Open International Journal of Informatics (OIJI)

Experiences in Software Development Model Selection and MVC Design Pattern Implementation for Kuala Lumpur City Hall Traffic Offence Management System

Othman Mohd Yusop, Md Nafees Mahbub, Saiful Adli Ismail , Azri Azmi , Nazri Kama , Haslina Md Sarkan

Razak Faculty of Technology and Informatics,

Universiti Teknologi Malaysia, Jalan Sultan Yahya Petra, 54100 Kuala Lumpur, Malaysia.

Article history

Received:
2 July 2019

Received in revised form:
25 August 2019

Accepted:
30 September 2019

Published online:
3 October 2019

Othman Mohd Yusop
othmanyusop@utm.my

Abstract

Traffic Offence Management system is a crucial system in facilitating information about traffic offenders. As highlighted by Kuala Lumpur City Hall (DBKL), repeated traffic offenders are exist due to semi-auto and manual management of information. Moreover the data is decentralised and scattered, resulting difficult for cross referencing and data retrieval on traffic offenders record. The existing system is less transparent and plausible to breed unfortunate circumstances such as mismanagement at both ends between officers and offenders. Towards a new development of Traffic Offence Management System, this paper findings focus on selecting software development model as the Software Development Life-Cycle (SDLC) and implementation of Model-View Controller (MVC) design pattern for a new enhanced system which will be highlighted through comparative study.

Keywords: Software Development Lifecycle, Software Development Model, Model View Controller Pattern, Traffic Offence Management System.

1. Introduction

Problems such as excessive time consumption in accessing decentralised records, repeated traffic offenders, mismanaged information and semi-auto in managing traffic offenders record are the weaknesses in the current DBKL Traffic Management system. These problems pose a challenge to surmount for the enforcers in maintaining laws and preventing traffic crimes. Decentralised and scattered data escalate the current issues further in retrieving and cross referencing traffic offender record. Existing semiautomated monitoring and traffic offence system has low deterrent mechanism application to restraint repeated traffic offenders, hence the traffic offenders are able to evade the higher penalty or compound as should be imposed on these errant drivers. Moreover, low transparency within the current system is plausible to breed some unlawful circumstances between the officers and offenders.

* Corresponding author. *E-mail address*: othmanyusop@utm.my

Developing a new enhanced system to combat the shortcomings of the current semiautomated traffic offence system seems crucial and inevitable. This study requires collaboration of multiple departments under KL City Hall to curb mismanagement and other weaknesses of the existing system. Several meetings with the authorities were conducted and some features were identified as mandatory to be implemented into the new enhanced system:

i. Should be in a web-based system that allows admin to search notices at optimum level.
ii. Should be able to create notices from several different departments. iii. Should be able to list down all notices based on their status. iv. Should be able to do audit trail through audit log files. v. Should be able to generate report.

Realising the above-mentioned requirements into a full-fledged software requires a proper study on existing software development model as the selected SDLC. A systematic software development model should deliver software products within a stipulated deadline and have acceptable quality [1]. Harshad S. M.(2017) states, the process works by lowering

the cost of development whereas at the same time improving quality and shortening production time through evaluating deficiencies exist in the current existing system.

In this paper, we will conduct a comparative study on several software development models and determine one of the models as the SDLC for the new development of an enhanced Traffic Offence Management system, elaborate the necessity of design pattern usage in implementation phase of the model.

2. Background Study of Several Methodologies

Sommerville I (2007) states the SDLC is a framework and comprises of phases. The process is initiated by a formal request from client to a developer team. The initial stage includes a comprehensive discussion between the client and developers on requirements or software needs and the workable software products. Kay (2002) elaborates on the development stages can be characterised and divided in different ways namely system initiation or planning, requirement analysis and specification, design, implementation, integration and testing, and maintenance and evidently most phases in software development process are either fully composed or partially consisted of these phases.

Many software development models are designed and used to improve their SDLC process. The models adhere to SDLC phases as a warrant for successful achievement in developing a software [4]. Numerous software development models have been created and claimed to be more efficient and better than other models [5]. Finding a software development model to resolve issues in current Traffic Offence Management system, requires the authors to conduct a comparative study in the existing software development models as explained in the next subsection.

2.1 Waterfall Model

Waterfall model consists of a sequence of software lifecycle phases and activities. Developing software based on this model will gradually evolve from one phase to another in downwards manner henceforth known as the waterfall model. Waterfall model phases are analysis, design, implementation, testing, and maintenance as shown in figure 1 below.

