IMPLEMENTASI *REQUIREMENT TRACEABILITY MATRIX* PADA SISTEM INFORMASI AKADEMIK UNIVERSITAS NEGERI JAKARTA

Skripsi

SITI NUR FITRIANI

5235134397

Skripsi ini Ditulis untuk Memenuhi Sebagian Persyaratan dalam
Memperoleh Gelar Sarjana

PROGRAM STUDI PENDIDIKAN TEKNIK INFORMATIKA DAN KOMPUTER

FAKULTAS TEKNIK

UNIVERSITAS NEGERI JAKARTA

2017

HALAMAN PENGESAHAN

NAMA DOSEN	TANDA TANGAN	TANGGAL
Widodo, M.Kom (Dosen Pembimbing I)	.....	14-08-17
Hamidillah Ajie, S.Si., M.T. (Dosen Pembimbing II)	.....	14-08-2017

PENGESAHAN PANITIA UJIAN SKRIPSI

NAMA DOSEN	TANDA TANGAN	TANGGAL
Bambang P. Adhi, M.Kom (Ketua Penguji)	.....	14-08-2017
Vina Oktaviani, M.T. (Sekretaris Penguji)	.....	14-08-2017
Fandy Septia Anggriawan, M.Pd.T (Dosen Penguji Ahli)	.....	14/08 17

HALAMAN PERNYATAAN

Dengan ini saya menyatakan bahwa:

1. Karya tulis skripsi saya ini adalah asli dan belum pernah diajukan untuk mendapat gelar sarjana, baik di Universitas Negeri Jakarta atau di perguruan tinggi lain;
2. Karya tulis skripsi saya ini adalah murni gagasan, rumusan, dan penelitian saya sendiri dengan mendapatkan arahan dari dosen pembimbing;
3. Tidak terdapat karya atau pendapat yang telah ditulis atau dipublikasikan oleh orang lain di dalam karya tulis skripsi saya ini, kecuali secara tertulis dengan jelas dicantumkan sebagai acuan dalam naskah dengan disebutkan nama pengarang dan dicantumkan di dalam daftar pustaka;
4. Pernyataan ini saya buat dengan sesungguhnya dan apabila di kemudian hari terdapat penyimpangan dan ketidakbenaran dalam pernyataan ini, maka saya bersedia menerima sanksi akademik berupa pencabutan gelar yang telah diperoleh karena karya tulis ini, serta sanksi lainnya sesuai dengan norma yang berlaku di Universitas Negeri Jakarta.

Jakarta, 31 Juli 2017

Yang Membuat Pernyataan

Siti Nur Fitriani

5235134397

ABSTRAK

SITI NUR FITRIANI, Implementasi *Requirement Traceability Matrix* Pada Sistem Informasi Akademik Universitas Negeri Jakarta. Pembimbing Widodo, M.Kom dan Hamidillah Adjie, S.Si, M.T.

Setiap sistem dibuat sesuai dengan kebutuhan atau *requirements* dari *user*, *clients*, *stakeholder*, dan pihak yang terlibat dalam pembuatan sistem. Sejumlah *requirements* dapat memiliki keterkaitan atau relasi dengan *requirements* lainnya. Relasi yang terbentuk bisa dikarenakan kemiripan kebutuhan yang dikelompokkan menjadi fungsi yang saling terkait dalam Siakad. Karena kebutuhan Siakad yang beragam, pengelompokkan fungsi yang ada membuat Siakad mengharuskan menggunakan sistem modul. Modul biasanya terdiri dari beberapa bagian kecil yang menangani kebutuhan tertentu pada suatu sistem. Untuk mengetahui keterkaitan pada *requirements*, haruslah dilakukan *tracing* pada sistem. Salah satu metode yang digunakan untuk melakukan *tracing* pada *requirements* dalam sistem adalah dengan menggunakan metode *Requirement Traceability*. Ada beberapa model yang digunakan dalam metode *Requirement Traceability*, salah satunya yaitu dengan menggunakan model *Requirement Traceability Matrix*. *Requirement Traceability Matrix* mempermudah dalam menunjukkan proses analisis (*forward* dan *reverse*). Hasil penelitian yang didapatkan yaitu *Requirement Traceability Matrix* bisa digunakan pada modul dosen siakad unj, terdapat beberapa *requirement* yang bisa dijadikan satu *requirement*, ada pula beberapa functional area yang bisa dijadikan satu *functional* area karena lebih efisien, dan *requirement* yang dihasilkan bisa dijadikan dokumentasi sistem untuk siakad.

Kata Kunci: *Requirements*, Modul Siakad, *Requirement Traceability Matrix*, Dokumentasi Sistem

ABSTRACT

SITI NUR FITRIANI, The Implementation Of Requirement Traceability Matrix For The Academic Information System State University Of Jakarta. Supervisor Widodo, M.Kom and Hamidillah Adjie, S.Si, M.T.

Each system created with requirements from user, clients, stakeholder, and other party. Some requirements can have relationship with other requirements. The relationship can cause by the matched of requirements which categorized to interrelated function in Siakad. Categorization of Siakad function use module system. Module generally consist of many small requirements in one system. To know relationship in requirements, tracing must used by system. Requirement Traceability is one of many method that can be use for tracing system requirements. Requirement Traceability Matrix (RTM) is one of may model that can be use for Requirement Traceability Method. RTM is useful to represent the analyze process (Forward or Reverse). The Result of research are RTM can be use for Siakad UNJ Lecturer Module, there are many requirement that can be unite, there are many functional area that can be unite for efficiency, and the result of requirement can be documentation for Siakad UNJ.

Keyword: Requirements, Modul Siakad, Requirement Traceability Matrix, Dokumentasi Sistem

KATA PENGANTAR

Puji dan syukur penulis panjatkan kehadirat Allah SWT yang telah melimpahkan rahmat kesehatan dan kemudahan kepada penulis dalam kelancaran menyelesaikan skripsi dengan judul "**Implementasi *Requirement Traceability Matrix* Pada Sistem Informasi Akademik Universitas Negeri Jakarta**" merupakan salah satu syarat untuk memperoleh gelar Sarjana pada Program Studi Teknik Informatika dan Komputer di Universitas Negeri Jakarta.

Selama proses pengerjaan skripsi ini, penulis tidak lepas dari bimbingan, bantuan serta dukungan dari berbagai pihak. Oleh karena itu pada kesempatan ini, penulis ingin menyampaikan terimakasih kepada :

1. Dr. Yuliatri Sastrawijaya, M.Pd., selaku Kepala Program Studi Pendidikan Teknik Informatika dan Komputer (PTIK) Fakultas Teknik Universitas Negeri Jakarta;
2. Bapak Prasetyo Wibowo Yunanto S.T., M.Eng. selaku pembimbing akademik;
3. Bapak Widodo, M.Kom., selaku Dosen Pembimbing I yang selalu meluangkan waktu untuk membimbing, memberikan saran dan motivasi kepada penulis;
4. Bapak Hamidillah Adjie, S.Si., M.T., selaku dosen Pembimbing II yang selalu meluangkan waktu untuk membimbing, memberikan saran dan motivasi kepada penulis;
5. Bapak Muhammad Ficky Duskarnaen, S.T,M.Sc., selaku ketua Unit Pelayanan Teknis – Teknologi Informasi dan Komunikasi (UPT TIK) Universitas Negeri Jakarta beserta staff;
6. Ibu Dr. Ir. Dra. Erda Kamaruddin, M.Pd. yang telah memberikan saran dan masukan kepada penelitian penulis;
7. Mamah, Mama, Bapa, Papah, kakak dan adik – adik yang selalu mendoakan dan mendukung keberhasilan pendidikan penulis;
8. Moctar Afendi, yang selalu mendoakan, mendukung dan meluangkan waktu untuk membantu penulis menyelesaikan pendidikan;
9. Teman – teman PTIK 2013 yang menjadi teman seperjuangan penulis selama menempuh pendidikan di PTIK UNJ terkhusus kepada Dios Athina Malonda, Dian Kurnia Setia Putri, Febrina Ramadani dan Gemawati Putri;
10. Lusyane Margaretha, Yaumadina Larasati, Lestari Nurreta Hartanti dan Muhammad Yoga Arifianto selaku teman dari masa SMA hingga sekarang yang selalu bisa diandalkan untuk berbagi macam hal; dan
11. Semua pihak yang terlibat secara langsung atau tidak langsung dan tidak bisa penulis sebutkan seluruhnya.

Penulis menyadari bahwa skripsi masih jauh dari kata sempurna, oleh karena itu penulis mohon maaf bila masih terdapat kekurangan. Semoga penelitian ini bermanfaat baik bagi pembaca atau bagi semua pihak yang membutuhkan acuan untuk penelitian sejenis.

Jakarta, 31 Juli 2017

Siti Nur Fitriani

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

Gambar Halaman

DAFTAR LAMPIRAN

BAB I

PENDAHULUAN

1.1. Latar Belakang Masalah

Teknologi saat ini digunakan untuk hampir setiap kegiatan di berbagai lapisan masyarakat bahkan dalam kegiatan di suatu perusahaan atau instansi tertentu. Teknologi juga diperlukan di dunia pendidikan, khususnya di tingkat perguruan tinggi. Teknologi yang dibutuhkan berupa sebuah sistem guna mengelola kegiatan perkuliahan di perguruan tinggi agar lebih terorganisir, efektif dan efisien. Sistem tersebut haruslah bersifat *online* dan *mobile*, agar bisa diakses dimana saja dan kapan saja secara *real time* oleh segenap *civitas academica*. Di Universitas Negeri Jakarta (UNJ), telah dikembangkan sistem tersebut yang dinamakan Sistem Informasi Akademik (Siakad). Siakad memuat berbagai data kegiatan perkuliahan, seperti pengisian data diri mahasiswa atau dosen, pengisian jadwal kuliah, pengisian nilai, dan lain sebagainya.

Setiap sistem dibuat sesuai dengan kebutuhan atau *requirements* dari *user*, *clients*, *stakeholder*, dan pihak yang terlibat dalam pembuatan sistem. Namun seiring bertambahnya waktu dan perubahan pada lingkungan, *requirements* pada suatu sistem yang sudah ada bisa saja berubah. Terdapat kemungkinan fungsi yang tadinya memang sudah ada pada sistem harus ditambahkan atau bisa juga dihilangkan, bahkan bisa terdapat fungsi yang seharusnya ada tetapi faktanya tidak ada pada sistem. Untuk mengetahui perubahan pada *requirements*, selain menanyakan langsung ke *user* dan berbagai pihak yang terlibat pada sistem, pengembang bisa juga melakukan pendekatan pakar atau *expert judgement* dari

sistem perangkat lunak dengan memperlihatkan sistem yang ingin dicari perubahannya ataupun langsung melihat dokumentasi dari sistemnya. Siakad saat ini tidak memiliki dokumentasi yang terstruktur. Tentu saja itu membuat Siakad tidak mudah untuk dikembangkan jika sewaktu – waktu ingin dilakukan perubahan. Perubahan pada suatu sistem memang harus dilakukan terutama pada sisi *requirements* atau fungsi apabila sistem sudah digunakan dalam waktu yang lama tanpa adanya perubahan.

Sejumlah *requirements* dapat memiliki keterkaitan atau relasi dengan *requirements* lainnya. Relasi yang terbentuk bisa dikarenakan kemiripan kebutuhan yang dikelompokkan menjadi fungsi yang saling terkait dalam Siakad. Karena kebutuhan Siakad yang beragam, pengelompokkan fungsi yang ada membuat Siakad mengharuskan menggunakan sistem modul. Modul biasanya terdiri dari beberapa bagian kecil yang menangani kebutuhan tertentu pada suatu sistem.

Untuk mengetahui keterkaitan pada *requirements*, haruslah dilakukan *tracing* pada sistem. Salah satu metode yang digunakan untuk melakukan tracing pada *requirements* dalam sistem adalah dengan menggunakan metode *Requirement Traceability*.

Menurut Gotel O.C.Z. dan Finkelstein A.C.W. (1994: 1), *Requirement Traceability* adalah kemampuan untuk menggambarkan dan mengikuti alur kehidupan *requirements* baik dengan alur mundur atau dengan alur maju. Penelusuran *requirements* juga dapat memberikan informasi mengenai hubungan antar *requirements*, yaitu *requirements* yang memiliki ciri yang serupa dengan *requirements* lainnya. Selain itu, dengan penelusuran *requirements* kita bisa

mengetahui kesalahan atau ketidaksesuaian *requirements* dan juga antar hubungan *requirements.*

Ada beberapa model yang digunakan dalam metode *Requirement Traceability*, salah satunya yaitu dengan menggunakan model *Requirement Traceability Matrix*. *Requirement Traceability Matrix* mempermudah dalam menunjukkan proses analisis (*forward* dan *reverse*). Model *Requirement Traceability Matrix* merupakan tabel *testing* untuk proses mendokumentasikan hubungan antar *requirements* dan memverifikasi kesesuaian *requirements*. Menurut Madan, dkk. (2016: 1), kelebihan dari *Requirement Traceability Matrix*, yaitu:

1. Melakukan cek pada *requirements* sistem perangkat lunak dan penghubung saat dilakukan *test case*;
2. *Requirement Traceability Matrix* digunakan untuk menemukan *requirements* yang hilang;
3. *Requirement Traceability Matrix* membantu proses *test case* setiap terjadinya perubahan pada *requirements*.

