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

Skripsi

HAFIEZ ARIEF RAHARJO

5235134427

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)	.....	14-8-17
Hamidillah Ajie, S.Si., M.T. (Dosen Pembimbing II)	.....	14-08-2017

PENGESAHAN PANITIA UJIAN SKRIPSI

NAMA DOSEN	TANDA TANGAN	TANGGAL
Drs. Bachren Zaini, M.Pd. (Ketua Penguji)	.....	14-08-2017
Bambang P. Adhi, M.Kom. (Sekertaris)	.....	14-08-2017
Vina Oktaviani, M.T. (Dosen Penguji Ahli)	.....	14-08-2017

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, 14 Agustus 2017

Yang Membuat Pernyataan

Hafiez Arief Raharjo

5235134427

ABSTRAK

HAFIEZ ARIEF RAHARJO, Perancangan Model GORE Menggunakan Metode KAOS untuk Proses *Reverse Engineering* Sistem Informasi Akademik Universitas Negeri Jakarta. Pembimbing Widodo, M.Kom. dan Hamidillah Ajie, S.Si., M.T.

Perubahan dan penambahan *requirement* pada suatu sistem aplikasi perangkat lunak membuat pengembangan pada sistem aplikasi terus dilakukan. 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 *requirement*

ABSTRACT

HAFIEZ ARIEF RAHARJO, GORE Design Models Using KAOS Method for Reverse Engineering Process In State University Of Jakarta's Academic Information System. Supervisor Widodo, M.Kom. and Hamidillah Ajie, S.Si., M.T.

Changes and attempt to add a *requirements* on a software application system for development of the said application system always continues to be done. This made documentation of system requierement more important for the effort of further system development and fulfillment of the requirements given by stakeholders. This research purpose is to create a reverse engineering model, using the *Goal Oriented Requirement Engineering* (GORE) model and *Keep All Objectives Satisfied* (KAOS) method as an aids to analyze and searching a functional requirement on availabele software system. The research resulting a diagram that explain the use of GORE model and KAOS method for reverse engineering model in a software application system. The stage of reverse engineering using GORE model and KAOS method consist of: (1) taking a main goal from the interface view of application system; (2) Represents the goal to parralellogram graph; (3) Develop the goals into subgoal; (4) Determine the expectation and obstacle based on the goal; (5) Decide which agent get involved in the expectation and goal; (6) Represents the expectation and goal as a requirement. The reverse engineering model which use GORE model and KAOS method successfully applied on the application sample of Jakarta State University Student Module, and get 110 requirement system.

Keywords: reverse engineering, GORE, KAOS, and requirement

KATA PENGANTAR

Puji dan syukur penulis panjatkan kehadirat Allah SWT yang telah memberikn rahmat, karunia, dan hidayah-Nya, sehingga penulis dapat menyelesaikan skripsi dengan judul **"Perancangan Model GORE Menggunakan Metode KAOS untuk Proses *Reverse Engineering* Sistem Informasi Akademik Universitas Negeri Jakarta"**, yang merupakan salah satu syarat untuk meraih gelar Sarjana Pendidikan Teknik Infromatika dan Komputer pada Fakultas Teknik, Universitas Negeri Jakarta.

Dalam merencanakan, menyusun, dan menyelesaikan skripsi ini, penulis banyak menerima bantuan, bimbingan, dan motivasi serta dukungan dari berbagai pihak. Oleh karena itu, pada kesempatan ini penulis bermaksud mengucapkan terima kasih kepada:

1. Ibu Dr.Yuliatri Sastrawijaya, M.Pd selaku Ketua Program Studi Pendidikan Teknik Informatika dan Komputer, Universitas Negeri Jakarta
2. Bapak Prasetyo Wibowo Yunanto S.T., M.Eng. selaku pembimbing akademik
3. Bapak Widodo, M.Kom., selaku dosen pembimbing yang telah memberikan bimbingan dan arahan sehingga penulis dapat menyelesaikan skripsi ini
4. Bapak Hamidilllah Ajie S.Si., M.T., selaku dosen pembimbing yang telah memberikan bimbingan dan arahan sehingga penulis dapat menyelesaikan skripsi ini
5. Orangtua dan pihak keluarga yang tiada henti memanjatkan doa dan memberikan semangat, dukungan, dan motivasi kepada penulis dalam menyelesaikan proposal penelitian Skripsi ini
6. Para sahabat mahasiswa program studi Pendidikan Teknik Informatika dan Komputer angkatan 2013 yang setia memberikan bantuan dalam menyelesaikan skripsi ini.

Penulis menyadari bahwa skripsi ini jauh dari kesempurnaan, untuk itu penulis mohon maaf apabila terdapat kekurangan dan kesalahan. Akhir kata penulis berharap agar penelitian dan penyususnan skripsi ini dapat bermanfaat bagi pembaca pada umumnya, dan bagi penulis khususnya.

Jakarta, 14 Agustus 2017

Penulis

DAFTAR ISI

Halaman

DAFTAR TABEL

Tabel Halaman

DAFTAR GAMBAR

Gambar Halaman

DAFTAR LAMPIRAN

Lampiran Halaman

BAB I

PENDAHULUAN

1.1. Latar Belakang Masalah

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 yang tidak dapat beradaptasi terhadap perubahan yang terjadi di sekitarnya, akan cenderung ditinggalkan oleh para penggunanya. Hal tersebut berkaitan dengan tingkat kepuasan pengguna dalam menggunakan sistem perangkat lunak. Pengguna yang merasa puas akan tetap menggunakan suatu sistem perangkat lunak untuk membantu aktivitasnya. Sebaliknya, pengguna yang merasa tidak puas terhadap pemenuhan kebutuhan yang dimiliki oleh sistem perangkat lunak, akan mencari sistem perangkat lunak lain yang dapat memenuhi kebutuhannya.

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.

Berperan sebagai sistem yang mengelola kegiatan administrasi akademik, membuat Siakad UNJ menjadi salah satu sistem perangkat lunak yang sangat fundamental bagi UNJ dalam melaksanakan kegiatan akademik. Pentingnya peran Siakad UNJ membuat proses pengembangan dan pemeliharaan sistem terus dilakukan.

Banyaknya pengguna dan bertambahnya kebutuhan pengguna, membuat munculnya *requirements* baru yang harus dipenuhi oleh *developer*. Hal ini membuat proses pengumpulan dan pemahaman *requirements* merupakan suatu proses yang penting dalam pengembangan perangkat lunak. Pada umumnya dalam pengembangan perangkat lunak, daftar *requirements* sistem aplikasi perangkat lunak ditulis di dalam dokumentasi sistem. Dokumen terkait *requirements*, merupakan dokumen yang penting agar proses pengembangan sistem dapat berjalan sesuai dengan *requirements* awal dan dapat memenuhi kebutuhan penggunanya.

Pada umumnya keberhasilan suatu sistem aplikasi perangkat lunak dapat dilihat dari seberapa besar sistem tersebut dalam memenuhi *requirements* dan tujuan pembuatannya. Semakin terpenuhinya *requirements* maka akan semakin tinggi tingkat keberhasilan dari suatu sistem perangkat lunak. Hal tersebut membuat proses pengumpulan dan pemahaman *requirements* sistem menjadi proses yang sangat krusial dalam proses pengembangan sistem perangkat lunak.

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”.

1.2. Identifikasi Masalah

Berdasarkan latar belakang yang telah dipaparkan sebelumnya, maka dapat diidentifikasi masalah yang terkait pada penelitian ini adalah:

1. Keinginan *stakeholder* untuk terus menggunakan suatu sistem aplikasi perangkat lunak menyebabkan proses pengembangan sistem terus dilakukan;
2. Sistem yang tidak dapat beradaptasi terhadap perubahan dan memenuhi kebutuhan *stakeholder* akan ditinggalkan oleh para penggunanya;
3. Siakad UNJ adalah sistem aplikasi yang kompleks karena terdiri dari beberapa modul atau bagian sistem, sehingga menyebabkan sulitnya proses pengembangan sistem;
4. Dalam proses pengembangan dan pemeliharaan Siakad UNJ diperlukan dokumentasi terkait *requirements* sistem;
5. *Reverse engineering* yang pada umumnya dilakukan hanya dengan mengambil langsung dari tampilan antarmuka sistem aplikasi perangkat lunak berpotensi mengalami *missing requirements* pada daftar *requirements* yang dihasilkan.

1.3. Pembatasan Masalah

Karena luasnya lingkup permasalahan yang diidentifikasi sebelumnya, maka pembatasan masalah perlu dilakukan. Pembatasan masalah dalam penelitian ini adalah merancang sebuah model untuk proses *reverse engineering* dalam melakukan penelusuran terhadap *requirements* dari suatu sistem aplikasi perangkat lunak dengan menggunakan metode KAOS serta menerapkannya pada Modul Mahasiswa dari Siakad UNJ untuk mendapatkan *functional requirements*.

1.4. Perumusan Masalah

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

"Bagaimana merancang model *Goal Oriented Requirements Engineering* (GORE) menggunakan metode *Keep All Objectives Satisfied* (KAOS) untuk proses *reverse engineering* Modul Mahasiswa Sistem Informasi Akademik Universitas Negeri Jakarta (Siakad UNJ)?"

1.5. Tujuan Penelitian

Tujuan dari penelitian ini adalah untuk merancang model GORE menggunakan metode KAOS untuk melakukan proses *reverse engineering* dan mendapatkan *functional requirements* dari Modul Mahasiswa Siakad UNJ.

1.6. Manfaat Penelitian

Berikut adalah manfaat dari penelitian ini, yaitu:

1. Menghasilkan model *reverse engineering* yang dapat digunakan untuk membantu penelusuran *requirements* suatu sistem aplikasi perangkat lunak;
2. Menghasilkan dokumentasi sistem terkait *functional requirements* dari Modul Mahasiswa Siakad UNJ;
3. Hasil *requirements* dari Modul Mahasiswa Siakad UNJ dapat membantu *develope*r UPT TIK-UNJ dalam mengembangkan Siakad UNJ;
4. Dapat menjadi acuan bagi pembaca apabila ingin melakukan *reverse engineering* pada suatu sistem perangkat lunak.

BAB II

TINJAUAN PUSTAKA

2.1. Kerangka Teoritik

2.1.1. *Reverse Engineering* Siakad UNJ

2.1.1.1. Definisi *Reverse Engineering*

Reverse engineering menurut McClure (1992), diacu dalam Rizqy (2016: 9) adalah proses analisis sebuah sistem perangkat lunak untuk merekonstruksi sebuah deskripsi komponen dan hubungan antar komponennya, deskripsi *high-level* dari sebuah program diperoleh dari *low-level*-nya. Tujuan dari *reverse engineering* adalah untuk mendokumentasi ulang sistem dan menemukan informasi desain sebagai alat bantu dalam meningkatkan kemampuan pemahaman suatu program.

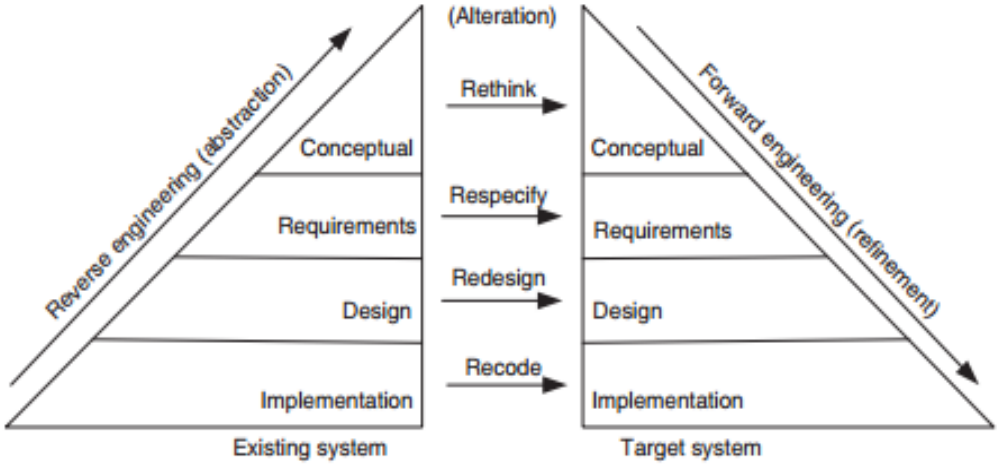
Reverse engineering menurut Simarta (2010: 404) diacu dalam Rizqy (2016: 9) adalah proses pengembangan sekumpulan spesifikasi untuk sistem perangkat keras yang rumit dengan urutan pengujian *specimen* dari sistem tersebut. Proses analisis sebuah sistem yang ada digunakan untuk mengidentifikasi komponen-komponen dan hubungannya, dan membuat representasi dari sistem dalam bentuk lain atau pada tingkat yang lebih tinggi dari abstrak.

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.

Jadi, dapat disimpulkan bahwa *reverse engineering* adalah suatu proses identifikasi dan analisis suatu sistem perangkat lunak yang sudah ada sebelumnya untuk mengidentifikasi komponen-komponen dan hubungan antara komponennya dengan tujuan mendokumentasi ulang, menemukan informasi sistem dan membuat representasi sistem dalam bentuk lain untuk meningkatkan kemampuan sistem.

2.1.1.2. 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*.

