Architectural Design of Sensor based Environmental Information Systems for Maintainability

Ruthbetha Kateule[1], and Andreas Winter[2]
Carl von Ossietzky University Oldenburg, Oldenburg, Germany
Department of Computer Science
kateule@se.uni-oldenburg.de, winter@se.uni-oldenburg.de

Keywords: Software Architecture, Sensor based Environmental Information Systems, Maintainability, Scenarios, Architecture Design Decisions.

Abstract: The achievement of software quality attributes contributes to the success of any system. Maintainability is one of the software quality attributes that plays a major role in attaining system quality, however, it is a time-consuming and expensive phase of system development life cycle. Sensor based environmental information systems have a long operational lifetime. For these reasons, It is very important for sensor based environmental information systems to possess the maintainability quality attribute in order to remain useful during their lifetime. However, the development process of such systems did not realize explicitly the maintainability requirements to sustain the operation of such systems. Since the fulfilment of quality attributes of the system has been increasing realized as a significant role of software architecture. This work extends the architecture of sensor based environmental information systems for maintainability, using road traffic control system as a case of study. Maintainability is assessed through the use of change scenarios. Architectural design decisions are applied in redesigning the architecture to improve maintainability.

1 INTRODUCTION

Sensor based environmental information systems utilize the sensing techniques to manage the data about the air, water, soil and other objects revolving around the world such as road traffic control, air pollution, and fire detection systems (Kateule and Winter, 2016). The utilization of such systems is increasingly worldwide. This motivates the development process of such systems to be initiated from the existing systems by reusing the developed artifacts: code and architectural designs (Graaf, 2004). However, the development of such systems emphasizes on the performance, reliability, and usability. The software quality attributes pertaining the accommodation of likely changes in the future i.e. growth and technology are never explicitly specified, measured or architected during the life cycle of the system.

Maintainability is an essential software quality attribute for the long-term success of software system. It is found that the maintenance phase consumes a large part of a system costs such as between 50 to 80 percent of the system total costs (Lientz and Swanson, 1980), (Clements and Kazman, 1998). The conventional sensor based environmental information systems are insufficient in terms of accommodating continuous changing of the requirements, caused by changes in the demands of various stakeholders and environment. This could be addressed by first understanding the system since 47 percent of the maintenance efforts are directly related to the system understanding (Clark and Boehm, 1995) i.e. software architecture.

The software architecture of a program or computer system is the structure or structures of the system, which comprise software components, the externally visible properties of those components, and the relationships among them (Clements and Kazman, 1998). Basically, the software architecture is regarded as a blueprint for the development process of a system. The software architecture facilitates the achievement of systems' functional and quality attributes. Software architecture provides an appropriate level of abstraction for evaluating, reasoning and managing the software quality attributes (Kaufmann, 2014). The desired level of software quality could be achieved via comprehensive specification and evaluation of a software quality attributes. System architecture should accommodate quality attributes by providing a foundation for achieving systems quality

attributes. Maintainability is one of software quality attributes that could be assessed in architectural level (Wall and Land, 2008). The architecture determines the efforts required to find and fix errors as well as moving the software to different platform or hardware. Therefore, the architecture evaluation of maintainability is essential to determine the ability of a system in accommodating new requirements or changes to avoid expensive rework in the future.

Since maintainability is crucial for the long-term success of sensor based environmental information systems. The software architecture plays an important role in achieving this mainly by incorporating new and changed requirements of the system in the early stages of system designs through various architectural design solutions. This results in systems with reduced risks, costs, and efforts in sustaining the effective and efficient operations of the system. However many researchers neglect the maintainability aspect while designing software architecture.

In the previous conference paper, we have provided a preliminary discussion on the essential viewpoints for the reference architecture of sensor based environmental information systems (Kateule and Winter, 2016). In this work, we extend the proposed architectural design (mainly conceptual) with maintainability perspective. The contribution of this paper is the proposed architectural design (viewpoint) that facilitate the development of maintainable sensor based environmental information system.

1.1 Road Traffic Control System

Before continuing with the related works section, the road traffic control system is introduced as a case of sensor based environmental information system. This system aims at maximizing the efficiency of road networks by minimizing traffic jams. The conceptual view of road traffic control system as presented in figure 1. The system employs sensors i.e. *InductiveloopSensor*s to count the number of vehicles in road lanes at the junctions, and then the information is sent to the *Server*. The *Server* integrates the information collected by various sensors located in different road lanes. Then the *Server* analyses the collected information and execute suitable control actions through *TrafficSignalActuator*s and *TrafficLightsActuator*s that maximize the traffic flow. The collected information from the sensors as well as control actions executed by the actuators are stored in the *TrafficDB*. A *UserInterface* displays the road traffic information of various junctions.

