Membangun Aplikasi Berdasarkan Konsep SaaS: Studi Kasus Sistem Billing Rumah Sakit

Roy D.H. Tobing[1], Sandro Hutapea[2], Relita Marpaung[3], Rista Sihombing[4]

[1,2,3,4]Program Studi Manajemen Informasi, Politeknik Informatika Del
Jl. Sisingamangaraja, Sitoluama, Laguboti, Tobasa, 22381
[1]roy.deddy@del.ac.id,{[2]if07050, [3]if07052, [4]if07026}@students.del.ac.id

Abstrak

Pertama, penulis menyatakan bahwa makalah ini disarikan dari tugas akhir D3 Politeknik Informatika Del[1]. Topik yang dipilih dalam riset terkait Total Cost of Ownership (TCO) sebuah software. Kepemilikan software secara tradisional relatif mahal karena biaya pengembangan software, biaya hardware dan software tambahan, biaya operasional dan perawatan. Software-as-a-Service (SaaS) merupakan salah satu yang dianggap sebagai solusi terhadap isu TCO tersebut. Konsep ini merubah paradigma pada model bisnis, arsitektur aplikasi dan struktur operasional. Riset ini bertujuan untuk membangun sebuah aplikasi berdasarkan konsep SaaS dan difokuskan pada domain sistem billing rumah sakit.

Keywords: software development, ICT, tugas akhir, Software as a Service, SaaS.

1 Pendahuluan

Rumah sakit adalah organisasi di bidang kesehatan yang memiliki kebutuhan untuk penggunaan *software* dalam mendukung efisiensi proses bisnisnya. Saat ini, tersedia banyak *software* yang siap dipakai untuk kebutuhan rumah sakit tersebut, belum lagi pengembangan *software* yang dilakukan secara internal oleh rumah sakit. Akan tetapi, kepemilikan akan sebuah *software* diikuti oleh biaya-biaya yang bukan hanya dikeluarkan pada saat pembelian dan pen-*deploy*-an *software* tersebut. *Software Total Cost of Ownership* (TCO), istilah populer terkait biaya-biaya kepemilikan sebuah *software*, juga terdiri dari biaya hardware, biaya pengembangan *software*, biaya lisensi dan *upgrade*, biaya operasional, dan biaya perawatan. Akibatnya, sebuah organisasi pada dasarnya mengeluarkan biaya yang relatif besar untuk memiliki *software-assisted* proses bisnis.

Software-as-a-Service (SaaS) muncul sebagai sebuah solusi. SaaS adalah konsep yang mengubah paradigma dalam model bisnis, arsitektur aplikasi, dan struktur operasional[2]. Pada dasarnya, SaaS merupakan *pay-as-you-go model* dengan metode berlangganan untuk menggunakan sebuah aplikasi[3] dan telah dibuktikan bahwa SaaS memiliki operasional yang jelas, *financial winner* dalam hal penghematan biaya, dan menyediakan manfaat *availability* dan *redundancy*[4][5].

Riset ini bertujuan untuk membuat sebuah sistem *billing* rumah sakit berdasarkan perubahan paradigma yang diperkenalkan oleh konsep SaaS. Akan tetapi, fokus utamanya adalah arsitektur aplikasi. Salah satu alasannya adalah aplikasi yang dibangun dengan menggunakan SaaS menawarkan fungsi generik kepada pelanggannya sehingga menjadi isu menarik untuk dieksplorasi selama riset terkait proses pengembangan *software* tersebut. Hanya beberapa isu terkait model bisnis dan struktur operasional yang didiskusikan pada riset ini karena kedua hal tersebut tergantung pada visi sebuah penyedia layanan tentang bagaimana aplikasi SaaS tersebut akan dipasarkan kepada kliennya. Pada makalah ini hanya disarankan sebuah model bisnis dan struktur operasional yang dapat diimplementasikan pada sistem *billing* rumah sakit.