Menurut Tripathy dan Naik (2015: 136) konsep *abstraction* dan *refinement* digunakan untuk membuat model pengembangan *software* berdasarkan suatu urutan tahapan, dimana pada setiap tahapan terdapat empat tingkatan spesifik, yaitu:

> *"(1) Conceptual Level, berada pada level abstraksi tertinggi. Perangkat lunak dijelaskan dengan konsep high-level dan membahas mengenai alasan keberadaan sistem (Why?) dengan kata lain Conceptual level membahas aspek "mengapa" dari sistem dengan menjawab pertanyaan: "Mengapa sistem tersebut dibuat?"; (2) Requirements Level, karakteristik fungsional dari sistem (What?) digambarkan dengan konsep high-level dan menghasilkan daftar karakteristik fungsional sistem secara detail dengan kata lain Requirements Level menjelaskan aspek "Apa" dari sistem dengan menjawab pertanyaan: "Apa yang dapat lakukan oleh sistem?"; (3) Design Level, karakteristik sistem (apa dan bagaimana?) komponen utama sistem, yaitu: gaya arsitektur, antar muka, algoritma, struktur data internal utama, dan database dijelaskan secara rinci dengan kata lain Design Level membahas lebih banyak aspek "what" dan "how" dari sistem dan menjawab pertanyaan: "Apa karakteristik sistemnya?" dan "Bagaimana sistem akan memiliki karakteristik untuk memberikan fungsionalitasnya?"; (4) Implementation Level adalah tingkatan abstraksi terendah dalam hirarki. Pada tingkatan ini, detail informasi sistem digambarkan dengan istilah dalam bahasa tingkat tinggi dengan kata lain, Implementation Level menjelaskan "How" bagaimana tepatnya sistem akan diimplementasikan."*

Software reengineering menggambarkan satu siklus pengambilan sebuah sistem yang ada dan menghasilkan sistem yang baru dengan penambahan beberapa perubahan untuk meningkatkan sistem. Sementara itu, proses evolusi perangkat lunak dapat terjadi secara terus–menerus. Secara keseluruhan, evolusi perangkat lunak dapat dilihat sebagai perangkat lunak dengan proses *software reengineering* secara berulang. *Software reengineering* diterapkan untuk mengubah sistem siap pakai yang lebih rendah atau sederhana menjadi sistem baru yang lebih baik.

2.1.1.3. Paradigma *Reverse Engineering Goals/Model/Tools*

Menurut Tripathy dan Naik (2015: 10) ketika melakukan *reverse engineering* pada suatu sistem yang kompleks, alat dan metodologi yang digunakan umumnya tidak stabil. Oleh karena itu, organisasi tingkat tinggi memungkinkan paradigma pengulangan proses sehingga *maintenance engineer* mempelajari tentang sistem.

Benedusi dkk. (1992), diacu dalam Tripathy dan Naik (2015: 10) mengusulkan paradigma berulang yang disebut *Goals*/*Model*/*Tools,* paradigma ini menggambarkan *reverse engineering* dalam tiga tahap berturut-turut, yaitu: *goal*, model, dan *tools*.

Berikut ini merupakan deskripsi mengenai tahapan dalam paradigma *Goals*/*Model*/*Tools* menurut (Tripathy dan Naik, 2015: 155 - 156), yaitu:

> *"(1) Goals, pada fase ini alasan untuk melakukan proses reverse engineering diidentifikasi dan dianalisis. Analisis dilakukan untuk mengidentifikasi kebutuhan informasi untuk proses reverse engineering dan abstraksi yang ingin dihasilkan dari proses reverse engineering. Contohnya, dokumen desain yang akan dihasilkan dari source code adalah sebagai berikut, yaitu: (a) low-level dokumen, dokumen yang memberikan gambaran dan deskripsi detail dari modul individu meliputi struktur modul dalam hal arus kontrol dan struktur data; (b) high-Level dokumen, dokumen yang memberikan gambaran umum dari produk perangkat lunak dan penjelasan rinci tentang penataan dalam hal modul, interkoneksi mereka, dan arus informasi antara modul. (2) Model, pada fase ini abstraksi yang diidentifikasi dalam tahap goals dianalisis untuk membuat representasi model. Representasi model termasuk informasi yang diperlukan untuk abstraksi berikutnya. Kegiatan pada fase ini adalah: (a) mengidentifikasi jenis dokumen yang akan dihasilkan; (b) menghasilkan dokumen-dokumen, mengidentifikasi informasi, dan hubungan mereka berdasarkan source code; (c) mendefinisikan model yang telah digunakan untuk mewakili informasi dan hubungan antar model yang telah diekstrak dari source code; (d) menghasilkan dokumen yang telah diinginkan dari model yang telah didefinisikan sebelumnya dan menentukan algoritma abstraksi untuk proses reverse engineering. Sifat penting dari model reverse engineering adalah kekuatan ekspresif, independensi bahasa, keutuhan, kekayaan konten informasi, granularitas, dan dukungan untuk transformasi informasi yang dipertahankan. (3) Tools, pada fase ini alat-alat yang diperlukan untuk proses reverse engineering diidentifikasi, diperoleh, dan/atau dikembangkan sendiri. Tools tersebut dikelompokkan menjadi dua kategori, yaitu: (a) alat untuk mengekstrak informasi dan menghasilkan representasi program sesuai dengan identifikasi model; (b) alat untuk mengekstrak informasi dan menghasilkan dokumen yang diperlukan."*

2.1.1.4. 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.

Siakad UNJ menyimpan seluruh informasi akademik mahasiswa aktif dan tidak aktif termasuk data mahasiswa yang telah lulus seperti Kartu Rencana Studi (KRS), Kartu Hasil Studi (KHS), pembayaran, jadwal kuliah, dan Daftar Hasil Studi (DHS). Siakad UNJ mengelola seluruh data akademik tersebut secara integritas sehingga setiap perubahan yang terjadi pada data akan berpengaruh terhadap data lainnya secara langsung. Contohnya jika seorang mahasiswa belum melakukan pembayaran uang kuliah, maka mahasiswa tersebut tidak dapat melakukan pengisian KRS. Sebaliknya, jika seorang mahasiswa telah membayar uang kuliah sesuai dengan tagihan maka mahasiswa dapat langsung melakukan pengisian KRS tanpa harus adanya *validasi* manual oleh *administrator* Siakad UNJ.

Siakad UNJ juga merupakan pusat informasi akademik utama bagi universitas. Pimpinan universitas dan *administrator* dapat mengambil data rekapitulasi akademik untuk keperluan universitas dengan mudah melalui Siakad UNJ. Adanya Siakad UNJ membuat data tersimpan secara rapi dan terstruktur tanpa memerlukan ruang yang besar untuk menyimpannya dan waktu yang lama untuk memperoleh atau menggunakan data tersebut.

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.

2.1.1.5. Modul pada Siakad UNJ

Suatu sistem aplikasi perangkat lunak pada umumnya terdiri dari beberapa bagian atau komponen yang dapat berdiri sendiri, tetapi dapat menunjang program sistem untuk menangani suatu kebutuhan tertentu. Bagian atau komponen sistem yang menangani suatu kebutuhan tertentu disebut dengan modul.

Siakad UNJ memiliki beberapa modul di dalamnya. Setiap modul dari Siakad UNJ dapat beridiri sendiri dan mendukung sistem untuk memenuhi kebutuhan pengguna. Pembagian modul pada Siakad UNJ dilakukan berdasarkan jenis pengguna Siakad UNJ. Oleh karena itu, salah satu modul utama yang terdapat pada Siakad UNJ adalah modul mahasiswa dan modul dosen.

Gambar 2.3 Halaman Beranda Modul Mahasiswa Siakad UNJ

Gambar 2.3 merupakan tampilan antarmuka dari halaman beranda Modul Mahasiswa Siakad UNJ. Halaman beranda Modul Mahasiswa Siakad UNJ menampilkan layanan yang terdapat di dalam Modul Mahasiswa Siakad UNJ, informasi jumlah pengunjung dan informasi *user* yang masuk ke dalam sistem.

Modul Mahasiswa merupakan salah satu modul yang ada di dalam Siakad UNJ. Modul ini dibuat dan digunakan untuk mahasiswa agar proses administrasi akademik dapat dilakukan secara komputerisasi. Berikut ini merupakan beberapa layanan yang ada di dalam modul mahasiswa, yaitu: (a) melakukan pengelolaan biodata; (b) melakukan perubahan kata sandi; (c) melakukan pengisian KRS pada awal semester; (d) melihat dan mengunduh KRS dan KHS pada tiap semester yang diikuti; (e) melihat DHS yang merupakan daftar keseluruhan hasil dari seluruh aktifias akademik yang sudah dikerjakan selama mahasiswa kuliah di UNJ; (f) melihat jadwal kuliah; (g) melihat status pembayaran pada masing-masing semester; (h) melakukan pendaftaran Praktik Kerja Mengajar (PKM); (i) melakukan pengisian evaluasi perkuliahan.

Selain Modul Mahasiswa, terdapat Modul Dosen di dalam Siakad UNJ. Modul Dosen dibuat untuk menunjang Modul Mahasiswa dalam proses administrasi akademik seperti persetujuan KRS oleh pembimbing akademik dan juga proses pemebrian nilai mata kuliah. Modul dosen digunakan oleh setiap dosen yang megajar di UNJ. Modul ini dibuat untuk membantu dosen dalam melakukan kegiatan administrasi akademik. Berikut ini adalah beberapa layanan yang ada di dalam Modul Dosen, yaitu: (a) melihat jadwal mengajar; (b) melakukan pengisian nilai untuk masing-masing mahasiswa pada masing-masing kode seksi; (c) melihat hasil evaluasi perkuliahan yang diisi oleh mahasiswa; (d) melihat jumlah sistem kredit semester (SKS) yang diizinkan untuk mengajar.

2.1.2. Model GORE

2.1.2.1. Definisi Model

Model menurut Kamus Besar Bahasa Indonesia (KBBI), didefinisikan menjadi kata benda atau nomina yang diartikan sebagai pola (contoh, acuan, ragam, dan sebagainya) dari sesuatu yang dihasilkan; orang yang dipakai sebagai contoh untuk dilukis (foto); orang yang (pekerjaannya) memperagakan contoh pakaian yang akan dipasarkan; barang tiruan yang kecil dengan bentuk (rupa) persis seperti yang ditiru.

Model banyak digunakan untuk menggambarkan objek yang ada di sekitar kita. Model menurut Gerald V. Post dan David L. Anderson, diacu dalam Rizqy (2016: 6) model adalah suatu penggambaran abstrak dan sederhana dari beberapa sistem dunia nyata. Beberapa model dapat ditulis sebagai persamaan matematika atau grafik, dan lainnya adalah deskripsi yang subjektif.

Jadi, dapat disimpulkan bahwa model adalah suatu pola atau representasi dari suatu objek yang ada pada alam nyata yang bersifat sederhana dan menyeluruh serta digunakan untuk menjadi acuan dalam menggambarkan suatu objek. Model dapat ditulis sebagai persamaan matematika atau grafik, dan lainnya dalam bentuk deskriptif yang subjektif.

2.1.2.2. Jenis Model

Model seringkali digunakan untuk membantu kegiatan manusia dalam memahami sesuatu. Saat ini untuk memudahkan proses pemahaman, berbagai hal dijelaskan dengan menggunakan model. Model terbagi menjadi beberapa jenis sesuai dengan objek yang akan digambarkan oleh model tersebut.

Berikut ini merupakan klasifikasi mengenai jenis-jenis model menurut Raymond McLeod, diacu dalam diacu dalam Rizqy (2016: 6):

> *"(1) Model Fisik, model ini merupakan gambaran tiga dimensi dari kesatuan itu sendiri. Model fisik digunakan dalam dunia usaha/bisnis, termasuk model skala pusat perbelanjaan (shopping centers) dan bentuk dasar (prototype) kendaraan bermotor baru. (2) Model Cerita, merupakan jenis model yang menggambarkan kesatuannya sendiri dengan berbicara atau tertulis. Pendengar atau pembaca dapat mengerti kesatuan tersebut dari cerita. Semua komunikasi dalam usaha adalah model cerita, yang membuat model cerita paling terkenal dari jenis model. (3) Model Grafik, jenis model lain yang penggunaannya tetap (constant) adalah model grafik. Sebuah model grafik mewakili kesatuannya dengan sebuah garis lambang atau bentuk-bentuk yang abstrak. (4) Model Matematika, merupakan rumus matematika atau persamaan matematika. Keuntungan terbesar dari model matematika adalah tingkat ketelitian dalam menggambarkan hubungan di antara bagian suatu objek."*

2.1.2.3. Manfaat Model

Sebagai perannya dalam membantu proses pemahaman mengenai suatu objek, model memiliki beberapa manfaat. Berikut ini merupakan manfaat model menurut Gaol (2008: 106-107) diacu dalam Rizqy (2016: 8), yaitu:

> *"(1) Memudahkan Pengertian, suatu model dapat lebih sederhana daripada kesatuannya sendiri. Kesatuan yang terbentuk lebih mudah dimengerti pada saat unsur dan hubungannya ditampilkan dalam cara yang mudah dimengerti apa adanya. (2) Memudahkan Komunikasi, bila suatu kelompok pemecah masalah dibentuk untuk memahami kesatuan tersebut, biasanya penting untuk mengomunikasikan pemahaman tersebut kepada yang lainnya. Mungkin sistem analis harus mengomunikasikan kepada manager atau kepada pemrogram atau mungkin manager harus mengomunikasikan kepada anggota lain dari kelompok pemecah masalah yang dibentuk tersebut."*

2.1.2.4. 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.

