SISTEM INFORMASI MANAJEMEN LABORATORIUM MENGGUNAKAN METODE AGILE DENGAN KONSEP MODEL-VIEW-CONTROLLER DATA ACCESS OBJECT

Reza Firsandaya Malik[1], Muhammad Fachrurrozi[2], Rahmanto Prabowo[2]
[1] Jurusan Sistem Komputer, Fakultas Ilmu Komputer, Universitas Sriwijaya
[2] Jurusan Teknik Informatika, Fakultas Ilmu Komputer, Universitas Sriwijaya

Jl. Raya Palembang-Prabumulih Km. 32 Indralaya, Ogan Ilir 30662, Sumatera Selatan
e-mail: rezafm@unsri.ac.id

Abstract- Management Information System in the organization is important to improve its performance. It is a reason for laboratory in Faculty of Computer Science, Universitas Sriwijaya, to develop the management information system laboratory. By implementing AGILE method with Model-View-Controller Data Access Object concept for developing management information system laboratory. Comparison with SDLC method is done. The results show AGILE method quicker than SDLC method in management information system laboratory development.

Keywords: Management Information System, AGILE Method, Model-View-Controller Data Access Object Concept

***Abstrak* – Sistem Informasi Manajemen dalam sebuah organisasi merupakan hal yang penting dalam menunjang kinerjanya. Hal tersebut yang menjadi landasan laboratorium Fakultas Ilmu Komputer Universitas Sriwijaya untuk membuat sistem informasi manajeman laboratorium. Dengan menerapkan metode AGILE dengan konsep Model-View-Controller Data Access Object dalam pembuatan sistem informasi laboratorium. Perbandingan dilakukan dengan metode SDLC. Hasil yang diperoleh adalah kecepatan dalam menyusun dan membuat sistem informasi manajemen laboratorium lebih baik dibandingkan dengan metode SDLC.**

Kata kunci : Sistem Informasi Manajemen, Metode AGILE, Konsep Model-View-Controller Data Access Object

I. PENDAHULUAN

1.1 LATAR BELAKANG

Pengelolaan laboratorium di sebuah perguruan tinggi menjadi penting untuk menghasilkan pembelajaran yang baik, terutama di bidang praktikum ilmu sains [1]. Di dalam struktur organisasi Fakultas Ilmu Komputer (Fasilkom) Universitas Sriwijaya (Unsri), terdapat unit laboratorium yang dikelola oleh seorang ketua unit dan membawahi beberapa kepala laboratorium. Pelaksanaan administrasi rutin dibantu oleh dua orang pegawai administrasi yang berada di kampus Indralaya dan Palembang. Terpisahnya kampus Unsri ini menjadi salah satu kesulitan dalam pengelolaan laboratorium. Tata kelola dan akses informasi menjadi kurang efisien dan kurang cepat. Pelaksanaan praktikum yang menjadi salah satu aktivitas rutin kurang terkontrol. Hal ini pula disebabkan kurangnya sumber daya manusia yaitu pegawai dan dosen. Tata kelola informasi dan pengawasan aktifitas laboratorium yang baik dan benar diharapkan mampu menjawab permasalahan-permsalahan tersebut.

Pemanfaatan teknologi dan informasi dapat menjadi salah satu solusi dalam membantu kebijakan dan aturan yang telah ditetapkan fakultas, terutama tentang pengelolaan laboratorium. Saat ini Fasilkom Unsri telah memiliki jaringan intranet dan internet yang baik. Kampus Indralaya dan Palembang sudah saling terhubung. Semua aktivitas yang berkaitan dengan teknologi dan informasi ditugaskan ke ketua unit *Information Technology and Communication* (ICT). Namun saat ini Fasikom belum memiliki sistem terkomputerisasi untuk pengelolaan laboratorium. Pengelolaan laboratorium meliputi pengelolaan praktikum, pengelolaan peralatan laboratorium dan pengawasan. Dengan demikian pengembangan sistem manajemen laboratorium diharapkan mampu membantu dalam menjalankan kebijakan dan tata kelola laboratorium yang efisien, cepat dan baik.