Dalam sebuah implementasi aplikasi, SaaS memanfaatkan teknologi lain, misalnya *Service Oriented Architecture* (SOA)[3]. Pada riset ini juga digunakan SOA ([6,7]) untuk melengkapi implementasi konsep SaaS. Dalam melakukan riset ini, langkah-langkah yang dilakukan untuk mencapai tujuan adalah: (1) mempelajari karakteristik dari konsep SaaS, (2) mengkonstruksi sebuah studi kasus, (3) menganalisis *software requirements*, (4) mendesain *software*, (5) membangun aplikasi sesuai dengan konsep SaaS yang dipelajari. Pengujian aplikasi juga merupakan bagian dari langkah ini.

2 Hasil

Studi Kasus Sistem Billing Rumah Sakit

Pada riset ini dilakukan pengkonstruksian studi kasus dengan domain sistem *billing* rumah sakit. Sistem *billing* rumah sakit dipilih karena Politeknik Informatika Del (PI Del)[8] memiliki sebuah proyek nyata untuk membangun sistem sejenis di dua rumah sakit nyata dan adanya kebutuhan untuk menguji apakah sistem *billing* rumah sakit dapat dibangun berdasarkan konsep SaaS. Studi kasus ini, sebut saja Rumah Sakit XYZ, adalah fiktif tapi relatif realistik dan sebagian besar berdasarkan proses bisnis pada kedua rumah sakit klien PI Del yang

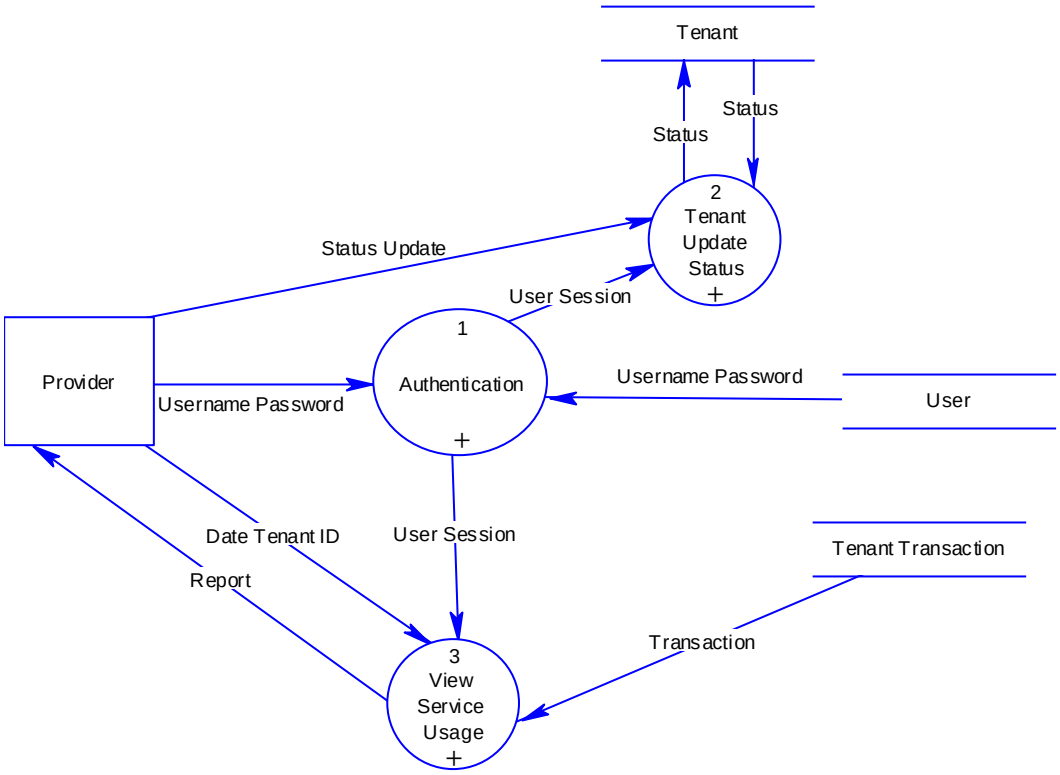
disebutkan tadi. Nama kedua rumah sakit tersebut tidak disebutkan untuk keperluan *anonymity*.