Requirements engineering menurut Zave (1997), adalah cabang dari rekayasa perangkat lunak yang berkaitan dengan tujuan (dunia nyata), fungsi, dan batasan-batasan pada sistem perangkat lunak. Hal ini juga berkaitan dengan hubungan faktor-faktor dalam menetapkan spesifikasi yang tepat dari suatu perangkat lunak, proses evolusinya baik berhubungan dengan masalah waktu maupun dengan perangkat lunak lain.

Jadi, dapat disimpulkan bahwa *requirements engineering* adalah cabang dari rekayasa perangkat lunak yang mempelajari penelusuran, perbaikan, analisis, penelusuran dan pemahaman mengenai *requirements,* fungsi, dan batasan-batasan pada sistem perangkat lunak. *Requirements engineering* mempelajari dan membahas mengenai cara mendapatkan dan memahami *requirements* yang di inginkan dalam pembuatan sistem perangkat lunak.

2.1.2.5. Definisi *Requirement*

Requirement menurut IEEE (1998), adalah sebuah pernyataan yang mengidentifikasi sebuah produk atau operasional proses, fungsi, karakteristik desain atau batasan, yang mana tidak ambigu, mudah dites atau diukur, serta dibutuhkan untuk produk atau dalam proses penerimaan oleh konsumen.

Requirement menurut Sommerrville (2011: 83), adalah deskripsi tentang apa yang harus dilakukan sistem, layanan yang diberikannya dan kendala dalam operasinya. *Requirement* ini Mencerminkan kebutuhan pelanggan terhadap sistem

yang melayani tujuan tertentu seperti mengendalikan perangkat, melakukan pemesanan, atau mencari informasi.

Jadi, dapat disimpulkan bahwa *requirement* adalah kalimat pernyataan yang tidak ambigu dan mudah dimengerti serta dapat menggambarkan tentang apa yang harus dilakukan sistem, operasional proses, fungsi, karakteristik desain atau batasan sistem sehingga dapat mencerminkan kebutuhan pelanggan atau pengguna terhadap sistem yang nanti akan melayani tujuan tertentu seperti mengendalikan perangkat, melakukan pemesanan atau mencari infromasi.

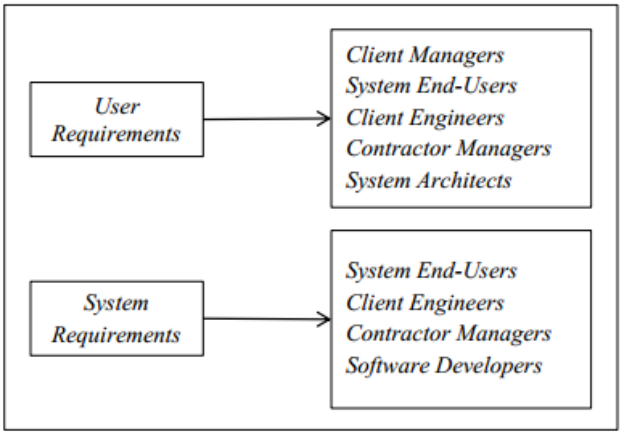
2.1.2.6. Klasifikasi *Requirement*

Pada proses *requirement engineering* seringkali timbul masalah dalam pemisahan hasil *requirements* berdasarkan tingkat deskripsi dan abstraksinya. Pemisahan *requirements* perlu dilakukan karena dokumentasi sistem tidak hanya dibaca oleh seorang *developer* tetapi juga *client* ataupun *stakeholder* yang kurang mengerti pemrograman.

Pembagian *requirements* dalam proses *requirement engineering* berdasarkan level abstraksinya menurut Sommervile (2011: 83) adalah sebagai berikut, yaitu:

> *"(1) User Requirements, pernyataan mengenai layanan sistem yang diharapkan dapat diberikan kepada pengguna sistem dan batasan di mana ia harus beroperasi yang ditulis dengan bahasa alami dan ditambahkan dengan diagram. (2) System Requirements, deskripsi yang lebih rinci dari sistem perangkat lunak mengenai fungsi, layanan, dan kendala operasional. Dokumentasi System Requirements terkadang disebut sebagai functional specification yang menjelaskan seperti apa sistem diimplementasikan. Dokumentasi System Requirements bisa menjadi bagian dari kontrak antara pemebli sistem dengan software developers.*

Pembagian *requirements* berdasarkan *level* abstraksi merupakan langkah untuk membuat komunikasi mengenai *requirements* antara *stakeholder* dan *developer* berjalan dengan baik. Hal tersebut juga dikarenakan bahwa pembaca dari daftar *requirements* tidak hanya dari golongan *programmer* saja melainkan *stakeholder* yang pada umumnya kurang mengerti pemrograman. Gambar 2.4 merupakan klasifikasi pembaca berdasarkan jenis *requirement*:



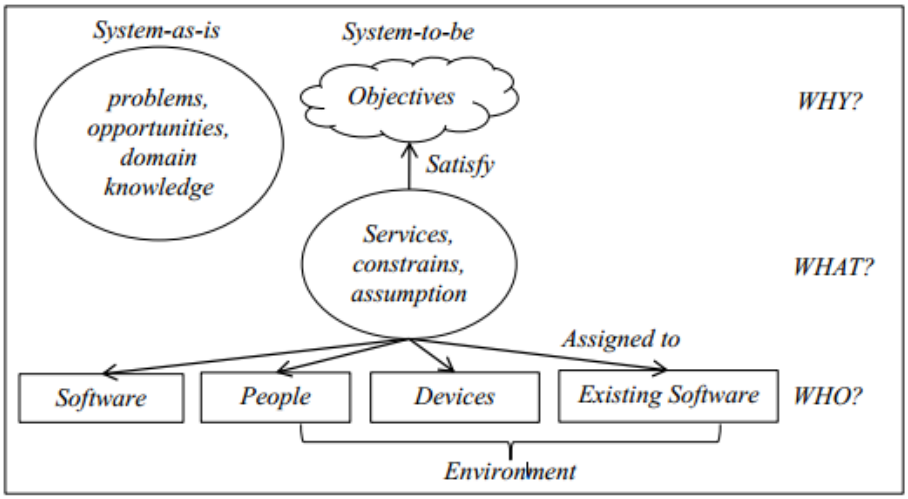
Gambar 2.4 Klasifikasi *requirement* (Sommerville, 2011: 84-85)

System requirements menurut Sommerville (2011: 85-86) seringkali diklasifikasikan dan dijelaskan kembali sebagai berikut:

> *"(1) Functional Requirements, pernyataan dari layanan sistem yang harus disediakan, bagaimana sistem merespon input secara partikular, dan bagaimana sistem menangani situasi partikular. Pada beberapa kasus, functional requirements mungkin juga berupa pernyataan tersirat yang tidak harus dikakukan sistem. (2) Non-Functional Requirements, kendala pada layanan atau fungsi yang ditawarkan oleh sistem. Kendala tersebut termasuk kendala waktu, kendala pada proses pembangunan, dan kendala yang dikenakan oleh standar. Kebutuhan non-functional requirements sering berlaku untuk sistem secara keseluruhan, bukan fitur sistem individual atau jasa.*

2.1.2.7. Dimensi dan Proses *Requirement Engineering*

Requirement engineering merupakan segala sesuatu yang berhubungan dengan kebutuhan untuk menemukan (*discover*), memahami (*understand*), memformulasikan (*formulate*), menganalisis (*analyse*), dan menyetujui (*agree*) pada masalah apa (*what*) yang harus diselesaikan, mengapa (*why*) masalah tersebut perlu diselesaikan, dan siapa (*who*) yang harus bertanggung jawab untuk menyelesaiakan masalah tersebut (Van Lamsweerde, 2009, diacu dalam Rizqy, 2016: 20). Berdasarkan hal tersebut w*hy*, *what*, dan *who* dikenal sebagai tiga dimensi dalam *requirements engineering*.



Gambar 2.5 Dimensi *Requirements Engieering*

Gambar 2.5 menjelaskan mengenai tiga dimensi dalam *requirement engineering*, dimensi *why* fokus pada alasan kontekstual terhadap usulan versi baru dari sistem yang secara eksplisit merupakan objektif (*objectives*) yang harus dipenuhi berdasarkan keterbatasan (*limitations*) dari *system-as-is* dan peluang peluang eksploitasi dari *system-as-is* tersebut (Van Lamsweerde, 2009, diacu dalam Rizqy, 2016: 21).

Dimensi *what* fokus pada layanan fungsional (*functional services*) *system to-be* yang harus disediakan untuk memenuhi objektif yang diidentifikasi pada dimensi *why* (Van Lamsweerde, 2009, diacu dalam Rizqy, 2016: 21). Beberapa layanan tersebut akan diimplementasikan oleh *system to-be*, dan sebagian yang lain akan direalisasikan melalui prosedur manual maupun *device*.

Layanan sistem, *constraint*, dan asumsi dapat diidentifikasi dari objektif sistem yang telah disetujui dan diformulasikan dengan istilah atau bahasa yang tepat atau cermat dimana semua pihak yang berkepentingan dapat memahami untuk memvalidasi dan merealisasikannya (Shofi dan Budiarjo, 2011:2).

Sedangkan dimensi *who* mengarah pada penugasan (*assignment*) tanggung jawab untuk mencapai objektif, layanan, dan *constraint* diantara komponen-komponen pada *system-to-be*, manusia, alat (*devices*), atau perangkat lunak (Van Lamsweerde, 2009 diacu dalam Rizqy, 2016: 21).

Menurut Li Jiang (2005), proses dalam *requirements engineering* terdiri dari lima fase, yaitu:

1. *Requirement Elicitation*

Requirement elicitation adalah proses mengidentifikasi kebutuhan, menjembatani komunikasi antara komunitas atau *user* yang terlibat dan menyaring atau menyeleksi *requirements* untuk dapat menghasilkan kesepakatan keinginan antara pihak-pihak terkait (SEI, 1991, diacu dalam Shabrina, 2016: 13). Berbagai sumber yang akan menjadi bahan dalam *requirements elicitation* antara lain: (a) Klien/*Stakholder*; (b) *Customer*; (c) Dokumentasi dari sistem yang sudah ada sebelumnya; (d) Pengguna sistem yang sudah ada; (e) Pengguna yang berpotensial dalam menggunakan sistem baru.

Bagian utama yang paling penting dalam fase *requirement elicitation* adalah, memastikan terciptanya komunikasi yang efektif antar *stakeholder* yang terlibat dengan pihak pengembang. *Output* dari fase *requirements elicitation* adalah mengumpulkan catatan atau dokumen-dokumen informal yang berisikan deskripsi cikal bakal *requirements* (Bray, 2002, diacu dalam Shabrina, 2016: 14). *Requirements* pada fase ini masih belum terproses dan terstruktur dengan baik, selain itu masih mengandung *requirements* yang tidak relevan, tidak konsisten, dan tidak tercantum.

Berbagai teknik yang bisa digunakan di dalam proses *requirements elicitation* adalah sebagai berikut (Bray, 2002, diacu dalam Shabrina, 2016: 14): (a) *Interview;* (b) Etnografi teknik yang diadopsi dari ilmu sosiologi, termasuk di dalamnya teknik observasi (Goguen, 1993, diacu dalam Shabrina, 2016: 14); (c) Pertanyaan yang berhubungan (Holtzblattt, 1993, diacu dalam Shabrina, 2016: 14); (d) Skenario, *use case*, dan teknik *goal-based* (Kavakli, dkk., diacu dalam Shabrina, 2016: 14).

2. Analisis *Requirements* dan Negosiasi

Analisisi *requirements* dan negosiasi adalah sebuah proses dimana *requiremets* mengalami tahapan analisis dan pemodelan serta tahap penyelesaian konflik-konflik akibat adanya negosiasi antara *stakeholder* yang terlibat (Li Jiang, 2005). Hasil dari *requirements elicitation* menjadi input dari proses ini. *Output* dari proses ini adalah sebuah *set requirements* yang konsisten dan lengkap. Beberapa teknik yang digunakan pada tahap analisis *requirements* dan negosiasi adalah: (a) *Unified Modelling Language* (UML), adalah teknik pemodelan sistem atau software. UML menyediakan kebutuhan *modelling* (Dennis, 2002, diacu dalam Shabrina, 2016: 15); (b) *Spesification and Description anguage* (SDL) (Spivey,

1998, diacu dalam Shabrina, 2016: 15); (c) *Structured Analysis Structured Design* (SASD), melibatkan beberapa teknik seperti teknik penguraian fungsi, *Data Flow Diagram* (DFD), dan *dictionary* yang berfungsi untuk memodelkan fungsi dan proses dari sistem secara keseluruhan (Yourdon, 1989, diacu dalam Shabrina, 2016: 15).