Penelitian ini, menunjukkan bagaimana implementasi model *requirement traceability matrix* pada Siakad UNJ. Karena selain menghasilkan format *requirement traceability matrix* untuk modul dosen Siakad, hasil penelusuran *requirements* juga nantinya bisa digunakan untuk menjadi pertimbanngan pada pengembangan Siakad dimasa mendatang.

Berdasarkan latar belakang yang telah dijelaskan, maka penelitian ini diberi judul "Implementasi *Requirement Traceability Matrix* Pada Sistem Informasi Akademik Universitas Negeri Jakarta".

1.2. Identifikasi Masalah

Berdasarkan latar belakang yang telah dijelaskan, identifikasi masalah pada penelitian ini adalah :

1. Perubahan pada lingkungan seperti pergantian pimpiman UNJ, menyebabkan Siakad UNJ senantiasa mengalami pengembangan;
2. Siakad UNJ tidak memiliki dokumentasi terstruktur yang menyebabkan perubahan dapat terjadi kapan saja;
3. Perlu dilakukan penelusuran *requirements* pada Siakad UNJ dalam rangka melakukan proses pengembangan yang lebih baik.

1.3. Batasan Masalah

Batasan masalah dalam penelitian ini adalah :

1. Melihat luasnya lingkup Siakad, objek dari penelitian ini adalah Modul Dosen dari Siakad UNJ;
2. Penelitian ini dilakukan pada *requirements* dari Modul Dosen Siakad UNJ;
3. Penelitian ini menggunakan model *Requirement Traceability Matrix*.

1.4. Rumusan Masalah

Berdasarkan uraian proses latar belakang, identifikasi, dan pembatasan masalah maka rumusan masalah yang akan dibahas pada penelitian ini adalah:

“Bagaimana Implementasi *Requirement Traceability Matrix* Pada Modul Dosen Sistem Informasi Akademik Universitas Negeri Jakarta?“

1.5. Tujuan Penelitian

Tujuan dari penelitian ini adalah untuk melakukan implementasi *Requirement Traceability Matrix* pada modul dosen Siakad UNJ.

1.6. Manfaat Penelitian

1. Hasil *requirement traceability matrix* yang digunakan bisa menjadi acuan pengembang sistem perangkat lunak Siakad UNJ untuk melakukan pengembangan;
2. Menghasilkan dokumentasi mengenai *requirements* pada Modul Dosen Siakad UNJ;
3. Hasil analisis dari *Requirement Traceability Matrix* yang digunakan bisa dijadikan acuan bagi pengembang sistem perangkat lunak lain yang ingin melakukan pengembangan pada sistem perangkat lunak yang dikelolanya.

BAB II

TINJAUAN PUSTAKA

2.1. Kerangka Teoritik

2.1.1. Sistem Informasi Akademik (Siakad) UNJ

Siakad merupakan sebuah sistem informasi yang dimiliki UNJ dengan basis web. Siakad mengelola segala kebutuhan akademik di Universitas Negeri Jakarta. Sistem ini menyimpan seluruh informasi akademik mahasiswa aktif dan non-aktif termasuk data mahasiswa yang telah lulus mulai dari pendaftaran Kartu Rencana Studi (KRS), pembayaran, jadwal kuliah, daftar hasil studi, dan lain sebagainya.

Sistem informasi ini mengelola seluruh data akademik secara terintegritas sehingga setiap perubahan yang terjadi akan selalu berpengaruh terhadap data lainnya secara langsung. Sebagai contoh, jika seorang mahasiswa belum melakukan pembayaran semester, maka mahasiswa tersebut tidak bisa melakukan pengisian KRS. Namun, apabila mahasiswa sudah melakukan pembayaran semester sesuai dengan tagihan yang ditentukan, maka mahasiswa sudah bisa melakukan pengisian KRS tanpa harus ada verifikasi manual oleh *administrator*.

Siakad juga menjadi pusat informasi utama bagi universitas, karena sistem ini merekam setiap kegiatan akademik yang terjadi di universitas. Pimpinan universitas atau *administrator* dapat mengambil data rekapitulasi akademik dengan mudah melalui Siakad. Sehingga, data tersimpan secara rapi dan terstruktur tanpa memerlukan ruang yang besar untuk menyimpannya dan waktu yang lama untuk menggunakannya.

Gambar 2.1 Tampilan awal Siakad UNJ

Gambar 2.1 menampilkan halaman awal dari Siakad UNJ. Informasi yang ditampillkan oleh Siakad ada yang bisa diakses tanpa harus *login* terlebih dahulu, ada juga informasi yang bisa dilihat jika mahasiswa sudah *login* di Siakad.

2.1.1.1. Modul

Dalam sebuah sistem, biasanya terdiri dari beberapa bagian kecil yang menangani kebutuhan tertentu. Hal ini dinamakan dengan modul. Sama halnya seperti sistem lainnya, Siakad juga memiliki modul – modul di dalamnya. Adapun modul utama yang terdapat pada Siakad adalah modul mahasiswa dan modul dosen.

2.1.1.2. Modul Mahasiswa

Mahasiswa merupakan salah satu pengguna yang menggunakan Siakad secara intensif. Setiap kegiatan akademik mahasiswa terekam jelas pada Siakad. Fungsi – fungsi yang termasuk dalam modul mahasiswa, yaitu: 1) Melakukan pengelolaan biodata mahasiswa; 2) Melakukan perubahan kata sandi; 3) Melakukan pengisian KRS pada awal semester; 4) Melihat dan mengunduh KRS dan Kartu Hasil Studi (KHS) pada masing-masing semester; 5) Melihat Daftar Hasil Studi (DHS) yang merupakan rekapitulasi dari seluruh aktifias akademik yang sudah dikerjakan selama mahasiswa kuliah di universitas; 6) Mencetak KRS setiap semester; 7) Mencetak KHS setiap semester; 8) Melihat jadwal kuliah yang berlaku; 9) Melihat status pembayaran pada masing-masing semester; 10) Melakukan pendaftaran Praktik Kerja Lapangan (PKL); 11) Melakukan pendaftaran Praktik Kerja Mengajar (PKM); 12) Melakukan pendaftaran cuti kuliah; 13) Melakukan pengisian evaluasi perkuliah di setiap akhir semester.

2.1.1.3. Modul Dosen

Selain mahasiswa, dosen juga merupakan pengguna intensif dari Siakad, karena tanpa adanya dosen maka perkuliahan tidak akan berlangsung. Fungsi – fungsi yang termasuk dalam modul dosen, yaitu: 1) Melihat jadwal mengajar; 2) Melakukan pengisian nilai untuk masing-masing mahasiswa pada masing-masing kode seksi; 3) Melihat hasil evaluasi perkuliahan yang diisi oleh mahasiswa; 4) Melihat jumlah sks yang diizinkan untuk mengajar; dan sebagainya.

2.1.2. *Requirements Traceability Matrix*

2.1.2.1. Definisi *Requirements*

Menurut Martin, C. & Dorf, M. (1990), *requirements* adalah sebuah kemampuan yang harus dimiliki dari suatu *software* atau perangkat lunak. Kemampuan ini bisa ditujukan sebagai pemecah masalah pembuatan sistem perangkat lunak atau sebagai kemampuan untuk memenuhi kebutuhan dalam pembuatan sistem perangkat lunak.

Menurut IEEE *Standard Glossary of Software Engineering Technology* (1990 : 62), *requirements* dapat diartikan sebagai berikut: 1) Suatu kondisi atau kemampuan yang diperlukan oleh *user* untuk memecahkan masalah atau mencapai tujuan; 2) Suatu kondisi atau kemampuan yang harus dipenuhi atau dimiliki oleh sistem atau komponen sistem untuk memenuhi kontrak, *standard*, spesifikasi atau dokumen formal lain; 3) Gambaran yang terdokumentasi dari kondisi atau kemampuan yang disebut pada 1 dan 2.

Ada beberapa faktor yang harus ada dalam sebuah *requirements*, menurut Wieringa (1995 : 4) faktor-faktor tersebut adalah : (1) *requirements* harus menggambarkan tujuan yang harus dipenuhi oleh sistem, dan (2) *requirements* yang dibuat untuk lebih mudah di observasi. Menurut Wieringa (1995 : 4), tujuan yang digambarkan dalam *requirements* adalah berupa spesifikasi dari sistem yang harus ada, yang nantinya tujuan tersebut dijadikan fungsi dalam sistem. Fungsi – fungsi yang terkumpul yang menjadi sarana interaksi antara sistem dan *user*.

Dalam siklus hidup sistem perangkat lunak, *requirements* digunakan sebagai sarana komunikasi antara beberapa *stakeholder* yang berbeda. Berikut ini pihak – pihak yang terlibat dalam pembuatan *requirements*, seperti yang

diungkapkan oleh Wieringa (1995 : 2) , yaitu: 1) *Developer* dan *client* menjadikan *requirements* sebagai fungsi yang akan dibuat dalam sistem perangkat lunak; 2) *Client* menggunakan *requirements* untuk memperoleh *acceptance test* untuk sistem perangkat lunak yang dibuat; 3) *Developer* yang berbeda menggunakan *requirements* sebagai acuan untuk *interface* di bagian yang mereka kerjakan; 5) *Maintenance staff* menggunakan *requirements* sebagai acuan untuk memperbaiki kerusakan dalam sistem perangkat lunak; 6) *Client* dan *maintenance staff* menggunakan *requirements* untuk mengettahui *requirements* mana yang akan dirubah dengan mengikuti kebutuhan dari user atau pasar.

Berdasarkan pendapat para ahli mengenai gambaran *requirements*, maka bisa disimpulkan bahwa r*equirements* merupakan gambaran kebutuhan *user* dari sistem perangkat lunak yang harus dipenuhi oleh *developer*. Misalnya, *user* bisa melakukan operasi tertentu dalam sistem. Seperti, *user* dapat melihat informasi, melakukan aktifitas *login* dan lain sebagainya sesuai dengan tujuan dari pembuatan sistem. *Requirements* juga merupakan batasan dalam suatu sistem perangkat lunak. *Requirements* bisa saja berubah sewaktu-waktu karena perbedaan pendapat dari para *stakeholder*, kebutuhan *user* atau pasar. Untuk itu, *requirements* harus terus di *update* untuk merespon perubahan yang terjadi pada sistem agar sistem bisa terus berjalan sesuai target yang diinginkan.

2.1.2.2. Klasifikasi *Requirements*

Requirements pada suatu sistem perangkat lunak tidak hanya terdapat satu macam yang nantinya menjadi fungsi dari suatu sistem, akan tetapi terdapat dua macam *requirements* yang mendukung berjalannya suatu sistem perangkat lunak. Dua macam *requirements* tersebut menurut Sommerville (2010: 83), yaitu:

(1) *User Requirements*

User requirements merupakan gambaran pelayanan yang disediakan untuk *user*. *User requirements* digambarkan dengan menggunakan *natural language*, diagram, dan notasi lain yang dapat dimengerti oleh *user*.

(2) *System Requirements*

System requirements merupakan dokumen yang mencantumkan detail fungsi dari sistem perangkat lunak, layanan dan dan cara kerja sistem perangkat lunak. Menjelaskan apa yang harus diterapkan, kemungkinan sudah menjadi bagian dari persetujuan antara *client* dengan *developer*. *System requirements* dibedakan menjadi dua, yaitu:

a. *Functional Requirements*

Functional Requirements adalah pernyataan dari layanan yang sistem perangkat lunak tersebut bisa lakukan. Menjelaskan bagaimana sistem bisa merespon terphadap input tertentu dan bagaimana sistem harus bereaksi terhadap situasi tertentu.

b. *Non – functional Requirements*

Non – functional requirements merupakan kendala pada layanan atau fungsi dari sistem seperti kendala waktu, kendala saat proses pengembangan, standarisasi, dan lain sebagainya.

2.1.2.3. *Requirements Engineering*

Requirements engineering adalah fase paling awal dari proses rekayasa perangkat lunak (*software engineering*), dimana kebutuhan dari *user* dan *customer* dikumpulkan, dipahami dan ditetapkan. Banyak kegagalan pengembangan *software* disebabkan karena adaya ketidakkonsistenan (*inconsistent*), ketidaklengkapan

(*incomplete*), maupun ketidakbenaran (*incorrect*) dari *requirements specification* (spesifikasi kebutuhan).

Menurut Thayer, R.H. and Dorfman, M. (1997) *requirement engineering* menyediakan mekanisme yang tepat untuk memahami apa yang *costumer* inginkan, analisis kebutuhan, uji kelayakan, menegosiasikan solusi yang masuk akal, menentukan solusinya dengan jelas, memvalidasi spesifikasi, dan mengelola *requirements* saat diubah menjadi operasi pada sistem.