Rumah sakit XYZ adalah rumah sakit swasta yang berkembang dari hanya satu kantor ketika pertama kali diresmikan hingga menjadi rumah sakit yang memiliki kantor pusat di Balige, Sumatera Utara dan beberapa cabang yang tersebar di daerah-daerah di Indonesia. Rumah sakit ini melayani Rawat Inap, Rawat Jalan, Unit Gawat Darurat, *Intensive Care Unit*, Ruang Operasi, Laboratorium, Farmasi, dan Instalasi Kamar Mayat. Setiap kelompok layanan rumah sakit ini terdiri dari *treatments* yang lebih detail. Rumah sakit menagih biaya terhadap pasien yang sudah teregistrasi dan menggunakan fasilitas layanan dari rumah sakit. Setiap unit layanan rumah sakit memiliki orang yang bertanggung jawab untuk mencatat semua informasi penggunaan layanan oleh pasien. Setiap pasien mendapat *treatment*, sebuah *record* baru akan ditambahkan ke rekam mediknya. Jika pasien *unregister* dari rumah sakit, maka si pasien tersebut harus membayar biaya pemakaian layanan rumah sakit yang dipakainya.

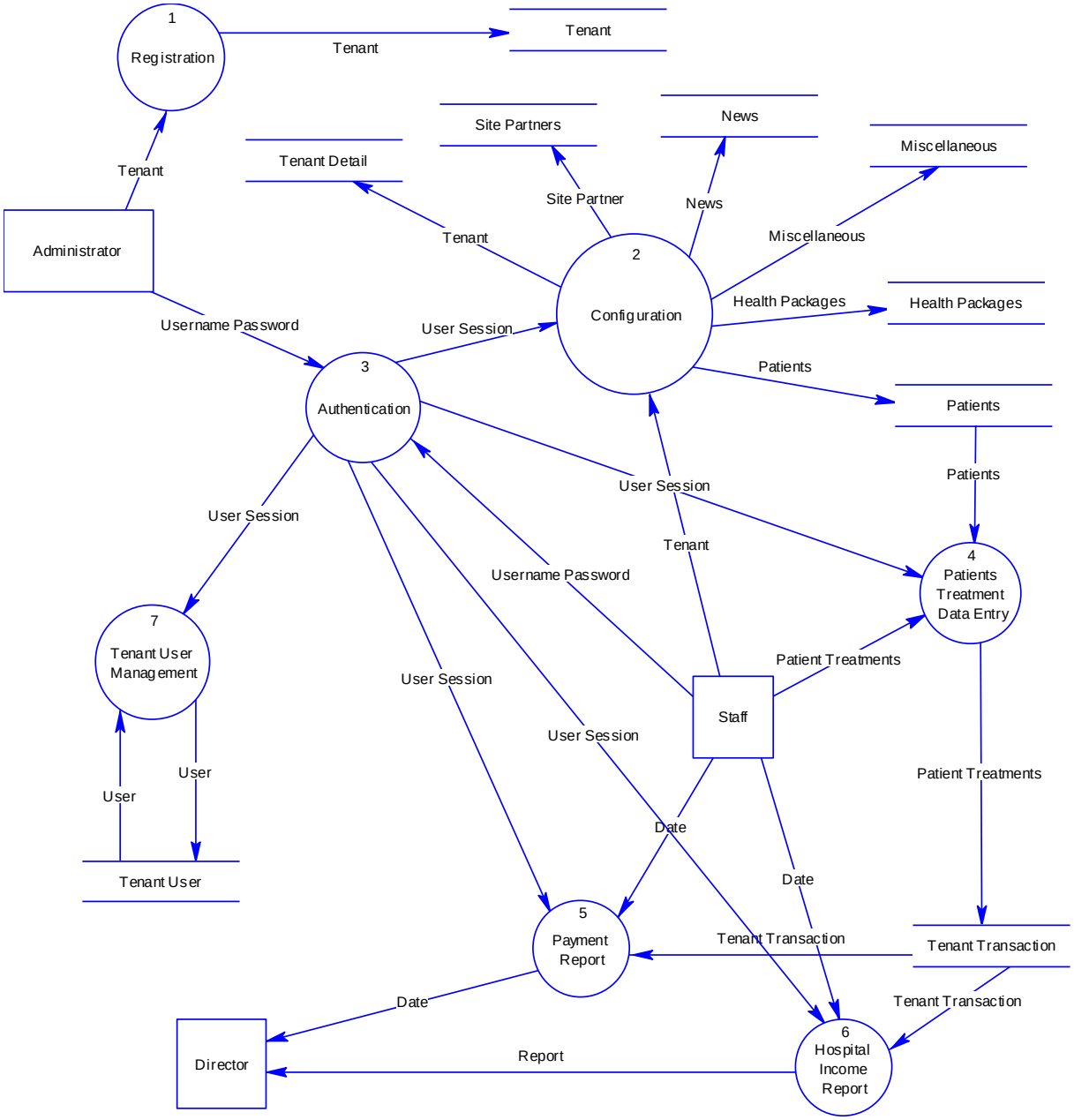
Cabang-cabang dari rumah sakit akan mengirimkan semua data transaksi pasien melalui pos atau email ke kantor pusat. Di kantor pusat, data tersebut akan diintegrasikan untuk mendapatkan informasi lebih lengkap tentang pembayaran yang telah dilakukan oleh pasien di Rumah Sakit XYZ.

