JURNAL PINTER
VOL 3. NO.1 JUNI 2019
e-ISSN: 2597- 4475
http://doi.org/10.21009/pinter.3.1.4

PERANCANGAN MODEL GORE MENGGUNAKAN METODE KAOS UNTUK PROSES *REVERSE ENGINEERING* SISTEM INFORMASI AKADEMIK UNIVERSITAS NEGERI JAKARTA

Hafiez Arief Raharjo[1], Widodo[2], Hamidillah Ajie[3]

[1,2,3] Pendidikan Teknik Informatika dan Komputer Fakultas Teknik
Universitas Negeri Jakarta

[1] hafiezraharjo@gmail.com, [2] widodo@unj.ac.id, [3] -hamidillah@unj.ac.id

Abstrak

Perubahan dan penambahan *requirement* pada suatu sistem aplikasi perangkat lunak membuat pengembangan pada sistem aplikasi terus dilkukan. Hal tersebut menyebabkan pentingnya dokumentasi *requirement* sistem dalam upaya pengembangan sistem lebih lanjut dan pemenuhan *requirement* yang diberikan oleh *stakeholder*. Penelitian ini bertujuan untuk merancangan model *reverse engineering* dengan menggunakan model *Goal Oriented Requirement Engineering* (GORE) dan metode *Keep All Objectives Satisfied* (KAOS) sebagai alat bantu untuk melakukan analisis dan penelusuran *functional requirement* pada sistem aplikasi perangkat lunak siap pakai. Hasil penelitian merupakan model *reverse engineering* dengan menggunakan model GORE dan metode KAOS dalam bentuk diagram yang menjelaskan tahapan untuk melakukan *reverse engineering* pada sistem aplikasi perangkat lunak. Tahapan *reverse engineering* dengan menggunakan model GORE dan metode KAOS tersebut, yaitu: (1) Mengambil *main goal* dari tampilan antarmuka sistem aplikasi (2) Merepresentasikan *goal* ke dalam *parralellogram graph* (3) Mengembangkan *goal* menjadi *subgoal* (4) Menentukan *expectation* dan *obstacle* berdasarkan *goal* (5) Menentukan *agent* yang terlibat dalam *expectation* dan *goal* (6) Merepresentasikan *expectation* dan *goal* yang merupakan suatu *requirement*. Model *reverse engineering* menggunakan model GORE dan metode KAOS berhasil diterapkan pada sampel aplikasi modul Mahasiswa siakad UNJ dan mendapatkan 125 *functional requirement*.

Kata kunci : ***Reverse Engineering*, GORE, KAOS dan *Requriement***

---

1. Pendahuluan

Proses pengembangan suatu sistem perangkat lunak senantiasa dilakukan selama sistem ingin terus digunakan oleh para penggunanya. Menurut Sommerville (2011: 235) pengembangan perangkat lunak tidak berhenti ketika sistem selesai dibuat tetapi terus selama sistem digunakan. Setelah sistem dibuat dan digunakan, sistem pasti akan mengalami perubahan jika sistem tersebut ingin tetap digunakan.

Pada umumnya pengembangan sistem perangkat lunak disebabkan oleh berbagai perubahan yang terjadi di sekitar lingkup sistem. Perubahan yang terjadi di sekitar lingkup sistem mendorong terjadinya perubahan di dalam sistem perangkat lunak. Perubahan bisnis dan pengalaman pengguna dapat menghasilkan *requirements* baru untuk perangkat lunak yang mengakibatkan keharusan bagi *developer* untuk memperbaiki sistem. Selain itu perubahan dapat terjadi karena beberapa hal seperti *eror code* atau adaptasi terhadap perubahan *hardware* dan *software platform* serta peningkatan terhadap kemampuan sistem.

Sistem Informasi Akademik Universitas Negeri Jakarta (Siakad UNJ) adalah sebuah sistem aplikasi yang hingga kini senantiasa dikembangkan. Siakad UNJ merupakan sistem perangkat lunak berbasiskan *website* milik UNJ yang digunakan untuk mengelola kegiatan administrasi akadaemik. Penggunaan Siakad UNJ membuat proses administrasi akademik berjalan secara komputerisasi.

Siakad UNJ terbagi menjadi beberapa modul atau bagian sistem. Pembagian modul umunya dikaitkan terhadap jenis pengguna Siakad UNJ. Modul yang ada di dalam Siakad UNJ diataranya adalah: Modul Mahasiswa, Modul Dosen, dan Modul Admin. Setiap modul memiliki layanan yang berbeda-beda dan telah disesuaikan dengan kebutuhan masing-masing jenis penggunanya.

Dalam proses pengembangannya sering kali penambahan fungsi Siakad UNJ untuk memenuhi *requirements* baru tidak disertai dengan dokumentasi yang baik. Hal ini menyebabkan semakin besarnya tingkat kesulitan dalam melakukan pemeliharaan atau pengembangan Siakad UNJ. Selain itu, sedikitnya dokumentasi dapat memperbesar terjadinya *eror* atau kegagalan sistem untuk

memenuhi tujuan awal. Untuk membantu proses pengembangan sistem yang memiliki sedikit dokumentasi diperlukan suatu metode *software reengineering* yang digunakan untuk menata dan membangun ulang sistem perangkat lunak.

Banyak metode dan model dikembangkan untuk membantu proses pengumpulan dan pemahaman *requirements* sistem perangkat lunak. *Requirements engineering* adalah salah satu proses yang digunakan untuk mengumpulkan dan memahami *requirements* sistem perangkat lunak.