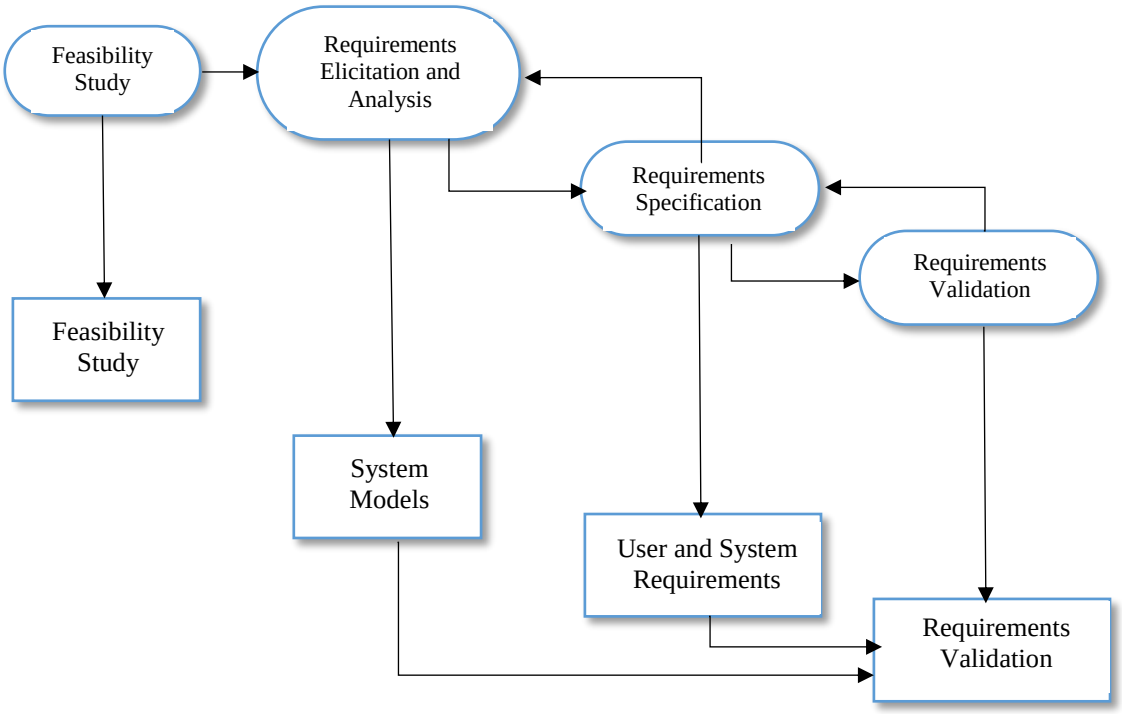
Ada pula pendapat lain menurut Sommerville (2010 : 36), *requirements engineering* merupakan proses pemahaman dan penjelasan apa layanan yang dibutuhkan dari sistem dan mengidentifikasi kendala dari pengoperasian dan pengembangan sistem.

Berdasarkan penjelasan mengenai *requirements engineering*, maka bisa disimpulkan bahwa *Requirements engineering* adalah cabang dari *software engineering* yang membahas masalah yang berhubungan dengan: tujuan, fungsi, dan batasan-batasan pada sistem perangkat lunak. *Requirements engineering* merupakan proses pengumpulan *requirements* dari *user* dan *customer* yang dipahami lebih dalam untuk ditetapkan sebagai fungsi yang akan diterapkan pada sistem.

Dalam tahap pengumpulan *requirements*, komunikasi secara berulang dengan sumber informasi seperti *customer*, *user*, dan *stakeholder* sangat perlu dilakukan. Bisa dengan dilakukannya wawancara mengenai *requirements* yang dibutuhkan hingga mendapatkan kesepakatan mengenai sistem yang ingin dibuat.

2.1.2.4. Proses *Requirements Engineering*

Requirements engineering dilakukan dengan beberapa proses, seperti yang ditunjukan oleh Gambar 2.2 berikut:



Gambar 2.2 Proses *Requirements Engineering*

Sumber : *Software Engineering* oleh Sommervile. (2010 : 38)

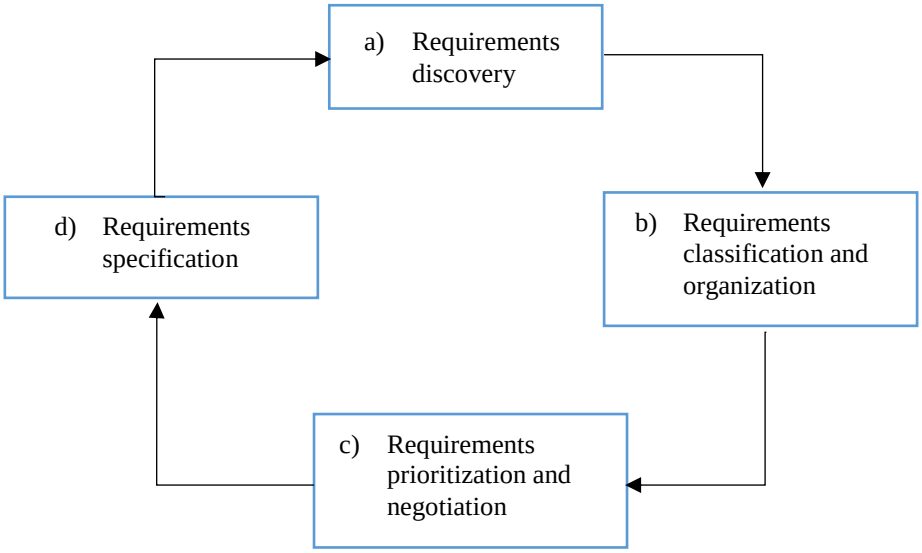
Gambar 2.2 menunjukan proses dari *requirements engineering* menurut Sommerville (2010 : 38), proses-proses tersebut yaitu :

(1) *Feasibility study*

Tahap *feasibility study* merupakan tahapan paling awal dalam proses *requirement engineering*. Tahap ini merupakan penilaian pada kebutuhan sistem yang berguna bagi bisnis. Seperti penerapan sistem dengan menggunakan teknologi yang ada saat ini, kesesuaian *budget*, integrasi sistem dengan sisem lain dalam perusahaan, dan lain sebagainya.

(2) *Requirements elicitation and analysis*

Tahapan pengumpulan kebutuhan apa saja yang di perlukan, serta kendala dalam pembuatan sistem. Dalam tahap ini, *developer* berinteraksi dengan *client* dan juga *system end-user*.



Gambar 2.3 Proses *requirements elicitation and analysis*

Sumber : *Software Engineering* oleh Sommervile. (2010: 101)

Gambar 2.3 menunjukan alur dari tahapan *requirements elicitation and analysis processes*, yaitu: a) *Requirements discovery*, *developer* mengumpulkan *requirements* dari *stakeholder;* b) *Requirementss classification and organization*, melakukan pengelompokan *requirements* secara tidak terstruktur, seperti pengelompokkan berdasarkan *requirements* yang berhubungan fungsinya; c) *Requirements prioritization and negotiation*, saat beberapa *stakeholder* terlibat, maka akan terjadi konflik pada *requirements* karena pemikiran yang berbeda. *Developer* dituntut untuk memprioritaskan *requirements* tertentu yang harus disetujui oleh

semua *stakeholder*; d) *Requirements spesification, s*emua *requirements* yang telah di kumpulkan dan disepakati didokumentasikan.

(3) *Requirements spesification*

Tahapan tranlasi dari informasi yang telah dikumpulkan saat proses *analysis* menjadi dokumen yang mendefinisikan *requirements.* Terdapat dua jenis *requirements* yang dihasilkan, yaitu: *user requirements* dan *system requirements.*

(4) *Requirements validation*

Tahapan pengecekan kebutuhan kepada kenyataannya di sistem, konsistensi dan kelengkapan *requirements.* Tahapan yang memastikan bahwa sistem sudah sesuai dengan keinginan *client*.

2.1.2.5. Definisi *Requirements Management*

Requirements pada suatu sistem perangkat lunak yang besar selalu berubah. satu alasan untuk perubahan *requirements* pada sistem peragkat lunak yang besar yaitu karena biasanya saat pengembangan ada masalah yang tidak dapat seluruhnya dipecahkan. Karena sejatinya masalah tersebut tidak akan seluruhnya selesai, bisa dipastikan akan ada masalah lain yang muncul. Setiap ada perubahan sistem dalam lingkungan dan diberlakukan, maka akan muncul *requirements* baru nantinya. Oleh karena itu, perlu adanya proses untuk mengatur, mengontrol dan menulusuri perubahan yang terjadi pada *requirements* yang disebut *requirement management.*

Menurut Sommerville (2010 : 111) *requirements management* merupakan proses pemahaman dan mengontrol perubahan *requirements*. *Requirements* harus terus di *tracing* satu per satu dan mempertahankan relasi diantara *requirements* yang saling bergantung agar bisa mengurangi efek dari perubahan *requirements*.

Menutrut Pressman (2010 : 124) *requirements management* adalah aktifitas yang membantu tim dalam suatu proyek perangkat lunak untuk mengidentifikasi, mengontrol, dan menelusuri *requirements* dan perubahan pada *requirements* setiap saat selama proyek dikerjakan.

Berdasarkan pendapat ahli yang telah disebutkan, perubahan *requirements* dalam suatu sistem perangkat lunak bisa saja terjadi. Baik saat proses pembuatan sistem, bahkan saat sistem perangkat lunak telah dibuat atau telah digunakan. Perubahan yang terjadi bisa bermacam – macam, yang paling memengaruhi yaitu bila terjadi perubahan pada lingkungan organisasi instansi atau perusahaan pemilik sistem. Untuk itu diperlukan proses dimana *requirements* yang telah berubah diatur sedemikian rupa agar tidak mengganggu *life cycle* dari sistem, terutama jika sistem tersebut merupakan bagian penting dalam aktivitas di instansi atau perusahaan tersebut. Proses yang bisa mengatur perubahan *requirements* tersebut yaitu proses *requirements management*. Mengatur *requirements* bisa dilakukan dengan berbagai cara, salah satu diantaranya adalah dengan melakukan *tracing* pada sistem.

2.1.2.6. Definisi *Requirements Traceability*

Requirements Traceability mengacu pada kemampuan untuk menggambarkan dan mengikuti alur hidup dari *requirements*, dari arah belakang atau dari arah depan (dari asalnya, sampai ke pengembangan dan spesifikasinya, berikut penyebaran dan kegunaannya, sampai pada berjalannya perbaikan dan perulangan di setiap fase (Gotel and Finkelstein, 1994 diacu dalam Pinheiro. F, (2004 : 2).

Requirements Traceability menurut Pinherio (1995: 93) mengacu pada kemampuan untuk mendefinisikan, menangkap, dan mengikuti jejak yang

ditinggalkan oleh *requirements* pada unsur-unsur lain di saat pengembangan sistem perengkat lunak dan jejak yang ditinggalkan oleh unsur – unsur dalam *requirements*. Menurut penjelasan dari Pinherio, lingkungan sekitar sistem perangkat lunak yang di maksud tidak hanya lingkungan *technical* saja, tetapi terdapat aspek sosial yang ikut terlibat dalam pengembangan sistem perangkat lunak. Lingkungan sosial tersebut seperti orang-orang, aturan, keputusan dan bahkan sampai pada konsep dan tujuan dari sistem perangkat lunak yang dibangun.

Requirement Traceability merupakan salah satu bagian dari *requirements management*. *Requirements Traceability* sangat bermanfaat terutama bagi pihak *stakeholders* dan *developer* dalam proses pengembangan sistem. Berikut ini adalah pihak – pihak yang membutuhkan *traceability*, yaitu: 1) *Project Management*, secara umum *tracing* membantu *project management* untuk mengetahui status dari projek yang dikerjakan jika *tracing* dilakukan saat *project* di kerjakan. Manfaatnya untuk *project management* seperti: dapat memperkirakan dampak dari perubahan *requirements*, dapat mengetahui lebih dulu jika ada konflik antar *requirements*, dapat mengestimasi waktu untuk memperbaiki beberapa *requirements* yang kurang memuaskan dan bisa mengurangi waktu dan upaya *maintenance* di masa depan; 2) *Client, tracing* dilakukan untuk mengetahui *requirements* apa saja yang sudah memuaskan *client* dan *test* apa yang harus dilakukan untuk menguatkan keberadaan *requirements* yang lainnya; 3) *Designer, tracing* pada *requirements* harus mendokumentasikan hasil dari *design* setelah dilakukan *tracing*, karena perubahan yang terjadi pada *requirements* bisa mempengaruhi *design* dari sistem; 4) *Maintener, maintener* membutuhkan *tracing* terhadap *requirements* untuk memperhitungkan pengaruh perubahan pada *requirements* ke *requirements* lainnya,

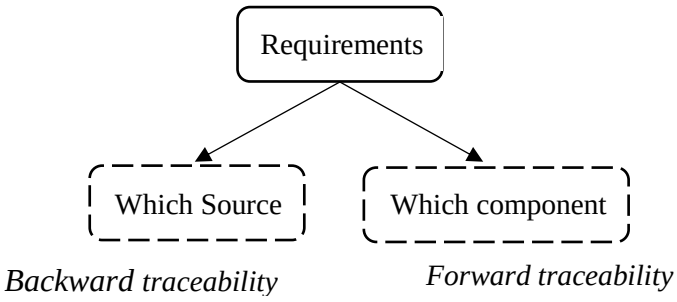
pengaruh perubahan *requirements* pada implementasi atau sistem, dan pengaruh perubahan *requirements* pada *requirementrequirements* yang tidak mengalami perubahan.