The remainder of this paper is organized as follows. The related works are given in Section 2. Section 3 presents maintainability aspect on software architecture. The software architecture description for maintainability is presented in Section 4. The paper ends with the conclusion in Section 5.

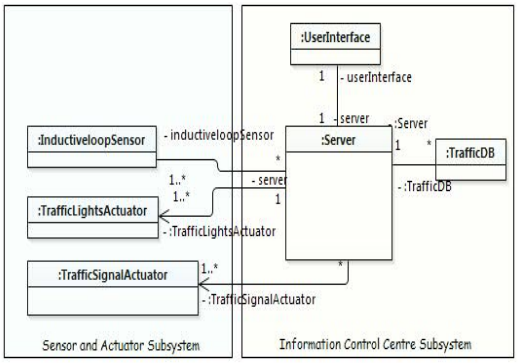

Figure 1: Conceptual View of Road Traffic Control System.

2 RELATED WORKS

One of the most challenging tasks in designing software architecture is not only to design the specified system functionality but also attaining a specific software quality attributes i.e. performance or maintainability that led to the quality of the system (Bosch and Bengtsson, 2002). Therefore the evaluation of software quality is crucial to both software engineers and business expertise perspectives.

In (Lindvall and Dennis, 2009), the Tactical Separation Assisted Flight Environment system (TSAFE) was analyzed and investigated over the accommodation of new features. The maintainability and flexibility issues were discovered and their impacts on the software system were analyzed. Then the observed issues were resolved via architecture design decisions that reduce maintenance effort of such system.

In (IEEE, 2011), different views of the software systems are proposed to address different concerns of various stakeholders of the system. A view conforms to a corresponding viewpoint. Maintainability being one of the concerns of various stakeholders is addressed by various viewpoints for instance module dependency viewpoint.

In (Bengtsson and Bosch, 1998), a method for reengineering software architecture was demonstrated. The method addressed the quality attributes of software architecture. This was achieved through the use of scenarios and then the design transformations were applied to improve the quality attributes of the required system. The method was illustrated via a beer can inspection system.

From the literature, it is found that scenario based technique is mostly used in assessing the maintainability from the architecture point of view. Also, most of the architectural designs that have been proposed belong to various system domains rather than sensor based environmental information systems. Hence this work addresses the issue of maintainability on the architectural level of sensor based environmental information systems.

3 MAINTAINABILITY ON SOFTWARE ARCHITECTURE

Any system intends to provide desired services based on the predetermined quality attributes. The sensor based environmental information systems is expected to possess longer operational lifetime. The stakeholders consider maintainability as the most important quality attributes.

Maintainability is the ability of the system to accommodate new or change requirements with a degree of ease (Christensen, 2003). In general, *maintainability is the capability of the software product to be modified* (ISO, 2001). This includes the addition or manipulation of functionalities, fixation of errors and fulfilment of new raised requirements to meet the demands of the business.

Maintainability is one of the software quality attributes which is highly affected by the architectural design of a software system. This is because the functionality of the system is decomposed into several components. Hence the introduction of change or new requirements led to the modification of the specified architecture by imposing the changes in several components to accommodate the introduced requirements. This situation could be handled effectively through the analysis and understandability of the software system i.e., if the designed software architecture is extended with the design decisions for maintainability then the system would be able to accommodate change and new requirements against minimal efforts.

4 SOFTWARE ARCHITECTURE DESCRIPTION FOR MAINTAINABILITY

The description of software architecture for sensor based environmental information system that supports maintainability adopts IEEE 42010 (Recommended Practice for software Architectural Description of Software Intensive Systems) (IEEE, 2011). The developers, architects, and sensor experts are identified as the main stakeholders with maintainability concerns. Since maintainability is one of the software quality attributes that could be expressed naturally through change scenarios, then a scenario-based approach is utilized for the identification of the desirable set of viewpoints. A scenario-based method is a technique of evaluating a software quality attribute of an architecture mainly maintainability by expressing the software quality attribute in terms of scenarios (R. Kazman and Webb, 1994). Therefore for our case, change scenarios in road traffic control system as a case of study are utilized as described in the following section.