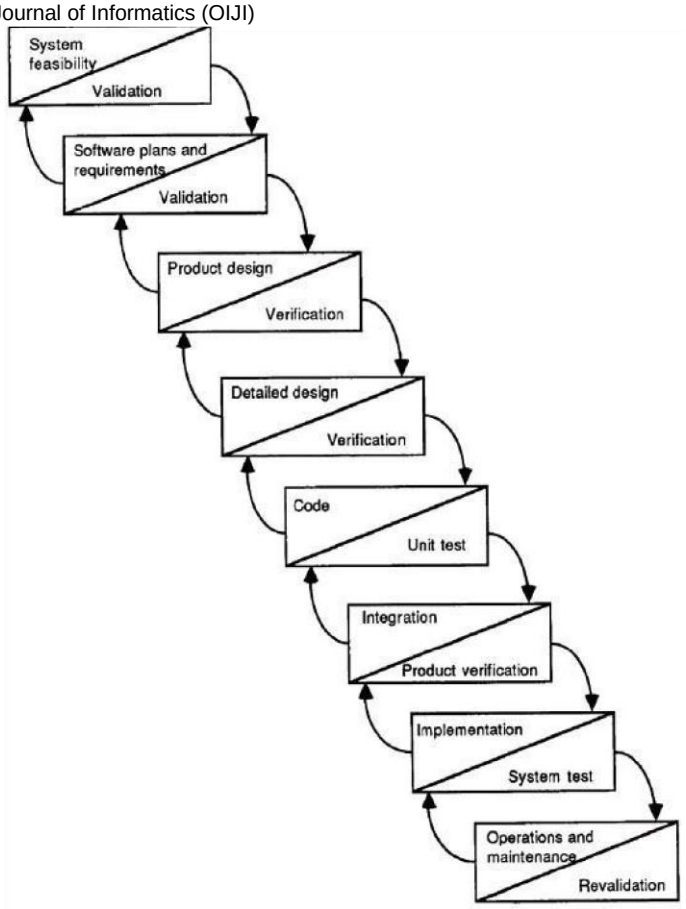

Figure 1: The Watefall Model of Software Development Life-Cycle [6]

All requirements are gathered during feasibility and requirement phases. These requirements are modelled using either structured or object oriented approach. The output from requirement phase will be the input to the design phase. Design model will be derived from the requirement model before the former is set as the input for the implementation phase. At this stage, the source codes have to conform to the design model. The developed source codes must be testing at several stages. These stages are unit testing, integration, system and user acceptance test. Once the software delivered, any amendment or requirement change, will be fallen under maintenance phase.

The model proceeds from one phase to another after the current phase is completed. However, there are different adapted waterfall models that might involve minor or major differences such as Modified Waterfall Model [7].

2.2 Agile Methodology

According to [8], there are distinguish dissimilarities among established software development models and Agile methodology. Agile emphasises people's influence in which both software developers and domain experts are playing significant roles during software development. Dated software development models i.e. [7][6] focus on domain

expert contributions in descriptive form at earlier phases and missed out other software development activities [9]. In Agile method, the domain experts work in tandem with developers in small team throughout the entire software development to ensure the efficiency and effectiveness of the collaboration. Agile makes used of iteration process. Each iteration comprises of high risk requirements which are ratified by domain experts and the developers as shown in figure 2 below.

