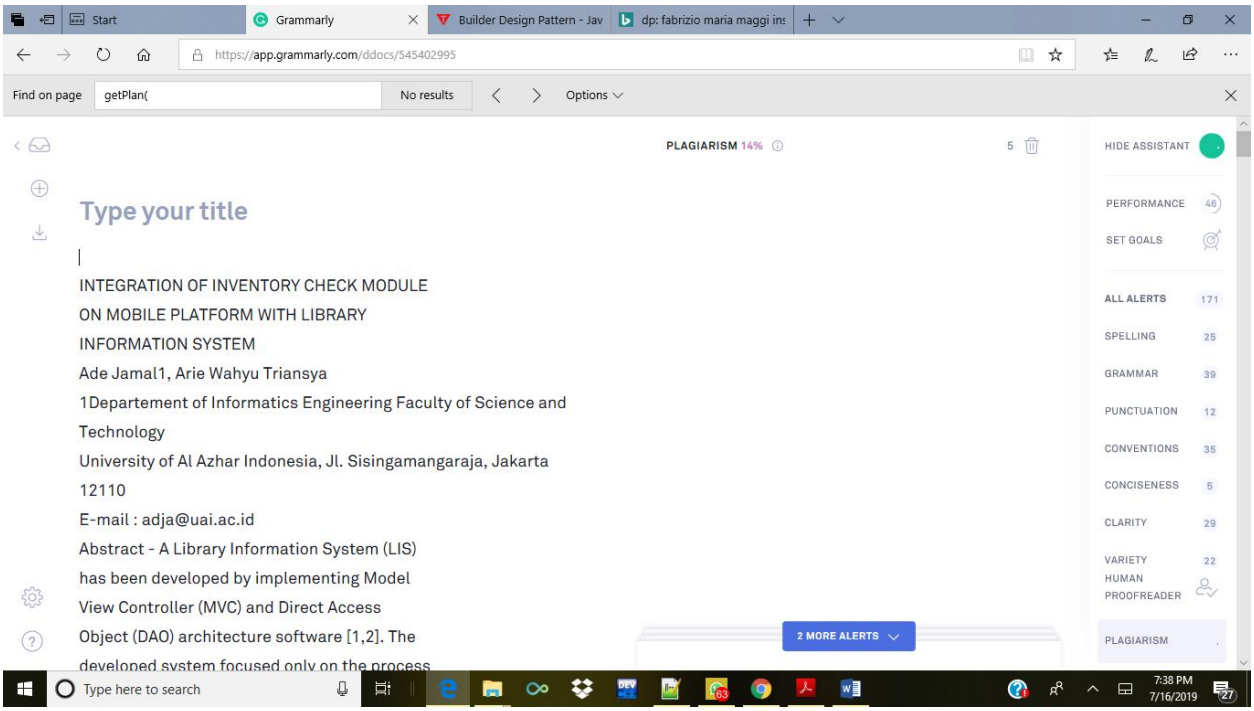

INTEGRATION OF INVENTORY CHECK MODULE ON MOBILE PLATFORM WITH LIBRARY INFORMATION SYSTEM

Ade Jamal[1], Arie Wahyu Triansya
[1]Departement of Informatics Engineering Faculty of Science and Technology
University of Al Azhar Indonesia, Jl. Sisingamangaraja, Jakarta 12110
E-mail : adja@uai.ac.id

Abstract - **A Library Information System (LIS) has been developed by implementing Model View Controller (MVC) and Direct Access Object (DAO) architecture software [1,2]. The developed system focused only on the process of circulation and implemented on desktop platform. This article will present an extension of this LIS by implementing an inventory check process of library collection on mobile platform technology. Since both applications are parts of the Library Information System, the two applications need to be integrated in order to communicate with each other. For this**

purpose Servlet and Remote Method Invocation (RMI) technology will be utilized for the integration process because both applications are possessed of different platform, namely desktop based and mobile based platform. However, during the implementation phase, several programming problem arose such as poor code structure. Hence some refactoring work has to be done on the already developed desktop based LIS application.
Keywords - *Integration System, Librarian Information System, Software Refactoring*

I. INTRODUCTION

n the previous publications [1,2], the development of a library information system has been presented by making use of MVC (Model View Controller) as software architecture model and DAO (Direct Access Object) as interface to its database layer. The system is built on desktopbased platform. This work is the first stage of LIS development project, hence this focus only on the library core business process namely the circulation module [2].

