ANALISIS DAN RANCANGAN PROTOTIPE MANAJEMEN DOKUMENTASI REKAYASA PERANGKAT LUNAK

Hanson Prihantoro Putro
Jurusan Teknik Informatika
Universitas Islam Indonesia
hanson@uii.ac.id

ABSTRACT

In our department, lectures found that students are difficult to create documentation, especially software documentation. Besides, software documentation is an important handbook in a software engineering. We need tools to help students or people that learn software engineering to create software documentation. This paper explains how to analyze and design a prototype for software documentation management. We propose three steps to do: problem analysis, prototype development and evaluation. First, we do the problem analysis by defining the component inputs, documentation management models and printing process. Then, the prototype is developed with object oriented software analysis and design. Finally, we create a traceability table and conduct a design testing in the evaluation step.At the result, we build analysis and design for a prototype to manage software engineering documentation with six use cases. The evaluation is conducted which well modeling the functional requirements. 26 errors are found and already refined in the documentation report. We ensure that this design is ready to implement.

Keyword: object oriented software engineering, software process documentation, document management.

1. Pendahuluan

Sebuah perangkat lunak (*software*) akan terdiri dari tiga komponen utama yaitu program, dokumentasi dan data (IEEE Computer Society, 2004). Sebagai penyelenggara pendidikan di bidang teknologi informasi, Jurusan Teknik Informatika Universitas Islam Indonesia (UII) juga mengajarkan konsep-konsep perangkat lunak kepada mahasiswanya. Pada kurikulum jurusan Teknik Informatika UII, pembuatan program dipelajari dalam berbagai mata kuliah pemrograman. Kemudian pengelolaan data dipelajari dalam beberapa mata kuliah basis data. Sedangkan dokumentasi perangkat lunak hanya dipelajari dalam satu mata kuliah Rekayasa Perangkat Lunak atau disingkat RPL (Teknik Informatika UII, 2011).

Dengan hanya menyertakan satu mata kuliah untuk mempelajari segudang ilmu rekayasa perangkat lunak, dokumentasi yang dihasilkan para mahasiswa Teknik Informatika UII menjadi kurang optimal. Dokumentasi ini cenderung beragam antar satu mahasiswa dengan mahasiswa lain. Padahal telah diberikan pedoman pembuatan dokumentasi RPL. Oleh karena itu, diperlukan sebuah perangkat (*tools*) yang dapat membantu para mahasiswa serta para *software engineer* untuk membuat dokumentasi perangkat lunak yang sistematis.

Pada penelitian sebelumnya, telah dikembangkan sebuah perangkat *generator* dokumen perangkat lunak secara umum (Roth, 2009), dokumen semantik perangkat lunak (Arantes dan Falbo, 2010) serta dokumen arsitektural perangkat lunak (Riva dan Yang, 2002). Namun

metode yang diberikan pada perangkat tersebut terlalu kompleks untuk para mahasiswa yang baru belajar RPL. Belum ada perangkat yang menyediakan sebuah pembuatan dan pengelolaan dokumen RPL secara sederhana dan mudah digunakan oleh para mahasiswa.Harapannya dengan adanya perangkat sederhana ini, mahasiswa dapat terbiasa mendokumentasikan proses rekayasa perangkat lunak dengan baik dan benar.

Hal yang akan disajikan dalam makalah ini adalah analisis dan rancangan sebuah prototipe manajemen dokumentasi RPL sebagai sarana kolaborasi antara dosen dan mahasiswa. Perangkat lunak yang dikembangkan harus dapat digunakan mahasiswa untuk mencetak dokumen dari proses RPL yang mereka lakukan. Perangkat lunak ini juga harus mampu mendaftarkan mahasiswa sebagai *member* yang mengelola dokumen RPL mereka masing-masing. Sedangkan bagi dosen, perangkat lunak ini harus mampu membuat skema dan proyek dokumentasi RPL dan mengubahnya sesuai kebutuhan.

2. Pekerjaan Terkait

Dokumen rekayasa perangkat lunak adalah sebuah dokumen tertulis yang penting bagi para *stakeholder* terkait untuk berkolaborasi dalam proses pengembangan dan rekayasa perangkat lunak (Arantes dan Falbo, 2010). Arantes dan Falbo, mengembangkan sebuah infrastruktur sebagai sarana untuk manajemen dokumen semantik, khususnya yang diolah dari dokumen RPL.Di sisi lain, dokumentasi perangkat lunak sangat diperlukan dalam pengembangan perangkat lunak. Sama halnya dengan proses pelaksanaan RPL, struktur dokumentasinya juga harus bersifat bisa digunakan sebagai standard acuan. Selain itu, sebuah dokumentasi juga harus mencatat konfigurasi perangkat keras dan perangkat lunak yang dikembangkan (Rochimah, 1999).