Requirements engineering merupakan suatu proses penelusuran dan pemahaman *requirements* sistem yang umumnya digunakan pada proses *forward engineering* dan *reverse engineering*. Pada proses *forward engineering, requirements engineering* digunakan untuk melakukan penelusuran terhadap *stakeholder* untuk memperoleh *requirements* awal dalam suatu pembuatan sistem perangkat lunak. Pada proses *reverse engineering, requirements engineering* dapat digunakan untuk melakukan penelusuran dan mendapatkan *requirements* sistem. Pada *reverse engineering* penelusuran pada umumnya dilakukan terhadap tampilan antar muka sistem aplikasi.

Reverse engineering adalah salah satu pendekatan yang dapat digunakan untuk menemukan *requirements* dari sistem perangkat lunak yang telah siap digunakan. Menurut Singhal dan Gandhi (2014: 3) kelemahan utama *reverse engineering* adalah terdapat batas-batas praktisi untuk sejauh mana sebuah sistem dapat diperbaiki melalui *reverse engineering*. Kelemahan tersebut menyebabkan munculnya berbagai cara yang digunakan untuk meningkatkan hasil proses *reverse engineering*. Perancangan dan pembuatan model pendekatan yang seefektif mungkin merupakan salah satu cara untuk mengatasi kelemahan tersebut.

Siakad UNJ sebagai sebuah sistem yang mengelola kegiatan administrasi akademik memiliki banyak layanan yang digunakan untuk memenuhi kebutuhan dalam melaksanakan kegiatan akademik di UNJ. Hal ini membuat Siakad UNJ menjadi sebuah sistem yang kompleks dan memiliki cakupan yang luas. Oleh karena itu, diperlukan suatu metode *reverse engineering* yang mampu secara terstruktur dalam melakukan penelurusan terhadap *requirements* sistem.

Pada umunya metode *reverse engineering* dilakukan dengan cara mengambil langsung *requirements* sistem berdasarkan tampilan antarmuka dari suatu sistem aplikasi perangkat lunak. Pengambilan langsung *requirements* berdasarkan tampilan tanpa adanya teknik dan prosedur yang terstruktur memiliki kelemahan. Hal ini dapat menyebabkan terdapatnya *missing requirement* pada daftar *requirements* sistem yang dihasilkan.

Pada *requirements engineering*, terdapat model pendekatakan yang berorientasi *goal* dan aktor. Model pendekatan ini melibatkan tujuan pembuatan sistem dengan kebutuhan pengguna dalam suatu sistem perangkat lunak. *Requirements* yang dihasilkan dengan orientasi *goal* dan aktor akan mengarah pada suatu tujuan-tujuan dalam pembuatan sistem. *Goal Oriented Requirements Engineering* (GORE) merupakan model yang digunakan untuk mengumpulkan dan menaganalisis *requirements* awal dalam pembuatan sistem perangkat lunak pada proses *forward engineering*. Hal tersebut menjadi keunggulan untuk GORE sebagai suatu model yang baik digunakan dalam penelusuran dan analisis *requirements*. Berdasarkan hal tersebut GORE dijadikan sebagi alat bantu dalam proses *reverse engineering*. Terdapat beberapa metode di dalam model GORE, yaitu: *Keep All Objectives Satisfied* (KAOS), I*/Tropos, dan *Goal-Based Requirements Analysis Method* (GBRAM).

Penelusaran *requirements* dari Siakad UNJ perlu dilakukan untuk membantu *developer* dalam mengembangkan sistem. Penelitian dalam bidang pengembangan model pendekatan dalam proses *reverse engineering* dapat dilakukan untuk membantu dalam proses penelusuran *requirements* Siakad UNJ. Metode KAOS merupakan salah satu metode dari model GORE yang dapat digunakan untuk membantu proses *reverse engineering*. Penelitian yang dilakukan bertujuan mengembangkan sebuah model untuk proses *reverse engineering* dengan metode KAOS untuk memperoleh *functional requirements* dari Siakad UNJ. Penelitian ini berjudul "Perancangan Model GORE Menggunakan Metode KAOS untuk Proses *Reverse Engineering* Sistem Informasi Akademik Univeristas Negeri Jakarta".

2. Dasar Teori

Pada bagian ini diuraikan landasan teoretis yang berhubungan dengan penelitian atau perancangan yang dilakukan.

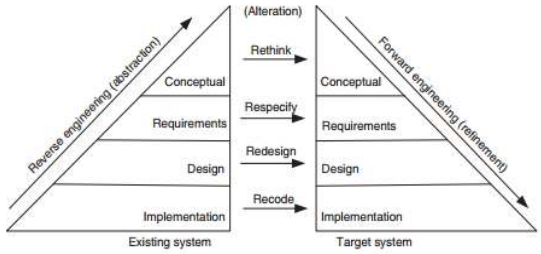
2.1. Definisi *Reverse Engineering*

Reverse engineering menurut Chikofsky (1993), diacu dalam Rizqy (2016: 9) dalam dunia perangkat lunak, *reverse engineering* digunakan untuk melakukan proses analisis sistem agar diperoleh identifikasi komponen sistem, hubungan antar komponen, dan membuat representasi sistem dalam bentuk lain atau melakukan abstraksi pada tingkat yang lebih tinggi.

2.2. Tingkat Abstraksi *Reverse Engineering*

Reverse engineering adalah salah satu tahapan dari *software reengineering*. *Reverse engineering* memiliki beberapa tingkatan abstraksi. Tingkat abstraksi akan menentukan sejauh mana suatu sistem aplikasi siap pakai akan di analisis.

Gambar 2.1 mengilustrasikan tingkat abstraksi dalam *software reengineering*.



Gambar 2.1. Model Umum dari *Software Reengineering* (Tripathy dan Naik, 2015: 138)

Gambar 2.1 menggambarkan proses untuk semua tingkat abstraksi yang ada di dalam *software reengineering*. Gambar 2.1 juga menunjukkan bahwa *software reengineering* terdiri dari tiga tahapan, yaitu: *reverse engineering*, *alteration*, dan *forward engineering*.