Inventory check is one of the activities in the libraries that calculate physically the collections and books inventories that are stored in book shelves and then matched with recorded inventory of books in the library system [3]. An inventory check module was not yet implemented in the previously developed LIS system. The LIS system needs to be further extended to implement the module for inventory check.

Inventory check activity in the library of Al Azhar Junior High School is rarely done, because the currently used library system does not support this process. This may cause big problems that will be occurred when the time comes to do the inventory check and highly probably, the library collection lost or damages will be thousands in number. Since the inception of the library in 1982 until now, only 2 times inventory check has been carried out namely in 1997 and 2012.

Considering that within inventory check work, librarians should do a checking process on the state of all collections in different places or bookshelves in the library, the mobility of the library staff is then very high during this inventory check. An application system-based on mobile device is therefore very appropriate to be applied here that enables librarians to perform their duties in

Figure 1. Business Process of manual inventory check

comfort manner because of mobile applications can be taken anywhere.

II. DEVELOPMENT AND INTEGRATION OF INVENTORY CHECK MODULE

2.1 Business Process Inventory Check

The Al Azhar Junior High School librarians still perform inventory checking manually as depicted in Fig. 1. The purpose of the inventory check process to maintain the stability of the system of data collection on the physical state of the book so that they stay in synchronise with data in the system. Inventory check activity is actually performing data matching between the existing collections in the book shelves and those persist in the library database system.

There are several state of collection should be recorded during the inventory check according to the physical condition of the collection, namely:

a. Existence
b. Borrowed
c. Broken
d. Old
e. Disappear

To perform this process manually, first the librarian should print out the collection data from the system. Then they check the collection physically in the book shelves and make notes on their states. Finally they have to put back these current states of the collection into the system. This manual process is shown in Fig. 1.

The inventory checking process could be done completely in computerized based system. The requirement solution of this issue is the need for computer device which not hinders librarians from moving around the
library from one location to another. A hand-held device such as a mobile smartphone may fulfil this need.

By using a mobile application that is integrated with the LIS System, the required data from the LIS system will be fully visible from the mobile application. The mobile application can immediately make changes to the states which will be sent and stored to a database through the Internet as depicted in Fig. 2.

2.2 System Integration

Mobile application to be developed based on Android platform [4, 5] will be integrated with the previously developed library information system (LIS). Hence it will use the same database from the problem domain model of LIS. There are some data that are required for the inventory check process as follow:

a. Classification code
b. Code Collection
c. Book Title
d. Author
e. Inventory State (Exist, Borrowed, Broken, Old, Lost)

These data will be read from LIS database by the mobile application for retrieving collection information via a webserver. The common servlet technology [6] used for the communication between the Android based mobile application and the webserver.

By letting a webserver as intermediate layer between the mobile device and the existing desktop based LIS, the LIS system is not tightly coupled with the new inventory check module in the mobile device. Another advantage of the proposed intermediate webserver layer is a readiness of the LIS for further extension in the multi-tier environment.

Since both systems to be coupled, namely the servlet based web-application and the desktop

based LIS are implemented using the same Java programming language, then Remote Method Invocation (RMI) technology [7] will be used for the integration. The remotely invoked method will make communications with database via the existing DAO. When the mobile application needs data from the LIS system, DAO will read these data from database.

To interchange these data between these three layers, i.e. mobile device, web-server and desktop-based application, a so called JSON (JavaScript Object Notation) format is used. JSON is a lightweight data-interchange text based format that is easy for human to read and write as well as for machine to parse and generate [8]. It is based on a subset of JavaScript but since it is text based format it is completely language independent. Hence JSON is an alternative datainterchange format in additional to the more

popular XML format.

2.3 Refactoring

The choice of RMI first seems to be a good alternative. However, when it came into the detail of implementation in invoking methods remotely, some poor code structures are found that obstruct applying RMI seamlessly. The solution to this problem is to perform refactoring of the exiting code. Indications of so-called "Code Smells" in the earlier system such as duplicated code, long methods and large classes are found in the LIS system.

The main goal of refactoring is to improve the overall design and structure of existing programs without changing the functionality of the system. This means when you are not steeled refactoring functions at the same time. But after refactoring has been done, it is easier to add new functionality to the system. [9]

Kent Beck and
Martin Fowler
have also developed a list of what they call "code smells" to help to determine when to refactor. These are things to look for in existing code that indicates that refactoring is in order. Here is a brief list of some of the smells they have identified [9]:

- Duplicate Code - duplicate code means you need to extract some methods.
- Long Method - too long is hard to understand, extract methods.
- Large Class - a class that does too much needs to be split.
- Long Parameter List - makes it hard to read, consider passing objects.
- Divergent Change - code degradation as a result of too many chaotic changes to a class.
- Shotgun Surgery - too many undisciplined changes to classes and attributes.
- Feature Envy - one class is interested in too many details of another class.
- Data Clumps - data that is used together everywhere should be in a class of its own.
- Primitive Obsession - a program can use too many primitive data types that should really be part of a class.
- Switch Statements - switch statements can mean you are not using polymorphism effectively.
- Parallel Inheritance Hierarchies - repeating class definitions in parallel classes is more duplication to eliminate.