Untuk membuat sebuah dokumen RPL diperlukan sistematika yang menjelaskan bahwa akan terdiri dari bagian apa saja dokumen RPL dibuat. Sistematika disusun dalam sebuah skema yang secara umum disediakan dalam sebuah pola (*template*). Pola ini sendiri memiliki banyak kegunaan, misalnya digunakan untuk *generator* kode program (Sarkar dan Cleaveland, 2001) dan juga *generator* aplikasi web (Parr, 2004).

Karena pentingnya dokumentasi rekayasa perangkat lunak, perusahaan-perusahaan *entreprise* skala besar juga ikut andil dalam memperhatikan permasalahan ini. GmbH, sebuah perusahaan yang bergerak di Eropa tengah telah menciptakan sebuah *generator* dokumentasi kode program (EasyCode GmbH, 2011). Selain itu, Oracle juga ikut serta mengambangkan *generator* laporan bagan dan proses bisnis (Oracle, 2011).

Dalam membuat sebuah dokumentasi RPL, Riva dan Yang (2002) telah membuat perangkat *generator* untuk dokumentasi arsitektural RPL menggunakan model skema XML. Melakukan *generate* dokumentasi RPL adalah proses membuat dokumentasi sistem pada berbagai tingkat abstraksi untuk penggunaan dan pengembangan sistem selanjutnya. Biasanya dokumen tersebut di-*generate* dari kode programnya sehingga perangkat ini tidak bisa digunakan pada tahap analisis dan perancangan.

Paten telah diberikan kepada sebuah sistem dan metode untuk *generator*dalam berbagai dokumen perangkat lunak (Roth, 2009). Sistem ini memiliki masukan berupa beberapa file yang menyimpan informasi dari perangkat lunak yang direkayasa. Lalu sistem ini menganalisis setiap masukan tersebut untuk diambil informasi yang diperlukan. *Generator* akan mengeluarkan satu atau banyak dokumen dengan format sesuai skema dokumen yang dipilih. Sistem ini cocok untuk pengembangan perangkat lunak yang besar.

3. Metodologi

Selain studi literatur yang telah disebutkan sebelumnya, penelitian ini dilakukan dalam tiga langkah. Langkah pertama adalah analisis permasalahan yang dilakukan untuk menganalisis rumusan masalah yang diangkat dan mengantarkannya kepada proses pengembangan perangkat lunak. Kemudian langkah kedua adalah proses pengembangan prototipe perangkat lunak. Dalam melakukan pengembangan perangkat lunak, metodologi yang digunakan adalah metodologi pengembangan rekayasa perangkat lunak dengan *Object Oriented Software Engineering*. Terakhir, langkah ketiga adalah evaluasi penelitian. Hal ini untuk memastikan apakah analisis telah dirancang dengan baik dan benar atau tidak.