3. *Requirements Documentation*

Requirements documentation adalah proses pendokumentasian dari keseluruhan *requirements* yang disetujui dan disepakati. *Input* pada fase ini adalah hasil dari proses analisis dan negosiasi sebelumnya, dan output dari proses ini adalah suatu struktur dan spesifikasi *requirements* yang teridentifikasi dengan baik walaupun tetap dibutuhkan proses verifikasi dan persetujuan setelahnya. Dalam pendokumentasian *requirements*, model yang digunakan bisa diantara tiga model yang disajikan sebelumnya, yaitu: informal (bahasa alami), semiformal (diagram, grafik), atau formal (notasi matematika). Akan tetapi juga dapat menggunakan model dokumentasi atau bahasa notasi yang lain, yang pasti dapat dipahami dan diemengerti oleh *stakeholder* yang terlibat dan bersinggungan di dalamnya.

4. *Requirement Verification and Validation*

Requirement verification and validation adalah proses untuk memastikan tidak adanya *requirment* yang ambigu, konsisten, dan lengkap sehingga *stakeholder* yang terlibat puas dan setuju dengan hasil akhir dari spesifikasi *requirement*. Karakteristik sebuah set *requirement* dikatakan baik (sesuai pada penjelasan definisi) bisa dijadikan sebuah panduan untuk memverifikasi dan menyetujui *requirement* tersebut.

5. *Requirement Management*

Requirement management proses mengidentifikasi, mengorganisir, mendokumentasi, dan menelusuri, perubahan-perubahan yang terjadi selama sistem atau *software* berjalan. Tahapan tersebut adalah sebuah *on going proccess* yang waktu pengerjaannya berlangsung selama siklus hidup sistem atau *software* tersebut.

2.1.2.8. Hasil *Requirement Engineering*

Output terakhir yang dihasilkan dari *requirements engineering* adalah daftar kebutuhan sistem, yaitu: *functional requirements* dan *non-functional requirements*. *functional requirements* dan *non-functional requirements* tersebut ditulis dalam satu buah dokumen yang disebut *Software Requirements Spesification* (SRS). Berdasarkan IEEE Std830 (984) diacu dalam Shabrina (2016: 12), SRS yang baik mengikuti beberapa karakteristik berikut ini, yaitu: (1) Tidak ambigu, Setiap *requirement*s yang ada di SRS hanya memiliki satu interpretasi; (2) Lengkap, semua *requirements* yang ada di SRS telah memiliki definisi, referensi, label, dan aturan yang sesuai dengan SRS; (3) Terverifikasi, semua *requirements* dalam SRS sudah terverifikasi; (4) Konsisten, tidak ada *requirements* dalam SRS yang memiliki konflik dengan *requirements* yang lain; (5) Mudah dimodifikasi, jika ada *requirements* yang membutuhkan untuk diubah atau diganti, maka perubahan tersebut dapat dilakukan dengan mudah; (6) Dapat dilacak atau ditelusuri, di dalam SRS, *requirements* yang *original* dan rasional memiliki hubungan satu sama lain meliputi antar *requirements*, desain, implementasi, dan *test case*, yang itu semua dapat ditelusuri keterkaitannya; (7) Dapat digunakan selama *software* dioperasikan atau pada tahap perbaikan.

2.1.2.9. Definisi *Requirement Traceability*

Requirement traceability adalah kemampuan menelusuri secara deskriptif dengan mengikuti siklus kehidupan *requirement*, menelusuri siklus sebelumnya maupun sesudahnya, menelusuri *requirement* yang masih *original* maupun yang sudah dikembangkan atau dispesifikasikan untuk dapat disesuaikan penggunaannya sesuai perubahan-perubahan yang terjadi pada siklus kehidupan suatu *requirement*. Penelusurannya dilakukan secara sekuensial (beruntun) maupun secara iteratif (pengulangan) (Got, 1994, diacu dalam Shabrina, 2016:21).

Requirement traceability menurut Aurum dan Wohlin (2005: 97) diacu dalam Rizqy (2016: 21) adalah kegiatan proses yang dilakukan untuk mendapatkan informasi keterkaitan antara objek-objek yang terlibat dalam *requirement*, sebagai contoh: (a) Hubungan keuntungan/kepuasan antara *requirement* dengan komponen sistem; (b) Hubungan antara objek sistem dengan *requirement*; (c) Perubahan proposal aplikasi dengan *requirement*; (d) Hubungan antara *test case* dengan *requirement* (sebagai bentuk validasi *requirement*, menyesuaikan kecocokan, keterkaitan, keterhubungan); (e) Hubungan antara komponen sistem dengan *resources* yang terlibat dan dibutuhkan untu diimplementasikan pada *requirement*.

Jadi, dapat disimpulkan bahwa *requirement traceability* adalah proses kegiatan yang memiliki kemampuan menelusuri secara deskriptif siklus kehidupan *requirement*, baik siklus sebelum maupun sesudah ataupun menelusuri *requirement* yang masih *original* maupun yang sudah dikembangkan untuk mendapatkan informasi keterkaitan antara objek-objek yang terlibat di dalam *requirement*.

2.1.2.10. *Requirement Traceability Matrix*

Requirement traceability matrix (RTM) adalah sebuah dokumen yang menghubungkan *requirement* selama proses validasi. RTM bagi tim validasi adalah sebuah alat untuk memastikan bahwa *requirement* tidak hilang selama proses validasi dalam pembuatan sistem sedangkan untuk tim auditor RTM adalah alat untuk meninjau dokumen validasi sistem.

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 2.1 *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

2.1.2.11. 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.

2.1.2.12. Konsep *Goal*

Goal menurut Anwer dan Ikram (2006: 2) diacu dalam Rizqy (2016: 26) adalah kondisi atau keadaan yang diinginkan dari sebuah sistem berdasarkan beberapa pertimbangan. Dalam hal ini, *agent* bertanggung jawab atas pemenuhan *goal*. Terdapat beberapa konsep mengenai *goal*, yaitu:

1. *Goal Type*

Goal Type adalah konsep yang didasarkan pada *functional requirements* dan *non-functional*. Terdapat tiga jenis *goal*, yaitu: *achievement goal* (terpenuhi jika kondisi target tercapai), *soft goal* (tujuan yang berkaitan dengan *non-functional requirements*), dan *maintenance goal* (tujuan yang harus memiliki kondisi konstan).

2. *Belief*

Belief adalah konsep yang berkaitan dengan *agent*. *Belief* merupakan cara pandang *agent* mengenai statusnya sendiri maupun lingkungannya.

3. *Constraint*

Constraint adalah konsep yang menjelaskan batasan pencapaian suatu *goal* pada sistem aplikasi yang telah ditentukan sebelumnya oleh *stakeholder*.

4. Level Abstraksi *Goal*

konsep ini menggambarkan level abstraksi atau tingkatan dari *goal*. Terdapat tiga level abstraksi atau tingkatan dari *goal*, yaitu: (a) *highest* level (berhubungan dengan *survival* atau pertahanan suatu perusahaan atau organisasi); (b) *high*-level (berhubungan dengan masalah strategi suatu perusahaan), (c) *low*-level (berhubungan dengan teknik yang digunakan suatu perusahaan dalam menanggapi masalah).

5. Taxonomi *Goal*

Taxonomi *goal* merupakan suatu konsep yang menjelaskan pola operasional yang memetakan spesifikasi *goal* menggunakan aturan-aturan turunan formal.

6. *Agent*

Agent adalah komponen aktif dari suatu sistem seperti manusia, alat, dan perangkat lunak. *Agent* sangat bertanggung jawab pada pencapaian *goal* yang dibuatnya. *Goal* lebih disempurnakan dengan menguraikannya menjadi beberapa *sub-goal* sehingga apabila ada agen tunggal, maka tidak keberatan apabila dierikan pertanggung jawaban atas *goal* tersebut.

7. *Requirements*

Requirements merupakan *goal* yang berada di bawah pertanggung jawaban *agent* tunggal di dalam perangkat lunak.

8. Asumsi

Asumsi merupakan g*oal* di bawah tanggung jawab *agent* tunggal di lingkungan perangkat lunak yang akan dibuat menjadi asumsi.

9. Atribut Goal

Sebuah *Goal* dapat ditandai dengan beberapa atribut, atribut tersebut dapat membantu untuk membuat keputusan penting. Atribut tersebut seperti nama, spesifikasi, prioritas, utilitas dan kelayakan.

10. *Goal Link*

Goal Link merupakan suatu konsep yang digunakan untuk menjelaskan perbedaan dari bermacam-macam *traceability*. *Link* dapat berupa *inter-goal contribution links* (yang meng-*capture* kontribusi *positive* atau *negative* dari satu *goal* ke *goal* yang lain), *operationalization links* (yang melengkapi kondisi *pre*, *post*, dan *trigger* untuk menspesifikasikan *goal*), *coverage links* (berhubungan dengan *goal* dengan skenarionya), *responsibility links* (yang merelasikan *goal agent* yang bertanggung jawab terhadapnya), dan *Wish links* (*link* ini membantu *agent* menentukan *goal*. Sebagai contoh, seorang *agent* tidak boleh diberikan *goal* yang diinginkan untuk memenuhi *goal* lain yang dapat menyebabkan konflik antar *goal*).

2.1.2.13. Metode Dalam GORE

Berikut ini merupakan penjelasan beberapa metode di dalam model pendekatan GORE menurut Lapouchnian (2005: 8-12), yaitu:

1. *i*/Tropos*

i/Tropos* adalah sebuah metode yang berorientasi *agents* dan dapat digunakan untuk *requirements engineering*, rekayasa ulang proses bisnis, analisis dampak organisasi, dan pemodelan proses perangkat lunak. Metode ini memiliki dua komponen utama, yaitu: *the Strategic Dependency* (SD) model dan Strategic Rationale (SR) model.

Tropos merupakan kerangka pemodelan yang memandang perangkat lunak dari empat perspektif yang saling melengkapi, yaitu: (a) Sosial - merupakan aktor yang relevan, apa yang mereka inginkan? Apa kewajiban mereka? Apa kemampuan mereka?; (b) Disengaja - apa tujuan yang relevan dan bagaimana mereka saling berhubungan? Bagaimana mereka terpenuhi dan oleh siapa?; (c)Proses-berorientasi - apa proses bisnis/komputer yang relevan? Siapa yang bertanggung jawab dan untuk apa?; (d) Berorientasi obyek - apa obyek dan kelas yang relevan, bersama dengan hubungan antar mereka?

2. *Goal-Based Requirements Analysis Method* (GBRAM)

Metode yang lebih menekankan pada identifikasi awal dan abstraksi *goal* dari berbagai sumber informasi. GBRAM mengasumsikan bahwa tidak ada *goal* yang telah didokumentasikan atau ditimbulkan dari *stakeholder*, oleh karena itu proses analisis dapat menggunakan beberapa data yang sudah ada berupa diagram, laporan tekstual, transkrip wawancara, dll. Metode GBRAM melibatkan kegiatan analisis tujuan dan perbaikan tujuan sistem perangkat lunak.

Serupa dengan pendekatan GORE lainnya, sistem dan lingkungan pada GBRAM direpresentasikan sebagai kumpulan *agent*. Di sini, *agent* didefinisikan sebagai entitas atau proses yang berusaha untuk mencapai tujuan dalam sebuah organisasi atau sistem berdasarkan tanggung jawab yang diasumsikan untuk *goal*. Dalam GBRAM, tujuan, *agent, stakeholder,* dan lain-lain ditentukan dalam bentuk tekstual pada skema *goal*. Anehnya, metode ini tidak memberikan notasi grafis untuk mewakili tujuan, perbaikan tujuan, *agent,* dan lain-lain. Sementara secara metode hal tersebut dilibatkan, misalnya menciptakan hubungan antara prioritas *goal*, hubungan tersebut lebih mudah dimengerti ketika diwakili secara grafis.

2.1.3. Metode KAOS

2.1.3.1. Definisi Metode KAOS

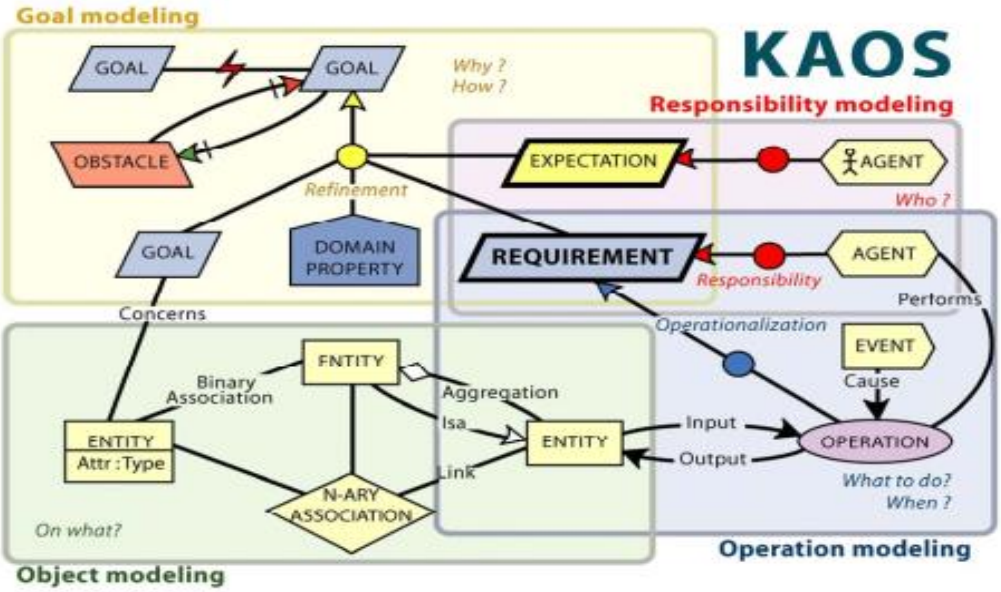
KAOS menurut Lapouchnian (2005: 10) adalah sebuah metodologi dari pendekatan *Goal Oriented Requirement Engineering* (GORE) yang menggunakan sekumpulan teknik analisis formal. KAOS adalah singkatan dari *Knowledge Acquisition in autOmated Specification* atau *Keep All Objects Satisfied*.

