PENATAAN ULANG *SOFTWARE REQUIREMENT SPECIFICATION* (SRS) SISTEM INFORMASI AKADEMIK DENGAN PENDEKATAN *REVERSE ENGINEERING*

Eko K. Budiardjo[1)] dan Yanti Andriyani [2)]
[1)] Fakultas Ilmu Komputer, Universitas Indonesia
[2)] Program Studi Teknik Informatika, STT-PLN, Jakarta
Email: [1)] eko@cs.ui.ac.id, [2)] yeandriyani@yahoo.com

ABSTRAK

Makalah ini menjelaskan bagaimana pendekatan Reverse Engineering dapat dipergunakan untuk menyusun kembali Software Requirement Specification (SRS) dari piranti lunak sistem informasi akademik yang telah operasional pada organisasi. Sistem informasi akademik (SIAK) STT-PLN merupakan studi kasus terpilih untuk memperlihatkan pendekatan ini. SRS yang diperoleh disusun berdasarkan evaluasi terhadap piranti lunak aplikasi yang sudah terbangun, yang telah berada pada tahapan construction pada siklus pengembangan piranti lunak. Evaluasi terhadap Code menhasilkan Use Case Realization sebagai artefak dari tahapan Analysis dan Design, yang merupakan tahapan sebelumnya. Pada akhirnya menghasilkan SRS, dengan pendekatan pemetaan satu-satu terhadap Use Case dan spesifikasinya, sebagai artefak pada tahapan Requirement Engineering.

Kata kunci: *Software Requirement Specification (SRS), Use Case (UC), Requirement Engineering (RE), Reverse Engineering*, Sistem Informasi AKademik (SIAK)

PENDAHULUAN

Proses pengembangan piranti lunak adalah proses yang tidak sederhana dan memerlukan pengetahuan dan keterampilan yang khusus. Dalam bidang ini telah berkembang berbagai teknik dan metodologi yang mempermudah proses pengembangan perangkat lunak yang berkualitas. Proses pengembangan piranti lunak sendiri terdiri dari beberapa tahapan dengan tahapan umum dimulai dari *analysis-design-coding-testing-implementation*, dengan berbagai variasinya bergantung metodologi pengembangan piranti lunak tersebut seperti *waterfall*, *iterative*, *Rational Unified Process* (RUP), *Extreme Programming* (XP), dan sebagainya. Dalam proses mengembangkan sebuah piranti lunak, pada domain masalah apapun, akan diawali oleh tahapan *Requirement Engineering* guna menghasilkan SRS.

Pada tulisan ini akan menjelaskan proses untuk menyusun kembali SRS berbasis pada UC. Sebuah piranti lunak Sistem Informasi Akademik (SIAK) merupakan studi kasus untuk menguji pendekatan yang diusulkan. Proses ini menggunakan pendekatan *Reverse Engineering* yang merupakan proses terbalik dari proses pengembangan perangkat lunak. Kegiatan *Reverse Engineering* yang dilakukan diawali dengan melakukan wawancara, observasi, evaluasi terhadap setiap siklus end-to-end dari setiap interaksi antara pengguna dan komputer. Pada akhirnya salah satu varian SRS dari piranti lunak yang sudah terbangun dapat dipetakan ke dalam bentuk SRS.

Domain masalah ini dipilih karena permasalahan ini belum terlalu mendapatkan perhatian dalam kegiatan pengembangan piranti lunak SIAK. Hal ini berdampak pada ketidak-lengkapan dokumen - dokumen pengembangan, termasuk SRS. Dapat dimengerti mengapa hal ini terjadi, karena sebagian besar masih mengembangkan piranti lunak SIAK secara *ad hoc.* Berbagai manfaat dapat diperoleh dengan adanya SRS, a.l. (a) pengembangan aplikasi selanjutnya dapat lebih terarah, karena *requirement* yang terlewatkan dapat diketahui; (b) penggunaan kembali requirements dari piranti lunak yang telah teruji dapat merupakan *best* practices; (c) penambahan *requirement* dapat dilakukan secara sistemik dan prioritas pembangunannya dapat disusun berdasarkan realitas.