Requirement traceability memiliki beberapa manfaat untuk siklus hidup dari sistem perangkat lunak, yaitu: 1) memberikan pernyataan yang kuat berupa dokumentasi bahwa semua *requirements* sudah benar-benar diterapkan pada sistem ; 2) untuk membantu mengidentifikasi efek dari perubahan beberapa *requirements*; 3) untuk membantu mengidentifikasi keterkaitan kerusakan jika terjadi kegagalan atau *error* pada sistem.

Banyak cara bisa dilakukan untuk melakukan *tracing* pada *requirements*. Menurut Pinherio (2004 : 3), yaitu :

(1) Backward and forward traceability

Jika dilihat dari arah *tracing*-nya, *requirements* bisa di-*tracing* dengan *forward* dan *backward traceability*.



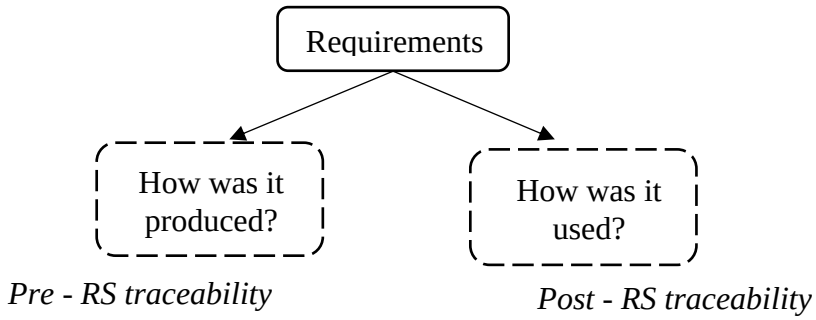
Gambar 2.4 *Backward dan Forward Traceability*

Berdasarkan Gambar 2.4, bisa diketahui bahwa *backward traceability* adalah kemampuan untuk menelusuri *requirement* dari sumber mana (which source) yang berkaitan dengan pembuatan sistem, seperti user, institusi, dan sebagainya. Sedangkan untuk penelusuran dengan cara *forward traceability* dilakukan dengan cara melakukan penelusuran langsung dari implementasi design

atau sistem yang mana (which component). Seperti melakukan penelusuran pada sistem aplikasi yang menjadi implementasi dari sistem tersebut.

(2) Pre and post – requirements spesification traceability

Jika *tracing* dilakukan dengan melihat evolusi dari *requirements* maka *tracing* dilakukan dengan cara *pre-post traceability*.



Gambar 2.5 *Pre and Post – requirements spesification traceability*

Jika mengacu pada Gambar 2.5, maka *pre-requirements traceability* yaitu requirements yang ada menunjukkan bagaimana sistem dibuat. Contohnya adalah saat ada perubahan pada *requirements* dan ingin mengetahui sumber orang yang bertanggung jawab untuk memvalidasi perubahan *requirements* tersebut. Sedangkan p*ost – requirements traceability* melakukan *tracing* berdasarkan fungsi ataupun kegunaan nya.

(3) Inter and extra-requirements traceability

Inter and extra-requirements traceability merupakan tracing requirements yang berkaitan dengan relasi antara *requirements* atau artefak lain. *Inter-requirements traceability* lebih mengacu kepada relasi penelusuran antar *requirements* pada satu artefak, sedangkan *extra-requirements* merupakan penelusuran *requirements* antara *requirements* dengan artefak lain.

2.1.2.7.*Requirements Traceability Matrix*

Menurut IEEE *Standard Glossary of Software Engineering Technology* (1990 : 78), *traceability matrix* adalah dokumentasi dari hubungan antara dua atau lebih produk dari proses pengembangan sistem. Seperti, *matrix* yang mendokumentasikan hubungan antara *requirements* dan *design* dari komponen *software*.

Dari berbagai macam cara dalam melakukan *tracing* pada *requirements*, *requirements traceability matrix* merupakan salah satu dari beberapa *tools* yang digunakan dalam melakukan *tracing requirements*. Keuntungan menggunakan *traceability matrix* diantaranya: 1) Untuk melakukan check terhadap *requirements* pada sistem, lebih baik jika *tracing* dilakukan sebelum sistem dibuat; 2) *Traceability matrix* digunakan untuk menemukan *requirements* yang hilang; 3) *Traceability matrix* membantu menemukan perubahan *requirements* saat *testing* pada sistem.

Tracing dengan *Requirements Traceability Matrix* bisa dilakukan dengan cara manual ataupun dengan cara otomatis. Langkah – langkah yang perlu dilakukan untuk melakukan *Requirements Traceability Matrix*, yaitu: 1) Megumpulkan *requirements* dari sistem yang akan dilakukan *tracing*; 2) Membuat daftar *requirements* dengan memberikan kode kepada setiap requirements yang sudah dikumpulkan; 3) Membuat matriks atau tabel keterkaitan antar *requirements* berupa *requirements traceability matrix*. Tabel berisi kode dari *requirements* di sebelah kiri dan atasnya; 4) Analisis keterkaitan antar *requirements*; 5) Jika terdapat *error* dalam sistem, analisa *requirements* yang terjadi *error*; 6) Prediksi dengan keterkaitan *requirements* yang ada, *requirements* apa lagi yang akan terkena

dampak dari *error* tersebut; 6) Dokumentasikan hasil analisis. Di bawah ini terdapat Tabel 2.1 yang merupakan contoh format dari tabel *Requirements Traceability Matrix*, yaitu:

Tabel 2.1 Contoh format *Requirement Traceability Matrix*

Feature Number	Subject Area				Domain Area					Relation_Req_Code			
	Admin	Atasan	Tata Usaha	Pegawai	Login	Data Kehadiran	Status Approval	Data Pembayaran	Pengajuan Lembur	FR001	FR002	FR003	FR004
F001													
F002													
F003													
F004													

Sumber: Model *Requirement Traceability* untuk metode pengembangan perangkat lunak *Feature Driven Development* (DFD) oleh Fildzah Shabrina, S.Pd

Keterangan :

- *Feature Number* : penomoran untuk tiap fitur yang terdapat pada sistem;
- *Subject Area* : *user* atau pengguna yang terlibat pada sistem;
- *Domain Area* : berisi hasil pengkategorian *requirements* berdasarkan kesamaan ciri dan dimensi;
- *Relation Req Code* : keterkaitan antar *requirement* dan *feature.*

Format dari *Requirements Traceability Matrix* yang di tunjukkan pada Tabel 2.1 hanya merupakan satu dari sekian banyak format tabel yang dibuat untuk penggunaan *Requirements Traceability Matrix*. *Requirements Traceability Matrix* tidak memiliki aturan baku dalam penerapan format tabel yang digunakan. Untuk itu, tabel bisa dibuat sedemikian rupa sesuai dengan kebutuhan dari dilakukannya *tracing requirements* pada sistem yang diinginkan.

2.2. Penelitian yang Relevan

Penelitian yang dilakukan penulis mengacu kepada beberapa penelitian lain yang relevan dan telah dilakukan oleh peneliti lain. Penelitian – penelitian yang relevan tersebut, yaitu:

1. Model *Requirement Traceability* untuk Metode Pengembangan Perangkat Lunak *Feature Driven Development* (FDD). Penelitian dilakukan oleh Fildzah Shabrina, S.Pd. yang melakukan penelitian di PT. Gapura Angkasa. Penelitian yang dilakukan adalah untuk membuat model penelusuran *Requirements Traceability Matrix* pada metode FDD. Model penelusuran *Requirements Traceability Matrix* yang tidak memiliki template khusus, memungkinkan untuk dibuat menjadi berbagai model penelusuran yang lebih efektik. Penelitian dibuat dengan menggunakan aplikasi sistem absensi PT. Gapura Angkasa. Model penelusuran *Requirements Traceability Matrix* berhasil diterapkan pada metode FDD. Model yang dibuat menghasilkan beberapa tahapan dalam penerapan model penelusuran *requirements* pada FDD. *Requirements* yang terdapat pada sistem dibuat menjadi general berdasarkan keterkaitan antar *requirements*. Dari hasil generalisasi, kemudian dijadikan *feature list*. *Feature list* ini lah yang menjadi bagian dari FDD yang diterapkan. Hasilnya, pembuatan model penelusuran *requirements* pada metode FDD dapat memudahkan penelusuran jika terjadi kesalahan dalam *testing* hasil akhir program dan dapat dijadikan sebagai acuan pembuatan fungsi dalam pengembangan proyek selanjutnya.
2. Model *Requirement Traceability* untuk Metodologi Pengembangan Perangkat Lunak *Extreme Programming*. Penelitian dilakukan oleh Nadya Fadillah Fidhyallah, S.Pd, penelitian dilakukan di PT. Gapura Angkasa. Penelitian yang dilakukan adalah untuk membuat model

penelusuran *Requirements Traceability Matrix* pada metode *Extreme Programming*. Model penelusuran *requirement traceability matrix* dipilih karena tidak memiliki template khusus, memungkinkan untuk dibuat menjadi berbagai model penelusuran yang lebih efektik termasuk pada metode *extreme programming*. Penelitian dibuat dengan menggunakan aplikasi sistem absensi PT. Gapura Angkasa sebagai objek penelitian. Model penelusuran *Requirements Traceability Matrix* berhasil diterapkan pada metode *extrme programming*. Berdasarkan model yang telah dibuat menghasilkan beberapa tahapan dalam penerapan model penelusuran *requirements* pada *extreme programming*. Tahapan penelitian tersebut, yaitu: 1) Mengumpulkan dokumentasi *requirement engineering;* 2) Mengumpulkan daftar Requirement dari Aplikasi dengan memberikan *Req_code* pada setiap *requirementnya*. Membuat *Requirement Traceability Matrix* berbentuk tabel *checklist* dari *raquirement* yang sudah dibuat sebelumnya; 3) Menggeneralisir *requirement* menjadi daftar *requirement* baru untuk dimasukan kedalam *requirement management* pada *eXtreme Programming* (XP); 4) Merevisi redaksi *requirement* agar menjadi lebih general dan dan membuat *requirement_code* baru yang kemudian digunakan untuk *requirement* XP; 5) Analisis kekuatan keterkatian pada *requirements_updated;* 6) Membuat *Requirement Traceability Matrix* untuk *eXtreme Programming*.

Manfaat dari penelitian yang dilakukan yaitu: 1) Model tidak hanya dapat dikembangkan dengan parameter kekuatan keterkaitan antar-

requirements saja, namun juga dapat dikembangkan degan parameter lain seperti keserasian atau keselarasan satu *requirements* dengan *requirements* yang lain; 2) Model ini dapat dikembangkan untuk metodologi pengembangan perangkat lunak modern yang lain, yang masih dalam lingkup *Agile Method*; 3) Dalam pembuatan *requirement taceability matrix* di fase awal model, tabel dapat dibuat menggunakan model tabel lain, yang terpenting tabel tersebut mengandung kolom keterkaitan antar *requirement*.

3. Analisis dan Perancangan Sistem Informasi Penerimaan Mahasiswa Baru Univeristas Negeri Jakarta. Penelitian dilakukan oleh Hanifa Fissalma, S.Pd, penelitian dilakukan di Universitas Negeri Jakarta (UNJ). Penelitian yang dilakukan adalah menganilisis sistem informasi yang telah berjalan sebelumnya dan merancangan sistem informasi yang baru untuk jalur Penerimaan Mahasiswa Baru (PENMABA) di UNJ. Perancangan yang dilakukan membantu *developer* dalam pembuatan sistem. Perancangan dilakukan dari pengumpulan *requirements* sampai pada dokumentasi *feature* pada sistem. Untuk *testing requirements*, peneliti *menggunakan Requirements Traceability Matrix* yang telah dimodelkan pada metode perangkat FDD. Penelitian dilakukan untuk meminimalisir masalah yang sering dihadapi *developer* sistem pada waktu PENMABA UNJ. Calon mahasiswa di UNJ tidak hanya masuk dari satu jalur pendaftaran saja, oleh karena itu sistem yang ada dibuat terpisah berdasarkan jalur masuk mahasiswa baru. Karena adanya ketidak seragaman data mahasiswa baru, penelitian tersebut merancang

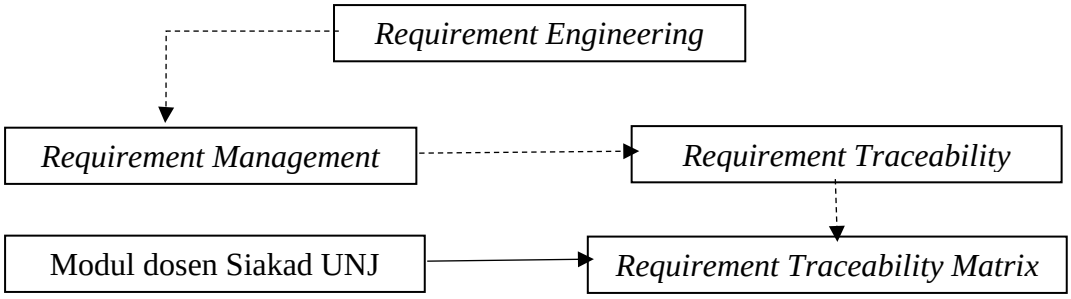
konsep agar ada satu sistem yang bisa terintegerasi pada seluruh sistem penerimaan mahasiswa baru UNJ dari semua jalur dan bisa memproses semua jenjang pendidikan yang ada di UNJ. Selain informasi perkuliahan, sistem juga menyediakan informasi mengenai tahapan pendaftaran sampai pembayaran Uang Kuliah Tunggal (UKT). Namun untuk jalur Mandiri UNJ, informasi hanya dibatasi sampai alur pendaftaran saja. Perancangan masih bersifat konseptual tidak untuk diterapkan pada sistem PENMABA UNJ 2016, namun bisa menjadi acuan untuk pengembangan sistem dimasa mendatang.