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.

Jadi, dapat disimpulkan bahwa KAOS merupakan suatu metode dalam model pendekatan GORE yang menggunakan analisis formal yang di dalamnya memiliki kerangka kerja *elicitation*, menspesifikasi dan menganalisis tujuan (*goals*), kebutuhan (*requirements*), skenario, dan tanggung jawab tugas.

2.1.3.2. 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 2.6 adalah ilustrasi dari seluruh area kerja yang ada di dalam metode KAOS.



Gambar 2.6 Area Kerja Metode KAOS (Respect-IT, 2007: 45)

Berdasarkan Gambar 2.6 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.

2.1.3.3. 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.*

2.1.3.4. Ketergantungan *Goal* pada Metode KAOS

Terdapat tiga jenis hubungan antar Goal dalam metode KAOS menurut Adikara, dkk. (2013: 230), yaitu:

> *(1) AND/OR-decomposition, merupakan sebuah hubungan yang menggambarkan hirarki dari goal dengan sub-goal-nya, menggambarkan bahwa goal dapat dipenuhi/dipuaskan jika seluruh sub-goal-nya terpuaskan (menggunakan AND decomposistion), atau minimal salah satu dari softgoal tersebut terpuaskan (menggunakan OR decomposistion). AND/OR-decomposition pada metode KAOS tidak digambarkan dengan simbol melainkan dijelaskan secara tekstual. (2) Potential Conflict, merupakan hubungan yang menggambarkan pada kondisi tertentu. Jika sebuah goal terpenuhi ternyata dapat menyebabkan goal yang lainnya tidak terpenuhi. Konflik ini biasanya bisa muncul karena adanya perbedaan sudut pandang dan kepentingan dari entitas yang berhubungan. Potential conflict pada metode KAOS dilambangkan dengan tanda petir yang berada ditengah garis penghubung antara goal yang konflik. Contoh Potential conflict dapat dilihat pada Gambar 2.6. (3) Responsibility Assignment, hubungan antara agent dengan sebuah goal. Agent yang terhubung tersebut mempunyai tanggung jawab agar goal dapat dipenuhi atau terpuaskan. Responsibility assignment pada metode KAOS dilambangkan dengan menggunakan garis dan penghubung berupa lingkaran berwarna merah. Contoh penulisan Responsibility assignment dapat dilihat pada Gambar 2.6.*

2.2. Penelitian Yang Relevan

Penelitian mengenai perancangan model untuk proses *reverse engineering* dalam melakukan penelusuran *requirement* dari suatu sistem aplikasi perangkat lunak mengacu kepada beberapa penelitian lain yang relevan dan telah dilakukan sebelumnya. Penelitian yang relevan tersebut, yaitu:

1. Perancangan Model *Goal Goal Oriented Requirements Engineering* (GORE) Untuk Proses *Reverse Engineering*

   Penelitian tersebut dilakukan oleh Fauziah Rizqy, S.Pd. di Laboratorium Komputer Jurusan Teknik Elektro Universitas Negeri Jakarta. Penelitian dilakukan dengan tujuan untuk merancang model GORE dengan menggunakan metode GSP untuk melakukan proses *reverse engineering*. Model GORE dengan metode GSP digunakan untuk membentuk suatu model pendekatan dalam melakukan *reverse engineering* yang bertujuan untuk mengatasi kelemahan *reverse* konvensional dalam melakukan *reverse* terhadap sistem aplikasi perangkat lunak. Model berhasil dibuat dan menghasilkan lima tahapan untuk melakukan *reverse engineering* dengan menggunakan metode GSP. Tahapan tersebut, yaitu: Mengambil *main goal* dari tampilan sistem aplikasi, merepresentasikan *goal* dalam bentuk *goal graph*, mengembangkan *goal* menjadi *sub goal*, membuat *set of alternatives* dari *sub go*al, dan mengambil *requirements* dari setiap *alternatives*. Model yang telah berhasil dibuat di uji dengan cara diimplementasikan terhadap sampel aplikasi “Sistem Informasi Penerimaan Mahasiswa Baru UNJ (SIPENMABA UNJ)”. Proses implentasi berhasil dilakukan dan menghasilkan 31 buah *requirements*.

2. Analisis dan Perancangan Sistem Informasi Penerimaan Mahasiswa Baru Universitas Negeri Jakarta

   Penelitian dilakukan oleh Hanifa Fissalma, S.Pd. di Unit Pelayanan Teknis Teknik Informatika dan Komputer Universitas Negeri Jakarta (UPT TIK – UNJ). Penelitian dilakukan dengan tujuan untuk menganalisis Sistem Infromasi Penerimaan Mahasiswa Baru (SIPENMABA) UNJ dan merancang ulang sistem. Pada penelitian tersebut digunakan *software reenginering* dan metode *Feature Driven Development* (FDD) dalam menganalisis dan merancang ulang sistem. Proses analisis sistem bertujuan untuk memperoleh dokumentasi berupa data *requirements* sistem. Data *requirements* yang diperoleh dianalisis dengan menggunakan *Requirement Traceability Matrix* (RTM) yang telah dimodelkan terlebih dahulu pada model FDD. Perancangan ulang sistem dilakukan untuk membantu *developer* dalam memperbaiki dan mengembangkan SIPENMABA. Analisis dan perancangan terhadap SIPENMABA berhasil dilakukan. Berdasarkan proses analisis diperoleh jumlah data *requirements* yang didapat adalah 256, dengan rincian 107 untuk *requirements* bagian portal SIPENMABA dan 129 untuk *requirements* bagian SIPENMABA Mandiri. Perancangan ulang SIPENMABA berhasil dilakukan dengan membuat *use case*, *activity diagram* dan *class diagram* dari SIPENMABA. Perancangan ulang yang dilakukan tersebut masih bersifat konseptual dan tidak untuk diterapkan pada sistem SIPENMABA UNJ 2016, namun bisa menjadi acuan untuk pengembangan sistem diwaktu yang akan datang.

3. Model *Requirement Traceability* untuk Metode Pengembangan Perangkat Lunak *Feature Driven Development* (FDD)

   Penelitian dilakukan oleh Fildzah Shabrina, S.Pd. di PT. Gapura Angkasa. Penelitian dilakukan dengan tujuan untuk membuat model penelusuran *Requirement Traceability Matrix* (RTM) pada metode FDD. Model penelusuran RTM adalah salah satu model yang tidak memiliki *template* tertentu atau khusus, sehingga memungkinkan untuk dibuat menjadi berbagai model penelusuran yang lebih efektif. Penelitian berhasil dilakukan dan model berhasil dibuat. Berdasarkan hasil penelitian tersebut diketahui untuk membuat suatu model penelusuran RTM pada metode FDD dilakukan 7 tahapan, yaitu: (1) Mengumpulkan dokumentasi *requirement engineering*; (2) Mengumpulkan daftar *requirement* dari aplikasi dengan memberika *Req_Code* pada setiap *requirement*; (3) Membuat *requirement tracebility matrix* berbentuk *table checklist* dari *requirement* yang sudah dibuat sebelumnya; (4) Menggeneralisir *requirement* menjadi daftar *requirement* baru untuk kemudian diubah ke dalam Bahasa domain; (5) Membuat *feature set* yang berisi nomor fitur, subject area (pengguna), dan daftar masing-masing fitur (*feature list*) (6) Analisis kekuatan keterkaitan pada *feature_list*; dan (7) Tahap akhir membuat RTM untuk FDD.

Relevansi dalam penelitian pada butir 1 sampai 3 dengan penelitian ini terletak pada pembuatan suatu model penelusuran *requirement* untuk suatu sistem aplikasi perangkat lunak dan melakukan uji coba *requirement* yang didapat dengan menggunakan model penelusuran RTM pada hasil *requirement* yang telah didapatkan.

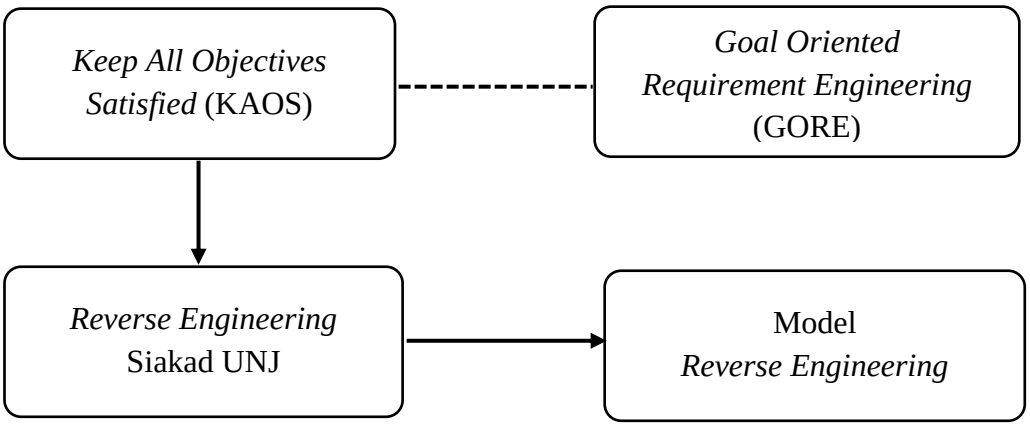
2.3. Kerangka Berpikir

Seringkali sistem yang telah siap pakai tidak memiliki dokumentasi terkait *requirement*. Hal ini dapat menyebabkan proses pengembangan sistem menjadi lebih lama dan sulit dikemudian hari. Proses penelusuran *requirement* dari sistem yang siap pakai bukan merupakan suatu proses yang mudah. Hal itu disebabkan tingkat kompleksitas sistem berbeda – beda. *Requirements engineering* adalah salah satu bidang dalam s*oftware engineering* yang mempelajari cara menelusuri, mengumpulkan, dan menganalisis *requirements*. Salah satu pendekatan dalam menemukan *requirement* adalah *Goal Oriented Requirements Engineering (GORE)*. Pendekatan ini berupaya menelusuri *requirement* sistem berdasarkan *goal* dan kepentingan dari *stakeholder*. Pendekatan ini membuat penelusuran menjadi lebih sistematis dan terarah.

Reverse engineering merupakan proses yang digunakan untuk menganalisis sistem yang telah siap pakai. *Reverse engineering* yang pada umunya dilakukan hanya dengan melihat dan menganalisis tampilan antarmuka tanpa adanya prosedur analisis akan kurang efektif apabila digunakan untuk sistem perangkat lunak yang kompleks karena tidak memiliki prosedur yang sistematis dan terarah. *Reverse engineering* dengan model GORE dan metode *Keep All Objects Satisfied* (KAOS) berusaha untuk melakukan proses penelusuran dan penggalian terhadap sistem perangkat lunak dengan berdasarkan tujuan utama dari *stakeholder*.

Sistem yang digunakan sebagai sampel untuk perancangan model *reverse engineering* dengan model GORE adalah Modul Mahasiswa Siakad UNJ. Proses *reverse engineering* dengan model GORE yang dilakukan akan dianalisis dengan menggunakan metode KAOS. Analisis dilakukan terhadap desain atau tampilan

antarmuka dari Siakad UNJ. Setelah proses *reverse engineering* terhadap Modul Mahasiswa Siakad UNJ dengan menggunakan metode KAOS berhasil dilakukan maka akan didapat suatu model *reverse engineering* dengan mengguanakan model GORE dan metode KAOS. Gambar 2.7 merupakan bagan kerangka berpikir dari penelitian ini.



Gambar 2.7 Bagan Kerangka Berpikir

Berdasarkan bagan kerangka berpikir pada Gambar 2.7 dapat dijelaskan bahwa garis besar kerangka berpikir penelitian yang dilakukan, yaitu: *Keep All Objectives Satisfied* (KAOS) merupakan salah satu dari metode yang ada di dalam Model *Goal Oriented Requirement Engineering* (GORE) akan dipetakan dan diterapkan ke dalam proses *reverse engineering* dengan menggunakan sampel aplikasi Modul Mahasiswa Siakad UNJ. Hasil dari pemetaan dan penerapan tersebut adalah suatu model *reverse engineering* dengan menggunakan metode KAOS yang dapat diterapkan untuk melakukan penelusuran *requirement* pada suatu sistem aplikasi perangkat lunak.