Requirement Engineering (RE)

Requirement dalam konteks rekayasa perangkat lunak adalah sebuah kemampuan yang harus dimiliki oleh sebuah piranti lunak [7, 9]. Kemampuan ini dapat ditujukan untuk memecahkan suatu permasalahan ataupun diperlukan untuk memenuhi ketentuan - ketentuan tertentu (seperti bakuan, keputusan manajemen, ataupun alasan-alasan politis). Kumpulan dari berbagai *requirement* digunakan dalam berbagai aspek pengembangan piranti lunak. Dalam tahap perancangan, *requirement* digunakan untuk menentukan berbagai fitur yang akan ada di dalam sistem. Pada penghujung sebuah *development effort*, himpunan *requirement* ini digunakan untuk melakukan *validation & verification* untuk memastikan perangkat lunak yang telah dibuat memang sesuai dengan

yang diinginkan. Bahkan selagi pengembangan berjalan, himpunan *requirement* ini terus dimodifikasi untuk menyesuaikannya dengan berbagai kebutuhan para *stakeholder* serta tenggat waktu dan dana yang tersedia.

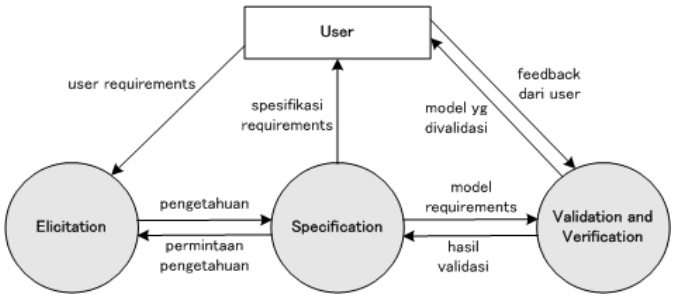
Secara luas, *software systems* (RE) adalah proses untuk menemukan suatu himpunan *requirement* yang tepat sehingga suatu perangkat lunak dapat memenuhi kegunaannya. Proses ini dilakukan dengan cara mengenali para *stakeholder* serta kebutuhan mereka serta mendokumentasikannya di dalam bentuk yang dapat digunakan untuk analisa, komunikasi, dan implementasi yang mengikutinya [10].

Zave memberikan salah satu definisi yang paling jelas dari RE:

"Is the branch of software engineering concerned with the real-world goals for, functions of, and constraints on software systems. It is also concerned with the relationship of these factors to precise specifications of software behavior, and to their evolution over time and across software families."

Requirements engineering adalah cabang dari software engineering yang mengurusi masalah yang berhubungan dengan: tujuan (dunia nyata), fungsi, dan batasan-batasan pada sistem software. Termasuk hubungan faktor-faktor tersebut dalam menetapkan spesifikasi yang tepat dari suatu software, proses evolusinya baik berhubungan dengan masalah waktu maupun dengan software lain.

Hasil dari fase dalam *requirement specification*. Requirements specification berisi kesepakatan bersama tentang permasalahan yang ingin dipecahkan antara pengembang dan customer, dan merupakan titik awal menuju proses berikutnya yaitu *software design*. Sistemisasi proses negosiasi pengembang dan customer dalam requirements engineering dibagi dalam 3 proses besar yaitu: *elicitation, specification, validation* dan *verification*. Formula ini kemudian juga dikenal dengan nama *The Three Dimensions of* yang dijelaskan pada gambar di bawah ini.



Gambar 1: Proses *Requirement Engingeering* (2)

Menurut Kotonya dan Sommerville, **Requirements Management (RM)** adalah kegiatan untuk mengelola perubahan terhadap requirements sistem yang akan dikembangkan. Menurut Alan Davis, requirement management adalah suatu pendekatan sistematis untuk mengidentifikasikan, mengorganisasikan, mengkomunikasikan dan mengelola perubahan terhadap requirement piranti lunak.

Kegiatan yang dilakukan dalam requirement management menyangkut beberapa hal, antara lain:

- Melakukan pengelolaan terhadap perubahan requirement yang sudah disepakati
- Melakukan pengelolaan terhadap keterhubungan antara beberapa requirement
- Mengelola dokumen yang dihasilkan pada tahapan requirements dengan dokumen yang dihasilkan pada tahapan lai dari pengembangan piranti lunak.