Analisis

Berdasarkan studi kasus, telah diidentifikasi proses bisnis yang terjadi di rumah sakit, yaitu pengentrian data *treatment* pasien dan pelaporan. Pada sistem *billing* rumah sakit yang dibangun, proses bisnis dibagi menjadi dua kelompok besar berdasarkan lokasi, yaitu proses bisnis di sisi penyedia layanan dan di sisi klien (*tenant*). Kebutuhan fungsional dan data dimodelkan dengan menggunakan *Data Flow Diagram* (DFD) pada saat analisis. Proses bisnis di sisi penyedia layanan ditunjukkan pada Gambar 1, sementara Gambar 2 menunjukkan DFD pada sisi klien. Model pada Gambar 2 dibuat setelah melakukan analisis terhadap proses generik yang cocok untuk lebih dari satu rumah sakit.



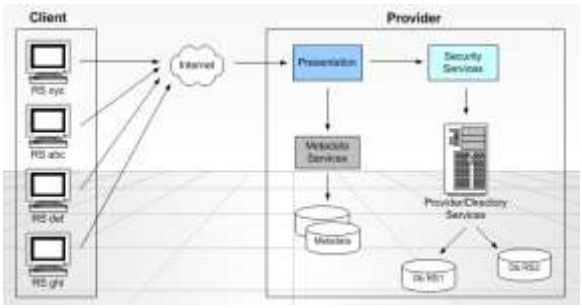
Gambar 1 DFD Dari Sisi *Provider*



Gambar 2 DFD Dari Sisi Klien

Proses 3 *View Service Usage* (lihat Gambar 1) muncul karena pada bagian ini *provider* dapat mengecek pemakaian layanan oleh klien dan menghitung biaya pemakaian aplikasi berdasarkan informasi tersebut.

Pada proses *Tenant Update Status* (lihat Gambar 2), status keanggotaan seorang klien akan di-*update* setelah si klien telah melakukan proses *Registration* (lihat Gambar 2) dan telah mentransfer uang pembayaran ke rekening bank si penyedia layanan (*provider*). *Provider* akan mengecek rekening banknya untuk mengkonfirmasi pembayaran oleh klien. Interaksi antara *provider* dan seluruh kliennya melalui Internet dapat dilihat pada Gambar 3.