2.3. Siakad UNJ

Sistem Informasi Akademik Universitas Negeri Jakarta (Siakad UNJ) adalah sebuah sistem aplikasi berbasis *website* milik UNJ yang digunakan untuk menangani kegiatan administrasi akademik di lingkup universitas. Siakad UNJ membuat proses administrasi akademik berjalan secara terkomputerisasi dan mengelola berbagai kebutuhan akademik di UNJ.

Gambar 2.2. Halaman Portal Siakad UNJ

Gambar 2.2. merupakan tampilan antarmuka dari halaman portal Siakad UNJ. Halaman portal adalah halaman yang digunakan oleh *user* untuk mengisi *form login* agar dapat masuk ke dalam sistem. Halaman portal juga menampilkan beberapa informasi seperti tanggal pembayaran kuliah, tanggal pengisian KRS dan jumlah pengunjung Siakad UNJ.

2.4. Definisi *Requirement Engineering*

Requirements engineering menurut Lapouchnian (2005:1), merupakan salah satu cabang dari rekayasa perangkat lunak yang berhubungan dengan penelusuran, perbaikan, analisis, dan lain-lain dalam *requirements* sistem perangkat lunak.

2.5. *Requirement Traceability Matrix*

RTM merupakan *tools* yang digunakan dalam melakukan penelusuran *requirement*. *Requirement traceability matrix* juga dapat digunakan untuk memeriksa kesesuaian antara *requirement* dengan implementasinya pada sistem yang sudah dibuat. Penggunaan *requirement traceability matrix* dapat membantu menemukan *requirement* yang hilang. Tabel 2.1 merupakan contoh sederhana format tabel RTM yang dapat digunakan untuk melakukan penelusuran *requirement* pada suatu sistem aplikasi.

Tabel 2.2. *Template Requirement Traceability Matrix* (Shabrina, 2016: 56)

Kode *Requirement*	*Subject* Area	*Functional* Area	Relasi *Requirement*

Keterangan Tabel:
a. Kode *Requirement* : penomoran *requirement*
b. *Subject* Area : pengguna yang terlibat dalam penggunaan aplikasi
c. *Functional* Area : tema dari generalisasi *requirement* seperti *login*
d. Relasi *Requirement* : *requirement* yang saling berhubungan

2.6. *Definisi GORE*

GORE merupakan salah satu pendekatan di dalam *requirements engineering* berorientasi *goal* dan aktor. GORE menurut Adikara, dkk., (2013: 229) merupakan salah satu model pendekatan *requirements engineering* yang merasionalisasikan berbagai *requirements* yang diperlukan oleh sebuah sistem yang akan dibuat berdasarkan ujuan-tujuan yang dirumuskan sehingga diharapkan *requirements* yang didapatkan bukan hanya berdasarkan data dan proses bisnis manual.

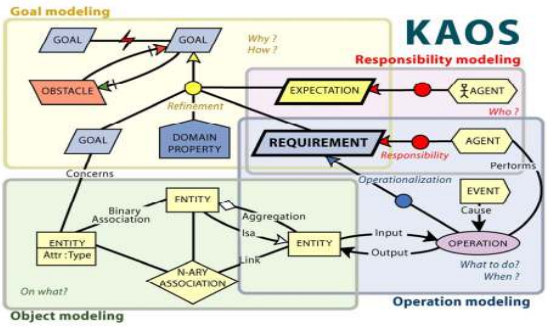
2.7. *Definisi Metode KAOS*

KAOS menurut Adikara, dkk., (2013: 230) dapat dideskripsikan sebagai sebuah kerangka kerja dari beberapa paradigma yang memungkinkan untuk mengkombinasikan beberapa tingkatan pemikiran berbeda dan disertai alasannya. Bahasa pemodelan KAOS merupakan bagian dari kerangka kerja KAOS untuk menggali (*elicitation*), menspesifikasi,

dan menganalisis tujuan (*goals*), kebutuhan (*requirements*), skenario, dan tanggung jawab tugas.

2.8. Area Kerja Pada Metode KAOS

Metode KAOS adalah salah satu metode yang terdapat di dalam model GORE. Berbeda dengan metode yang lainnya GORE memiliki area kerja yang masing-masing fokus dalam melakukan analisis pemodelan sistem. Gambar 2.3 adalah ilustrasi dari seluruh area kerja yang ada di dalam metode KAOS.



Gambar 2.3. Area Kerja Metode KAOS (Respect-IT, 2007: 45)

Berdasarkan Gambar 2.3 di dalam metode KAOS terdapat empat area kerja yang dijelaskan sebagai berikut, yaitu:

1. *Goal Modelling*
   Goal Modelling, adalah domain proses dari metode KAOS yang digunakan untuk melakukan penelusuran terhadap tujuan sistem beserta *requirements*-nya. Dalam lingkup ini juga dilakukan proses penelusuran terkait hambatan (*obstacle*) dan *expectation* dari setiap *goals* dan *sub goals*.
2. *Responsibillity Modelling*
   Responsibility Modelling, adalah domain dari metode KAOS yang digunakan untuk menelusui *agent* yang terhubung dengan *requirement* dan *expectation*.
3. *Operation Modelling*
   Operation Modelling, adalah domain dari metode KAOS yang digunakan untuk menelususri dan menganalisis *operation* yang harus dilakukan sistem terhadap *requirements* sistem. Domain ini juga menganalisis *operation* pada entitas sistem dan juga *performs agent* terhadap *operation*.
4. *Object Modelling*
   Object *Modelling,* adalah area kerja dari metode KAOS yang digunakan untuk menelusuri dan menganalisis entitas yang ada di dalam sistem dan relasi yang terdapat diantara entitas tersebut.