4. Analisis Permasalahan

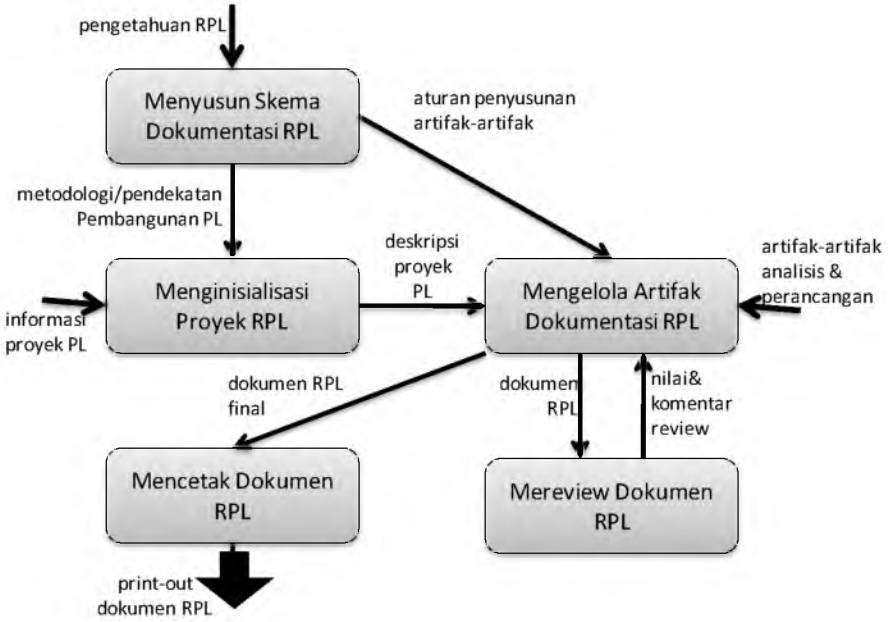
Dalam analisis permasalahan, dilakukan penentuan input komponen, penentuan model dokumentasi dan menentukan proses cetak dokumen.Yang diperlukan sebagai input komponen dalam proses rekayasa perangkat lunak (RPL) adalah model-model (jenis-jenis diagram) yang diperlukan dalam dokumentasi rekayasa perangkat lunak ditambah dengan spesifikasi dan deskripsi tambahan mengenai perangkat lunak tersebut serta artifak-artifak yang diisikan sebagai model dokumentasi terkait proyek yang dibangun. Secara umum, model-model ini nantinya akan menjadi komponen yang akan dimasukkan ke dalam Prototipe Manajemen RPL yang akan dikembangkan dalam penelitian ini. Input-input ini dapat berupa teks satu baris (beberapa kata), teks dalam beberapa baris (tabel), teks dalam bentuk deskriptif (paragraf) atau diagram (deskripsi dan gambar). Gambar 1 memperlihatkan input-input komponen yang dipilih dalam dua pendekatan yang sering dilakukan dalam pengembangan perangkat lunak.

Pendekatan Terstruktur	Pendekatan Berorientasi Obyek
Nama perangkat lunak	Nama perangkat lunak
Deskripsi perangkat lunak	Deskripsi perangkat lunak
Deskripsi lingkungan sistem	Deskripsi lingkungan sistem
Fungsionalitas perangkat lunak	Kebutuhan fungsionalitas
Kebutuhan input	Kebutuhan non fungsionalitas
Kebutuhan proses	Daftar aktor
Kebutuhan output	Daftar *use case*
Diagram konteks	Diagram *use case*
Diagram alir data (DFD)	Skenario tiap *use case*
Spesifikasi Tiap Proses (PSpec)	Diagram aktivitas
Diagram relasi entitas (ERD)	Diagram sekuens
Kamus Data (DD)	Diagram kelas

Gambar 1. Input komponen dalam dua pendekatan pengembangan perangkat lunak

Kemudian dalam penentuan model proses manajemen dokumentasi, disusun sebuah model yang dapat memfasilitasi dosen dan mahasiswa untuk berkolaborasi mengelola dokumentasi RPL. Pertama-tama, dosen perlu membuat skema dokumentasi RPL yang sesuai dengan kebutuhan di lingkungan akademisnya. Kemudian dosen membuat suatu deskripsi

proyek RPL yang akan diserahkan kepada mahasiswa untuk didokumentasi. Selanjutnya mahasiswa diminta untuk membuat dokumentasi proses RPL terkait proyek tersebut sesuai skema yang ditentukan. Setelah dokumentasi selesai dilengkapi, dosen akan me-*review* dokumen RPL tersebut untuk diberikan komentar dan masukan perbaikan. Hasil review ini kemudian digunakan oleh mahasiswa untuk mengelola dan memperbaiki artifak-artifak dokumentasi yang telah dibuat. Gambar 2 menunjukkan model proses manajemen dokumentasi RPL seperti yang diharapkan ini.