Metode pengembangan perangkat lunak Agile memiliki karakteristik cepat untuk skala proyek pekerjaan medium. Namun demikian, model ini memerlukan sumber daya manusia yang lebih banyak dengan kemampuan yang sama baiknya [2]. Metode ini berfokus pada keinginan pengguna yang relatif dinamis, mengalami perubahan permintaan yang cepat. Hal ini membuat metode ini diklaim sangat fleksibel terhadap perubahan yang sering muncul [3]. Di sisi lain, metode ini harus memiliki karakteristik serupa dengan metode tradisional. Setiap aktivitas harus terencana dan mudah ditelusuri. Jaminan kualitas hasil harus terukur. Semua masalah yang muncul pada saat proses pengembangan harus dapat diidentifikasi secara dini [4].

Di dalam paradigma pemrograman berorientasi objek (PBO), aplikasi dibangun dari objek-objek. Objek merefleksikan keadaan benda di dunia nyata, yang berisikan data dan perilaku [5]. Konsep *Model-View-Controller* (MVC) telah banyak dikenalkan oleh peneliti dan pengembang perangkat lunak. Glenn E. Krasner dan Stephen T. Pope membangun antarmuka untuk sistem Smalltalk-80. Konsep MVC ini mengusung pemisahan tiga kajian paradigma sesuai dengan peran dan tugas setiap objek dalam perangkat lunak [6]. Di dalam perkembangannya, konsep ini dibawa ke platform lain, yaitu salah satunya platform web. Pengembangan perangkat lunak berbasis web saat ini telah menyediakan layanan yang bervariasi. Paradigma PBO pun juga telah diperkenalkan di bahasa pemrograman berbasis web,

antara lain PHP Hypertext-Preprocessor [7]. Konsep MVC dikembangkan menjadi sebuah *framework* untuk pengelolaan basis data. MVC *Framework* ini belum memisahkan aktifitas antara aplikasi dan basis data. Para pengembang perangkat lunak tradisional terbiasa menggabungkan objek model dan basis data. Pada penelitian ini dikembangkan konsep MVC dengan memisahkan aktivitas aplikasi dan *Data Access Object* (DAO) untuk pengembangan sistem manajemen laboratorium di Fasilkom Unsri.

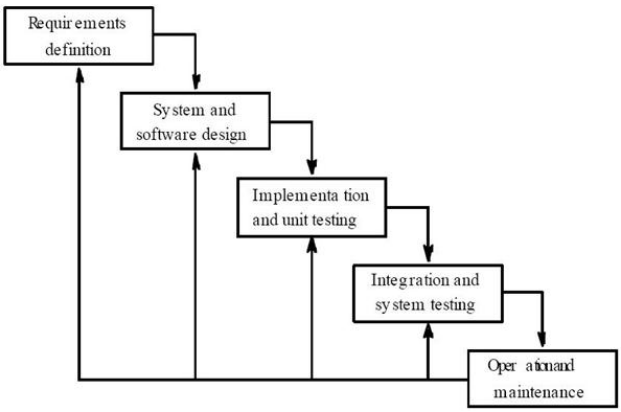
II. TINJAUAN PUSTAKA

2.1. Siklus Hidup Pengembangan Sistem

Siklus hidup pengembangan sistem (System development life cycle / SDLC) adalah tahapan aktivitas yang harus dikerjakan oleh pengembang sistem untuk menghasilkan sebuah sistem yang dapat dioperasikan pada organisasi pemakai sistem [8]. Metode SDLC adalah tahap-tahap pengembangan sistem informasi yang pertama kali dikembangkan yang dilakukan oleh analisis sistem dan programmer untuk membangun sebuah sistem informasi. Metode SDLC ini seringkali dinamakan sebagai proses pemecahan masalah, yang langkah langkahnya adalah sebagai berikut:

2.1.1. Analisis Kebutuhan.

Seluruh kebutuhan perangkat lunak harus bisa didapatkan dalam fase ini, termasuk di dalamnya kegunaan perangkat lunak yang diharapkan pengguna dan batasan perangkat lunak. Informasi ini biasanya dapat diperoleh melalui wawancara, survei atau diskusi. Informasi tersebut dianalisis untuk mendapatkan dokumentasi kebutuhan pengguna untuk digunakan pada tahap selanjutnya. Seperti terlihat pada Gambar 2.1.