BAB III

METODOLOGI PENELITIAN

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

3.2.1. Alat

Alat yang digunakan dalam penelitian ini adalah sebagai berikut:

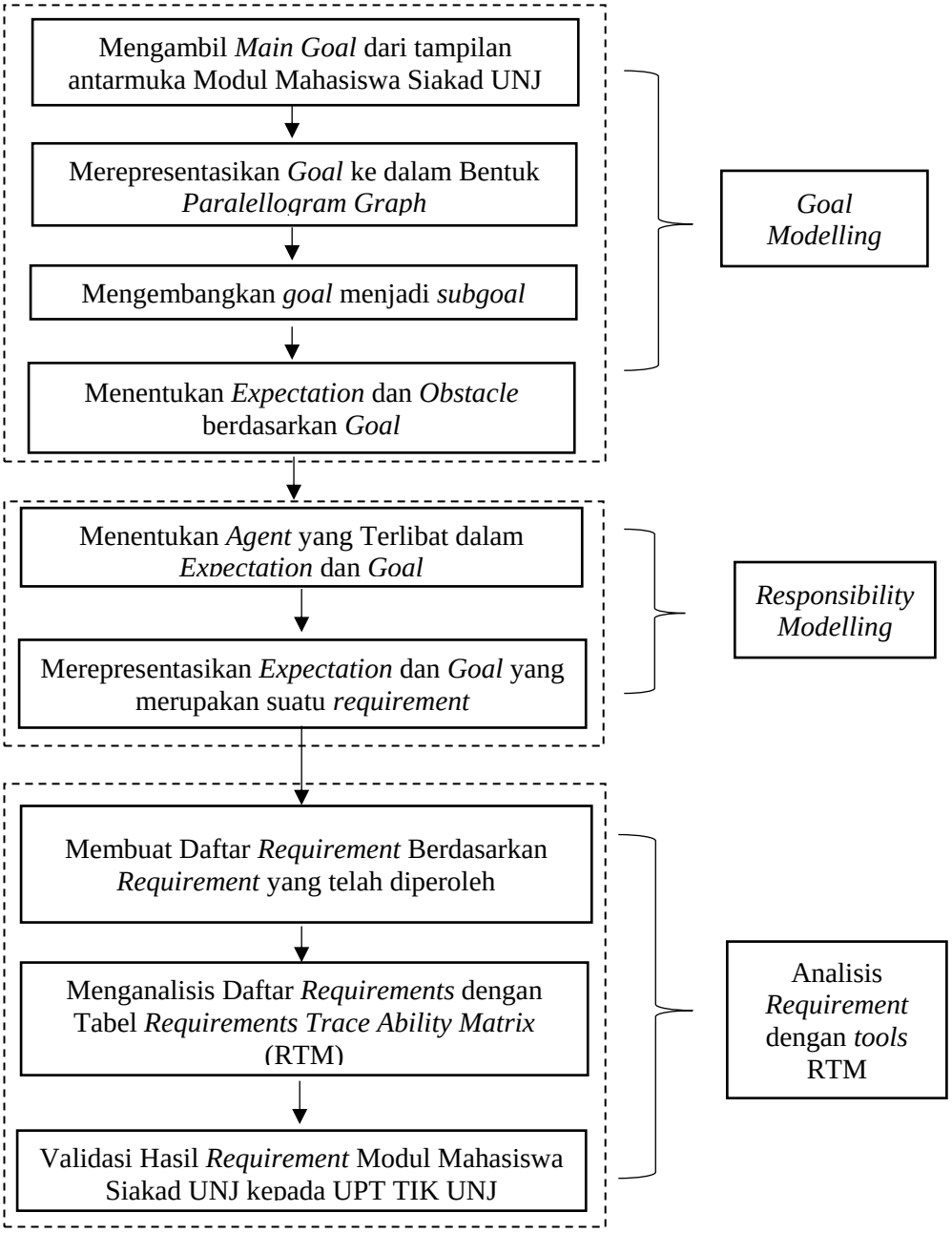
1. Perangkat Keras (*Hardware*)
    a. *Processor* Intel ® Celeron ® CPU N2840 @ 2.16 GHz;
    b. *Memory* RAM 2 GB DDR3.
2. Perangkat Lunak *(Software*)
    a. *Windows 8.1 Pro* 64-bit © 2013;
    b. *Microsoft Office Professional Plus* 2016 64 bit versi *en-us*;
    c. *Edraw Max* 7.9.2

3.2.2. Bahan

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.

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 3.1 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 dan Prosedur Pengumpulan Data

Teknik yang digunakan dalam pengumpulan data pada penelitian ini adalah dengan melakukan studi literatur dan pengamatan secara langsung dan cermat terhadap tampilan antar muka dari Modul Mahasiswa Siakad UNJ.

Studi literatur digunakan untuk mengumpulkan informasi mengenai proses *reverse engineering* untuk mendapatkan *requirement* dari suatu sistem aplikasi dan penggunaan metode KAOS dalam melakukan analisis terhadap *requirement* sistem. Pada saat proses pengamatan terhadap tampilan antarmuka dari Modul Mahasiswa Siakad UNJ, digunakan model pendekatan *reverse engineering* dengan metode KAOS yang telah dirancang dan diilustrasikan pada Gambar 3.1. Prosedur penelitian dilakukan sesuai dengan tahapan pada model tersebut.

3.5. 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 3.1 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

Tabel RTM yang digunakan pada Tabel 3.1 terdiri dari kolom kode *requirement*, pengguna, area fungsional dan relasi requirements. Kode *requirement* merupakan sebuah kode yang digunakan untuk mewakili satu data *requirement* yang sudah diperoleh. Kolom pengguna, terdiri dari *stakeholder* yang terdapat dari sistem yang dianalisis. Kolom area fungsional, terdiri dari area fungsional yang terdapat dari sistem yang dianalisis. Kolom relasi *requirement* terdiri dari setiap kode *requirement* yang sudah didapat. Pengisian kolom pengguna, area fungsional dan relasi *requirement* dilakukan dengan memberikan tanda centang sesuai dengan data yang sudah didapat.

Data yang dimasukan ke dalam Tabel 3.1 akan memberikan informasi setiap hubungan ketergantungan antar *requirement* beserta *functional* area dan subjek areanya. Diketahuinya hubungan tersebut akan memberikan informasi berupa penelusuran *requirement* jika suatu saat terjadi gangguan pada salah satu *requirement* yang ada.

BAB IV

HASIL PENELITIAN

4.1. Deskripsi Hasil Penelitian

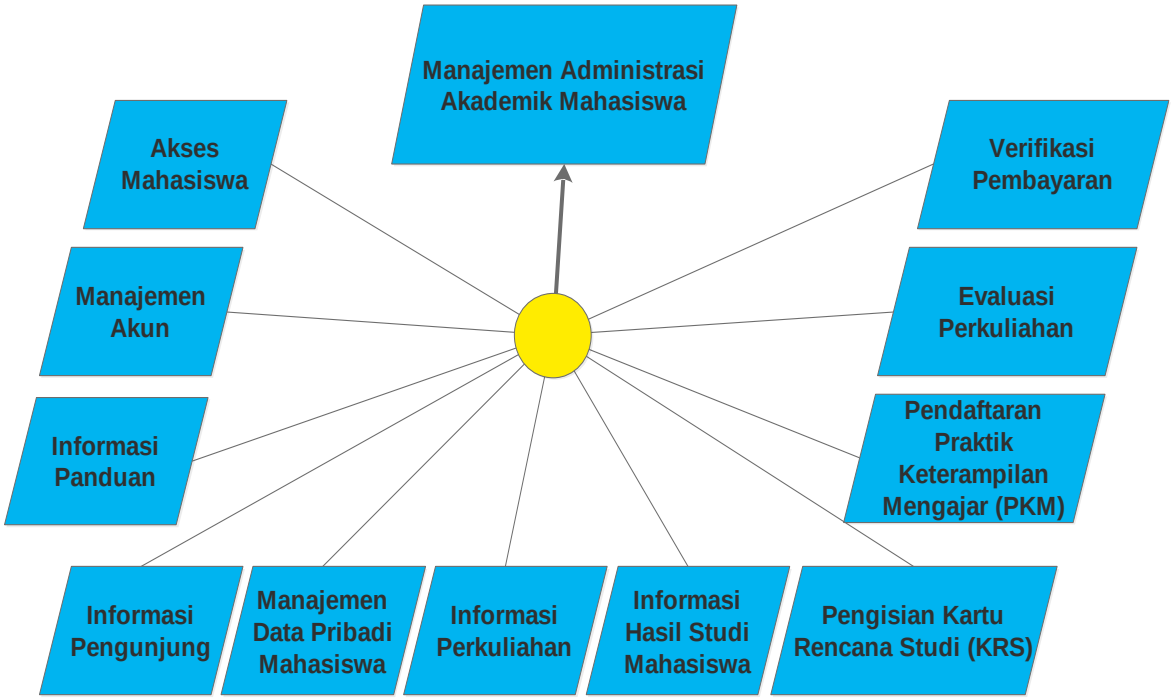
Hasil penelitian yang telah dilakukan pada penelitian ini adalah sebuah model *reverse engineering* dengan menggunakan model GORE dan metode KAOS untuk melakukan analisis dan penelusuran terhadap sistem siap pakai untuk mendapatkan *requirement functional* sistem. Berikut ini adalah hasil dari langkah-langkah yang dilakukan dalam perancangan model *reverse engineering* dengan menggunakan model GORE dan metode KAOS:

4.1.1. 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.

4.1.2. Hasil Represetasi *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*

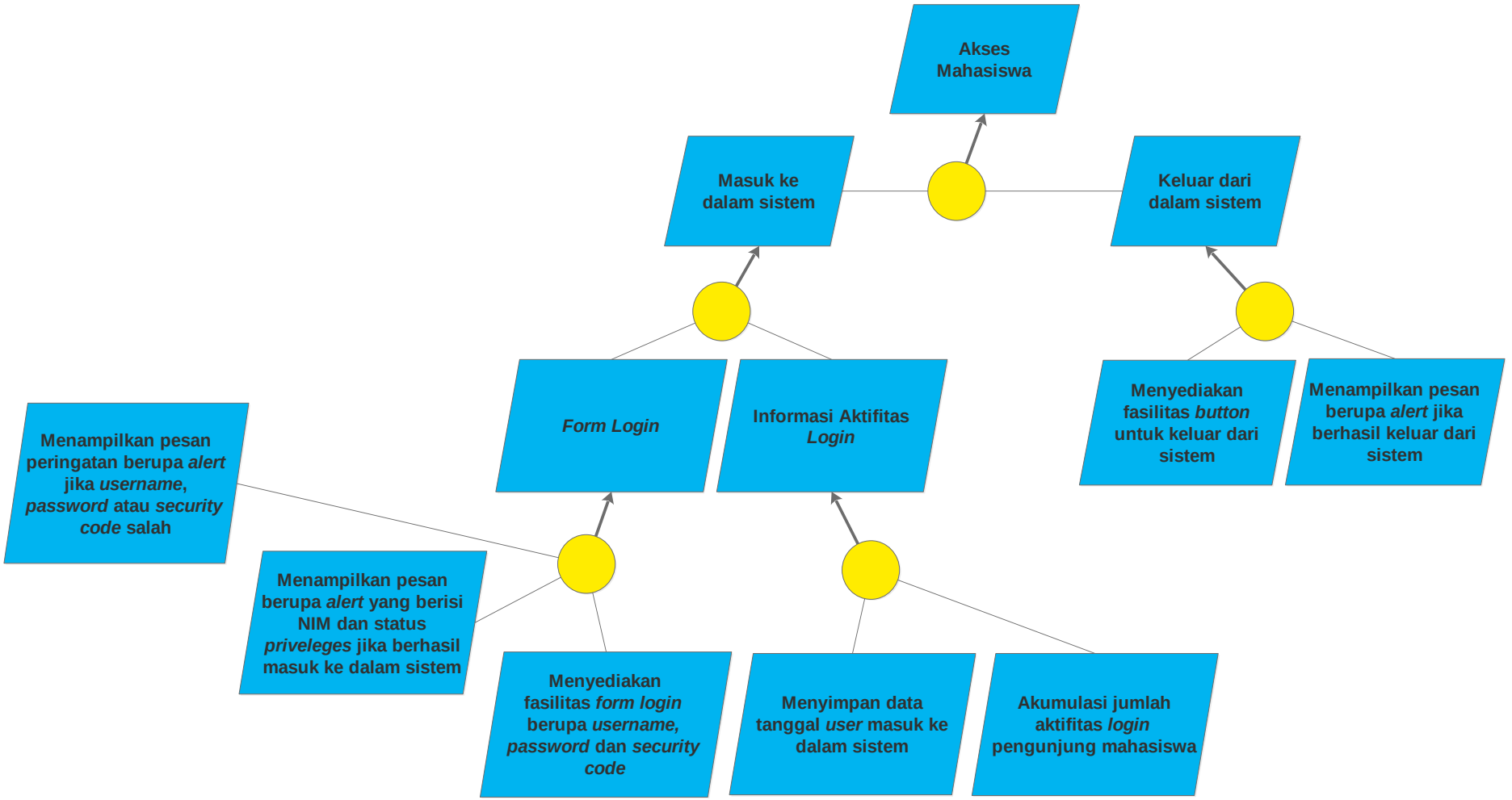
Parralellogram graph pada Gambar 4.1 merupakan hasil dari *requirements elicitation* pada tampilan anatarmuka aplikasi siap pakai yaitu Modul Mahasiswa Siakad UNJ. Berdasarkan Gambar 4.1 dapat diketahui untuk memenuhi tujuan "Manajemen Administrasi Akademik Mahasiswa" Siakad UNJ harus memenuhi sebelas *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.