Kegiatan dalam requirements management terkait dengan dokumen requirements. Dokumen digunakan untuk menyajikan dan menyimpan Informasi requirements. Jika terjadi perubahan terhadap requirements berarti informasi yang terdapat pada dokumen pun berubah. Kegiatan requirements management melakukan pengelolaan terhadap perubahan yang dilakukan terhadap dokumen ini. Terdapat beberapa dokumen tersebut di antaranya adalah:

- **Dokumen Visi dan Ruang Lingkup (Vision and Scope Document)** [8, 9]**.** Dokumen ini berisi bussiness requirement dari proyek yang sedang berlangsung, biasanya pada dokumen ini terkandung informasi mengenai latar belakang dan peluang bisnis mengapa proyek dilakukan, visi dan tujuan dari proyek yang dilakukan, batasan dan penjelasan produk yang dikembangkan, dan siapa yang menjadi pelanggan dari proyek yang dilakukan.
- **Dokumen Use Case.** Use case merupakan suatu pendekatan dalam rangka mendapatkan requirements yang mendeskripsikan cara seorang pengguna akan berinteraksi dengan sistem dengan cara memodelkan requirement. Use case berkaitan dengan actor, yaitu pengguna atau sistem lain yang berhubungan dengan sistem yang dikembangkan. Satu use case

menggambarkan runtunan interaksi antara pengguna dengan sistem. Urutan interaksi disebut dengan *basic flow.* Dan interaksi alternatif lainnya disebut dengan alternative flow. Dan sebelum menjalankan urutan interaksi tersebut, juga paerlu diinformasikan kondisi sebelum dan sesudah dilalui use case, yang sering disebut *precondition* dan *postcondition.*

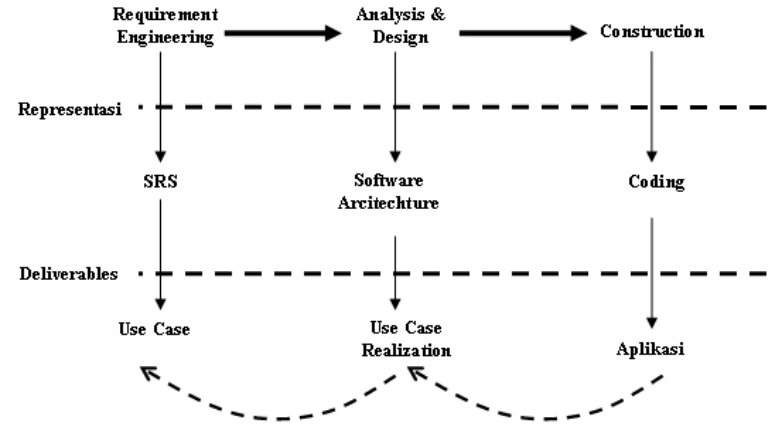
- **Dokumen Software Requirement Specification (SRS).** Dokumen Software Requirement Specification (SRS) ini menguraikan fungsi-fungsi dan kemampuan dari sistem yang harus dikembangkan dan harus dipenuhi. Selain itu SRS menguraikan batasan-batasan pengembangan yang harus dipatuhi. SRS adalah landasan dari tahapan selanjutnya dalam pengembangan piranti lunak. SRS ini merupakan rangkuman dari beberapa dokumen yang telah disebutkan sebelumnya

Metoda *Reverse Engineering* Yang Diusulkan

Pada Gambar 2. menjelaskan proses alur *reverse engineering* yang ditunjukkan pada panah putus-putus pada bagian bawah. Adapun tiga proses utama dari pengembangan perangkat lunak antara lain:

a. Requirement Engineering

Pada tahapan ini adalah tahapan dalam melakukan identifikasi kebutuhan (requirements) perangkat lunak yang akan dibangun. Yang nantinya akan direpresentasikan secara detail dalam bentuk dokumen SRS. Dan selain SRS, bentuk deliverable yang akan dijadikan acuan untuk tahapan berikutnya adalah pemodelan kebutuhan dalam bentuk Use case, yang diklasifikasikan dalam bentuk use case modelling dan use case description.