Gambar 2.1 Software Developmet Life Cycle [8]

2.1.2. Desain Sistem

Tahap ini dilakukan sebelum melakukan pengkodean. Tahap ini bertujuan untuk memberikan gambaran yang seharusnya dikerjakan dan bagaimana tampilannya. Tahap ini membantu dalam memberikan spesifikasi kebutuhan perangkat keras dan sistem, serta mendefinisikan arsitektur sistem secara keseluruhan.

2.1.3. Implementasi

Tahap ini dilakukan pemrograman. Pembuatan perangkat lunak dipecah menjadi modul-modul kecil yang nantinya digabungkan dalam tahap berikutnya. Selain itu dalam tahap ini juga dilakukan pemeriksaaan terhadap modul yang dibuat, apakah sudah memenuhi fungsi yang diinginkan atau belum.

2.1.4. Integrasi dan Pengujian.

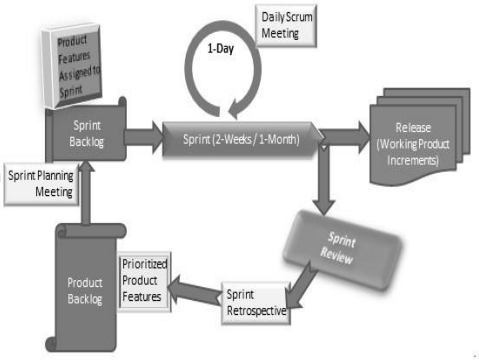
Di tahap integrasi ini dilakukan penggabungan modul-modul yang sudah dibuat dan yang telah dilakukan. Tahapan pengujian dilakukan untuk mengetahui apakah perangkat lunak yang dibuat telah sesuai dengan desainnya dan masih terdapat kesalahan atau tidak.

2.1.5. Operasi dan Perawatan.

Perangkat lunak yang sudah jadi dijalankan serta dilakukan pemeliharaan. Pemeliharaan termasuk dalam memperbaiki kesalahan yang tidak ditemukan pada langkah sebelumnya. Perbaikan implementasi unitsistem dan peningkatan jasa sistem sebagai kebutuhan baru.

2.2. Metode Agile

Metode Agile adalah salah satu metodologi dalam tahap pengembangan sebuah perangkat lunak yang efektif. Kata Agile berarti bersifat cepat, ringan, bebas bergerak, waspada. Agile juga merupakan model yang lebih efektif dari pada model tradisional yang tidak cukup bagus dan tidak efektif. Namun, metode agile bukan suatu proses yang bersifat menentukan, dengan kata lain tidak mendefinisikan prosedur secara detail untuk bagaimana membuat tipe model yang telah diberikan, meskipun terdapat cara bagaimana untuk menjadi suatu modeler yang efektif.



Gambar 2.2 Metode Agile SCRUM [9]

Berikut ini adalah beberapa model yang termasuk dalam metode agile seperti ditunjukkan pada Gambar 2.2:

1. *Extreme Programming* (XP)
2. *Adaptive Software Development* (ASD)
3. Metode *Scrum*, ilustrasi seperti pada Gambar 2.
4. Dynamic System Development Method (DSDM)
5. Metode *Crystal*
6. *Feature Driven Development* (FDD)

7. *Lean Software Development* (LSD).