Gambar 3 Gambaran Interaksi Antara *Provider* dan Klien

Pada SaaS, seperti telah disebutkan sebelumnya, mengubah paradigma di tiga area yang saling terhubung. Cara berpikir yang baru pada model bisnis berarti klien (rumah sakit) tidak memiliki *software*, infrastruktur dan manajemen teknologi berada di sisi *provider*. Di aspek struktur operasional, *provider* harus selalu memonitor ketersediaan layanan dan *shared-services*. Terakhir, arsitektur aplikasi terkait dengan level kedewasaan layanan, layanan metadata, dan *authorization*.

Ada tiga atribut yang diajukan untuk mengidentifikasi SaaS yang dibangun dengan baik dan membedakan level kedewasaan layanan, yaitu (1) *scalable*, (2) *multi-tenant-efficient*, dan (3) *configurable* [2]. Sistem *billing* rumah sakit yang dibangun memenuhi level III tingkat kedewasaan aplikasi SaaS yang ditandai oleh karakteristik *configurable* dan *multi-tenant-efficient*. Mengacu kepada *multi-tenancy*, *provider* harus membuat sebuah aplikasi klien yang dapat memenuhi proses bisnis yang sama bagi banyak rumah sakit. Oleh karenanya, sistem *billing* yang dibangun pada riset ini juga harus tersedia dan cocok bukan hanya pada Rumah Sakit XYZ, tapi juga untuk rumah sakit lain yang ingin berlangganan ke layanan tersebut. Walaupun demikian, tidak tertutup kemungkinan bahwa sebuah klien yang ingin bergabung juga dapat mengadaptasikan proses bisnisnya supaya cocok dengan yang ditawarkan oleh *provider*. Untuk setiap kasus tersebut, *provider* dituntut untuk menyediakan informasi yang jelas tentang layanan yang disediakannya. Sebagai tambahan, Rumah Sakit XYZ, seperti yang disebutkan sebelumnya, memiliki beberapa kantor cabang. Akan tetapi, rumah sakit yang ingin bergabung dan berlangganan ke layanan sistem *billing* ini tidak diharuskan untuk memiliki kantor cabang. Klien dengan satu buah kantor dimungkinkan untuk memakai layanan tersebut.

Lebih lanjut lagi, sistem *billing* rumah sakit memiliki layanan metadata, yaitu:

1. Fitur yang memberi kebebasan kepada tiap klien untuk mengubah logo, slogan, dan profilnya.
2. Tiap klien dapat secara independen mengatur data dan informasi terkait layanan kesehatan yang disediakan di rumah sakitnya.

Untuk *authorization*, setiap klien dapat secara independen satu sama lain untuk mengatur akun penggunanya dan *role* yang di-*assign* ketiap pengguna tersebut.

Desain

Untuk cetak biru (*design*) yang digunakan pada proses *coding*, digunakan pendekatan *Object Oriented Design* (OOD). Arsitektur *software* didesain agar sesuai dengan *framework* Model-View-Controller (MVC)[9]. Pada *High-Level Architecture*, SaaS memiliki kesamaan dengan aplikasi yang dibangun dengan mengikuti prinsip desain berorientasi service [6][7] dan hal ini menjadi salah satu alasan untuk memanfaatkan SOA untuk implementasi aplikasi SaaS.

Web services dibuat berdasarkan fungsi yang diturunkan dari proses yang tergambar pada DFD dari sisi klien (lihat Gambar 2) dan setiap *web service* tersebut memiliki *service* yang lebih detil. Pihak *provider* yang membuat *services* tersebut dapat diakses oleh aplikasi klien. Beberapa fungsi yang dipetakan ke *web service* dan *detailed service*-nya dapat dilihat pada Tabel 1.

Tabel 1 Contoh *Mapping* Antara Fungsi DFD ke *Web Service* dan *Services* Detilnya

Fungsi	
Authentication	
Web Service	***Services***
saas_service_login	a. checkUser
Fungsi	
Patients Treatment Data Entry	
Web Service	***Services***
saas_service_pasien	getPasien getPasienAll insertNewLog registerPasien
Fungsi	
Payment Report	
Web Service	***Services***
saas_service_transact ion	getDetailLog addTransactionData
Fungsi	
Hospital Income Report	
Web Service	***Services***
saas_service_report	getDetailLogPasien getDetailAllTransModule

Semua *web services* yang dibuat pada aplikasi *billing* rumah sakit memiliki prefiks *saas_service_* sebelum namanya dengan tujuan standardisasi penamaan *web service* di antara *programmers*.