Gambar 2: Alur proses *reverse engineering*

b. Analysis and Design

Tahapan berikut dalam melakukan pengembangan perangkat lunak yang sudah mengklasifikasikan secara detail arsitektur dari aplikasinya berdasarkan komponen kebutuhan dari tahapan requirement sebelumnya.

c. Construction

Tahapan dalam melakukan pembangunan aplikasi adalah tahap konstruksi atau masuk ke dalam tahapan pengkodean. Yang nantinya akan menghasilkan aplikasi sesuai dengan kebutuhan, analisa dan desain yang sudah dilakukan sebelumnya.

Makalah tidak membahas alur pengembangan perangkat lunak secara *forward engineering*, namun akan dibahas secara *reverse engineering*. Dari aplikasi yang sudah digunakan dan terimplementasi (teruji) lalu dilakukan identifikasi kembali untuk mendapatkan sebuah SRS. Dengan melakukan proses analisa hasil dari *construction* berupa sebuah aplikasi, untuk dikenali sebagai sebuah Use case realization. Dan nantinya hasil dari UC realization akan diasumsikan 1 : 1 [7, 8, 9], artinya satu buah UC realization akan dipetakan menjadi satu basic use case, dan akan dijelaskan secara detail dalam dokumen SRS

Pemahaman Domain Masalah Pada Sistem Informasi Akademik

Penerapan metoda ini tidak dapat dilakukan secara tepat guna, tanpa terlebih dahulu memahami domain masalah pada studi kasus, Sistem Informasi Akademik (SIAK). Pada umumnya SIAK memberikan fokus lebih kepada pelayanan berbasis komputer untuk bidang akademik. SIAK merupakan tiang utama dalam mengatur segala hal yang berkaitan dengan penyelenggaraan

perkuliahan maupun hal-hal lainnya. Sistem ini secara khusus dirancang untuk memenuhi kebutuhan Perguruan Tinggi yang menginginkan layanan pendidikan yang terkomputerisasi untuk meningkatkan kinerja, kualitas layanan, daya saing serta kualitas SDM yang dihasilkannya. Pengembangan SIAK dilakukan secara berkesinambungan dan intensif, mengikuti perkembangan teknologi dan kebutuhan mendasar perguruan tinggi. SIAK dirancang untuk mampu mendukung semua hal yang berkaitan dengan penyelenggaraan perkuliahan maupun hal-hal spesifik lainnya. Segala hal yang berhubungan dengan urusan kampus dan perkuliahan dapat diselesaikan berbantuan SIAK, sehingga dapat memudahkan mahasiswa, dosen, dan staf kampus untuk mengolah data atau mendapatkan informasi yang dibutuhkan.

Dalam penelitiannya, Fernandez (4, 5) menjelaskan manajemen akademik menjadi 2 bagian utama dan 1 bagian yang mempertemukan keduanya, yang meliputi:

- ***Course Registration***, merupakan proses registrasi akademik yang didalamnya terdapat proses registrasi mahasiswa mendaftar ke dalam matakuliah yang diikutinya dan mengolah data siapa saja mahasiswa yang mengikuti kuliah tersebut
- ***Grade Management***, merupakan proses manajemen nilai yang dilakukan oleh dosen
- ***Course Management***, merupakan proses yang didalamnya menjelaskan dua aspek, registrasi dan evaluasi. Dan juga termasuk di dalamnya manajemen matakuliah yang diselenggarakan, manajemen dosen pengajar dan dosen pembimbing.

Dalam studi kasus ini, SIAK terbagi menjadi beberapa bagian prosesnya sebagai berikut [10]:

- **Proses Registrasi Akademik**, dimana dalam proses registrasi akademik terdapat beberapa proses utama seperti proses pembuatan KRS (Kartu Rencana Studi), proses pembatalan matakuliah pada KRS (Drop KRS), penambahan mata kuliah pada KRS (Add KRS), sampai dengan proses bukti mahasiswa bisa mengikuti perkuliahan pada semester yang berjalan.
- **Proses Manajemen Perkuliahan**, pada proses ini mendeskripsikan proses manajemen proses perkuliahan, seperti alokasi mata kuliah, alokasi jadwal perkuliahan, alokasi dosen pengajar, alokasi ruangan dan fasilitas yang digunakan, serta manajemen absensi perkuliahan.
- **Proses Administrasi Keuangan**, proses ini berkaitan dengan proses pembayaran administrasi perkuliahan mahasiswa, yang nantinya akan berhubungan dengan proses registrasi akademik
- **Proses Manajemen Nilai**, proses yang menjelaskan pengelolaan nilai mahasiswa yang diberikan oleh dosen sebagai hasil evaluasi belajar mahasiswa disetiap semesternya. Proses Kartu Hasil Studi (KHS) dan Transkrip Nilai termasuk di dalam proses ini.