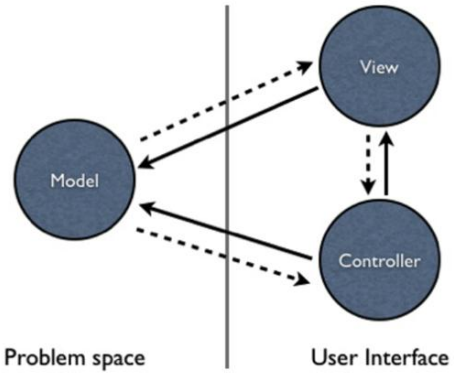
10 prinsip untuk mencapai proses yang termasuk dalam agility yaitu sebagai berikut [10];

1. Prioritas tertinggi adalah memuaskan pelanggan melalui penyerahan awal dan perangkat lunak yang bernilai.
2. Menerima perubahan kebutuhan meskipun perubahan tersebut diminta pada akhir pengembangan.
3. Memberikan perangkat lunak yang sedang dikerjakan dengan sering, beberapa minggu atau bulan, dengan pilihan waktu yang paling singkat.
4. Pihak bisnis dan pengembangan harus bekerja sama setiap hari selama pengembangan berjalan.
5. Bangun proyek dengan individu-individu yang bermotivasi tinggi dengan memberikan lingkungan dan dukungan yang diperlukan, dan mempercayai mereka sepenuhnya untuk menyelesaikan pekerjaannya.
6. Metode yang paling efektif dan efisien dalam menyampaikan informasi kepada tim pengembangan adalah dengan komunikasi langsung *face to face*.
7. Perangkat lunak yang dikerjakan merupakan pengukuran utama kemajuan.
8. Proses agile memberikan proses pengembangan yang dapat ditopang. Sponsor, pengembangan, dan pengguna harus bisa menjaga kekonsistenan langkah yang tidak pasti.
9. Perhatian yang harus menerus terhadap rancangan dan teknik yang baik meningkatkan *agility*.
10. Kesederhanaan seni untuk meminimalkan jumlah pekerjaan adalah penting.

2.3 *Model-View-Controller (MVC)*

Model-View-Controller (MVC) adalah sebuah konsep yang diperkenalkan oleh penemu *Smalltalk* [6] untuk meng-enkapsulasi data bersama dengan pemrosesan (model), mengisolasi dari proses manipulasi (*controller*) dan tampilan (*view*) untuk direpresentasikan pada sebuah user interface. MVC mengikuti pendekatan yang paling umum dari Layering. Layering hanyalah sebuah logika yang membagi kode kita ke dalam fungsi di kelas yang berbeda. Pendekatan ini mudah dikenal dan yang paling banyak diterima. Keuntungan utama dalam pendekatan ini adalah penggunaan ulang (reusability) kode. Definisi teknis dari arsitektur MVC dibagi menjadi tiga lapisan [11], yaitu:

Model, digunakan untuk mengelola informasi dan memberitahu pengamat ketika ada perubahan informasi. Hanya model yang mengandung data dan fungsi yang berhubungan dengan pemrosesan data. Sebuah model meringkas lebih dari sekedar data dan fungsi yang beroperasi di dalamnya. Pendekatan model yang digunakan untuk komputer model atau abstraksi dari beberapa proses dunia nyata. Hal ini tidak hanya menangkap keadaan proses atau sistem, tetapi bagaimana sistem bekerja. Sebagai contoh, programmer dapat menentukan model yang menjembatani komputasi *back-end* dengan *front-end* GUI (*Graphical User Interface*).



Gambar 2.3 MVC Layer [11]

Pada Gambar 2.3 menunjukkan bahwa *View*, bertanggung jawab untuk pemetaan grafis ke sebuah perangkat. *View* biasanya memiliki hubungan 1-1 dengan sebuah permukaan layar dan tahu bagaimana untuk membuatnya. *View* melekat pada model dan me-*render* isinya ke permukaan layar. Selain itu, ketika model berubah, *view* secara otomatis menggambar ulang bagian layar yang terkena perubahan untuk menunjukkan perubahan tersebut. Terdapat kemungkinan beberapa *view* pada model yang sama dan masing-masing *view* tersebut dapat me-*render* isi model untuk permukaan tampilan yang berbeda.

Controller, menerima masukan dari pengguna dan menginstruksikan model dan *view* untuk melakukan aksi berdasarkan masukan tersebut. Sehingga, *controller* bertanggung jawab untuk pemetaan aksi pengguna akhir terhadap respon aplikasi. Sebagai contoh, ketika pengguna mengklik tombol atau memilih item menu, *controller* bertanggung jawab untuk menentukan bagaimana aplikasi seharusnya merespon.