Relevan dalam penelitian-penelitian pada point 1-3 terletak pada penggunaan *Requirements Traceability Matrix* sebagai media *testing* pada penelitian yang dilakukan.

2.3. Kerangka Berpikir

Suatu sistem bisa saja berubah seiring berkembangnya kebutuhan *user*, termasuk pada modul dosen Siakad UNJ. Perubahan tersebut bisa mempengaruhi beberapa *requirements*. Untuk itu, penelusuran *requirements* perlu dilakukan secara berkala sebagai bentuk pemeliharaan dan untuk pengembangan pada sistem

Teknik penelusuran bisa dilakukan dengan berbagai cara, salah satunya dengan menngunakan teknik *requirement traceability*. *Requirement traceability* merupakan bagian dari *requirement management* yang merupakan proses untuk mengatur perubahan pada *requirements.* Teknik penelusuran *requirements* yang dilakukan yaitu menggunakan model *Requirements Traceability Matrix*. Gambar 2.6 menunjukkan kerangka berpikir pada penelitian ini:



Gambar 2.6 Kerangka berpikir penelitian

Pada Gambar 2.6, bisa dilihat bahwa bagaimana alur untuk melakukan implementasi *requirement traceability matrix* pada modul dosen Siakad UNJ. Mulai dari pemilihan sistem yang dijadikan objek penelitian, dalam hal ini objek tersebut adalah modul dosen Siakad UNJ hingga menganalisa tabel *requirement traceability matrix.* Proses awal sebelum bisa menerapkan *requirement traceability matrix* pada modul dosen Siakad UNJ yaitu proses pengumpulan *requirements* pada modul dosen Siakad UNJ yang termasuk pada proses *requirement engineering*. Dalam *requirement engineering*, terdapat proses dimana perubahan pada *requirements* diatur dan di kontrol yang disebut dengan proses *requirement management*. Cara mengatur dan mengontrol perubahan pada *requirements* tersebut yaitu dengan melakukan penelusuran *requirements* atau disebut *requirement traceability*. Pada proses penulusuran *requirements* tersebut terdapat *tools* yang bisa digunakan yaitu dengan menggunakan *requirements traceability matrix* agar bisa dianalisis keterkaitan *requirements*-nya.

Jadi, *requirements* yang didapat dari modul dosen Siakad UNJ hingga bisa dianalisis keterkaitan *requirements*-nya dengan *requirements traceability matrix* harus melewati alur seperti yang digambarkan pada Gambar 2.6.

BAB III

METODOLOGI PENELITIAN

3.1. Tempat dan Waktu Penelitian

Penelitian ini dilakukan di Gedung D, Kampus A Unit Pelayanan Teknis Teknologi Informasi dan Komunikasi Universitas Negeri Jakarta (UPT TIK - UNJ) yang berlokasi di Jl. Rawamangun Muka, Jakarta 13220. Penelitian dilaksanakan mulai bulan Maret 2017 hingga bulan Juni 2017.

3.2. Alat dan Bahan Penelitian

3.2.1. Alat

Alat yang digunakan dalam penelitian ini adalah sebagai berikut:

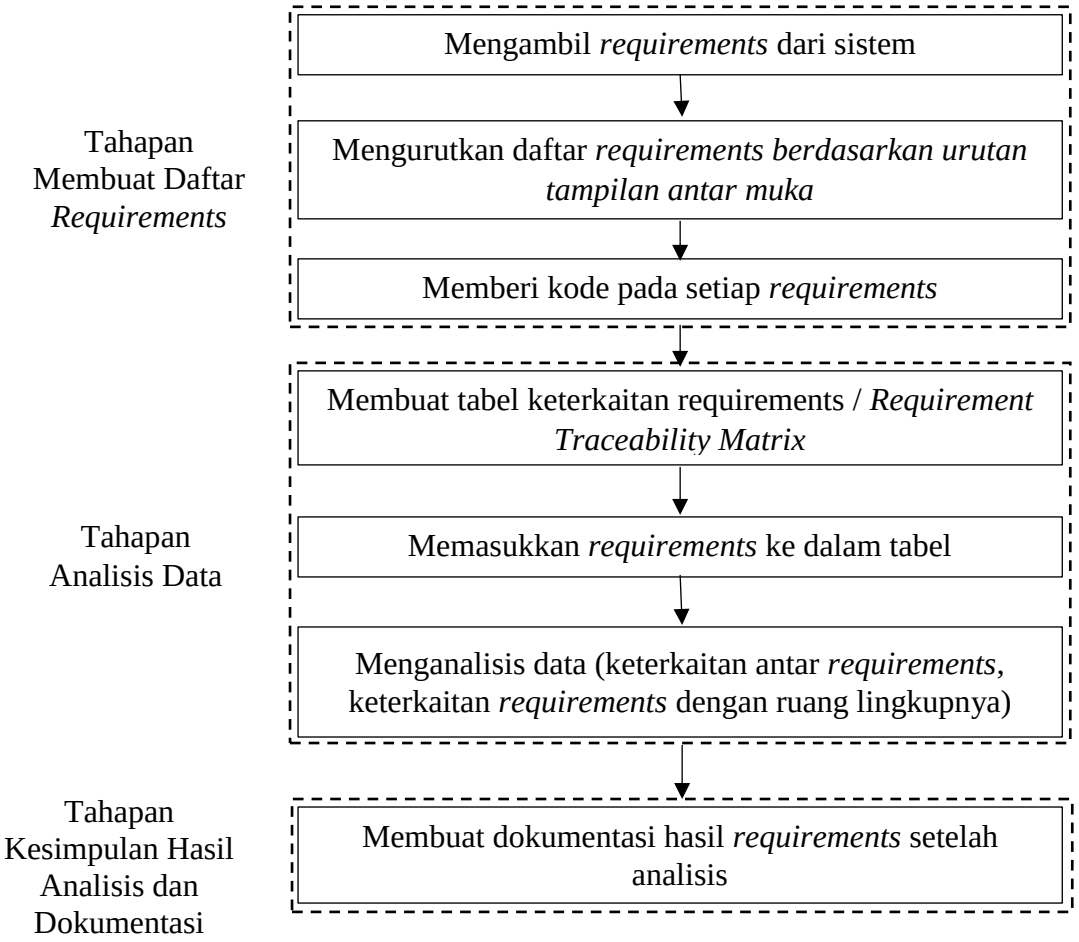
1. Perangkat Keras (*Hardware*) :
    a. Acer Aspire One 722 *Processor* AMD C – 60 APU *with* Radeon™ HD *Graphics* 1.00 GHz;
    b. *Memory* RAM 4 GB DDR3.
2. Perangkat Lunak *(Software*)
    a. *Windows 8.1 Pro* 64-bit © 2013;
    b. *Microsoft Office Professional Plus* 2013 64 bit.

3.2.2. Bahan

Penelitian ini menggunakan Siakad UNJ sebagai bahan penelitian. Bagian yang digunakan adalah *requirements* pada Modul Dosen Siakad UNJ.

3.3. Diagram Alir Penelitian

Diagram alir penelitian yang dilakukan yaitu implementasi *Requirement Traceability Matrix* pada modul dosen Siakad bisa dilihat pada Gambar 3.1:



Gambar 3.1. Diagam Alir Penelitian

3.3.1. Tahapan Membuat Daftar *Requirements*

Tahapan ini merupakan tahapan dilakukannya proses pengambilan *requirements* dari Modul Dosen Siakad UNJ sampai pada mengurutkan *requirements* yang didapat. Proses tahapannya yaitu: 1) Mengambil *requirements* dari sistem; 2) Mengurutkan *requirements*; 3) Memberikan kode pada setiap *requirements*.

Pengambilan *requirements* dilakukan dengan pengambilan langsung dari sistem, peneliti melihat situs lalu menganalisis apa saja fungsi yag terdapat pada sistem yang nantinya dijadikan sebagai *requirements* dari sistem. Hal ini dilakukan karena tidak adanya dokumentasi tertulis yang resmi dari sistem. Penulisan *requirements* menggunakan kata kerja dan setiap satu *requirements* menggunakan satu kata kerja.

Setelah mengambil *requirements* yang dibutuhkan, *requirements* lalu di urutkan dalam sebuah tabel sesuai dengan urutan dari sub-menu pada sistem. Diurutkan dalam tabel, karena setiap *requirements* nantinya diberikan kode *requirements*. Jadi, pada tahapan ini setiap *requirements* telah memiliki kode *requirements* sendiri sebagai pengganti nama *requirements* yang akan dicantumkan di tabel *Requirement Traceability Matrix* nantinya.

3.3.2. Tahapan Analisis Data

Pada tahapan ini, terdapat beberapa langkah yang dilakukan dalam analisis data. Langkah – langkah tersebut yaitu: 1) Membuat tabel keterkaitan/ tabel *Requirement Traceability Matrix*; 2) Memasukkan daftar *requirements* ke dalam tabel *Requirement Traceability Matrix*; 3) Menganalisis data.

1.3.3. Tahapan Kesimpulan Hasil Analisis dan Pembuatan Dokumentasi

Dari proses analisis yang dilakukan, bisa ditarik kesimpulan. Kesimpulan yang dibuat menentukan daftar *requirements* yang baru. Daftar *requirements* yang baru nantinya bisa di dokumentasikan sebagai acuan untuk pengembangan sistem dimasa mendatang.

3.4. Teknik dan Prosedur Pengumpulan Data

3.4.1. Teknik Pengumpulan Data

Pada penelitian kali ini, cara untuk melakukan *tracing* yang digunakan yaitu *forward traceability*. Karena pengumpulan data dilakukan dengan melakukan pengamatan langsung terhadap sistem melalui tampilan antar muka dari Modul Dosen Siakad UNJ.

Selain menggunakan *forward traceability*, pada penelitian ini juga menggunakan *post-requirements spesification traceability* karena penelusuran *requirements* dilakukan berdasarkan fungsi atau kegunaan dari sistem .

3.4.2. Prosedur Pengumpulan Data

Pada proses pengumpulan data, ada beberapa prosedur atau tahapan yang dilakukan. Alur prosedur pengumpulan data yang dilakukan bisa dilihat pada Gambar 3.2.

Gambar 3.2 Prosedur Pengumpulan Data

Berdasarkan Gambar 3.2, sebelum sampai pada tahap pengamatan, penelitian diawali dengan melakukan pengamatan dengan berinteraksi langsung pada Siakad UNJ. Setelah melihat keseluruhan tampilan dari Siakad UNJ dari sebelum dan sesudah *login* ke modul dosen, semua tampilan didokumentasikan dengan cara melakukan screenshot pada tampilan yang ada. Screenshot dilakukan

pada Siakad untuk modul dosen sendiri pengumpulan *requirements* bisa dilakukan diluar UPT TIK. Modul dosen tidak bisa diakses di luar UPT TIK, karena berkaitan dengan privasi akses untuk akun dosen yang digunakan. *Screenshot* yang dikumpulkan dikelompokkan berdasarkan menu yang terdapat pada tampilan Siakad untuk selanjutnya dibuat *requirements*-nya dan dikelompokkan sesuai dengan tampilan Siakad. Setelah itu, beri kode pada setiap *requirements*. Pemberian kode dilakukan untuk proses analisis data selanjutnya.

3.5. Teknik Analisis Data

Data yang telah dikumpulkan dan telah diberi kode pada setiap *requirements*, kemudian diolah dengan cara dianalisis keterkaitannya. Proses awal pengolahan *requirements* yaitu membuat tabel keterkaitan atau tabel *Requirement Traceability Matrix*. Format tabel *Requirement Traceability Matrix* yang digunakan seperti yang terlihat pada Tabel 3.1**.**

Tabel 3.1 Tabel Requirement Traceability Matrix yang digunakan

Kode req	*user*		*Functional area*					*Relation req*				
	Umum	Dosen	a	b	c	d	...	RD001	RD002	RD003	RD004	...
RD001												
RD002												
RD003												
RD004												
...												

Keterangan :

- Kode req : penomoran untuk tiap *requirements* yang ada pada sistem;
- *user* : *user* atau pengguna yang terlibat pada sistem;
- *Functional Area* : berisi hasil pengategorian *requirements* berdasarkan kesamaan ciri dan dimensi;
- *Relation Req* : keterkaitan antar *requirement* dalam sistem.