4.1 Scenarios

A scenario represents an action or sequence of actions that might occur as related to the system. It describes a certain maintenance task. Some of the potential change scenarios of road traffic control system that could arise as new requirements in the future are as described below;

- Hardware change: The sensor, servers, user interfaces and actuators hardware in the system might be changed or added to increase the accuracy or performance of the system hence the corresponding software need to be updated. For instance, the system employed inductive loop detectors and there was a need of adding CCTV camera sensor to increase accuracy. Also, initially the system utilized traditional traffic lights as actuators, the emergence of smart traffic lights enforce the system to replace the traditional traffic lights with new smart traffic lights to increase the performance of the system.
- The change of road traffic algorithms, database or sensor measuring types. Normally the road traffic algorithms serve the traffic demands in the lanes, hence when the traffic demands change mainly due to weather, road maintenance activities, accidents, and others, then there is a need of changing the algorithms to suit the demands. Change of one database schema to another i.e. from SQL to Oracle requires the road traffic control system to be updated. The sensor measuring types could be changed to meet the traffic demands and increasing the accuracy of the system, for instance, the extension of measuring only vehicle counts to include other measuring types such as length, and speed of vehicles.
- Extension of the system with some external systems via data exchange. Rise of new requirements

or some events such as travellers need to be updated on the status of road traffic to plan their trips and also occurrence of accidents and some other criminal acts on the roads require the road traffic control system to be extended or provide some information to other systems such as traveller information, emergence and security control systems.

These scenarios are representative scenarios for the maintenance of sensor based environmental information systems.

4.2 Effects on the Architecture

The aforementioned change scenarios are evaluated based on their impacts on the software architecture of road traffic control systems described in section 1.1. The impacts of those change scenarios are as follows:

- The introduction of new or change of hardware in the road traffic control system requires the change of all concerned components since the components are directly connected to each other.
- The road traffic algorithm change affects the *Server* component, database change affects the *TrafficDB* component, user interface change affects the *UserInterface* and *Server* components and also the change of sensor measuring types affect the *Sensor* and *TrafficDB* components.
- To accommodate the extension with new systems the *Server* component need to be reconfigured.

The realization of these scenarios in the architecture of road traffic control system presented in section 1.1 revealed that the accommodation of those scenarios impose the reconfiguration of the whole architecture since many components are affected. This implies that the demonstrated architecture of road traffic control system posses low maintainability.The main objective of software architectural design is to optimize the potential of the designed architecture in order to fulfil the software quality requirements (Bengtsson and Bosch, 1999).Therefore there is a need of redesigning the software architecture of road traffic control system taking into consideration the maintainability as the crucial requirement.

4.3 Proposed Architectural View for Maintainability

The effects of those scenarios on the described road traffic control system revealed that the system possesses low maintainability. The main objective of system architecture redesign is to utilize the architectural design decisions to accommodate those requirements.

An architectural design decision concerns with the application of the architectural styles and patterns in the system to satisfy the system requirements (Jansen and Bosch, 2005). The following are critical architectural design decisions that have been applied to extend the architecture design of road traffic control system with maintainability as illustrated in figure 2;

- **Addition of New Classes:** This handles the clear distribution of activities or functionality of particular component. For instance, a new class *Controller* implements the *Server* is responsible with coordinating the activities of all other components in the system, hence reduce inter-components coupling. The expansion of road traffic control system with other systems or functionality could be executed by *Controller*. *Analyser* responsible with analysing the collected traffic information, executing specified algorithm that optimise traffic flow. *Parser* handle various database schemas by facilitating the insertion and updation of database.

- **Interfaces:** Each component has been designed with an *Interface* class that facilitates the inter-components communication. This implies that the hardware and software changes (communication protocols, user-interfaces etc.) could be easily accommodated through interfaces.

- **Design Patterns:** Client-Server Architectural Style is employed to decouple the GUI from the program logic by separating the program logic *Server* Interface from the display *Client*.

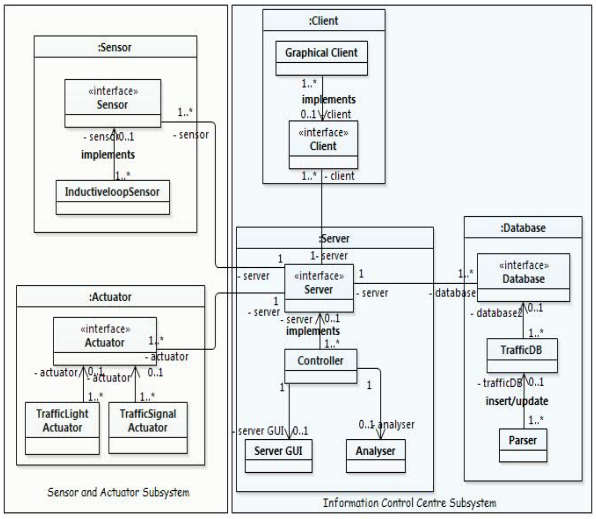

Figure 2: Redesigned View of Road Traffic Control System.