Selanjutnya, dalam kajian ini, fokus diberikan pada proses registrasi akademik (*course registration*) yang merupakan bagian dari SIAK, untuk itu observasi yang dilakukan menghasilkan identifikasi sebagai berikut:

- Dukungan terhadap kegiatan Pengisian KRS, proses yang menjelaskan alur pendaftaran dan pembuatan KRS melalui prosedur konsultasi bimbingan dengan dosen pembimbing.
- Dukungan terhadap kegiatan Perbaikan KRS, yang meliputi:
    - Proses Perbaikan KRS (Batal), menjelaskan proses perbaikan KRS dalam hal pembatalan pengambilan matakuliah sehingga mengurangi beban SKS yang sudah diinputkan ke dalam data KRS sebelumnya.
    - Proses Perbaikan KRS (Tambah), menjelaskan proses perbaikan KRS dalam hal penambahan pengambilan matakuliah sehingga menambah beban SKS yang sudah diinputkan ke dalam data KRS sebelumnya.
- Dukungan terhadap kegiatan Mencetak Kartu Studi Mahasiswa (KSM), menjelaskan proses pencetakan KSM yang digunakan sebagai bukti yang sah bagi mahasiswa untuk mengikuti perkuliahan pada semester yang berjalan. Proses pengambilan matakuliah belum dikatakan sah jika belum mendapatkan KSM.

Penerapan Metoda Pada Studi Kasus Sistem Informasi Akademik

Setelah memahami domain masalah, penerapan metoda ini dilakukan dengan melakukan meng-identifikasi interaksi aktor dengan komputer pada kegiatan registrasi akademik, yang bersifat end-to-end. Setiap interaksi ini, merupakan sebuah *Use Case Realization (UCR)*. Dengan mengenali *Basic Flow* (*The Happy Path*) merupakan cara yang mendasar untuk menemukan UCR. Setelah mengenali *Basic Flow*, barulah kegiatan dilanjutkan dengan mengenali *Alternate Flow*, dari Basic Flow tersebut. Pada studi kasus ditemukan sejumlah UCR antara lain seperti yang diperlihatkan pada Tabel 1.

Tabel 1: Basic Flow dan Alternate Flow dari UCR Mengisi KRS

Basic Flows :			
Actor Action		**System Response**	
1.	Aktor memasukkan user name dan password		
		2.	System melakukan validasi terhadap user name dan password
3.	Aktor melakukan pilihan input data KRS		
		4.	Sistem meminta aktor memasukkan NIM
5.	Aktor melakukan input NIM		
		6.	Sistem menampilkan form input KRS dan meminta input Kode MK yang diambil mahasiswa
		7.	Sistem melakukan validasi kapasitas kelas dan jadwal matakuliah yang dipilih
		8.	Sistem meminta Kode MK berikutnya
9.	Kembali ke basic flow 5		
		9.	Sistem melakukan penghitungan jumlah SKS yang diambil
		10.	Sistem menampilkan detail data KRS yang sudah diinputkan
		11.	Sistem melakukan update database KRS
Alternate Flows: 1. Jika matakuliah yang diambil bermasalah dengan jadwal yang bentrok atau kelas yang sudah penuh atau total SKS yang diambil melebihi batas maksimum, maka sistem menampilkan pesan status dari setiap permasalahan. Dan mahasiswa harus melakukan pengaturan pengambilan kelas secara manual untuk dilakukan input kembali oleh staff jurusan 2. Sistem melakukan penyimpanan otomatis jika masih terdapat KRS yang bermasalah, setiap KRS yang bermasalah sistem akan manampilkan status untuk melakukan proses batal/tambah.			

Dengan dilakukan pendekatan (asumsi) bahwa sebuah UCR merupakan realisasi dari sebuah UC, yang selanjutnya dikenal sebagai memiliki pemetaan 1 : 1. Maka dapat disusun Spesifikasi UC Mengisi KRS lengkap, seperti yang diperlihatkan pada Tabel 2.