2.9. Elemen Pada Metode KAOS

Elemen yang terdapat di dalam metode KAOS menurut Adikara, dkk., (2013: 230) meliputi istilah berikut, yaitu: (1) Tujuan (*goal*), tujuan (*goal*) adalah kumpulan perilaku/keadaan yang harus dipenuhi atau dapat diterima oleh sistem dalam sebuah kondisi yang ditetapkan. Definisi *goal* harus jelas sehingga dapat diverifikasi apakah sistem mampu memenuhi atau memuaskan *goal* tersebut. (2) *Softgoal*, digunakan untuk mendokumentasikan perlaku alternative dari sistem, sehingga tidak secara tegas dapat diverifikasi tingkat kepuasannya. Tingkat kepuasan dari *softgoal* akan dibatasi menggunakan limitasi yang ditetapkan. (3) *Agents*, adalah sebuah jenis dari objek yang bertindak sebagai pemroses kegiatan operasional. Agent merupakan komponen aktif dapat berupa manusia, perangkat keras, perangkat lunak, dan lainnya yang mempunyai peran spesifik dalam memuaskan sebuah tujuan.

3. Metodologi

3.1. Tempat dan Waktu Penelitian

Penelitian ini dilakukan di Gedung D, Unit Pelayanan Teknis Teknik Informatika dan Komputer Universitas Negeri Jakarta (UPT TIK – UNJ) yang berlokasi di Jl. Rawamangun Muka Jakarta Timur 13220. Penelitian ini dilakukan pada bulan Desember 2016 hingga bulan Mei 2017.

3.2. Alat dan Bahan Penelitian

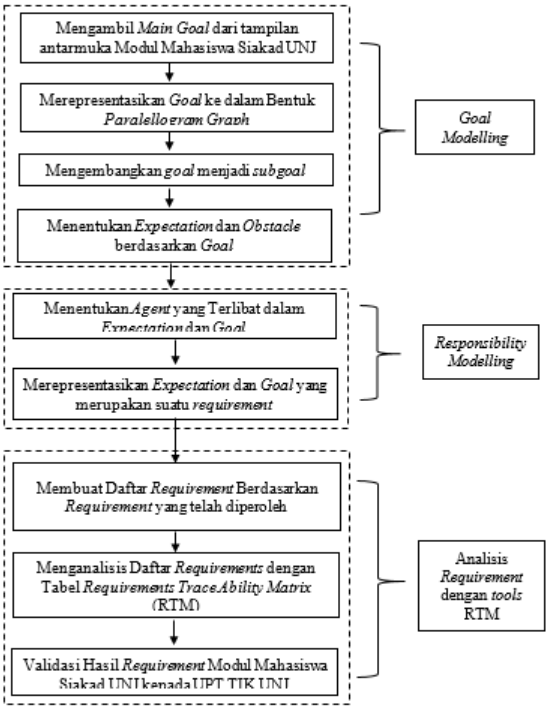
Penelitian ini bertujuan untuk mendapatkan *functional requirement* dari Siakad UNJ yang berkaitan dengan fungsi Siakad UNJ bagi jenis pengguna mahasiswa, sehinga yang menjadi objek atau bahan penelitian ini adalah Modul Mahasiswa Siakad UNJ.

Alat yang digunakan dalam penelitian ini adalah sebagai berikut:

1. Perangkat Keras (*Hardware*)
   a. *Processor* Intel ® Celeron ® CPU N2840 @ 2.16 GHz;
   b. *Memory* RAM 2 GB DDR3.
2. Perangkat Lunak *(Software*)
   a. *Windows 8.1 Pro* 64-bit © 2013;
   b. *Microsoft Office Professional Plus* 2016 64 bit versi *en-us*;
   c. *Edraw Max* 7.9.2

3.3. Diagram Alir Penelitian

Gambar 3.1 adalah diagram alir penelitian untuk model *reverse engineering* dengan model GORE dan metode KAOS untuk mendapatkan *functional requirement*:



Gambar 0.3. Diagram Alir Penelitian

Berikut ini merupakan deskripsi dari tiap tahapan dari diagram alir penelitian yang telah diilustrasikan pada Gambar 3.1:

1. Mengambil *Main Goal*
   Mengambil *main goal* adalah sebuah kegiatan menganalisis tampilan antar muka dari sistem aplikasi perangkat lunak dan menyimpulkan secara umum layanan yang diberikan sistem kepada *stakeholder*. *Main goal* merupakan dasar dari *requirements* yang muncul untuk memenuhi *goal* tersebut. *Main goal* dalam suatu sistem aplikasi perangkat lunak secara sederhana dapat dilihat dari menu yang ada pada tampilan antar muka sistem perangkat lunak dan generalisasi dari fitur yang ada.
2. Merepresentasikan *Goal* dalam Bentuk *Paralelogram Graph*
   Setelah mendapatkan *main goal* representasikan *main goal* ke dalam bentuk *paralelogram graph*. *Paralelogram graph* merupkan grafik yang menyerupai bentuk pohon bercabang yang akan memetakan jangkauan sebuah *goal* pada suatu sistem aplikasi. *Graph* tersebut akan menjadi hasil dekomposisi dari *main goal* yang ada pada sistem. *Goal* akan berbentuk sebagai *paralelogram* yang dapat menjadi sebuah *node* atau *leaf* pada grafik yang kemudian akan di dekomposisi kembali menjadi subgoal.
3. Mengembangkan *Goal* Menjadi Subgoal
   Pada tahapan ini *Main goal* atau *goal* yang sudah didapat sebelumnya dikembangkan atau didekomposisi menjadi *subgoal*. *Subgoal* merupakan sebuah *goal* yang diturunkan dari *parent goal*, dimana pemenuhan semua *subgoal* tersebut harus dilakukan agar *parent goal* dapat tercapai. Pada setiap *goal* memiliki kemungkinan untuk didekomposisi menjadi satu *subgoal* atau lebih atau bahkan tidak dapat didekomposisi. *Subgoal* analisis dihasilkan dari perluasan *main goal* yang sebelumnya sudah dijelaskan. *Subgoal* secara sederhana memperluas jangkauan *goal* hingga sampai kepada rincian *task* yang dilakukan oleh *stakeholder*. Proses dekomposisi pada *subgoal* akan berhenti apabila sudah mendapat tujuan yang dapat didelegasikan kepada komponen sebuah sistem aplikasi.
4. Menentukan *Expectation* dan *Obstacle*
   Tahap selanjutnya adalah menentukan *obstacle* dan *expectation* pada setiap *goal* yang sudah didapat sebelumnya. *Obstacle* adalah sebuah kondisi yang dapat mencegah pencapaian suatu *goal*. *Obstacle* juga dapat dikatakan sebagai suatu keadaan yang tidak diinginkan terjadi pada sistem. *Expectation* adalah salah satu jenis *goal* yang berupa sebuah kondisi yang dinginkan oleh *agent* yang merupakan bagian dari lingkungan sistem. Pada metode KAOS *obstacle* ditandai dengan bentuk *paralellogram* berwarna merah, sedangkan *expectation* ditandai dengan bentuk *paralellogram* berwarna kuning. Penentuan *obstacle* bertujuan untuk mempersiapkan solusi terkait kendala yang akan muncul sedangkan penentuan *expectation* bertujuan untuk mendapatkan *requirement* sesuai dengan *agent* yang berada pada lingkungan sistem.
5. Menentukan *Agent* yang Terlibat dalam *Expectation* dan *Goal*
   Pada tahap ini *expectation* dan *goal* yang sudah didapat sebelumnya dihubungkan dengan *agent* yang ada di dalam sistem aplikasi. Tujuan dari penentuan *agent* ini adalah agar setiap *expectation* dan *goal* dapat ditelusuri *agent* yang terlibat di dalam suatu *expectation*.
6. Merepresentasikan *Goal* Menjadi *Requirements*
   Goal yang menjadi sebuah *requirements* adalah sebuah *goal* yang menjadi *leaf* dalam *parralelogram graph* dan berhubunan dengan *agent*. *Expectation* termasuk ke dalam jenis *goal* yang berhubungan dengan *agent* dan dapat menjadi sebuah *requirement*. *Goal* yang menjadi *requirements* direpresentasikan dengan *parralelogram* yang memiliki garis pinggir tebal.
7. Membuat Daftar *Requirement*
   Setelah pembutan *parallelogram graph* selesai dan tahapan *responsibility modelling* selesai, tahapan selanjutnya adalah membuat *daftar requirements*. *Goal* dan *expectation* yang sudah didapat pada *parallelogram graph* di berikan kode dan dibuat menjadi *daftar requirements*. Data yang ada di dalam daftar *requirement* meliputi kode, *requirement*, agent dan area fungsional dari *requirement*.
8. Analisis Daftar *Requirement* Menggunakan RTM
   Tahapan selanjutanya adalah menganalisis daftar *requirements* yang sudah dibuat menggunakan

tabel RTM. Data yang ada di dalam daftar, dimasukan ke dalam tabel RTM. Tabel RTM yang digunakan dapat dilihat pada Tabel 3.1. Penggunaan tabel RTM dapat memberikan informasi setiap keterkaitan *requirement* yang ada di dalam sistem terhadap pengguna, area fungsional dan antar *requirement*.

9. Validasi
   Setelah proses analisis dengan menggunakan tabel RTM, proes selanjutnya adalah memvalidasi hasil *requirements* dan menyerahkan hasil analisis dengan menggunakan tabel RTM kepada pihak UPT TIK UNJ. Proses ini akan dilakukan bersama dengan Kepala dan *developer* UPT TIK UNJ.
10. Kesimpulan
    Setelah proses validasi selesai dilakukan dan hasil *requirement*s yang didapat telah disetuji oleh pihak UPT TIK UNJ, tahapan selanjutnya adalah menentukan kesimpulan dari proses yang telah dilakukan untuk mendapatkan model *reverse engineering* dengan menggunakan model GORE dan metode KAOS.

3.4. Teknik Analisis Data

Proses analisis data yang digunakan dalam penelitian ini adalah dengan menggunakan *Requirement Traceability Matrix* (RTM). Data *requirements* yang telah diperoleh, kemudian diberikan kode untuk dilakukan analisis keterkaitan antar *requirements*. Berikut ini adalah format tabel *Requirement Traceability Matrix* yang digunakan untuk menganalisis data dalam penelitian ini:

Tabel 2.4. Tabel *Requirement Traceability Matrix* (Shabrina, 2016: 56)

Kode *Requirement*	*Subject* Area	*Functional* Area	Relasi *Requirement*

Keterangan Tabel:

a. Kode *Requirement* : penomoran *requirement*
b. *Subject* Area : pengguna yang terlibat dalam penggunaan aplikasi
c. *Functional* Area : tema dari generalisasi *requirement* seperti *login*
d. Relasi *Requirement* : *requirement* yang saling berhubungan

4. Hasil dan Analisis

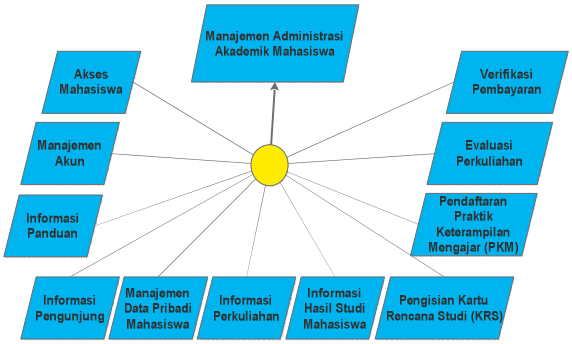
Hasil penelitian yang telah dilakukan pada penelitian ini adalah sebuah model *reverse engineering* dengan menggunakan model GORE dan metode KAOS untuk melakukan analisis dan penelusuran terhadap sistem siap pakai untuk mendapatkan *requirement functional* sistem.