4.4 Generalised Viewpoint

The generic architectural design (viewpoint) of sensor based environmental information systems for maintainability is derived from the representation of the road traffic control system architecture view presented in the previous section. The viewpoint of sensor based environmental information system as shown in figure 3, consists of the following components *Sensor*, *Actuator*, *Server*, *Database* and *Client*. For Maintainability, the architectural design employs new classes for specific functionalities, interfaces for facilitating the communication between the aforementioned components and client-server architectural style for separating the processing of information from the actual displaying functionality. Therefore the development of any maintainable sensor based environmental information system could be facilitated by deduction of concrete system architecture view from this proposed viewpoint.

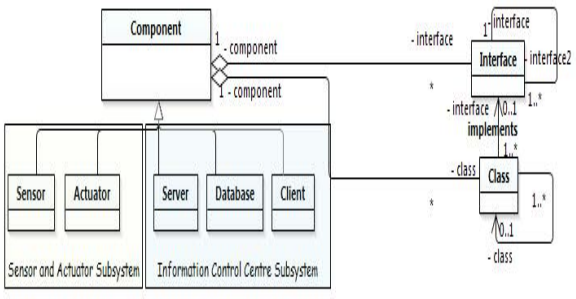

Figure 3: Conceptual Viewpoint of Sensor based Environmental Information Systems

5 CONCLUSION

The architecture of sensor based environmental information systems has been identified as an important aspect in the development of sensor based environmental information system. A well-defined software architecture led into the fulfilment of software quality attributes mainly maintainability. The incorporation of maintainability demands in an early phase of system design is crucial for achieving a well-functional and reasonable cost system since the resulted system will be able to accommodate the changes or new requirements throughout its lifetime.

This paper presents the architectural design of sensor based environmental information systems for maintainability. The approach utilizes a scenario-based technique and employs architectural design decisions to achieve the maintainability software quality attributes. The selected scenarios are aligned with stakeholders' concerns on the aspect of maintainability. An architectural view of improving maintainability of road traffic control system is proposed. And finally, the viewpoint for maintainable sensor based environmental information systems is derived.

REFERENCES

Bengtsson, P. and Bosch, J. (1998). Scenario-based software architecture reengineering. *Proceedings of 5th International Conference on Software Reuse*.

Bengtsson, P. and Bosch, J. (1999). Architecture level prediction of software maintenance. *Proceedings of 3rd European Conference on Software Maintenance and Re-engineering*, pages 139–147.

Bosch, J. and Bengtsson, P. O. (2002). Assessing optimal software architecture maintainability. *Proceedings of Fifth European Conference on Software Maintenance and Reengineering*, pages 168–175.

Christensen, H. B. (2003). Using software architectures for designing distributed embedded systems. *Technical Report, University of Aarhus, Denmark*.

Clark, C. G. A. A.-A. B. and Boehm, B. (1995). On the definition of software system architecture. *Technical report: USC/CSE-95-TR-500*.

Clements, L. B. P. and Kazman, R. (1998). *Software Architecture in Practice*. MA: Addison Wesley Longman.

Graaf, B. (2004). Maintainability through architecture development. *European Workshop on SA*.

IEEE (2011). Recommended practice for architectural description of software-intensive systems 42010.

ISO (2001). International organization for standardization , iso 9126-1:2001.

Jansen, A. and Bosch, J. (2005). Software architecture as a set of architectural design decisions. *Proceedings of 5th Working IEEE or IFIP Conference*.

Kateule, R. and Winter, A. (2016). Viewpoints for sensor based environmental information systems. *EnviroInfo: 30th edition, Berlin-Germany*.

Kaufmann, M. (2014). Relating system quality and software architecture. *Elsevier Inc.*, pages 41–73.

Lientz, B. and Swanson, E. (1980). *Software maintenance management*. Mass.: Addison-Wesley.

Lindvall, C. A. . M. and Dennis, G. (2009). Redesign for flexibility and mantainability: A case study. *Proceedings of European Conference on Software Maintenance and Re-engineering*.

R. Kazman, L. Bass, G. A. and Webb, M. (1994). Saam: A method for analyzing the properties software architectures. *Proceedings of the 16th International Conference on Software Engineering, Sorrento, Italy*.

Wall, M. L. C. N. A. and Land, R. (2008). Importance of software architecture during release planning. *Proceedings of the Seventh Working IEEE/IFIP Conference on Software Architecture*, pages 253–256.