***Coding* Dan Pengujian**

Sistem *billing* rumah sakit diimplementasikan dengan menggunakan **ASP.Net**. *Tools* pengembangan lainnya (*software*) dtunjukkan pada Tabel 2.

Tabel 2 *Tools* Pengembangan - *Software*

#	Item	*Tools*
1	Web Server	IIS
2	IDE	**Microsoft Visual Studi 2005**
3	Database	**SQL Server 2005**
4	Web browser	**Internet Explorer**, **Mozilla Firefox**

Disebabkan oleh batasan waktu, pengujian hanya dilakukan dengan mengaplikasikan *blackbox testing*[10]. Dalam pendekatan pengujian tersebut, mekanisme internal dari fungsi dihiraukan dan hanya difokuskan kepada pengecekan apakah jika data di-*input* ke dalam

sebuah fungsi, maka keluaran (*output*) sesuai dengan yang diharapkan. Data yang di-*input* pada saat pengujian adalah data yang valid dan tidak valid. Fungsi yang diuji adalah fungsi-fungsi seperti pada Tabel 1.

3 Diskusi

Pada sistem *billing* yang dibangun, setiap klien memiliki database masing-masing tapi dengan struktur data yang sama. Selain itu, klien dengan satu kantor pusat dan memiliki beberapa kantor cabang menggunakan hanya satu database. Metode ini dipakai untuk mengakomodasi aspek *multi-tenancy* dan *configurable* dari sistem *billing* rumah sakit. Penggunaan metode ini juga bertujuan untuk memisahkan dan mengisolasi data antara satu klien dengan klien lain.

Ketika sebuah klien berlangganan ke layanan, semua fungsi dapat diakses melalui Internet menggunakan *web browser*. Untuk memiliki web browser relatif mudah, murah dan bahkan gratis sehingga biaya untuk *software* tambahan untuk penggunaan aplikasi SaaS termasuk murah, bahkan dapat diabaikan. Akan tetapi, koneksi internet di daerah terpencil di Indonesia menjadi isu yang muncul pada saat penggunaan aplikasi SaaS. Kecepatan internet mempengaruhi ketersediaan layanan yang ditawarkan yang dapat berimplikasi pada *down-time* sistem *billing* rumah sakit yang relatif tinggi dan kurangnya *reliability*.

Layanan keamanan pada aplikasi SaaS terdiri dari:

1. *Delegated administration*, dimana klien memiliki kemampuan untuk membuat dan mengatur akun pengguna di tempatnya.
2. *Authorization*, yang berarti membuat *role* kepada setiap pengguna terkait tugasnya.

Untuk sistem *billing* rumah sakit yang dibangun, semua aspek keamanan yang disebutkan tersebut sudah diimplementasikan. Proses pendelegasian administrasi kepada klien dilakukan setelah *provider* telah menerima pembayaran atas registrasi klien yang diikuti peng-*update*-an status klien menjadi *registered customer*. Pada saat pertama kali klien *log-in* ke aplikasi setelah statusnya sudah di-*update*, klien akan diarahkan oleh sistem ke sebuah *form* di mana data *customer* tersebut dapat diubah.

Diskusi Aspek Pedagogi

Makalah ini disarikan dari tugas akhir mahasiswa Diploma 3 (D3) Politeknik Informatika Del (PI Del) [1]. Penulis pertama makalah ini merupakan dosen PI Del yang bertindak sebagai dosen pembimbing bagi tugas akhir yang dilakukan oleh tim yang terdiri dari penulis kedua, ketiga, dan keempat. Pada prosesnya, mahasiswa mengikuti jadwal yang sudah ditentukan untuk proyek ini oleh komite akademik PI Del. Pengerjaan tugas akhir ini dilakukan selama 2 semester, yaitu Tugas Akhir 1 (TA1) dan Tugas Akhir 2 (TA2), yang masing-masing dilakukan pada semester lima dan semester enam.