III. METODE PENELITIAN

3.1. *Metode Penelitian*

Secara keseluruhan, tahapan metode penelitian ini adalah sebagai berikut :

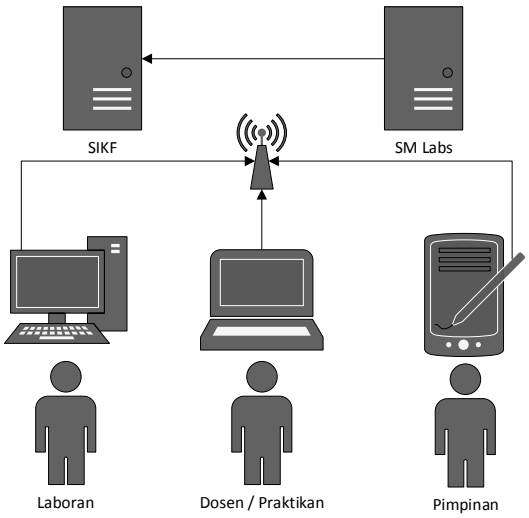
1. Mengumpulkan data kebutuhan fungsional dari pengguna.
2. Menganalisis kebutuhan fungsional yang dituangkan ke dalam dokumen kebutuhan.
3. Merancang basis data relasional dengan mengacu kepada kebutuhan pengguna.
4. Merancang antarmuka per modul berdasarkan konsep MVC DAO.
5. Mengimplementasikan hasil rancangan per modul ke dalam kode program.
6. Melakukan pengujian langsung ke pengguna dengan metode blackbox.

7. Melakukan perbaikan per modul sampai dengan selesai.
8. Mengimplementasikan perangkat lunak ke Sistem Informasi Kinerja Fakultas (SIKF).

IV. IMPLEMENTASI

4.1. *Arsitektur Umum Sistem*

Adapun rancangan arsitektur umum sistem informasi manajemen laboratorium seperti pada Gambar 4.1.



Gambar 4.1. Arsitektur Umum SIM Laboratorium

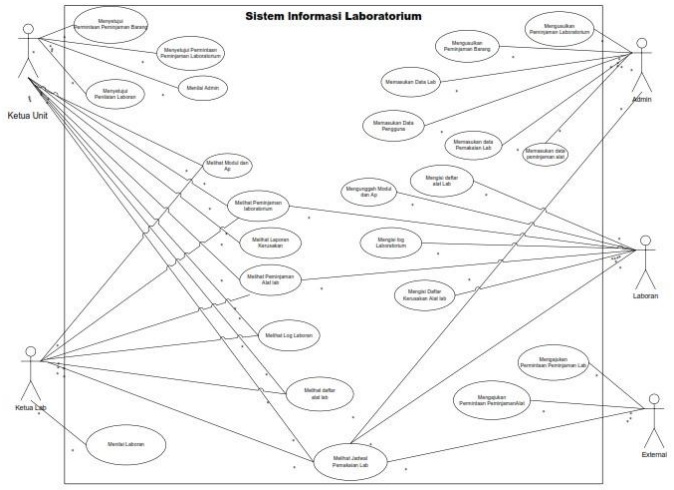
Kebutuhan Fungsional :

1. Sistem dapat mengelola proses manajemen laboratorium
2. Sistem dapat mengelola proses peminjaman ruangan laboratorium
3. Sistem dapat mengelola proses peminjaman barang
4. Sistem dapat menampilkan daftar peminjaman barang
5. Sistem dapat menampilkan jadwal kegiatan laboratorium
6. Sistem dapat mengelola laporan log kegiatan laboran dan menampilkannya
7. Sistem dapat mengelola proses penilaian laboran

Kebutuhan Non-Fungsional :

1. Sistem tidak dapat menerima file unggah langsung ke direktori sistem, file harus diunggah ke *cloud drive* lain dan memberikan tautannya ke sistem
2. Sistem hanya bisa di buka melalui jaringan intranet Fasilkom Unsri
3. Sistem dapat compatible dengan API *Single Sign On* Fasilkom Unsri.