- Lazy Class - a class should do enough to pay its own way or be eliminated.
- Speculative Generality - designing for future flexibility before it is needed can increase complexity unnecessarily.
- Message Chain - too many messages in a chain are hard to follow.

Figure 2. Designing Process Mobile Based Inventory Check

- Middle Man - sometimes it is better to work with an object directly.
- Inappropriate Intimacy - classes shouldn't need to know too much about each other.
- Incomplete Library Class - sometimes you can't get it all and need to do some of yourself.
- Data Class - classes need something to do.
- Refused Bequest - subclasses should use most of what their parents give them.
- Comments - could a comment be eliminated by providing a better name for a method or variable?

Noted that, in the present work, not all type of "code smell" is indicated in the developed LIS system. The first three of code smell are very obviously indicated, namely duplicate code, long method and large class [10].

Duplicate code and long method are indicated when the LIS system making connections with the database. This is refactored by introducing a new class whose responsibility to perform all connection tasks.

Large class is indicated in DAO classes where there were three responsibilities performed by DAO class, namely, Problem Domain Model (PDM) and Table Model for user interface as inner classes and the DAO itself. These classes are split as depicted in Fig. 3.

2.4 System Architecture

Fig. 4 shows the system architecture where the web-server put in the middle of mobile device and the desktop based LIS system. In this figure is also shown that internet is the network media between these three sub-systems, but it can be replaced by just a local area network if the whole system is used in the small single library.

How this 4-tiers system architecture is physically deployed can be different. Fig. 5 shows the recommended configuration where three

computer servers are utilized.

III. CONCLUDING REMARKS AND RECOMMENDATION

A newly developed mobile based application for inventory check has been integrated to the previously developed library Information System which is a desktop based application. The 3-tier architecture system is used for the integration where a webserver as a middle layer between the mobile device and the desktop application is deployed in order to make a loose integration rather than direct and tight link to the existing database.

Although the already developed library information system was implemented using MVC and DAO technique to provide an easy extendible system, the practical problem rose up when this application to be integrated with the web application using RMI technology because of a number of "code smells" were indicated in the computer program. The presented work has shown that refactoring works can solve the problem.

For future development such as searching and cataloguing system module, the middle layer web application can be further utilized to communicate with the library information system.

IV. BIBLOIGRAPHY

[1] Syahid, M. 2012. System Informasi Perpustakaan dengan Mengimplementasi-kan Metode MVC dan DAO. Jakarta : Universitas Al Azhar Indonesia.

[2] Jamal, A., Syahid, M, 2012 Perancangan Model-View-Controller pada Aplikasi Perpustakaan Sekolah, proceeding SINAPTIKA 2012, Jakarta

[3] Sutarno, NS. 2003. Perpustakaan dan Masyarakat. Jakarta : Yayasan Obor Jakarta. Hal 7.

Figure 3. Splitting of Large Class

[4] Stelee, James., Nelson. 2011. The Android Developer's Cookbook Building Application with Android SDK. Boston : Pearson Education, Inc.

[5] Safaat, H., Nazarudin. 2012. Pemrograman Aplikasi Mobile Smartphone dan Tablet PC Berbasis Android. Bandung : Penerbit Informatika.

[6] Wijono, S. H., Suharto, H. B., Wijono, M, S. 2006. Pemrograman Java Servlet dan JSP dengan NetBeans. Yogyakarta : Penerbit Andi.

[7] Clara, Santa. 2004. Java Remote Method

Invocation Specification. California: Sun Microsystems, Inc.
[8] http://www.json.org/ accessed 21 June 2012.
[9] Fowler, M. 2002. Refactoring: Improving the Design of Existing Code. Addison-Wesley.
[10] Triansyah, A. W. 2012, Integrasi Desktop Application dengan Mobile Application untuk Implementasi Modul Stock Opname pada Sistem Informasi Perpustakaan, Jakarta : Universitas Al Azhar Indonesia.
Figure 4. n-tier System Architecture

Figure 5. Deployment Diagram