Ketika melakukan TA1, mahasiswa mengerjakan topik tugas akhir tersebut dan melaporkannya melalui dokumen yang berisi informasi studi literatur, studi kasus yang diformulasikan, dan hasil analisis yang telah dilakukan. Mahasiswa juga diizinkan untuk memasukkan sebagian kecil dari desain aplikasi yang sudah dilakukan ke laporan tersebut. Selanjutnya pada TA2, mahasiswa melengkapi proses desain, membangun aplikasi dan melakukan pengujian terhadap aplikasi tersebut. Penjaminan mutu TA1 dan TA2 dilakukan melalui dua sesi pengujian dan penilaian di tengah dan akhir setiap semester pengerjaan tugas tersebut. Pengujian dilakukan oleh dua dosen PI Del lainnya. Selain itu, mahasiswa harus membuat laporan aktivitas yang dilakukan selama pengerjaan tugas akhir dan diperiksa oleh petugas administrasi yang telah ditunjuk.

Mengacu ke klasifikasi Dennings tentang informatika[11] yang didasarkan pada aspek teori, aspek abstraksi, dan aspek desain, topik tugas akhir ini dapat dimasukkan ke klasifikasi Rekayasa Perangkat Lunak. Pada dasarnya, mahasiwa melakukan (1) mempelajari konsep baru yang belum diajarkan di kelas, dan (2) menguji konsep tersebut dengan membangun sebuah aplikasi. Aplikasi ini dibatasi ke sebuah domain yang spesifik, dalam kasus ini, sistem *billing* rumah sakit. Hal ini dilakukan karena mahasiswa hanya perlu menunjukkan kemampuannya dalam implementasi terkait karakteristik dan aspek konsep baru yang sudah dipelajarinya melalui pembuatan *software* daripada mengerjakan porsi terbesar yang difokuskan ke fungsi yang dimiliki *software* tersebut. Sepeti yang telah disebutkan sebelumnya, riset ini berfokus pada arsitektur aplikasi sementara model bisnis dan struktur operasional didiskusikan dengan porsi yang kecil. Hal ini dilakukan karena mahasiswa dapat menunjukkan kemampuannya dalam menggunakan pengetahuan dan keahlian menggunakan *tools* yang sudah dipelajari di kelas kebanyakan melalui bagian arsitektur aplikasi SaaS. Keahlian ini didemonstrasikan misalnya pada saat pendekatan yang dipakai pada proses rekayasa perangkat lunak (berdasarkan *Software Development Life Cycle*), penggunaan *tools* untuk analisis seperti DFD (lihat Gambar 1 dan Gambar 2), penggunaan *tools* untuk desain, seperti *object-oriented design tools*, penggunaan *tools* pemrograman dan database(lihat Tabel 2), dan implementasi pengujian dengan metode *blacbox testing*. Sementara itu, model bisnis dapat dilihat sebagai aspek ekonomi dari SaaS, sehingga diskusi hanya terkait kepemilikan sistem *billing* rumah sakit dan biaya operasionalnya. Untuk struktur operasional, diskusi terkait *deployment* dari aplikasi.

Mahasiswa menyatakan bahwa materi terkait konsep SaaS termasuk relatif mudah ditemukan, khususnya di Internet. Buku juga menjadi sumber informasi yang digunakan. Menurut pendapat penulis satu, cakupan topik tugas akhir ini relatif luas dan mahasiswa mengerjakannya dengan ruang lingkup yang relatif luas, walaupun studi kasus sistem *billing* yang dipakai hanya

memiliki beberapa fungsi utama yang relatif kecil. Tingkat kesulitan riset ini dianggap termasuk level sedang karena selain aspek pengembangan *software*, mahasiswa juga melakukan riset terkait model bisnis dan struktur aplikasi yang dapat diimplementasikan pada sistem *billing* rumah sakit ini, serta implementasi level kedewasaan SaaS.

