iterative modifications and testing in which the user can be directly involved.

References

[1] Centers for Medicare & Medicaid Services (CMS) Office of Information Service (2008), "Selecting a development approach", Webarticle, United States Department of Health and Human Services (HHS), Re-validated: (2008) March 27, Retrieved: **(2018)** March 8.

[2] http://en.wikipedia.org/wiki/Software_development_process, Retrieved: **(2018)** March 9.

[3] http://en.wikibooks.org/wiki/Introduction_to_Software_Engineering/Process/Methodology, Retrieved: **(2018)** March 9.

[4] T. Reenskaug and J. O. Coplien, "The DCI Architecture: A New Vision of Object-Oriented Programming", http://www.artima.com/articles/dci_vision.html, (2009) March 20, Retrieved: **(2018)** March 12.

[5] http://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller, Retrieved: **(2018)** March 11.

[6] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal, "Pattern-Oriented Software Architecture", A System of Patterns Hardcover, Wiley, vol 1, **(1996)** August 16, ISBN-13: 978-0471958697, ISBN-10: 0471958697.

[7] S. Burbeck, "Applications Programming in Smalltalk-80: How to use Model–View–Controller (MVC), **(1992)**, https://web.archive.org/web/20120729161926/http://st-www.cs.illinois.edu/users/smarch/st-docs/mvc.html.