Requirements yang dimasukkan kedalam tabel adalah kode *requirements* yang telah dibuat pada proses pengumpulan data. Kode *requirements* yang dimasukkan bisa dilihat di Tabel 3.1. Setelah *requirements* dimasukkan, maka dilakukanlah analisis berupa relasi-relasi keterkaitan *requirements*. Untuk relasi keterkaitan yang dibuat, penelitian ini menggunakan jenis relasi dari *inter requirements traceability*. Artinya, relasi yang dibentuk merupakan relasi antar requirement dalam modul dosen Siakad UNJ saja.

Dari data yang telah dimasukkan kedalam tabel *Requirement Traceability Matrix*, akan diketahui relasi-relasi yang terkait pada *requirements*. Dengan diketahuinya keterkaitan *requirements*, maka akan jelas sumber dari suatu *requirements* yang *error* dan akan diketahui *requirements* apa saja yang akan terkena dampak dari *error*-nya suatu *requirements*.

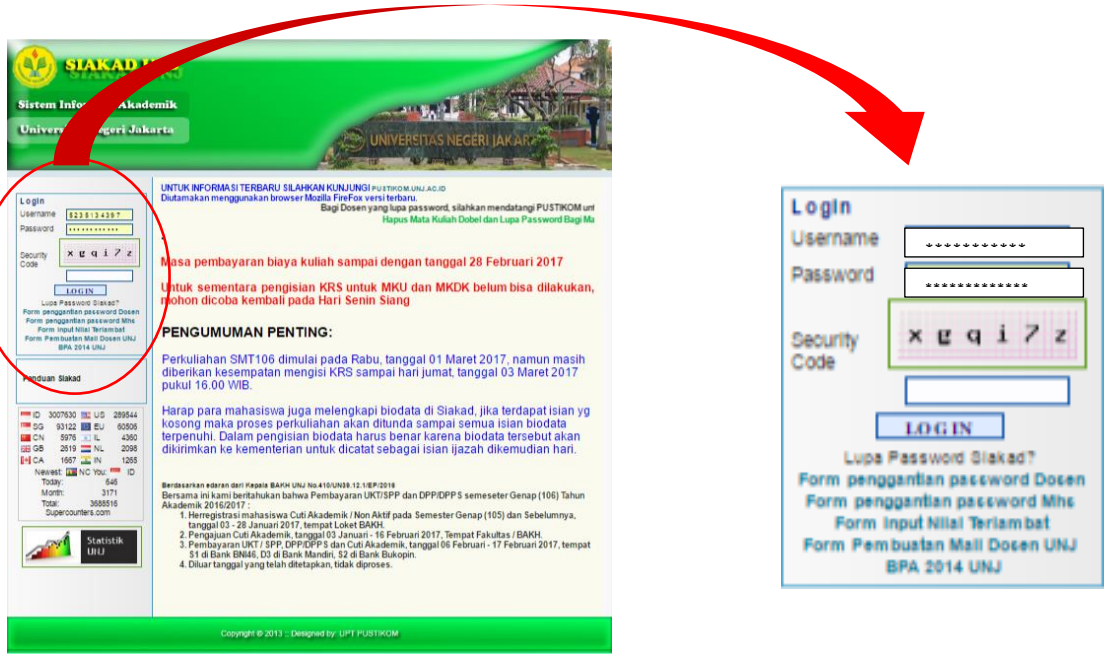
BAB IV

HASIL PENELITIAN

4.1. Deskripsi Hasil Penelitian

4.1.1. Mengambil *Requirements* dari Siakad UNJ

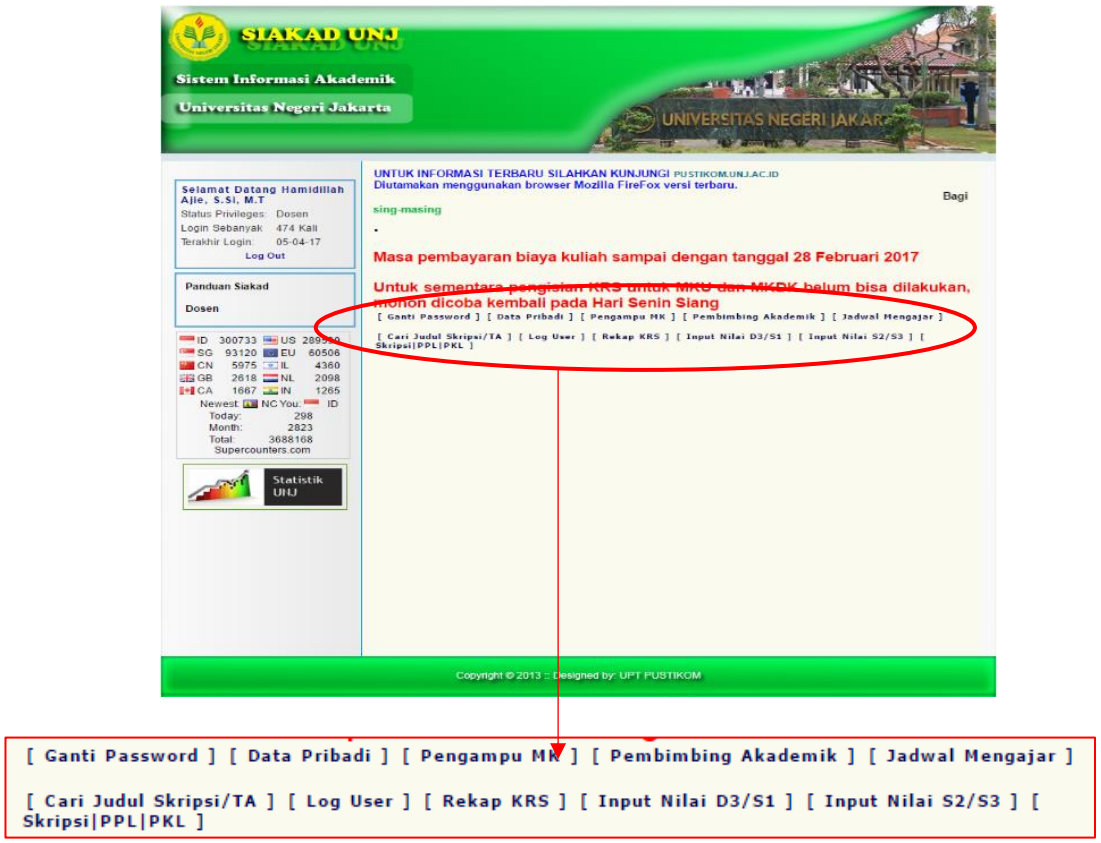
Requirements pada modul Dosen Siakad UNJ yang dijadikan bahan untuk penelusuran *requirements* yang dilakukan. Proses pengumpulan *requirements* diawali dengan mengakses http://siakad.unj.ac.id/ untuk bisa *login* dengan *username* dan *password* yang dimiliki oleh dosen. Kolom *login* yang terdapat pada halaman awal dari Siakad UNJ bisa dilihat pada Gambar 4.1 berikut :



Gambar 4.1 Kolom *login* di halaman awal Siakad UNJ

Pada halaman awal, terdapat beberapa bagian atau juga tulisan yang bisa dijadikan sebuah *requirements*. Contohnya seperti pada bagian "*form login*", bisa dijadikan *requirements* yaitu seperti "menyediakan fasilitas *form login* berupa *username* dan *password*". Ada juga contoh *requirements* lain seperti "menampilkan

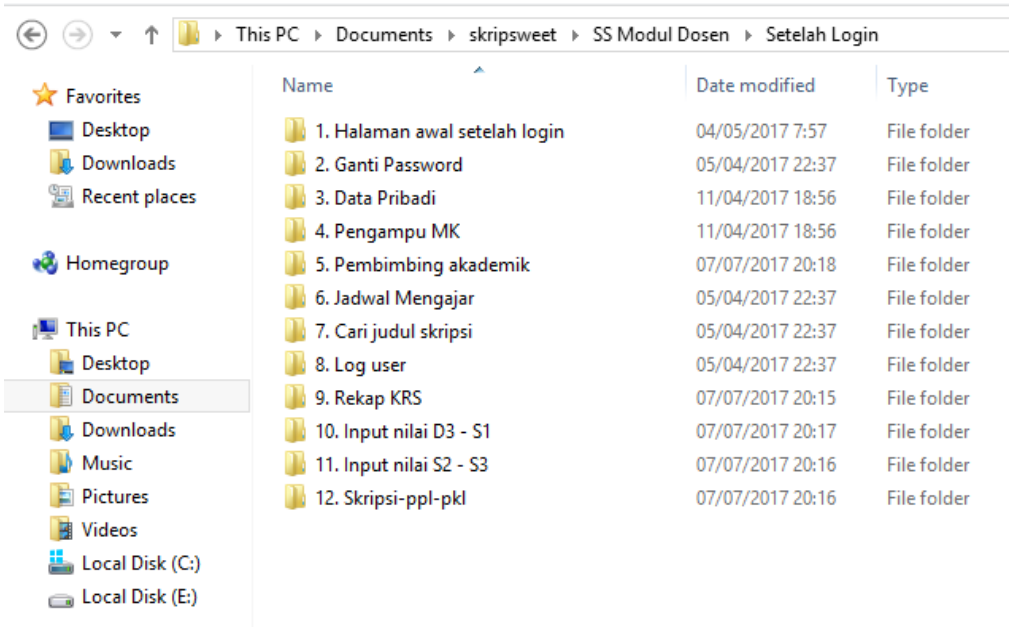
informasi seputar tanggal-tanggal penting yang berkaitan dengan pembayaran kuliah untuk semua *user*". *Requirements* tersebut bersifat umum karena semua *user* (dosen, mahasiswa, dan umum) bisa melihat tampilan yang dituliskan pada *requirements* tersebut. Seluruh elemen yang berkaitan dengan fungsional dari Siakad UNJ khusus modul dosen dijadikan *requirements*. Dikarenakan akses yang terbatas, pengumpulan *requirements* dengan menggunakan akses akun dosen tidak bisa dilakukan diluar gedung UPT TIK. Cara lain untuk mengumpulkan *requirements* yaitu dengan melakukan *screenshoot* pada seluruh tampilan Siakad UNJ yang diperlukan, seperti pada Gambar 4.2 :



Gambar 4.2 Menu pada modul dosen Siakad UNJ

Gambar 4.2 merupakan menu yang terdapat pada tampilan modul dosen Siakad UNJ. *Screenshot* dibagi ke dalam beberapa folder berdasarkan fungsinya

dan mengutkan folder berdasarkan urutan menu yang terdapat pada tampilan Siakad UNJ. Urutan menu diambil dari menu "[Ganti Password]" lalu dilanjutkan ke menu "[Data Pribadi]" dan selanjutnya hingga menu "[Skripsi/PPl/PKL]". Jadi, urutan folder *screenshoot* juga dibuat sesuai urutan menu pada Gambar 4.2.



Gambar 4.3 Contoh folder penyimpanan *screenshoot*

Pengurutan folder penyimpanan untuk *screenshoot* tampilan modul dosen Siakad UNJ dilakukan untuk memudahkan penyusunan *requirements* dan membuat *requirements* menjadi lebih terstruktur. Karena, menu yang terdapat pada modul dosen Siakad UNJ nantinya akan menjadi generalisasi atau pengelompokkan dari beberapa *requirements*. Setelah bahan untuk mengumpulkan *requirements* lengkap, maka dibuat lah *requirements* dari seluruh tampilan sistem. Tidak ada aturan khusus pada penulisan *requirements*, yang diperlukan yaitu konsistensi pada penulisan *requirements*. Jika *requirements* pertama ditulis dengan diawali kata kerja terlebih dahulu, maka sebaiknya seluruh *requirements* harus menggunakan konsep yang sama. Begitu juga dengan konsep penulisan lainnya, bisa dilakukan jika penulisan

seluruh *requirements* konsisten. Setiap satu fungsi dalam sistem dituliskan sebagai satu *requirements*. Contoh penulisan *requirements* yang konsisten bisa dilihat pada Gambar 4.4.

No	Requirements
1	Menampilkan informasi seputar tanggal-tanggal penting yang berkaitan dengan pembayaran kuliah untuk semua user
2	Menampilkan informasi seputar tanggal-tanggal penting yang berkaitan dengan pengisian kartu rencan studi (KRS) untuk semua user
3	Menampilkan panduan langkah-langkah pemakaian Siakad UNJ untuk semua user
4	Menampilkan Flag Counter dari Siakad UNJ untuk semua user
5	Menyediakan tautan menuju ke supercounters UNJ untuk semua user

Gambar 4.4 Contoh penulisan *requirements*

Keterangan :

- No : nomor urut *requirements*, nomor terakhir bisa menjadi jumlah seluruh *requriements* yang terdapat pada sistem;
- *Requirements* : nama *requirements.*

Penulisan *requirements* tidak harus terlalu detail, yang penting dalam penulisans *requirements* yaitu konsistensi penulisan dan bahasa yang digunakan mudah dimengerti oleh *developer* atau pihak yang nantinya akan menggunakan dokumentasi dari sistem perangkat lunak.