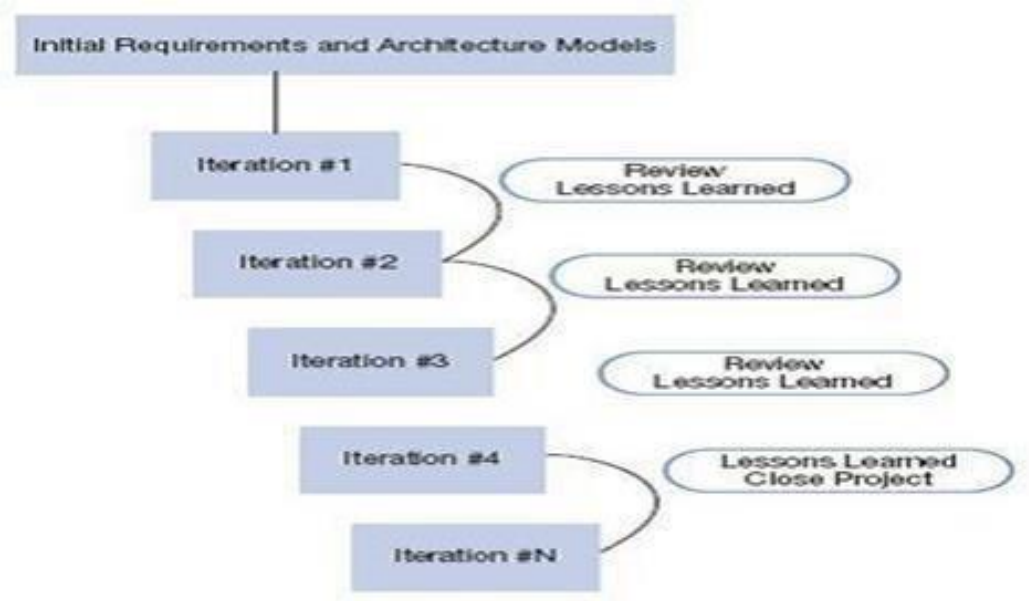

Figure 2: The Agile Methodology [8]

2.3 Spiral Model

The spiral model is also known as the spiral life cycle model. The spiral model combines prototyping feature and waterfall model. Larger companies adopted this model for expensive and complicated projects [6]. Figure 3 below shows the illustration of the spiral model.

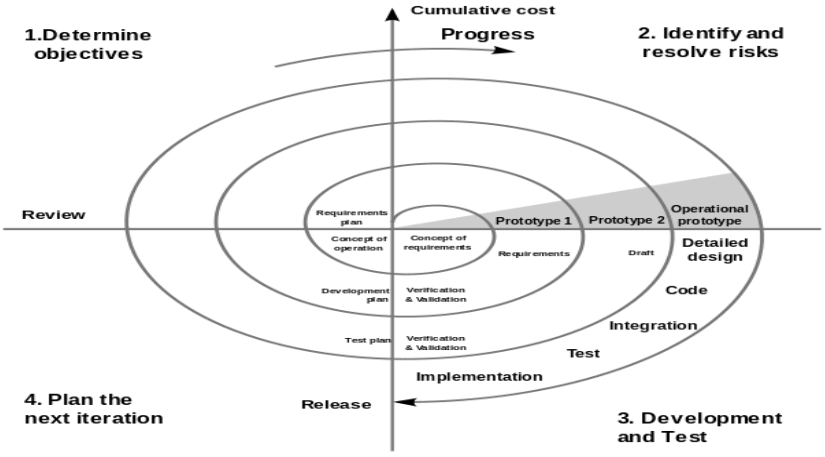

Figure 3: Spiral Model [6]

The project progress spans into four quadrants namely *objectives*, *risks*, *development and test*, and *iteration*. The cycle is repeated to completion, and *validation and verification* took place during a review the next cycle begins. Each cycle yields corresponding prototype and the more cycles are required, the higher cost will incurred by an organisation.

2.4 Discussion on the Software Development Models

Despite the claims being archaic and demise models, these models are the rudimentary SDLC of the modern development models. Through five years periods with 200 practitioners and several thousand projects [10] study shows the traditional software development model especially the waterfall model is still preferable over modern software development model. Some advantages and disadvantages were highlighted by several researchers [11][12] are shown in table 1 below.

Table 1: Advantages and Disadvantages of Several Software Development Models

	Advantages	Disadvantages
	Waterfall Model [12]	
1.	Simple and easy to use like Waterfall Model.	The biggest disadvantage of V-model is that it is the least flexible when comes to requirement changes
2.	In V-Model Tester role will be involved in the requirement phase itself.	If any changes happen mid-way, not only the requirements documents but also the test documentation needs to be updated.