4.1. Hasil Pengambilan *Main Goal*

Berdasarkan tampilan antarmuka Modul Mahasiswa dari aplikasi Siakad UNJ, dapat diketahui bahwa tujuan utama dari Modul Mahasiswa siakad UNJ adalah "Manajemen Administrasi Akademik Mahasiswa" yang mencakup *main goal* sebagai berikut, yaitu: (1) Akses mahasiswa; (2) Manajemen akun; (3) Informasi panduan; (4) Informasi pengunjung; (5) Manajemen data pribadi mahasiswa; (6) Informasi perkuliahan; (7) Informasi hasil studi mahasiswa; (8) Pengisian kartu rencana studi (KRS); (9) Pendaftaran praktik keterampilan mengajar (PKM); (10) Evaluasi perkuliahan; (11) Verifikasi pembayaran.

4.2. Hasil Representasi *Goal* dalam Bentuk *Parallelogram Graph*

Main goal yang sudah didapat berdasarkan tampilan antarmuka Modul Mahasiswa dari Siakad UNJ dibuat ke dalam bentuk *parralellogram graph* yang diilustrasikan pada Gambar 4.1.



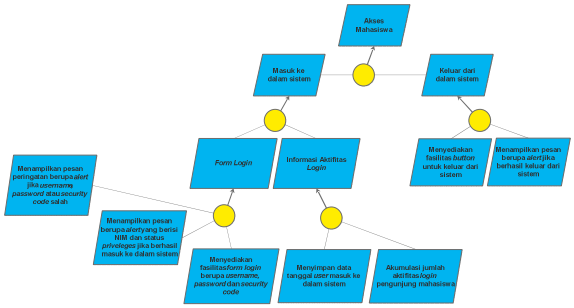
Gambar 4.1 Representasi *Main Goal* ke dalam *Parralellogram Graph*

Parralellogram graph pada Gambar 4.1 merupakan hasil dari *requirements elicitation* pada tampilan anatarmuka aplikasi siap pakai yaitu Modul Mahasiswa Siakad UNJ. Berdasarkan Gambar 4.1 dapat diketahui untuk memenuhi tujuan "Manajemen Administrasi Akademik Mahasiswa" Siakad UNJ harus memenuhi sebelas *goal*. Empat *goal* diantaranya, yaitu: akses mahasiswa, manajemen akun, informasi panduan, dan informasi pengunjung adalah *goal* yang ada karena munculnya Siakad UNJ. Tujuh *goal* lainnya adalah proses administrasi akademik yang akan ditangani oleh sistem. Jika terdapat satu saja goal yang tidak tercapai maka tujuan "Manajemen Administrasi Akademik Mahasiswa" juga tidak akan tercaai. Hasil representasi *main goal* pada Gambar 4.1 akan menjadi *graph* dasar yang kemudian setiap *main goal* akan di dekomposisikan kembali menjadi *subgoal* hingga mendapatkan *requirement* sistem.

4.3. Hasil Analisis Subgoal

Berdasarkan tampilan antarmuka Siakad UNJ, *subgoal* "akses mahasiswa" yang ditangani oleh sistem terdiri dari akses masuk ke sistem dan keluar dari sistem. Akses masuk ke dalam sistem dicapai dengan memenuhi *goal* *"form login"* dan *goal* "informasi aktifitas *login*. *Form login* yang harus dicapai sistem, yaitu: (1) Menyediakan fasilitas *form login* berupa *username*, *password*, dan *security code;* (2) Menampilkan pesan berupa *alert* yang berisi NIM dan *password* jika berhasil masuk ke dalam sistem; (3) Menampilkan pesan peringatan berupa *alert* jika *username, password atau security code* salah.

Untuk informasi aktifitas *login,* yang harus dicapai sistem, yaitu: (1) Menyimpan data tanggal *user* masuk ke dalam sistem; (2) Akumulasi jumlah aktifitas masuk ke dalam sistem yang dilakukan oleh *user*. Akses keluar dari dalam sistem yang harus dipenuhi oleh sistem meliputi: (1) Menyediakan fasilitas *button* untuk ke luar dari dalam sistem; (2) Menampilkan pesan berupa *alert* jika berhasil keluar dari dalam sistem. Gambar 4.2 merupakan *parralelogram graph* hasil dari *refinement subgoa*l akses mahasiswa:



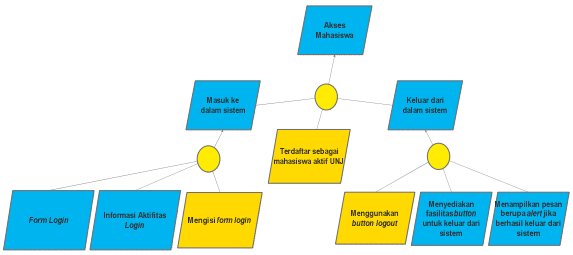
Gambar 4.2 Analisis *Subgoal* Akses Mahasiswa

4.4. Hasil Analisis *Expectation* dan *Obstacle*

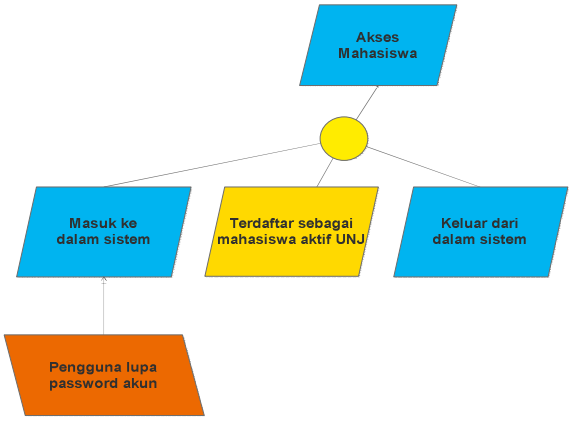
Berdasarkan tampilan antarmuka Siakad UNJ, *goal* "akses mahasiswa" memiliki tiga *expectation* yang harus dilakukan oleh mahasiswa sebagai *user* agar *goal* "akses mahasiswa" dapat tercapai. *Expectation* tersebut, yaitu: (1) Terdaftar sebagai mahasiswa aktif UNJ; (2) Mengisi *form login*; (3) Menekan *button logout*.