4.1.2. Mengurutkan *Requirements*

Proses selanjutnya setelah *requirements* dikumpulkan yaitu dengan mengurutkan *requirements* berdasarkan urutan menu yang terdapat pada modul dosen Siakad UNJ seperti yang ditunjukkan oleh Gambar 4.2. Tidak hanya menu yang terdapat pada tampilan modul dosen Siakad UNJ yang dijadikan sebagai *functional area* (pengelompokkan *requirements* berdasarkan fungsi) sistem,

terdapat beberapa *functional area* lainnya yang nantinya akan dimasukkan dalam tabel uji *Requirement Traceability Matrix.* Seperti misalnya pada tampilan antar muka pada halaman awal dari Siakad UNJ terdapat kolom untuk mengisi *username, password* dan *security code*. Dari kolom tersebut bisa dibuat menjadi beberapa *requirements* yang dapat disatukan dalam satu *functional area* "*Login*". Contoh lain penentuan *functional area*, bisa dilihat pada Gambar 4.5.

Requirements	Keterangan	Functional Area
Menampilkan informasi seputar tanggal-tanggal penting yang berkaitan dengan pembayaran kuliah		Informasi umum
Menampilkan informasi seputar tanggal-tanggal penting yang berkaitan dengan pengisian kartu rencan studi (KRS)		
Menampilkan panduan langkah-langkah pemakaian Siakad UNJ untuk dosen		
Menampilkan Flag Counter dari Siakad UNJ	Requirement Tambahan	
Menyediakan tautan menuju ke supercounters UNJ	Requirement Tambahan	
Menyediakan tautan menuju ke statistik UNJ	Tidak Berfungsi	

Gambar 4.5 Contoh functional area

Keterangan :

- *Requirements* : nama *requirements;*
- Keterangan : detail khusus pada suatu *requirements*;
- *Functional Area* : generalisasi fungsi *requirements*.

Gambar 4.5 menampilkan contoh lain *requirements* mana saja yang bisa disatukan dalam satu *functional area* selain *requirements* yang termasuk dalam menu pada tampilan modul dosen Siakad UNJ dan *functional area* "*Login*". Artinya *requirements* yang terkumpul dalam suatu *functional area* merupakan kumpulan beberapa *requirements* yang memiliki kemiripan fungsi satu sama lain atau fungsi *requirements*-nya saling berkaitan. Sehingga, *requirements* tersebut bisa disatukan

dalam suatu generalisasi fungsi yang disebut *functional area.* Setelah seluruh *requirements* telah terurut berdasarkan *functional areanya*, baru lah bisa melanjutkan ke proses selanjutnya.

4.1.3. Membuat Kode Pada Setiap *Requirements*

Tahap selanjutnya yaitu pemberian kode pada setiap *requirements*. Pemberian kode dilakukan untuk mempermudah dalam proses pengolahan data saat diuji dalam tabel *Requirement Traceability Matrix*. Jadi, setiap *requirements* yang telah diurutkan diberi kode satu persatu. Contoh pemberian kode pada *requirements* bisa dilihat pada Gambar 4.6.

Kode	Requirements	Keterangan	Functional Area
RD001	Menampilkan informasi seputar tanggal-tanggal penting yang berkaitan dengan pembayaran kuliah		Informasi umum
RD002	Menampilkan informasi seputar tanggal-tanggal penting yang berkaitan dengan pengisian kartu rencan studi (KRS)		
RD003	Menampilkan panduan langkah-langkah pemakaian Siakad UNJ untuk dosen		
RD004	Menampilkan Flag Counter dari Siakad UNJ	Requirement Tambahan	
RD005	Menyediakan tautan menuju ke supercounters UNJ	Requirement Tambahan	
RD006	Menyediakan tautan menuju ke statistik UNJ	Tidak Berfungsi	
RD007	Menyediakan tautan untuk mengunduh panduan siakad bagi dosen		Unduh Berkas
RD008	Menyediakan tautan untuk mengunduh panduan siakad bagi mahasiswa		

Gambar 4.6 Kode *Requirements*

Keterangan :

- Kode : penomoran untuk tiap *requirements* yang ada pada sistem;
- *Requirements* : nama *requirements;*
- Keterangan : detail khusus pada suatu *requirements*;
- *Functional Area* : hasil pengategorian *requirements* berdasarkan kesamaan ciri dan dimensi.

Dalam pemberian kode, sama seperti pada penulisan *requirements* yang tidak memiliki aturan baku dalam penulisannya. Hal yang bisa dijadikan acuan dalam penentuan kode *requirements* yaitu penentuan konsep kode yang akan digunakan. Konsep yang bisa digunakan untuk penentuan kode *requirements* yaitu sebaiknya kode *requirements* bisa menggambarkan *requirements* dari sistem yang mana. Seperti halnya pada Gambar 4.6, contoh kode *requirements* "RD001". Arti dari kode *requirements* "RD001" yaitu :

- RD : *Requirements* Dosen, *requirements* yang diambil dari modul dosen Siakad UNJ;
- 001 : angka yang mengurutkan *requirements*, skala 3 berarti kemungkinan jumlah *requirements* tidak akan melebihi 999.

Kode *requirements* yang digunakan dianggap mampu menggambarkan sistem yang menjadi acuan dilakukannya pengujian dengan *Requirement Traceability Matrix.*

4.2. Analisis Data Penelitian

4.2.1. Membuat Tabel *Requirement Traceability Matrix*

Pada tahap ini, *requirements* yang telah dikumpulkan bisa digunakan untuk dilakukan analisis. Sebelum *requirements* dapat diolah seluruhnya, dibuat terlebih dahulu format tabel dari *requirement traeability matrix* yang akan digunakan. Dari beberapa referensi tabel *requirement traceability matrix* yang ada, lalu dibuat contoh format tabel *requirement traceability matrix* yang sesuai dengan kebutuhan analisis untuk Modul Dosen Siakad UNJ. Format tabel yang dibuat seperti yang ditunjukkan oleh Tabel 4.1.

Tabel 4.1 Tabel *Requirement Traceability Matrix*

Kode req	*user*		*Functional area*					*Relation* req				
...												

Keterangan :

- Kode : penomoran untuk tiap *requirements* yang ada pada sistem;
- *User* : *user* atau pengguna yang terlibat pada sistem;
- *Functional Area* : hasil pengategorian *requirements* berdasarkan kesamaan ciri dan fungsi;
- *Relation Req* : keterkaitan antar *requirements* dalam sistem.

Tabel 4.1 merupakan contoh format tabel *requirement traceability matrix* yang digunakan untuk melakukan analisis keterkaitan antar *requirements*. Tabel berisi kolom Kode Req yang nantinya akan diisi oleh kode *requirements* yang telah dibuat. Selanjutnya ada kolom *user*, *user* disini merupakan *user* dimana *requirements* tersebut berada. *User* dibedakan berdasarkan sebelum dan setelah user melakukan *login* kedalam Siakad UNJ. Dalam kolom selanjutnya yaitu terdapat kolom *functional area* yang merupakan generalisasi fungsi dari beberapa *requirements* yang memiliki ciri dan fungsi yang sama. Pada kolom selanjutnya yaitu kolom *relation req* yang merupakan kolom untuk menganalisa relasi keterkaiatan antar *requirement*. Relasi keterkaitan antar *requirement* – nya bisa digambarkan dengan menghubungkan kode *requirements* dengan *requirements* yang terdapat pada kolom *relation* req. Seluruh kolom yang terdapat pada Tabel 4.1 dirasa sudah cukup sebagai bahan untuk analisis keterkaitan *requirements* dengan *requirement traceability matrix.*

4.2.2. Memasukkan *Requirements* ke dalam Tabel

Tahapan selanjutnya yaitu memasukkan *requirements* ke dalam tabel *requirement traceability matrix*. *Requirements* yang dimasukkan yaitu berupa kode *requirements* beserta jenis *user* dan *functional area* yang telah dibuat pada tahapan sebelumnya. Hasil memasukkan requirements pada tabel *requirement traceability matrix* bisa dilihat pada Tabel 4.2.

Tabel 4.2 Tabel *Requirement Traceability Matrix* yang digunakan

Kode req	*user*		*Functional area*		*Relation req*				
	Umum	Dosen	Login	...	RD001	RD002	RD003	RD004	...
RD001									
RD002									
RD003									
RD004									
...									

Requirements dari Modul Dosen Siakad UNJ seluruhnya terkumpul sebanyak 125, *requirements* telah divalidasi kepada pihak *developer* dari Siakad UNJ. Dari 125 *requirements* yang didapatkan, terdapat dua *user* yang terlibat dalam akses ke Siakad UNJ.

User dibedakan berdasarkan perbedaan hak akses pada aktifitas yang bisa dilakukan di dalam Siakad UNJ. *User* yang masuk kategori umum dikaitkan dengan *requirements* yang hanya bisa diakses dihalaman awal sebelum *login* dan *user* yang masuk kategori dosen merupakan *user* yang *requirements* – nya ada setelah *login* dilakukan.

Functional area yang telah dibuat tidak hanya *functional area* yang menjadi menu di modul dosen, tapi ada beberapa *functional area* yang dibuat berdasarkan kesamaan fungsi atau *requirements* yang saling mendukung *requirements* lain dalam suatu *functional area* tertentu. Sehingga *functional area* yang terkumpul

menjadi delapan belas *functional area* termasuk dengan sebelas *functional area* yang berasal dari menu modul dosen.

Relation req dalam tabel *requirements traceability matrix* merupakan kode *requirements* yang telah dikumpulkan sebanyak 125 *requirements*. Daftar *requirements* yang ada di kolom *relation req* merupakan acuan untuk menentukan relasi antar *requirements*. Jadi, *requirements* yang terdapat pada kolom kode req jika memiliki keterkaitan antar *requirements* akan di hubungkan dengan *requirements* yang memiliki relasi keterkaitan tersebut pada kolom *relation req*.

Jika tabel *requirement traceability matrix* telah terisi faktor-faktor analisisnya, maka relasi keterkaitan *requirements* siap untuk dilakukan analisis.

4.2.3. Menganalisis Data

Proses analisis data merupakan proses yang memerlukan ketelitian, karena proses dilakukan secara manual. Kemungkinan terjadi kesalaham karena human *error* sangat besar. Selain itu, proses analisis juga memakan waktu yang tidak sebentar, karena perlu dilakukan verifikasi langsung kepada *developer* sistem yang bersangkutan. Untuk itu, proses analisis memakan waktu paling banyak diantara proses yang lainnya.

Tabel 4.1 menunjukkan tabel *requirement traceability matrix* yang sudah diisi *requirements* dan semua bahan yang diperlukan untuk siap dilakukan analisis. Tabel *requirement traceability matrix* diisi dengan menggunakan tanda cek (v) pada kolom yang sesuai dengan *user* yang mana, *functional area* yang mana dan yang memiliki relasi antar *requirements*.

4.2.3.1 Cek Kolom *User*

Cara pertama yang dilakukan adalah melakukan cek dan memberikan tanda ceklis (v) pada kolom *requirements* yang ada pada *user* tertentu, seperti pada Gambar 4.7.

Kode Req	User	
	Umum	Dosen
RD013	v	
RD014	v	
RD015	v	
RD016	v	
RD017	v	
RD018	v	
RD019		v
RD020		v
RD021		v
RD022		v

Gambar 4.7 Cek kolom *user*

Pada Gambar 4.7 RD013 sampai RD018 memiliki tanda ceklis pada *user* "Umum" dan RD019 sampai RD022 memiliki tanda ceklis pada *user* "Dosen". Artinya *requirements* RD013 sampai RD018 bisa dilihat atau ada jika *user* belum melakukan *login*, semua *user* bisa melihat *requirements* tersebut. baik itu mahasiswa, dosen, bahkan *user* umum yang bukan bagian dari kedua *user* tersebut. Untuk *requirements* RD019 sampai RD022 bisa dilihat atau ada jika *user* telah melakukan *login* kedalam sistem dengan jenis *user* sebagai dosen. Seluruh *requirements* melalui proses pembagian *user* ini.

4.2.3.2 Cek Kolom *Functional Area*

Proses selanjutnya yaitu memberikan tanda pada kolom *functional area* yang sesuai dengan *requirements*-nya. Kode *requirements* yang ada dikaitkan dengan *functional area* yang menaunginya, seluruh *requirements* dilakukan hal yang sama. Seperti contoh pada Gambar 4.8

Kode Req	Functional Area																	
	Informasi Umum	Unduh Form	Panduan	Login	Status User	Logout	Akses Dosen	Ubah Password	Data Pribadi	Pengampu Mata Kuliah	Pembimbing Akademik	Jadwal Mengajar	Cari Judul Skripsi	Log User	Rekap KRS	Input Nilai D3/S1	Input Nilai S2/S3	Skripsi / PPL / PKL
RD001	v																	
RD002	v																	
RD003	v																	
RD004	v																	
RD005	v																	
RD006		v																
RD007		v																
RD008		v																
RD009		v																
RD010		v																
RD011			v															
RD012			v															
RD013			v															
RD014				v														
RD015				v														
RD016				v														
RD017				v														
RD018				v														
RD019	v																	
RD020	v																	
RD021	v																	

Gambar 4.8 Cek kolom *Functional area*

Pada Gambar 4.8 menunjukkan pemberian tanda pada *functional area* tertentu dimana *requirements* dikelompokkan atau darimana *requirements* berasal.