3.	Test activities and planning are done before coding, which saves time.	Development is dine only during the implementation phase so, not early prototype is made.
4.	Easy to keep track of progress.	Client can only see final product, not intermediate modules.
5.	Suitable for small and medium sized projects.	Not suitable for big and complex projects.
	Agile Methodology [12]	
6.	The most important advantages of then using the agile model is agile model is the ability to respond certainly profitable, but if it is a to the changing requirements of the large project, then it becomes project. difficult to judge the efforts and the time required for the project in the software development life cycle.	
7.	There is no guesswork between the development team and the customer, as there is face to face communication and continuous inputs from the client.	For the project to become a success all the stake holders should maintain a constant communication with each other, which may be not possible in certain situations.
8.	Possible to develop workable prototypes.	Difficult to measure progress compared to waterfall because progress happens across several cycles.
9.	Developers can improve their coding skills based on QA feedback.	In agile short sprints does not leave time for design process as a result, designers have to redevelop over and over due to negative feedback.
	Spiral Model [11]	
10.	High amount of risk analysis.	Can be a costly model to use.
11.	Good for large and missioncritical projects.	Risk analysis requires highly specific expertise.
12.	Software is produced early in the software life cycle.	Project's success is highly dependent on the risk analysis phase.
13.	It is suitable for high risk projects, where business needs may be unstable	It is not suitable for low risk projects
14.	Project estimates in terms of cost etc become more realistic as the project moves forward and loops in spiral get completed	May be hard to define objective, schedule, become more verifiable milestones.

The developer for the Traffic Offence Management system is a XYZ Malaysian company. The company was awarded a tender to develop the enhanced version of Traffic Management System for Kuala Lumpur City Hall. As the client, DBKL requires a proper

documentation for all software artefacts. The company opted for the traditional software development model which is the Waterfall Model. The model provides better documentation for each of its phases and the documentation standards would be based on IEEE documentation standards. Nasution, M. F. F., & Weistroffer, H. R. (2009) describe the Waterfall Model helps keeping track development activities through software document and further stated, a well-planned and documented systems development project is more likely to result in a system that meets the expectations of both; the software engineers and the domain experts/ intended users.

Taya, S. (2011) further listed down characteristics of different software development model as shown in table 2 below:

Table 2: Software Development Models Characteristics [11]

Characteristics	Waterfall	Spiral	Agile
Activities Workflow	Progressive movement from on phase to another with back step.	Risk Driven flow.	Simple Iteration.
Simplicity	Simply understanding of the process.	Hard to understand due enormous process.	Easier to understand & dealing with teamwork.
Documentation	*Each phase must be documented.*	*Each phase has one or more specific documentation.*	*Documentation is not important*
Flexibility	Less flexible if requirements change too much.	Flexible with the iteration and progressive process.	Supporting fast development.

Spiral and Agile methodologies are lacked of documentation work. As Spiral, the documentation is required at specific phase and almost no documentation in Agile methodology. The Waterfall Model emphasises on documentation for each of SDLC phases which is more practical and suitable for development of the new enhanced Traffic Offence Management system.

2.5 Discussion on the Model-View Controller (MVC) Design Pattern

The MVC is an architecture for software which is used frequently for web application development [14]. The architecture is designed to ensure efficiency and consistency throughout the development stage. In tradition programming approaches, where UI coding, business logic and application data was written in a single file, may create difficulty in terms of maintainability, testability as well as scalability of the application. MVC provides a loose coupling among the elements as shown in figure 4 below. The MVC is a suitable pattern for web applications as it combines several technologies usually

split into a set of layers. The separation of layers in MVC sends specific views to different types of user-agents [15].

The MVC design pattern provides separation of concern. Kalelkar, M., Churi, P., & Kalelkar, D. (2014) stated the separation of concern is regarded to separating application modules into individual Model, View and Controller, thus resulting each developer to work on their respective modules without worrisome over confliction. Moreover, MVC separates the business logic for the view and any changes applied at the front end interface will not affect the other part of application under development.