Expectation "terdaftar sebagai mahasiswa aktif UNJ" merupaka sebuah skenario atau kondisi berupa fakta yang harus dilakukan oleh jenis *user* mahasiswa untuk memenuhi goal "akses mahasiswa". *Expectation* "mengisi form login" merupakan kondisi atau fakta yang harus dilakukan *user* agar dapat memenuhi *goal* "masuk ke dalam sistem". Expectation "menekan *button logout*" adalah skenario atau kondisi berupa fakta yang harus dilakukan oleh *user* agar dapat memenuhi *goal* "keluar dari dalam sistem".

Goal "masuk ke dalam sistem" memiliki *obstacle* atau kondisi yang menghalangi pencapaian *goal,* yaitu: pengguna lupa *password* akun. Gambar 4.3 dan Gambar 4.4 berikut ini adalah *parralelogram graph* hasil dari analisis *expectation* dan *obstacle* dari *goal* "akses mahasiswa".



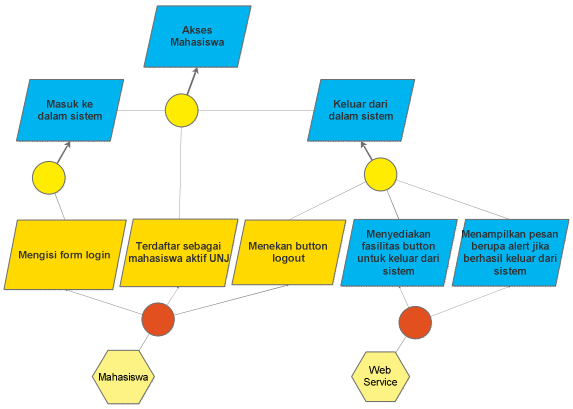
Gambar 4.3 Analisis *Expectation* pada *goal* Akses Mahasiswa



Gambar 4.4 Analisis *Obstacle* pada *goal* Akses Mahasiswa

4.5. Hasil Analisis *Agent*

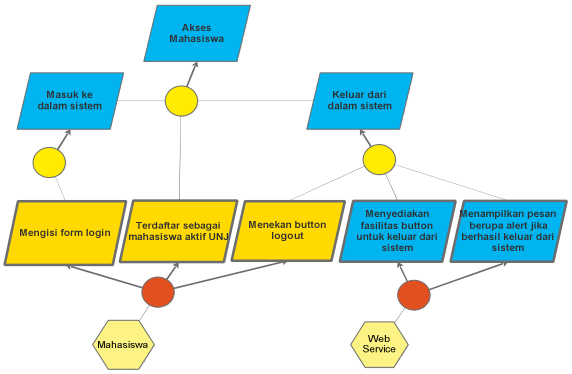
Berdasarkan tampilan anatarmuka sistem aplikasi Modul Mahasiswa Siakad UNJ dan proses analisis dengan menggunakan metode KAOS didapatkan hasil *agent* yang berperan dalam *functional requirement* sistem aplikasi siap pakai Modul Mahasiswa Siakad UNJ adalah *web service* dan mahasiswa. *Agent* di dalam metode KAOS adalah subjek yang bertanggung jawab dalam memenuhi suatu *requirement*. *Agent* di dalam metode KAOS digambarkan dengan menggunakan bentuk *hexagon* berwarna kuning muda. Hasil analisis *agent* untuk setiap *subgoal* tercantum di dalam lampiran. Gambar 4.3 berikut ini adalah contoh *parralellogram graph* hasil analisis *agent*:



Gambar 4.5 Analisis *Expectation* Akses Mahasiswa

4.6. Hasil Representasi *Requirement*

Requirement di dalam *parralelogram graph* merupakan sebuah *goal* yang berupa tugas yang harus dilakukan oleh *agent* untuk memenuhi pencapaian suatu *goal*. *Requirement* di dalam *parralelogram graph* digambarkan dengan menggunakan garis border tebal pada *goal* yang menjadi suatu *requirement* sistem. Hasil representasi *requirement* untuk setiap *subgoal* tercantum di dalam lampiran. Gambar 4.6 adalah contoh *parralellogram graph* hasil representasi requirement.



Gambar 4.6 Hasil Representasi *Requirement* pada *Goal* Akses Mahasiswa

4.7. Hasil Model *Reverse Engineering*

Berdasarkan diagram alir penelitian yang telah dilakukan dan hasil dari setiap tahapan, maka didapat hasil dari penelitian ini adalah model *reverse engineering* dengan menggunakan model GORE dan metode KAOS. Berikut ini merupakan model *reverse engineering* untuk mendapatkan *requirement* dari sistem perangkat lunak siap pakai yang dibuat dengan menggunakan model GORE dan metode KAOS:

Gambar 4.7 Hasil Representasi *Requirement* pada *Goal* Akses Mahasiswa

4.1 Pembahasan

Setelah melalui proses *reverse engineering* dengan menggunakan model GORE dan metode KAOS dengan tahapan yang diilustrasikan pada Gambar 3.1 didapatkan hasil *parralelogram graph* yang menggamabarkan *goal* dan *functional requirement* sistem. *Parralelogram graph* dibuat berdasarkan tampilan antarmuka sistem aplikasi Modul Mahasiswa Siakad UNJ dengan menganalisis menu dan fitur yang disediakan oleh sistem.