4.1.3. Hasil Analisis *Subgoal*

4.1.3.1. Hasil Analisis *Subgoal* Akses Mahasiswa

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

4.1.3.2. Hasil Analisis *Subgoal* Manajemen Akun

Berdasarkan tampilan antarmuka Siakad UNJ, *subgoal* "manajemen akun" yang ditangani oleh sistem terdiri dari mengubah *password* dan mengunggah foto profil akun. Mengubah *password* ditangani oleh sistem dengan *goal* "*form* ubah *password*" yang meliputi: (1) Menyediakan fasilitas *form* ubah *password* berupa *username*, *password* lama, *password* baru dan verifikasi *password* baru; (2) Menampilkan pesan berupa *alert* yang berisi *password* baru jika berhasil mengubah *password*; (3) Menampilkan pesan berupa *alert* jika gagal mengubah *password*. Unggah foto profil akun ditangani oleh sistem dengan *goal* "*form* unggah foto profil" yang meliputi: (1) Menyediakan fasilitas *form* unggah *file* foto; (2) Menampilkan pesan peringatan mengenai jenis dan ukuran dari file foto berupa teks; (3) Menampilkan pesan berupa alert jika berhasil mengubah foto; (4) Menampilkan pesan berupa alert jika gagal mengubah foto; (5) Hanya *file* gambar bertipe png, jpg dan gif berukuran tidak lebih dari 75 KB yang dapat di unggah. *Parralelogram graph* hasil dari *refinement* "*subgoa*l akses mahasiswa" dicantumkan di dalam lampiran.

4.1.3.3. Hasil Analisis *Subgoal* Informasi Panduan

Berdasarkan tampilan antarmuka Siakad UNJ, *subgoal* "informasi panduan" yang ditangani oleh sistem meliputi: (1) Cara penggunaan sistem; (2) Dokumen panduan sistem; (3) *Form* administrasi sistem; (4) Dokumen panduan akademik.

Goal "cara penggunaan sistem" ditangani oleh sistem dengan menampilkan langkah-langkah pemakaian Siakad UNJ. *Goal* "dokumen panduan sistem" ditangai oleh sistem dengan *goal* "panduan siakad dosen" dan *goal* "panduan siakad

mahasiswa". Pemenuhan *goal* tersebut ditangani oleh sistem dengan menyediakan (1) Menyediakan tautan untuk unduh panduan siakad dosen (2) Menyediakan tautan untuk unduh panduan siakad mahasiswa. *Goal* "*form* administrasi sistem" ditangani oleh sistem dengan beberapa *goal*, yaitu: (1) *Form* penggantian password dosen; (2) *Form* penggantian *password* mahasiswa; (3) *Form input* nilai terlambat; (4) *Form* pembuatan email dosen UNJ. Tiap *goal* tersebut dipenuhi oleh sistem dengan menyediakan tautan, yaitu: (1) Menyediakan tautan untuk unduh *form* penggantian *password* dosen; (2) Menyediakan tautan untuk unduh *form* penggantian *password* mahasiswa; (3) Menyediakan tautan untuk unduh *form input* nilai terlambat; (4) Menyediakan tautan untuk unduh *form* pembuatan email dosen UNJ. *Goal* "dokumen panduan akademik" ditangani oleh sistem dengan menyediakan tautan untuk mengunduh BPA UNJ tahun 2014. *Parralelogram graph* hasil dari *refinement* "*subgoa*l informasi panduan" dicantumkan di dalam lampiran.

4.1.3.4. Hasil Analisis *Subgoal* Informasi Pengunjung

Berdasarkan tampilan antarmuka Siakad UNJ, *subgoal* "informasi pengunjung" yang ditangani oleh sistem meliputi: (1) Data pengunjung mahasiswa; (2) Jumlah pengunjung. *Goal* "data pengunjung mahasiswa" yang ditangani oleh sistem meliputi: (1) Nama pengunjung mahasiswa; (2) Status *priveleges* pengunjung mahasiswa (3) Jumlah aktifitas *login* pengunjung mahasiswa; (4) Tanggal terakhir *login* pengunjung mahasiswa; (5) Foto profil pengunjung mahasiswa. Tiap goal tersebut dipenuhi dengan: (1) Menampilkan nama pengunjung mahasiswa yang telah melakukan login ke dalam sistem; (2) Menampilkan status priveleges pengunjung mahasiswa yang telah melakukan login ke dalam sistem; (3) Menampilkan kalkulasi jumlah total aktifitas masuk ke dalam

sistem yang telah dilakukan; (4) Menampilkan tanggal mahasiswa melakukan aktifitas masuk ke dalam sistem; (5) Menampilkan foto profil mahasiswa. *Goal* "jumlah pengunjung" dipenuhi oleh sistem dengan: (1) Menampilkan jumlah pengunjung berdasarkan negara yang dihitung oleh *plugin* supercounters; (2) Menampilkan jumlah pengunjung berdasarkan waktu hari ini, bulan, dan total yang dihitung oleh plugin supercounters; (3) Menyedikan tautan menuju ke supercounters. *Parralelogram graph* hasil dari *refinement* "*subgoa*l informasi panduan" dicantumkan di dalam lampiran.

4.1.3.5. Hasil Analisis *Subgoal* Manajemen Data Pribadi Mahasiswa

Berdasarkan tampilan antarmuka Siakad UNJ, *subgoal* "manajemen data pribadi mahasiswa" yang ditangani oleh sistem meliputi: (1) Menampilkan data mahasiswa; (2) Ubah data pribadi mahasiswa; (3) Ubah data asal sekolah; (4) Ubah data kontak; (5) Ubah data orangtua.

Goal "ubah data pribadi" ditangani oleh sistem dengan goal "*form* ubah data pribadi" yang meliputi: (1) Menampilkan data pribadi yang sudah diisi sebelumnya; (2) Menyediakan fasilitas *form* ubah data pribadi mahasiswa; (3) Menyediakan fasilitas untuk memeriksa kolom yang belum terisi pada *form* ubah data pribadi; (4) Menampilkan pesan berupa *eror* jika terdapat kolom yang belum terisi pada *form* ubah data pribadi.

Goal "ubah data asal sekolah" ditangani oleh sistem dengan goal "*form* ubah data pribadi" yang meliputi: (1) Menampilkan data asal sekolah yang sudah diisi sebelumnya; (2) Menyediakan fasilitas *form* ubah data asal sekolah; (3) Menyediakan fasilitas untuk memeriksa kolom yang belum terisi pada *form* ubah

data asal sekolah; (4) Menampilkan pesan berupa *eror* jika terdapat kolom yang belum terisi pada *form* ubah data asal sekolah.

Goal "ubah data kontak" ditangani oleh sistem dengan goal "*form* ubah data pribadi" yang meliputi: (1) Menampilkan data kontak yang sudah diisi sebelumnya; (2) Menyediakan fasilitas *form* ubah data kontak; (3) Menyediakan fasilitas untuk memeriksa kolom yang belum terisi pada *form* ubah data kontak; (4) Menampilkan pesan berupa *eror* jika terdapat kolom yang belum terisi pada *form* ubah data kontak.

Goal "ubah data orangtua" ditangani oleh sistem dengan goal "*form* ubah data pribadi" yang meliputi: (1) Menampilkan data orangtua yang sudah diisi sebelumnya; (2) Menyediakan fasilitas *form* ubah data orangtua; (3) Menyediakan fasilitas untuk memeriksa kolom yang belum terisi pada *form* ubah data orangtua; (4) Menampilkan pesan berupa *eror* jika terdapat kolom yang belum terisi pada *form* ubah data orangtua. *Parralelogram graph* hasil dari *refinement subgoa*l "manajemen data pribadi mahasiswa" dicantumkan di dalam lampiran

4.1.3.6. Hasil Analisis *Subgoal* Informasi Perkuliahan

Berdasarkan tampilan antarmuka Siakad UNJ, *subgoal* "informasi perkuliahan" yang ditangani oleh sistem meliputi: (1) Pengumuman pembayaran; (2) Pengumuman pengisian KRS; (3) Jadwal Kuliah.

Goal "pengumuman pembayaran" ditangani oleh sistem dengan menampilkan informasi seputar tanggal-tanggal yang berkaitan dengan pembayaran kuliah. *Goal* "Pengumuman pengisian KRS" ditangani oleh sistem dengan menampilkan informasi seputar tanggal-tanggal yang berkaitan dengan pengisian KRS.

Goal "jadwal kuliah" ditangani oleh sistem dengan *goal* "*form* pilih semester" dan "daftar mata kuliah". *Goal* "*form* pilih semester" yang disediakan oleh sistem meliputi: (1) Menyediakan fasilitas untuk melihat jadwal mata kuliah berdasarkan semester; (2) Menyediakan fasilitas *form* untuk memilih jadwal mata kuliah yang diinginkan; (3) Menyediakan pilihan semester sejak tahun ajaran 1999/2000 sampai semester yang sedang aktif. *Goal* "daftar mata kuliah" yang disediakan oleh sistem meliputi: (1) Menampilkan daftar seluruh jadwal mata kuliah yang dibuka berdasarkan kode seksi pada tiap semester; (2) Menghitung jumlah keseluruhan mata kuliha yang dibuka berdasarkan kode seksi pada tiap semester (3) Menampilkan jumlah keseluruhan mata kuliah yang dibuka berdasarkan kode seksi pada tiap semester; (4) Menyediakan fasilitas untuk mengkalkulasi jumlah peserta pada setiap kode seksi mata kuliah; (5) Menampilkan hasil kalkulasi jumlah peserta pada setiap kode seksi mata kuliah.

Goal "Pengumuman pengisian KRS" ditangani oleh sistem dengan menampilkan informasi seputar tanggal-tanggal yang berkaitan dengan pengisian KRS. *Parralelogram graph* hasil dari *refinement subgoa*l "informasi perkuliahan" tertera dicantumkan di dalam lampiran.

4.1.3.7. Hasil Analisis *Subgoal* Informasi Hasil Studi Mahasiswa

Berdasarkan tampilan antarmuka Siakad UNJ, *subgoal* "informasi hasil studi mahasiswa" yang ditangani oleh sistem meliputi: (1) Daftar hasil studi; (2) Kartu Hasil Studi (KHS).

Goal "Daftar hasil studi" yang ditangani oleh sistem meliputi: (1) Data mahasiswa; (2) Hasil perhitungan IPK; (3) Tabel daftar hasil studi seluruh mata kuliah. *Goal* "Data mahasiswa" ditangani oleh sistem dengan menampilkan data

mahasiswa berupa nama, NIM, fakultas, jurusan dan prodi. *Goal* "Perhitungakn IPK" ditangani oleh sistem dengan: (1) Menghitung total bobot yang diperoleh dari seluruh mata kuliah yang diambil; (2) Menampilkan total bobot yang diperoleh dari seluruh mata kuliah yang diambil; (3) Menghitung total sks yang diperoleh dari seluruh mata kuliah yang telah diambil; (4) Menampilkan total sks yang diperoleh dari seluruh mata kuliah yang telah diambil; (5) Menghitung indek prestasi kumulatif (IPK) dari seluruh mata kuliah yang telah diambil; (6) Menampilkan indeks prestasi kumulatif dari seluruh mata kuliah yang telah diambil. Goal "tabel daftar hasil studi seluruh mata kuliah" ditangani oleh sistem dengan: (1) Menampilkan daftar hasil studi seluruh mata kuliah yang telah diambil meliputi nama mata kuliah, nama dosen, sks, nilai, bobot, semester, dan status terurut berdasarkan semester; (2) Mengkonversi seluruh nilai pada setiap mata kuliah yang diambil ke dalam skor; (3) Menghitung nilai bobot yang diperoleh pada setiap mata kuliah yang telah diambil berdasarkan sks dan nilai yang diperoleh; (4) Menampilkan nilai bobot yang diperoleh pada seluruh mata kuliah yang telah diambil berdasarkan sks dan nilai yang diperoleh.

Goal "kartu hasil studi mahasiswa" yang ditangani oleh sistem meliputi: (1) Form pilih semester; (2) Tabel daftar hasil studi mahasiswa; (3) Hasil perhitungan indeks prestasi semester; (4) Unduh KRS dan KHS. *Goal* "Form pilih semester" ditangan oleh sistem dengan: (1) Menyediakan fasilitas untuk melihat kartu hasil studi berdasarkan semester; (2) Menyediakan fasilitas form untuk memilih KHS pada semester yang diinginkan; (3) Menyediakan pilihan semester sejak semester mahasiswa kuliah sampai semester yang sedang aktif. *Goal* "Tabel daftar hasil studi semester" ditangan oleh sistem dengan: (1) Menghitung jumlah mata kuliah yang

terdapat di dalam KHS; (2) Menampilkan jumlah mata kuliah yang terdapat di dalam KHS; (3) Menampilkan daftar hasil studi mata kuliah sesuai dengan daftar mata kuliah yang telah dimasukkan pada KRS berupa kode seksi, nama mata kuliah, nama doesn, nilai, sks, dan bobot pada masing-masing semester; (4) Menampilkan nilai, sks dan hasil studi mata kuliah pada masing-masing semester setelah megisi form instrumen evaluasi mata kuliah; (5) mengkonversi seluruh nilai pada setiap mata kuliah yang terdapat pada KHS pada tiap semester; (6) Menampilkan nilai bobot yang diperoleh pada seluruh mata kuliah yang tredapat pada KHS pada tiap Semester; (7) Menghitung nilai bobot yang diperoleh pada seuruh mata kuliah yang terdapat pada kHS pada tiap semester. *Goal* "hasil perhitungan IPS" ditangan oleh sistem dengan: (1) Menghitung indeks prestas semester (IPS) berdasrkan mata kuliah yang terdapat pada KHS; (2) Menampilkan IPS berdasarkan mata kuliah yang terdapat pada KHS; (3) Menghitung total bobot yang diperoleh dari seluruh mata kuliah yang terdapat pada KHS pada tiap semester; (4) Menampilkan total bobot yang diperoleh dari seluruh mata kuliah yang terdapat pada KHS pada tiap semester; (5) Menghitung total sks yang diperoleh berdasarkan mata kuliah yang terdapat pada KHS pada tiap semester; (6) Menampilkan total sks yang diperoleh berdasarkan mata kuliah yang tredapat pada KHS pada tiap semester. *Goal* "unduh KRS dan KHS" ditangan oleh sistem dengan: (1) Menyediakan tautan unduk mengunduh KRS pada semester yang dipilih; (2) Menyediakan tautan untuk mengunduh KHS pada semester yang dipilih. *Parralelogram graph* hasil dari *refinement subgoa*l " informasi hasil studi mahasiswa" dicantumkan di dalam lampiran.