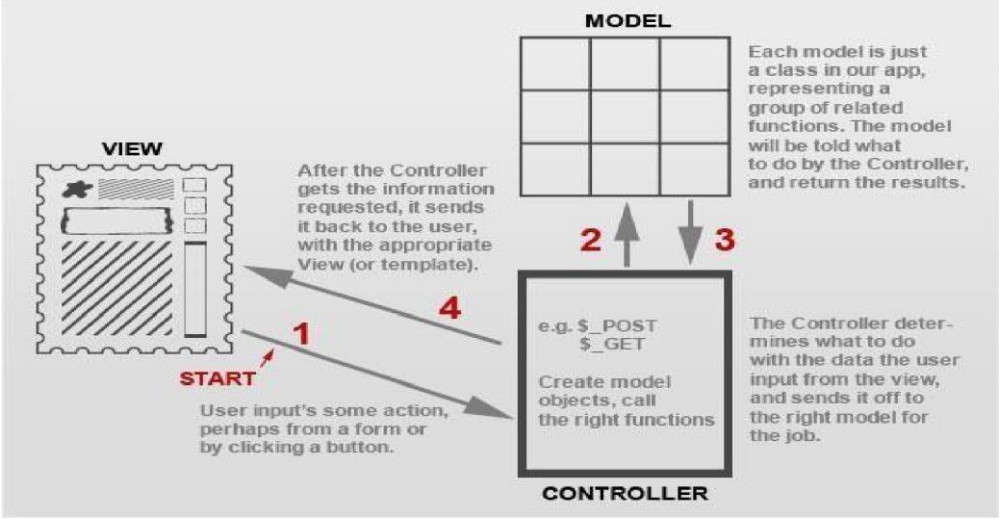

Figure 4: MVC Diagram [14]

The XYZ company decided to implement MVC design pattern due to its characteristic for the developer team to facilitate the segregated development modules and its deployment without putting halt on the online Traffic Offence Management system, and besides combating issues of decentralised and scattered traffic offenders record.

3 Limitation

There are number of limitations in this study. Due to non-disclosure agreement between the XYZ company and the authors, some of the outcomes of the study have no permission to be highlighted due to DBKL proprietary on the software artefacts under development. Therefore the experience is revolved around the selection of software development processes and the implementation or justification of MVC design pattern.

4 Conclusion

Our experiences on SDLC selection and design pattern implementation were presented in this paper, emphasises on the importance of SDLC selection based on the problem domain and client's (DBKL) needs, i.e. proper documentation standard and conform to a IEEE documentation standard.

Despite the fact that the archaic and dated software development models were selected, these models are the underlying fundamental of modern software development models and in this study it proves through existing research study, that the traditional software development models are still relevant to current software engineering practices.

References

[1] Harshad S. Modi, Nikhil Kumar Singh, Harsha Pradeepbhai Chauhan, " Comprehensive Analysis of Software Development Life Cycle Models" in International Research Journal of Engineering and Technology, 2017, vol 4(iss 6) pp. 117-122.

[2] Sommerville, I. (2007). *Software engineering* (pp. I-XXIII). Addison-wesley.

[3] Kay, R. (2002). QuickStudy: system development life cycle. *Computerworld, May, 14*

[4] Ron Burback, " Software Engineering Methodology: The WaterSluice," *Thesis Stanford University*, 1998.

[5] M.Sundararajan & S. Balaji, " Succeeding with Agile Software Development*″ IEEEInternational Conference On Advances In Engineering, Science And Managemen*t, pp. 162- 165, 2012.

[6] Barry W Boehm, " *A spiral model of software development and enhancement*," *Computer*, IEEE, pp. 61-72, 1998. [7] Royce, W. W. (1987, March). "Managing the development of large software systems: concepts and techniques". In Proceedings of the 9th international conference on Software Engineering (pp. 328-338). IEEE Computer Society Press.

[8] Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Kern, J. (2001). *Manifesto for agile software development*.

[9] Nerur, S., Mahapatra, R., & Mangalaraj, G. (2005). Challenges of migrating to agile methodologies. *Communications of the ACM*, *48*(5), 72-78.

[10] Laplante, P. A., & Neill, C. J. (2004). The demise of the waterfall model is imminent and other urban myths. *ACM Queue*, *1*(10), 10-15.

[11] Taya, S. (2011). Comparative Analysis of Software Development Life

[12] Balaji, S., & Murugaiyan, M. S. (2012). Waterfall vs. V-Model vs. Agile: A comparative study on SDLC. *International Journal of Information Technology and Business Management*, *2*(1), 26-30.

[13] Nasution, M. F. F., & Weistroffer, H. R. (2009, January). Documentation in systems development: A significant criterion for project success. In *System Sciences, 2009. HICSS'09. 42nd Hawaii International Conference on* (pp. 1-9). IEEE.
[14] Kayla. K. (2012). A Detailed Overview of the Model-View-Controller (MVC) Coding Structure
[15] Pop, D. P., & Altar, A. (2014). Designing an MVC model for rapid web application development. *Procedia Engineering*, *69*, 1172-1179.
[16] Kalelkar, M., Churi, P., & Kalelkar, D. (2014). Implementation of ModelViewController Architecture Pattern for Business Intelligence Architecture. *International Journal of Computer Applications*, *102*(12).