Parralelogram graph adalah *graph* berbetuk jajar genjang dan terssusun seperti pohon. *Graph* ini digunakan untuk menguraikan *goal* yang ingin dicapai sistem hingga menjadi sebuah *requirement* yang harus dilakukan oleh *agent*. *Agent* di dalam metode KAOS merupakan sebuah subjek yang bertanggung jawab untuk memenuhi suatu *requirement*. *Agent* di dalam KAOS tidak hanya *user* yang menggunakan sistem, melainkan juga *function* atau *web service*. Berdasarkan tampilan antarmuka Modul Mahasiswa Siakad UNJ diketahui *agent* yang terdapat di dalam sistem adalah *mahasiswa* dan *web service*.

Parralelogram graph adalah *graph* yang terdiri dari *goal*, *expectation*, dan *obstacle*. *Goal* adalah tujuan yang hendak dicapai sistem dan digamabarkan dengan bentuk jajar genjang berwarna

biru. *Expectation* adalah sebuah skenario atau tugas yang harus dilakukan oleh *agent* agar suatu *goal* dapat diselesaikan. Sebuah *goal* dapat diuraikan dan dihubungkan dengan parent goal dengan menggunakan penghubung lingkaran berwarna kuning. *Expectation* digambarkan dengan bentuk jajar genjang berwarn kuning. *Obstcale* adalah sebuah kondisi yang membuat pencapaian suatu *goal* menjadi terganggu. *Obstacle* digambarkan dengan bentuk jajar genjang terbalik berwarna merah. Goal dan expecatation dihubungkan dengan agent menggunakan penghubung berbentuk lingkaran berwarna merah.

Hasil utama dari penelitian ini adalah model *reverse engineering* dengan menggunakan model GORE dan metode KAOS yang didapat setelah melakukan penerapan model GORE dan metode KAOS pada sebuah sistem aplikasi siap pakai, yaitu Modul Mahasiswa Siakad UNJ. Model yang dihasilkan diilustrasikan dalam bentuk diagram pada Gambar 4.5.

5. Kesimpulan dan Saran

5.1. Kesimpulan

Berdasarkan hasil pembahasan penelitian, maka dapat disimpulkan bahwa perancangan model GORE dengan menggunakan metode KAOS untuk proses reverse engineering dapat dibuat dan diterapkan untuk mendapatkan *functional requirement*. Hasil model GORE dengan menggunakan metode KAOS yang telah dibuat diterapkan pada Modul Mahasiswa Siakad UNJ dengan tujuan untuk mendapatkan *functional requirement* dan membantu *developer* dalam proses pengembangan Siakad UNJ.

Berikut ini adalah tahap pembuatan model GORE dengan menggunakan metode KAOS untuk proses *reverse engineering*, yaitu:

1. Menentukan sistem aplikasi perangkat lunak yang akan dianalisis
2. Mengambil *main goal* dari tampilan antarmuka sistem aplikasi perangkat lunak
3. Merepresentasikan *goal* ke dalam *paralellogram graph*
4. Mengembangkan *goal* menjadi *subgoal*
5. Menentukan *expectation* dan *obstacle* berdasarkan *goal*
6. Menentukan *agent* yang terlibat dalam *expecatation* dan *goal*
7. Merepresentasikan *expectation* dan *goal* yang merupakan suatu *requirement*
8. Membuat daftar *requirement* berdasarkan hasil *parralelogram graph*
9. Menganalisis keterkaitan antar *requirement* dengan menggunakan RTM
10. Validasi hasil *requirement* kepada UPT TIK UNJ

5.2. Saran

Berdasarkan hasil pembahasan dan kesimpulan pada skripsi ini, kepada peneliti lain yang akan melakukan penelitian sejenis disarankan, yaitu:

1. Melakukan penelitian dalam bidang *reverse engineering* dengan menggunakan metode yang berbeda
2. Melakukan perbandingan hasil *requirement* yang diperoleh dengan menggunakan metode – metode yang ada di dalam model GORE
3. Melakukan penelitian dengan menggunakan seluruh area kerja yang ada di dalam metode KAOS yaitu *Goal Modelling, Responsibility Modelling, Object Modelling, dan Operation Modelling*
4. Melakukan penelitian pada modul dosen, admin dan modul lainnya untuk memperoleh *requirement* Siakad UNJ secara utuh.

Daftar Pustaka:

Adikara, F.; Sitohang, B.; Hendradjaya, B. (2013). *Penerapan Goal Oriented Requirements Engineering (GORE) Model (Studi Kasus: Penggembangan Sistem Informasi Penjaminan Mutu Dosen (SIPMD) Pada InstitusiPendidikan Tinggi*, Seminar Nasional Informasi Indonesia.

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

Lapouchnian, Alexei. (2005). *Goal-Oriented Requirements Engineering: An Overview of the Current Research*. Department of Computer Science, University of Toronto.

[Respect-IT] Respect-IT. (2007). A KAOS Tutorial.

Rizqy, F. 2016. Perancangan Model *Goal Oriented Requirements Engineering* (GORE) Untuk Proses *Reverse Engineering* [Skripsi]. Jakarta: Fakultas Teknik, Universitas Negeri Jakarta.

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

Sommerville, I. (2011). *Software Engineering: 9th Edition*. Boston: Addison- Wesley.

Tripathy, Priyadarshi & Naik, Kshirasagar. 2015. *Sofware Evolution and Maintenance*. New Jersey: John Wiley & Sons.