Gambar 2. Model Manajemen Dokumentasi RPL

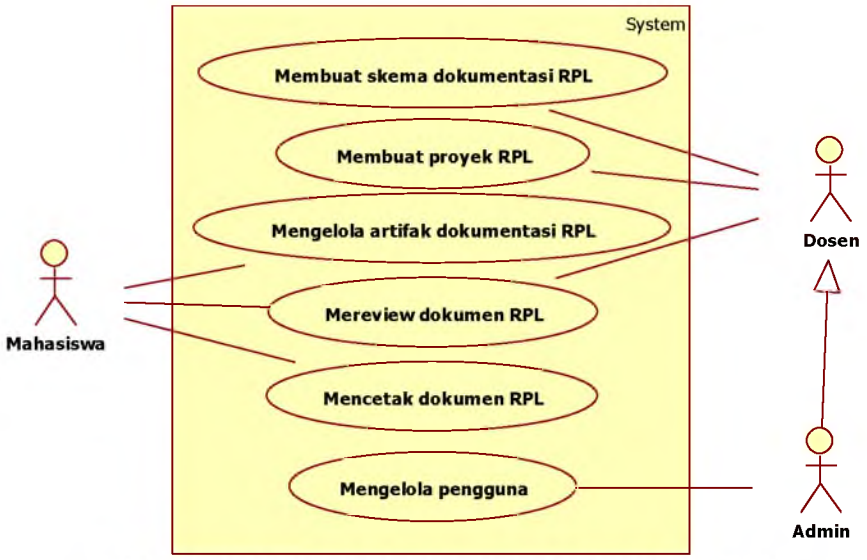
Selanjutnya setelah dokumentasi selesai direview dan diperbaiki maka dokumen RPL yang telah dibuat tersebut dapat dicetak. Dokumen RPL adalah keluaran dari proses dokumentasi rekayasa perangkat lunak. Ini menjadi tahap ketiga dalam langkah analisis permasalahan dan juga tahap terakhir dalam model pengelolaan dokumentasi yang dibuat dalam penelitian ini. Dokumentasi diperlukan dalam bentuk *print-out* untuk keperluan para *developer* ataupun *client* yang ingin membaca proses pengembangan secara lepas tanpa harus berada di depan komputer (*offline*).

5. Pembangunan Prototipe

Pembangunan perangkat lunak yang dijelaskan pada makalah ini merupakan proses rekayasa perangkat lunak (RPL) terhadap *tools* Prototipe Manajemen RPL yang dikembangkan pada penelitian ini. Dalam pelaksanaanya, metodologi pembangunan perangkat lunak yang digunakan adalah metodologi *waterfall* sedangkan pendekatan yang digunakan adalah pendekatan berorientasi obyek. Dengan demikian, dokumentasi yang dihasilkan (Putro, 2014), diharapkan dapat menyerupai hasil *generate* dari *tools* Prototipe Manajemen RPL setelah digunakan, khususnya tahap analisis dan perancangannya.

Dokumentasi perangkat lunak dimulai dari penentuan nama prototipe perangkat lunak beserta deskripsinya. Nama prototipe perangkat lunak yang diputuskan adalah **Prasasti Kutai**. **Prasasti Kutai,** atau cukup disebut **Prasasti** adalah sebuah *tools* yang akan memudahkan para *software engineer* untuk melakukan proses rekayasa perangkat lunak.

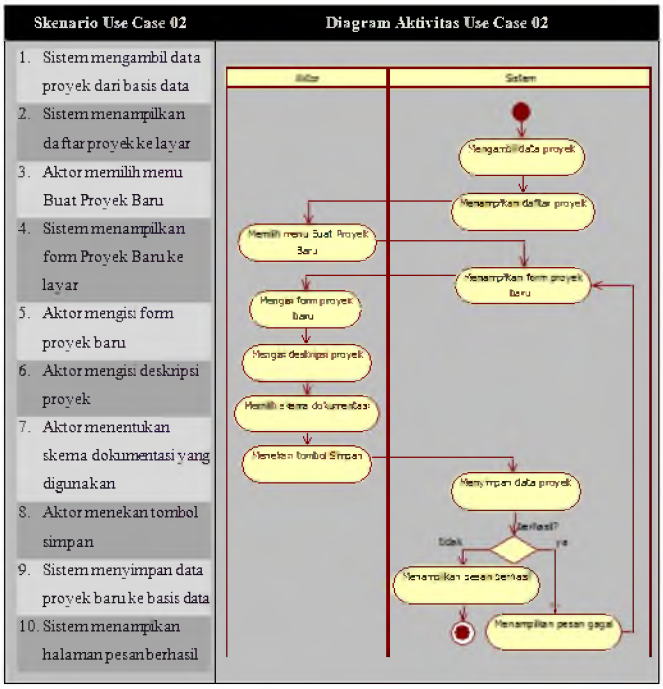
Perangkat lunak **Prasasti** dibangun dalam domain pendidikan. Lingkungan sistem dari perangkat lunak **Prasasti** adalam lingkungan sistem pembelajaran mata kuliah program studi Teknik Informatika, khususnya di Universitas Islam Indonesia. Selanjutnya dari analisis permasalahan yang telah disampaikan sebelumnya, dilakukanlah tahap analisis untuk perangkat lunak **Prasasti** ini.