4.2.3.3 Cek Kolom *Relation Req*

Setelah memberikan tanda pada *functional area*, proses selanjutnya yaitu memerikan tanda pada kolom *relation req* untuk menunjukkan terdapat keterkaitan antar *requirements*. Prosesnya sama seperti kedua proses sebelumnya, yaitu memerikan tanda ceklis pada kolom yang *requirements*-nya memiliki relasi keterkaitan, seperti yang ditunjukkan pada Gambar 4.9.

Kode Req	RD014	RD015	RD016	RD017	RD018	RD019	RD020	RD021	RD022	RD023	RD024	RD025	RD026	RD027	RD028	RD029
RD013																
RD014		v	v	v	v											
RD015	v			v	v											
RD016	v	v		v	v	v	v	v	v	v	v	v	v	v	v	v
RD017	v	v	v		v									v		v
RD018	v	v	v	v									v			v
RD019			v													v
RD020			v													v
RD021			v													v
RD022			v													v
RD023			v													v
RD024			v													v
RD025			v													v
RD026			v		v											v
RD027			v	v												v
RD028			v													v
RD029			v	v	v	v	v	v	v	v	v	v	v	v	v	v

Gambar 4.9 Cek *relation req*

Gambar 4.9 menunjukkan terdapat satu *requirements* yang memiliki banyak keterkaitan atau ada juga *requirements* yang tidak memiliki relasi keterkaitan sama sekali.

Seluruh *requirements* dimasukkan kedalam tabel *requirements traceability matrix* hingga melewati 3 proses pengecekkan dan pemberian tanda di masing – masing kolom. Seluruh proses dalam tabel *requirements traceability matrix* telah dilampirkan dan telah divalidasi oleh pihak *developer* dari UPT TIK UNJ dimana sistem dikembangkan.

4.3. Pembahasan

Tabel *requirements traceability matrix* yang telah terisi lengkap di sudah bisa digunakan untuk keperluan pengembangan sistem. namun pada sub bab ini, akan dijelaskan bagaimana membaca tabel *requirements traceability matrix* tersebut. Misalnya pada Gambar 4.10 tedapat *requirements* yang diambil dari functional area "*Login*".

RPD014	Menyediakan fasilitas *form login* berupa *username, password* dan *security code*	***Login***
RPD015	Menampilkan pesan peringatan berupa alert jika *username, password*, atau *security code* yang di isi salah	
RPD016	Menampilkan pesan berupa *alert* jika *user* berhasil *login* ke dalam sistem	
RPD017	Mencatat tanggal saat melakukan *login*	
RPD018	Mengakumulasi jumlah aktifitas *login* dari awal *user* menggunakan Siakad	

Gambar 4.10 Kolom *requirements functional area login*

Pada Gambar 4.10 menunjukkan *requirements* yang dikelompokkan kedalam *functional area login*. *Requirements* yang tergabung pada *functional area login* terdapat lima *requirements* dari kode *requirement* RD014 sampai kode *requirement* RD018. Jika digambarkan kedalam tabel *requirements traceability matrix* maka akan menjadi seperti yang ditunjukkan oleh Gambar 4.11.

Kode Req	RD014	RD015	RD016	RD017	RD018
RD014		v	v	v	v
RD015	v			v	v
RD016	v	v		v	v
RD017	v	v	v		v
RD018	v	v	v	v	

Gambar 4.11 *Requirement traceability matrix functional area login*

Pada Gambar 4.11, terdapat contoh pengisian tabel *requirement traceability matrix* bagi *functional area* "*login*". Jika diambil baris tabel pada *requirements* RD014 saja, maka relasi yang terbentuk yaitu :

RD014	RD015				RD014 yang tidak diisi dengan benar akan membuat RD015 muncul
	RD016				RD014 yang sudah diisi dengan benar akan membuat RD016 muncul
	RD017				Jika RD014 yang telah diisi dengan benar sampai muncul RD016 maka RD017 akan dicatat
	RD018				Jika RD014 yang telah diisi dengan benar sampai muncul RD016 maka RD018 akan diakumulasikan dari jumlah RD018 sebelumnya
	RD019 - RD125				Jika sudaj mengisi RD014 dengan benar, maka *user* dapat mengakses requirements dari RD019 - RD125

Gambar 4.12 Tabel relasi *requirement* RD014

Pada Gambar 4.10 dijelaskan bahwa RD014 merupakan subtitusi dari *requirement* "Menyediakan fasilitas *form login* berupa *username*, *password* dan *security code*". Jika dilihat dari keterangan *requirement* RD014 tersebut, bisa diartikan bahwa selulruh aktifitas *login* dimulai dari RD014. Sehingga RD014 memiliki relasi dengan seluruh *requirements* yang ada pada *functional area* "*Login*".

Sesuai dengan Gambar 4.10, contoh analsisa bisa dilakukan pada RD014. Pertama – tama, dilihat antara RD014 dengan RD015. RD015 merupakan subtitusi dari *requirement* "Menampilkan pesan peringatan berupa *alert* jika *username*, *password*, atau *security code* yang di isi salah". Artinya RD015 ada jika pada proses *input form login* salah dalam hal ini digambarkan dengan RD014. Jadi jika dilihat

dari kedua *requirement* tersebut, bisa ditarik relasi bahwa pengisian RD014 yang tidak benar akan membuat RD015 muncul sesuai dengan keterangan yang terdapat pada Gambar 4.12. Namun, RD014 tidak tergantung dengan RD015. Karena jika RD015 tidak ada, tidak akan mempengaruhi fungsi dari RD014. Namun sebaliknya, RD015 terikat denga RD014. Karena RD015 akan muncul jika dilakukan aksi pada RD014.

Selain dengan RD015, RD014 juga memiliki relasi dengan RD016. RD016 merupakan subtitusi dari *requirement* "Menampilkan pesan berupa *alert* jika *user* berhasil *login* ke dalam sistem". Jika disandingkan dengan RD014, RD016 tergantung pada RD014. RD016 akan muncul jika *user* mengisi form *login* (RD014) dengan benar, RD016 tidak akan muncul jika RD014 tidak diisi dengan benar. Tapi, jika RD016 tidak ada tidak akan mempengaruhi fungsi dari RD014.

Terdapat dua *requirement* yang belum di deskripsikan relasinya dengan RD014, yaitu RD017 dan RD018. RD017 yaitu subtitusi dari *requirement* "Mencatat tanggal saat melakukan *login*". Fungsi dari *requirement* ini yaitu sebagai *requirement* yang nantinya menampilkan waktu terakhir *login* di *functional area* status *user* yang terdapat pada tampilan Siakad UNJ setelah *login*. Relasinya dengan RD014 yaitu RD017 akan mencatat tanggal *login* jika *user* mengisi form *login* dengan tepat. Namun sampai seperti itu masih belum cukup, masih ada syarat untuk berhasil *login* yaitu munculnya pemberitahuan atau peringatan bahwa *user* telah berhasil *login*. Jika pemberitahuan atau dalam Siakad biasanya berupa *alert* merupakan RD016, maka RD017 memiliki relasi dengan dua *requirement*. Yaitu RD014 dan RD016. Agar RD017 bisa berfungsi maka *user* harus mengisi *form login* (RD014) hingga muncul *alert* berhasil *login* (RD016). Tetapi, jika tidak ada

RD016 pun tidak akan mempengaruhi fungsi dari RD017. Jadi, relasinya bias langsung dari RD014 ke RD17 atau bisa menjadi RD014 -> RD016 -> RD17.

Sama halnya dengan *requirement* RD018 yang merupakan subtitusi dari "Mengakumulasi jumlah aktifitas *login* dari awal *user* menggunakan Siakad UNJ". RD018 merupakan *requirement* yang nantinya berhubungan dengan *requirement* di *functional area* status *user* yaitu *requirement* yang menampilkan jumlah *login user* ke Siakad. Jadi untuk bisa mengakumulasi jumlah aktifitas *login*, *user* harus melakukan *login* sampai muncul *alert* (jika sistem memasang fungsi *alert*) yang menyatakan bahwa *user* berhasil *login*. Berarti, terdapat dua *requirement* yang terlibat dengan RD018, yaitu *requirement* yang menyediakan fasilitas untuk mengisi *login* form atau RD014 dan *requirement* yang memunculkan *alert* yaitu RD016. Sama halnya dengan relasi dari tiga requirement sebelumnya, bahwa RD016 tidak mempengaruhi kinerja fungsi dari RD018.

Requirement yang lain juga dihubungkan dengan cara yang sama, semua daftar *requirement*, tabel *requirement traceability matrix* dan tabel relasi sudah terlampir.

4.4. Aplikasi Hasil Penelitian

Aplikasi dari hasil *requirement traceability matrix* yang dilakukan yaitu untuk menjadi acuan *developer* sistem perangkat lunak Siakad UNJ untuk melakukan pengembangan sistem dimasa mendatang. Relasi-relasi yang terbentuk bisa dijadikan acuan untuk antisipasi jika terjadi *error* pada sistem dimasa mendatang.

BAB V

KESIMPULAN DAN SARAN

5.1. Kesimpulan

Berdasarkan hasil penelitian dan pembahasan yang telah dipaparkan, maka dapat disimpulkan bahwa *requirement traceability matrix* melewati tahap pengerjaan dari pengumpulan *requirements*, mengurutkan *requirements* dan pemberian kode pada *requirements*, penentuan *requirements* yang tergolong *functional area* tertentu, membuat tabel *requirements traceability matrix*, hingga menganalisis keterkaitan *requirements*-nya yang membuat *requiremen traceability matrix* bisa digunakan pada Modul Dosen Siakad UNJ.

Selain itu, penelitian ini juga menghasilkan beberapa informasi baru. Seperti pengumpulan *requirements* menghasilkan seratus dua puluh lima *requirements* dengan delapan belas *functional area*. Juga, terdapat beberapa *requirements* yang bisa dijadikan satu *requirement*, *requirements* sub menu nilai dan cetak nilai bisa dijadikan satu *requirement* menjadi input nilai dan bisa langsung dicetak. Ada pula beberapa *functional area* yang bisa dijadikan satu *functional area* karena lebih efisien dan *requirement* yang dihasilkan bisa dijadikan dokumentasi sistem untuk Modul Dosen Siakad UNJ.

5.2. Saran

Berdasarkan kesimpulan, maka disarankan :

1. Melakukan dokumentasi sistem yang lebih terstruktur pada sistem perangkat lunak Siakad UNJ;
2. Melakukan dokumentasi sistem yang lebih terstruktur pada sistem perangkat lunak lain yang ada di UNJ;
3. Tabel *requirement traceability matrix* yang dihasilkan bisa dijadikan acuan untuk melakukan *tracing requirements* pada sistem perangkat lunak yang lain;

4. Melakukan *tracing requiments* dengan menggunakan *requirement traceability matrix* pada sistem perangkat lunak lain yang ada di UNJ;
5. Melakukan *tracing requirements* dengan menggunakan *tools* lain selain *requiremnet traceability matrix.*

DAFTAR PUSTAKA

Fidhyallah, N. F. (2016). Model *Requirement Traceability* untuk Metode Pengembangan Perangkat Lunak *Modern Extreme Programming* [skripsi]. Jakarta : Fakultas Teknik, Universitas Negeri Jakarta.

Fissalma, H. (2016). Analisis dan Perancangan Sistem Informasi Penerimaan Mahasiswa Baru Universitas Negeri Jakarta [skripsi]. Jakarta : Fakultas Teknik, Universitas Negeri Jakarta.

[FT] Fakultas Teknik. 2012. Buku Pedoman Skripsi/Komprehensif/ karya Inovatif (S1). Jakrta : Fakultas Teknik, Universitas Negeri Jakarta.

Gotel, O. C. Z. & Finkelstein, A. C. W. (1994). "*An analysis of the requirements traceability problem." Proceeding of* ICRE94, 1st *International Conference on Requirements Engineering*. Colorado Springs, Co: IEEE CS Press, 1994.

IEEE *Standard Glossary of Software Engineering Technology* (IEEE Std 610.12-1990) [IEEE-610.12].

Madan, M., Dave, M., & Tandon, A.. (2016). *Need and Usage of Traceability Matrix for Managing Requirements. International Journal of Engineering Research Volume* No.5, *Issue* No.8, pp : 666-668.

Martin, C. & Dorf, M. (1990) *Interleukin-6 production by murine macrophage cell lines* P388D1 and J774A.1: *stimulation requirements and kinetics, Cell. Immunol. 128,* 555-568.

Pinherio, F. A. C. (2004). *Requirements Traceability: Chapter 5 Perspective on Software Requirements*. (91 – 113).

Pressman, R. S. (2010). *Software Engineering : A Practitioner's Approach 7th edition*. Boston.

Shabrina, F. (2016). Model *Requirements Traceability* untuk Metode Pengembangan Perangkat Lunak *Feature Driven Development* (FDD) [skripsi]. Jakarta : Fakultas Teknik, Universitas Negeri Jakarta.

Sommerville, I. (2010). *Software Engineering: 9th edition.* Pearson Education, Inc.

Thayer, R.H. and Dorfman, M. (1997). *Software Requirements Engineering*, 2nd ed., IEEE Computer Society Press

Wieringa, Roel. (1995). *An Introduction to Requirements Traceability*. (1 :1 – 24)