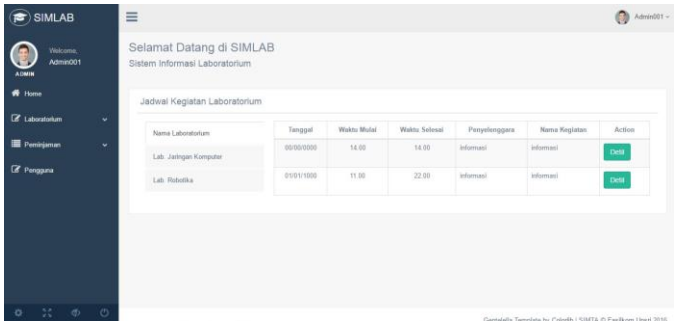
Sehingga paparan use case pada Gambar 4.2 pada sistem informasi manajemen laboratorium ini adalah:



Gambar 4.2. Use Case Sistem Informasi Manajemen Laboratorium

Dengan mengacu pada *use case* ini metode AGILE dengan konsep MVC DAO diterapkan. Pada pekerjaan yang biasa dilakukan dengan metode tradisional yaitu SDLC maka metode AGILE dengan konsep MVC DAO bisa menghasilkan kecepatan hasil dalam membangun sistem informasi manajemen laboratorium Fasilkom Unsri.

Pembuktian awal metode ini baru dilihat dari sisi kecepatan dalam pengerjaan sistem informasi manajeman laboratorium. Sehingga implementasinya telah dapat menghasilkan draft hubungan antar muka pengguna seperti yang ditunjukkan pada Gambar 4.3.



Gambar 4.3 Draft Hubungan Antar Muka Pengguna

V. KESIMPULAN DAN SARAN

5.1. Kesimpulan

Dari makalah penelitian ini dapat disimpulkan sebagai berikut:

a. Sistem informasi manajemen laboratorium berhasil diterapkan menggunakan metode AGILE dengan konsep MVC DAO
b. Metode AGILE dengan konsep MVC DAO bisa menghasilkan kecepatan hasil dalam membangun sistem informasi manajemen laboratorium dibandingkan metode SDLC.

REFERENSI

[1] Indrajit, R. E., dan Djokopranoto, R. (2006). Manajemen Perguruan Tinggi Modern.

[2] Papadopoulos, G. (2015). Moving from Traditional to Agile Software Development Methodologies Also on Large, Distributed Projects. Procedia - Social and Behavioral Sciences, 175, 455–463.

[3] Brhel, M., Meth, H., Maedche, A., dan Werder, K. (2015). Exploring principles of user-centered agile software development: A literature review. Information and Software Technology, 61, 163–181.

[4] Kupiainen, E., Mäntylä, M. V., dan Itkonen, J. (2015). Using metrics in Agile and Lean software development - A systematic literature review of industrial studies. Information and Software Technology, 62(1), 143–163.

[5] Pop, D. P., dan Altar, A. (2014). Designing an MVC model for rapid web application development. Procedia Engineering, 69, 1172–1179.

[6] Krasner, G. E., dan Pope, S. T. (1988). A Cookbook for Using the Model-View-Controller User Interface Paradigm in Smalltalk-80. Joop Journal Of Object

[7] Chanchai Supaartagorn. (2011). Php Framework for Database Management Based on Mvc Pattern. International Journal of Computer Science dan Information

[8] Sommerville, I (2011). Software Engineering Edisi ke-9. Addison-Wesley, Amerika Serikat.

[9] Tutorials Point. (2015). SCRUM. Agile framework for completing complex projects, 36. Diakses dari http://www.tutorialspoint.com/scrum/scrum_tutorial.pdf

[10] Fowler, M., dan Highsmith, J. (2001). The agile manifesto. Software Development, 9 Agustus, 28–35.

[11] Bragge, M. (2013). Model-View-Controller architectural pattern and its evolution in graphical user interface frameworks, 34. Diakses dari http://www.doria.fi/bitstream/handle/10024/92156/Model-View-Controller architectural pattern and its evolution in graphical user interface frameworks.pdf?sequence=2