Gambar 3. Diagram use case perangkat lunak Prasasti

a. Analisis Kebutuhan

Tahap analisis perangkat lunak yang dilakukan pada penelitian ini terbagi atas beberapa bagian. Bagian tersebut yaitu analisis tekait: kebutuhan fungsionalitas, kebutuhan non-fungsionalitas, idenfikasi aktor, identifikasi *use case*, diagram *use case*, skenario tiap *use case*, diagram aktivitas dan diagram kelas tahap analisis.



Gambar 4. Pemetaan skenario ke diagram aktivitas untuk *Use Case* 02

Kebutuhan fungsionalitas pada **Prasasti** didefinisikan sebanyak enam kebutuhan yang kesemuanya kemudian dimodelkan dalam enam *use case* yang bersesuaian. Sedangkan kebutuhan non-fungsionalitasnya didefinisikan sebanyak empat kebutuhan yang masing-masing menjadi pendukung fungsionalitas perangkat lunak tersebut. Selanjutnya dalam identifikasi aktor, sesuai analisis permasalahan yang telah dilakukan, ditentukan tiga aktor dalam **Prasasti**. Ketiga aktor tersebut kemudian dapat melakukan enam *use case* yang telah diidentifikasi (Gambar 3).

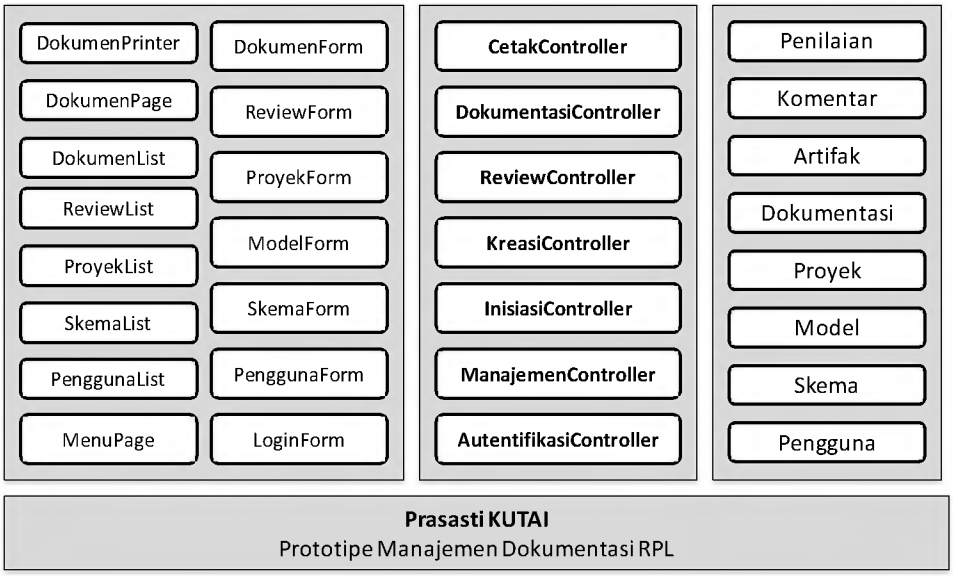
Selanjutnya dari keenam *use case* yang ada, masing-masing dianalisis untuk kemudian ditentukan skenario untuk setiap *use case*. Dari skenario tersebutlah dimodelkan diagram aktivitas untuk memperlihatkan interaksi antara aktor dan sistem dalam sebuah urutan dari awal hingga akhir. Kesemua itu adalah dalam dokumentasi lengkap dari laporan penelitian yang telah dilakukan (Putro, 2014). Salah satu contoh pemetaanya dari skenario *use case* ke diagram aktivitas diperlihatkan pada Gambar 4.

Bagian terakhir dalam tahap analisis ini adalah analisis entitas yang menghasilkan sebuah diagram kelas tahap analisis. Pada bagian ini, kelas-kelas diidentifikasi dari obyek-obyek yang ditemukan di tiap langkah dalam setiap *use case*.Dari identifikasi tersebut, ditemukan delapan kelas yang saling berelasi agregasi satu sama lain. Kedelapan kelas tersebut adalah: Skema, Model, Proyek, Dokumentasi, Artifak, Pengguna, Komentardan Penilaian.