Tabel 2: Spesifikasi UC Mengisi KRS

Use Case Name: **Mengisi KRS**
Primary Actor: Staff Jurusan
Brief Description: Merupakan interaksi *Actor* dengan komputer untuk melakukan input data mata kuliah yang akan diambil oleh mahasiswa pada semester aktif. Data tersebut tertera Form RPM (Rencana Pengambilan Matakuliah) yang sudah diisi dan disetujui sebelum oleh dosen pembimbing.
Key Scenario: 1. Staff jurusan melakukan input KRS 2. Sistem menyimpan data KRS 3. Mahasiswa melakukan proses konsultasi dengan dosen pembimbing akademik 4. staff jurusan mencetak KRS untuk proses (Batal/Tambah) dan mendapatkan KSM
Precondition: 1. Data alokasi jadwal sudah diintegrasikan dengan data KRS 2. Proses manual konsultasi dosen PA sudah dilakukan oleh mahasiswa 3. Form RPM yang sudah disetujui oleh PA baru dapat dilakukan input data KRS

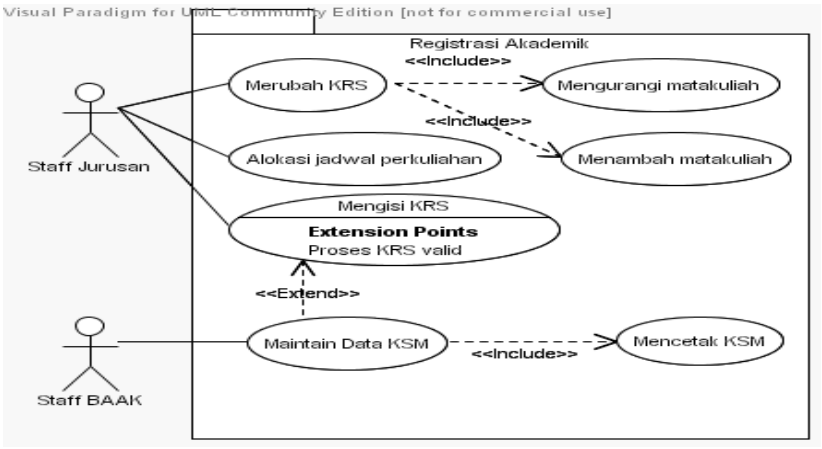
Postcondition:
1. Data KRS terupdate ke dalam database
2. Data KRS terintegrasi dengan data KBT dan KSM
3. Cetak data KRS untuk proses selanjutnya

Basic Flow :

Actor Action		System Response	
1.	Aktor memasukkan user name dan password		
		2.	System melakukan validasi terhadap user name dan password
3.	Aktor melakukan pilihan input data KRS		
		4.	Sistem meminta input NIM
5.	Aktor melakukan input NIM		
		6.	Sistem menampilkan form input KRS dan meminta input Kode MK yang diambil mahasiswa
		7.	Sistem melakukan validasi kapasitas kelas dan jadwal matakuliah yang dipilih
		8.	Sistem meminta Kode MK berikutnya
9.	Kembali ke basic flow 5		
		9.	Sistem melakukan penghitungan jumlah SKS yang diambil
		10.	Sistem menampilkan detail data KRS yang sudah diinputkan
		11.	Sistem melakukan update database KRS

Alternate/Exceptional Flows:
1. Jika matakuliah yang diambil bermasalah dengan jadwal yang bentrok atau kelas yang sudah penuh atau total SKS yang diambil melebihi batas maksimum, maka sistem menampilkan pesan status dari setiap permasalahan. Dan mahasiswa harus melakukan pengaturan pengambilan kelas secara manual untuk dilakukan input kembali oleh staff jurusan
2. Sistem melakukan penyimpanan otomatis jika masih terdapat KRS yang bermasalah, setiap KRS yang bermasalah sistem akan manampilkan status untuk melakukan proses batal/tambah.

Aktivitas serupa dilanjutkan terhadap keseluruhan interaksi aktor dengan komputer dalam lingkup kegiatan registrasi akademik. Dalam studi kasus pada makalah ini, ditemukan 7 (Tujuh) buah use case, yang selanjutnya dapat disusun kedalam sebuah UC Diagram, seperti yang terlihat pada Gambar 3.