4 Kesimpulan

Pada riset ini, sebuah sistem *billing* rumah sakit dibangun. Ada dua buah aplikasi sebagai hasilnya, yaitu: aplikasi di sisi penyedia layanan dan klien. Aplikasi tersebut dibangun berdasarkan konsep SaaS sebagai paradigma pengembangannya dan memanfaatkan SOA dalam implementasi arsitektur aplikasinya. Proyek ini menunjukkan bahwa SaaS dapat diterapkan untuk membangun sistem *billing* rumah sakit. Tiga atribut utama SaaS diimplementasikan pada sistem *billing* tersebut dan sesuai dengan level ketiga tingkat kedewasaan aplikasi SaaS (memenuhi *configurable*, *multi-tenant-efficiency*). Pada riset ini juga diajukan sebuah model bisnis yang dapat diformulasikan pada pengembangan sistem *billing* tersebut. Di pengembangan di masa yang akan datang, fitur yang diharapkan akan ditambahkan yaitu integrasi dengan media pembayaran online, manajemen pengguna di level klien yang lebih baik, dan aplikasi aspek keamanan sistem *billing* yang lebih baik.

Untuk aspek pedagogi, riset ini dianggap sudah cocok dan *applicable* untuk tugas akhir mahasiswa D3 karena topiknya yang masih dalam klasifikasi ilmu komputer, proporsi yang relatif masih cukup antara ruang lingkup dan cakupan pengerjaan terhadap batas waktu, serta mahasiswa dapat menunjukkan kemampuannya dalam menggunakan ilmu dan keahlian yang telah dipelajari di kelas.

5 Daftar Pustaka

[1] H. Sandro, M. Relita, S. Rista. 2010. Membangun Aplikasi dengan Konsep Software as a Service (SaaS) pada Sistem Billing Rumah Sakit. Dokumen Tugas Akhir D3 Politeknik Informatika Del, Sitoluama.

[2] Building Distributed Applications: Architecture Strategies for Catching the Long Tail. Accessed on November 16th, 2009 from http://msdn.microsoft.com/en-us/library/aa479069.aspx.

[3] The Overlapping Worlds of SaaS and SOA. Accessed on June 2nd, 2010 from http://cloudcomputing.sys-con.com/?q=node/1047073.

[4] SaaS-TCO, How Web-hosted Software-as-a-Service (SaaS) Lowers the Total Cost of Ownership(TCO) for Electronic Access Control Systems. Accessed on June 1st, 2010 from http://www.brivo.com/user_data/white_papers/12634870 43_mkt-doc-113-tcowhitepaper.pdf.

[5] Software as a Service: Strategic Backgrounder. 2001. Software & Information Industry Association. Taken from http://www.siia.net/estore/pubs/SSB-01.pdf on June 7th, 2010.

[6] Gold, Nicolas, et.al; *Understanding Service Oriented Software; IEEE Computer Society*, 2007.

[7] Latha Srinivasan, Jem Treadwell. *An Overview of Service-Oriented Architecture, Web Service and Grid Computing*, 2005

[8]Politeknik Informatika Del. Accessed on June 12th, 2010 from http://www.pidel.org.

[9] ASP.NET MVC Overview (C#). Accessed on June 10th, 2010 from http://www.asp.net/mvc/tutorials/asp-net-mvc-overview--cs.

[10] Testing Overview and Black-Box Testing Techniques. Accessed on June 12th, 2010 from http://agile.csc.ncsu.edu/SEMaterials/BlackBox.pdf.

[11] Bab I Bidang Ilmu Informatika-Komputer. Accessed on June 12th, 2010 from http://elista.akprind.ac.id/staff/catur/SIJK/Pengantar%20Teknologi%20Informasi%20PDF/Bab_01_Bidang_Ilmu_Informatika_Komp.pdf.