b. Model Perancangan

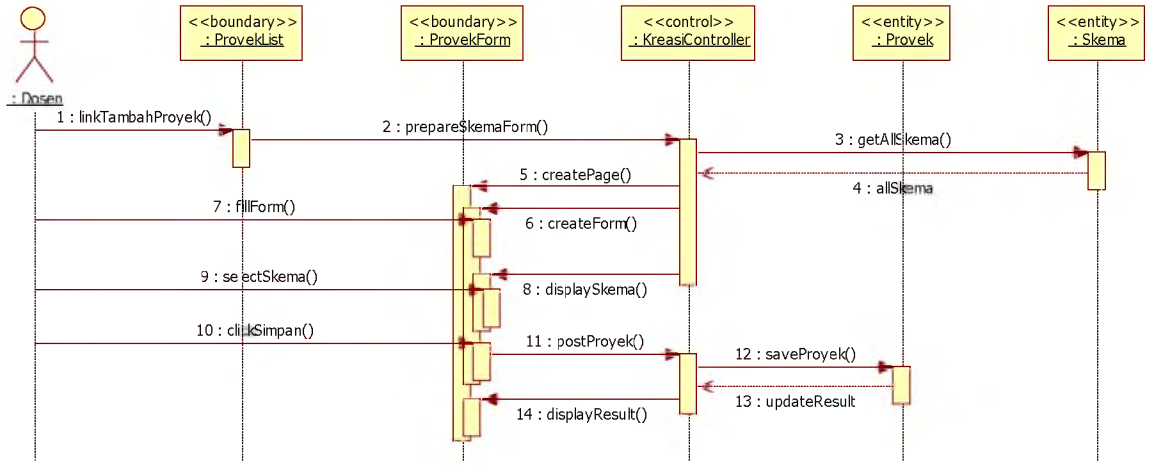
Pembangunan perangkat lunak **Prasasti** dilanjutkan dengan pembuatan model perancangan. Model perancangan dalam penelitian ini terdiri dari empat bagian yaitu: rancangan arsitektur, rancangan perilaku, rancangan basis data dan rancangan antarmuka.

Pembangunan **Prasasti** dirancang dengan menggunakan arsitektur MVC (*Model-View-Controller*). Oleh karena itu, dari analisis yang sudah dibuat, perlu diidentifikasi kelas-kelas *Model, View* dan juga *Controller*.Dari *class diagram* pada tahap analisis, kedelapan kelas tersebut menjadi bagian dari kelas *Model* pada arsitektur MVC. Sedangkan, kelas *View* dan *Controller* diidentifikasi kembali dari setiap *use case* yang ada. Dengan demikian, diperoleh setiap *use case* memiliki kelas *Model, View* dan juga *Controller*, dengan jumlah yang berbeda-beda bergantung dari kompleksitas setiap *use case*. Semua kels MVC dikolaborasikan ke dalam sebuah sistem Prasasti sehingga diperoleh diagram arsitektur seperti diperlihatkan Gambar 5.



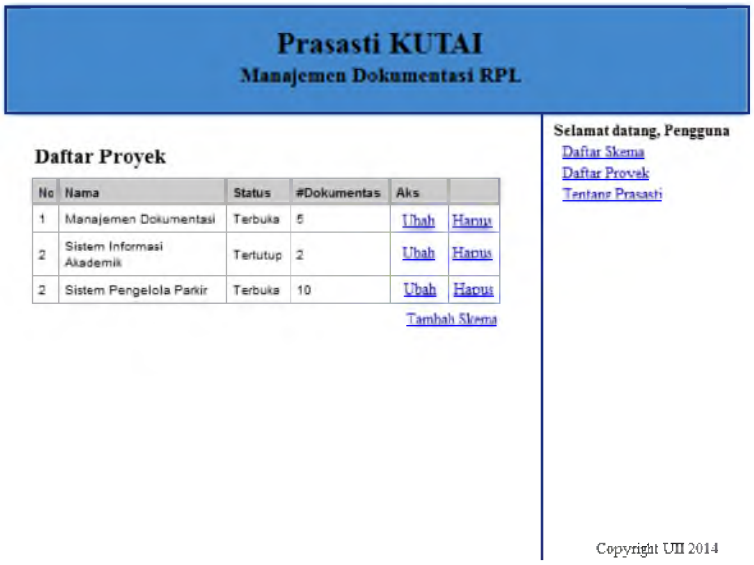
Gambar 5. Diagram arsitektur perangkat lunak Prasasti

Kelas MVC di tiap *use case* tersebut menjadi modal untuk membuat diagram sekuens. Diagram sekuens yang dimodelkan pada tahap perancangan ini menggambarkan interaksi antara aktor dan kelas-kelas MVC tersebut. Langkah-langkah interaksi ini dirancangan dari hasil analisis dalam diagram aktivitas, di mana aktivitas-aktivitas sistem dalam diagram aktivitas didetailkan lagi ke dalam message pasing kelas-kelas dalam sistem. Interaksi ini dalam pemrograman berorientasi obyek akan menjadi *method* dari kelas yang diinvokasi. Gambar 6 menunjukkan diagram sekuens yang dirancang dari hasil analisis diagram aktivitas pada *use case* yang sama (Gambar 4).