Gambar 3: Diagram UC pada kegiatan registerasi akademik

KESIMPULAN

Kegiatan *requirement engineering* terdiri atas tiga proses yang meliputi (a) *elicitation;* (b) *specification;* (c) *validation and verification. Requirements specification* berisi kesepakatan bersama tentang permasalahan yang ingin dipecahkan antara pengembang dan customer, dan merupakan titik awal menuju proses berikutnya yaitu *analysis & design*. Dalam hal proses *reverse engineering* dilakukan aktivitas kebalikannya *(backward analysis)*. Analisis terhadap aplikasi yang sudah dipergunakan (*proven*), dapat dikenali interaksi end-to-end antara aktor dan sistem (komputer), yang selanjutnya disetarakan sebagai *use case realization*. Selanjutnya dengan pendekatan pemetaan 1 : 1, UCR dapat disetarakan sebagai UC

Penataan ulang SRS, dengan pendekatan usulan penulis, dapat tersusun SRS dalam bentuk UC Diagram & UC Specification. SRS tersebut nantinya dapat digunakan sebagai dasar dalam pengembangan atau pembangunan sebuah aplikasi kembali tanpa harus melakukan kegiatan *requirement engineering* secara komprehensif. Bila mana kasus serupa diperbanyak, terhadap pada domain yang sama dapat disusun menjadi sebuah SRS Patterns.

DAFTAR PUSTAKA

1. Bleistein, Steven J.; Karl Cox; June Verner; et all. *Requirements Engineering for e-Business Advantage*. London: Springer-Verlag. Sep. 2005.
2. Budiardjo, Eko K., Kodrat Mahatma. *Pendekatan Software Requirement Pattern untuk Prasyarat dan Spesifikasi Sistem Informasi Keanggotaan di Rumah Sakit.* Yogyakarta: Prosiding Seminar nasional riset teknologi informasi (SRITI) 2006. Jul. 1, 2006.
3. Budiardjo, Eko K.; Zakky S. Balbeid. *Analisis Fitur Sistem Informasi Loyality Program Sebagai Bagian Dari CRM Perhotelan*. Bandung: Prosiding Konferensi Nasional Sistem Informasi. Feb. 2007.
4. Fernandez, Eduardo B.; X. Yuan. *Semantic Analysis Patterns.* Florida: Dept. of Computer Science and Eng., Florida Atlantic University, Boca Raton. 2000.
5. Fernandez, Eduardo B. *An Analysis Pattern for Course Management.* Florida: Departement of Computer Science and Engineering, Florida Atlantic University Boca Raton. 2001.
6. Fowler, Martin. *UML Distilled 3rd ed: A Brief Guide to the Standard Object Modeling Language*. Boston: Addison-Wesley. 2002.
7. Kotonya, Gerald and Ian Sommerville, Requirement Engineering Processes and techniques, Jhon Willey & Sons, Ltd, England, 1998
8. Larman, Craig, Applying UML and Pattern, and Introduction to Object Oriented Analysis and Design, Second Edition, Prentice Hall PTR, 2002
9. Leffingwell, Dean; Don Widrug. *Managing Software Requirements, A Use Case Approach, Second Edition.* Addison-Wesley. 2003.
10. Nuseibeh, Bashar; Steve Easterbrook. *Requirement Engineering: A Roadmap*. London: Department of Computing, Imperial College. 2000.
11. PLN, STT - PLN, Buku Petunjuk dan Pedoman Akademik STT-PLN, Sekolah Tinggi Teknik PLN, 2007
12. Rational Development Company, Rational Unified Process, Best Practices For Software Development Teams, Rational Software White Paper, TP026B, Rev 11/01
13. Wahono, Romi Satria, Analyzing Requirements Engineering Problems, Department of Information and Computer Sciences, Saitama University, Lembaga Ilmu Pengetahuan Indonesia (LIPI)-Pusat Dokumentasi Informasi Ilmiah (PDII), 2003
14. Yuan, X; Fernandez; Eduardo B*. An analysis pattern for course management*. Procs. EuroPLoP'03. 2003.