Gambar 6. Diagram sekuens untuk *Use Case* 02

Pada langkah-langkah akhir perancangan, dilakukan perancangan basis data dan perancangan antarmuka sesuai kelas *Model* dan kelas *View* yang diperoleh sebelumnya. Dirancang sembilan tabel basis data dari delapan kelas *Model* yang diidentifikasi sebelumnya. Satu kelas tambahan diperoleh dari relasi antara kelas Skema dan kelas Model yang perlu disimpan dalam satu tabel tersendiri karena memiliki relas n ke n. Kemudian dirancang pula 17 antarmuka dari 17 kelas *View* yang diidentifikasi sebelumnya. Contoh rancangan antarmuka dari perangkat lunak Prasasti ini diperlihatkan pada Gambar 7.



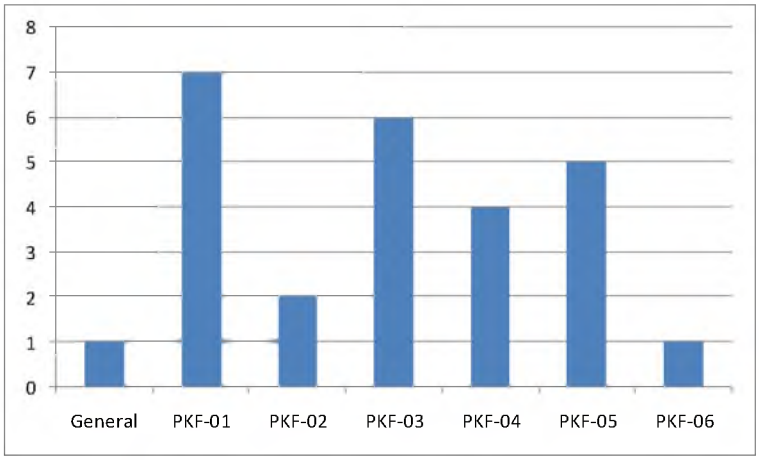
Gambar 7. Rancangan antarmuka halaman Proyek List, bagian dari Use Case 02

6. Evaluasi

Evaluasi prototipe perangkat lunak Manajemen Dokumentasi RPL diawali dengan terlebih dahulu melakukan penelusuran (*tracing*) terhadap artifak-artifak dokumentasi yang telah dibuat untuk sistem **Prasasti**. Penelusuran dimulai dari tahap analisis kebutuhan dalam kebutuhan perangkat lunak, *use case* dan diagram aktivitas hingga tahap perancangan model dalam diagram kelas, diagram sekuens, rancangan tabel dan rancangan antarmuka. Dari penelusuran ini, diperoleh bahwa keenam kebutuhan fungsioanalitas memiliki representasi model analisis dan model rancangannya sehingga setiap kebutuhan tersebut siap untuk diimplementasikan.

Selanjutnya dilakukan evaluasi dengan menyerahkan hasil analisis dan perancangan kepada *programmer* yang kemudian akan mengimplementasikan model tersebut. Proses evaluasinya adalah proses pengujian hasil analisis dan perancangan **Prasasti**, apakah bisa diimplementasikan dengan baik dan benar sesuai harapan atau tidak. Dari hasil pengujian tersebut, dinyatakan secara langsung oleh *programmer* bahwa analisis dan perancangan yang telah dibuat ini cukup mudah untuk dipahami.

Namun demikian, masih terdapat26 poin di mana *programmer* masih bertanya dan memerlukan tambahan informasi karena kurang jelasnya analisis dan perancangan tersebut. Kurang jelasnya informasi ini membuat hasil dokumentasi analisis dan perancangan menjadi tidak sempurna sehingga kemudian muncul kesalahan. Jumlah kesalahan di tiap kebutuhan fungsional **Prasasti** (PKF) diperlihatkan pada Gambar 8. Tindak lanjut dari evaluasi ini adalah perbaikan untuk 26 kekurangan dalam analisis dan perancangan pada dokumentasi versi selanjutnya. Setelah diperbaiki, dokumen analisis dan perancangan tersebut telah bisa digunakan untuk implementasi prototipe manajemen dokumentasi RPL ini dengan baik dan benar.



Gambar 8. Jumlah kesalahan dokumentasi analisis dan perancangan tiap kebutuhan

7. Kesimpulan

Dari penelitian yang telah dilakukan, telah berhasil dirancang sebuah prototipe manajemen dokumentasi RPL sebagai sarana kolaborasi antara dosen dan mahasiswa. Dari sisi kesesuaiannya, rancangan tersebut berhasil memodelkan kebutuhan mahasiswa yang terdaftar untuk mencetak dokumen dari proses RPL dan memodelkan kebutuhan dosen untuk membuat skema dokumentasi dan melihat dokumentasi yang dibuat mahasiswanya. Hasil rancangan ini telah siap untuk dikembangkan lebih lanjut karena permodelannya cukup mudah untuk dipahami. Kemudian dalam evaluasinya, telah dilakukan pengujian dokumentasi hasil analisis dan perancangan ini, yang diperoleh sebanyak 26 *review* kesalahan. Namun demikian, evaluasi tersebut telah ditindaklanjuti sehingga hasil rancangan ini siap untuk diimplementasikan dalam sebuah *tools* perangkat lunak manajemen dokumentasi RPL.

8. Pekerjaan ke Depan

Dari prototipe yang sudah dirancang, akan diimplementasikan perangkat lunak manajemen dokumentasi perangkat lunak yang bisa digunakan oleh mahasiswa dan dosen sebagai alat bantu perkuliahan Rekayasa Perangkat Lunak (RPL). Publikasi akan disajikan, melaporkan proses implementasi dan pengujian perangkat lunak manajemen dokumentasi sesuai analisis dan perancangan dari makalah ini.

9. Daftar Pustaka

Arantes, L.O. dan Falbo, R.A. 2010. *An Infrastructure for Managing Semantic Documents*. Prosiding Konferensi: Enterprise Distributed Object Computing Conference Workshops. Institute of Electrical and Electronics Engineers. Australia.

Brinkkemper, Sjaak; Hong, Shuguang; dkk. 1995. *Object Oriented Analysis and Design Methods: A Comparative Review*. Universitas Twente. Enschede. Belanda.

EasyCODE GmbH. 2012. *EasyCode Documment Generator*. EasyCODE GmbH. Jerman. (*http://www.easycode.de/en/products/v85-cc/add-on-document.html*

diakses pada 26 Maret 2012).

Riva, C dan Yang, Y. 2002. *Generation of Architectural Documentation using XML*. Prosiding Konferensi: Working Conference on Reverse Engineering, Institute of Electrical and Electronics Engineers. Amerika Serikat.

IEEE Computer Society. 2004. *Software Engineering Body of Knowledge*. Institute of Electrical and Electronics Engineers. California. Amerika Serikat.

Oracle. 2012. *Oracle Bussiness Intelligence Publisher*. Oracle. Amerika Serikat. (*http://www.oracle.com/technetwork/middleware/bi-publisher/overview/index.html*diakses pada 1 April 2012).

Parr, T. 2004. *Enforcing Strict Model-View Separation in Template Engines*. Prosiding Konferensi: World Wide Web. Association for Computing Machinery. Amerika Serikat.

Putra, Hanson Prihantoro. 2014. *Laporan Penelitian: Prototipe Manajemen Dokumentasi Rekayasa Perangkat Lunak*. Universitas Islam Indonesia. Yogyakarta. Indonesia.

Roth, M.L. 2009. *Software Document Generator*. US Patent No. US 758184 B1. Amerika Serikat.

Sarkar, S dan Cleaveland, C. (2001). *Code Generation using XML Based Document Transformation*. The Server Side – Your J2EE Community.

Rochimah, Siti. 1999. *Pengembangan Dokumentasi Standard untuk Tahapan Rekayasa Perangkat Lunak*. Lembaga Penelitian, Institut Teknologi Bandung. Bandung. Indonesia.

Teknik Informatika UII. 2011. Buku Panduan Akademik: Jurusan Teknik Informatika, Fakultas Teknologi Industri, 2011/2012. Yogyakarta. Universitas Islam Indonesia. Yogyakarta. Indonesia.