4.1.3.8. Hasil Analisis *Subgoal* Pengisian KRS

Berdasarkan tampilan antarmuka Siakad UNJ, *subgoal* "pengisian KRS" yang ditangani oleh sistem meliputi: (1) *Form* pengisian KRS; (2) Jumlah SKS yang diperbolehkan; (3) Daftar mata kuliah; (4) Keterangan pembimbing akademik; (5) Unduh KRS.

Goal "*form* pengisian KRS" ditangani oleh sistem dengan: (1) Menyediakan fasilitas pengisian KRS pada waktu yang telah ditentukan; (2) Menyediakan fasilitas *form* untuk memilih pengsisian KRS pada semester yang diinginkan; (3) Menyediakan pilihan semester untuk mengisi KRS hanya sampai awal semster mahasiswa mulai kuliha hingga semester yang sedang aktif; (4) Menonaktifkan pilihan semester yang sudah berlalu; (5) Menyediakan fasilitas *form* pengisian kode mata kuliah; (6) Menampilkan pesan berupa *alert* jika proses penambahan maa kuliah berhasil; (7) Menampilkan pesan berupa *alert* jika proses penghapusan ata kuliah yang telah dimasukkan ke dalam KRS berhasil; (8) Menyediakan fasilitas menghapus mata kuliah yang telah dimasukkan ke dalam KRS.

Goal "Jumlah SKS yang diperbolehkan" ditangani oleh sistem dengan: (1) Menampilkan IPS sebelumnya dari semester yang sedang aktif; (2) Menghitung jumlah SKS yang diperbolehkan berdasarkan IPS sebelumnya; (3) Menampilkan jumlah SKS yang diperbolehkan berdasarkan IPS sebelumnya; (4) Menghitung total mata kuliah telah dimasukkan ke dalam KRS; (5) Menampilkan total jumlah sks dari mata kuliah yang telah dimasukkan ke dalam KRS.

Goal "Daftar mata kuliah" ditangani oleh sistem dengan: (1) Menampilkan daftar mata kuliah yang telah dimasukan ke dalam KRS; (2) Menghitung jumlah mata kuliah yang telah dimasukan ke dalam KRS.

Goal "Keterangan pembimbing akadmeik" ditangani oleh sistem dengan menampilkan keterangan dosen pembimbing akademik terhadap KRS. *Goal* "Unduh KRS" ditangani oleh sistem dengan menyediakan tautan untuk mengunduh KRS. *Parralelogram graph* hasil dari *refinement subgoa*l "pengisian KRS" tertera di dalam lampiran.

4.1.3.9. Hasil Analisis *Subgoal* Pendaftaran PKM

Berdasarkan tampilan antarmuka Siakad UNJ, *subgoal* "pendaftaran PKM" yang ditangani oleh dengan goal "form pendaftaran PKM" yang meliputi: (1) Menyediakan fasilitas form pendaftaran PKM pada semester yang sedang berlangsung pada waktu yang ditentukan; (2) Menyediakan fasilitas *form* pendaftaran PKM pada semester yang sedang berlangsung pada waktu yang ditentukan; (3) Menampilkan pesan peringatan jika waktu pendaftaran telah selesai. *Parralelogram graph* hasil dari *refinement subgoa*l "pendaftaran PKM" dicantumkan di dalam lampiran.

4.1.3.10. Hasil Analisis *Subgoal* Evaluasi Perkuliahan

Berdasarkan tampilan antarmuka Siakad UNJ, *subgoal* "informasi pengunjung" yang ditangani oleh sistem meliputi: (1) Form evaluasi; (2) Status evaluasi mata kuliah; (3) Panduan evaluasi.

Goal "*form* evaluasi" dipenuhi oleh sistem dengan: (1) Menyediakan fasilitas *form* evaluasi mata kuliah berdasarkan semester yang sedang berlangsung; (2) Menampilkan daftar mata kuliah yang diikuti sesuai dengan KRS fi form evaluasi perkuliahan; (3) Menghitung total sks dari mata kuliah yang diikuti sesuai dengan KRS di form evaluasi perkuliahan; (4) Menampilkan jumlah sks dari mata kuliah

yang telah didaftarkan pada KRS di *form* evaluasi perkuliahan; (5) Menampilkan pesan berupa *alert* jika berhasil melakukan pengisian *form* evaluasi mata kuliah; (6) Menampilkan pesan berupa *alert* jika terdapat kolom yang belum terisi pada *form* evaluasi.

Goal "status evaluasi mata kuliah" dipenuhi oleh sistem dengan: (1) Memeriksa status pengisian evalasi mata kuliah; (2) Menampilkan status pengisian evaluasi mata kuliah.

Goal "panduan evaluasi" dipenuhi oleh sistem dengan: (1) Menampilkan teks petunjuk pengisian evaluasi mata kuliah; (2) Evaluasi perkuliahan yang telah diisi tidak dapat diubah. *Parralelogram graph* hasil dari *refinement* "*subgoa*l informasi panduan" dicantumkan di dalam lampiran.

4.1.3.11. Hasil Analisis *Subgoal* Verifikasi Pembayar*an*

Berdasarkan tampilan antarmuka Siakad UNJ, *subgoal* "verifikasi pembayaran" ditangani oleh sistem dengan: (1) Menghitung jumlah transaksi pembayaran yang telah dilakukan berdasarkan semester; (2) Menampilkan jumlah transaksi pembayaran yang telah dilakukan berdasarkan semester; (3) Menampilkan jumlah nominal pembayaran yang telah dilakuka berdasarkan jenis pembayaran pada tiap semester; (4) Menghitung total nominal pembayaran yang telah dilakukan berdasarkan jenis pembayaran pada tiap semester; (5) Menampilkan jumlah total nominal pembayaran yang telah dilakukan berdasarkan semester *parralelogram graph* hasil dari *refinement subgoa*l "verifikasi pembayaran" dicantumkan di dalam lampiran.

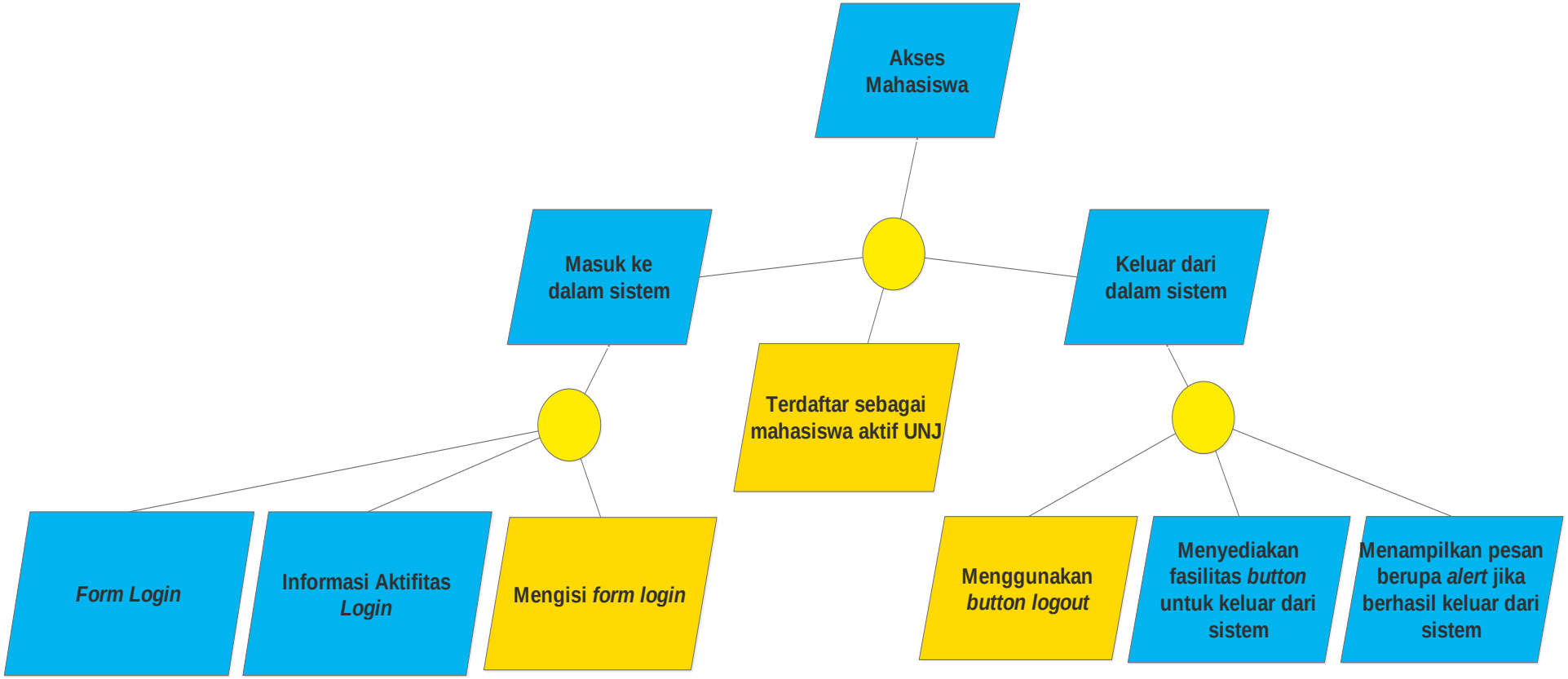
4.1.4. Hasil Analisis *Expectation* dan *Obstacle*

4.1.4.1. Hasil Analisis *Expectation* dan *Obstacle Akses* Mahasiswa

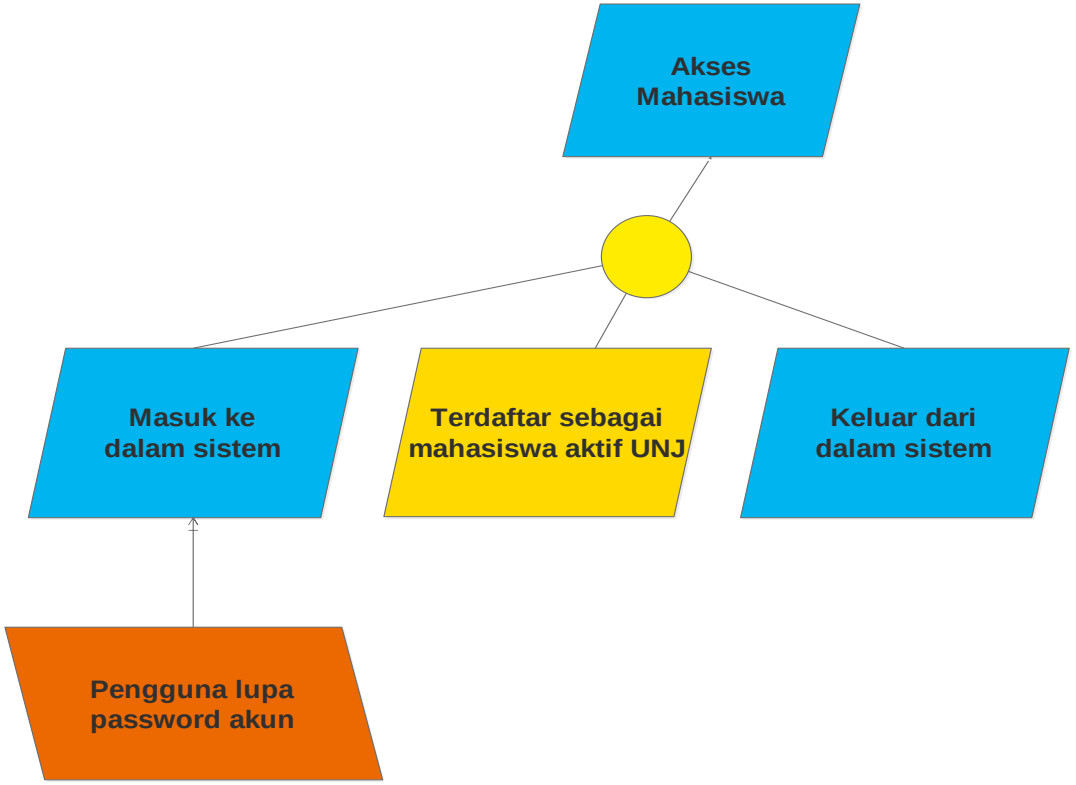
Berdasarkan tampilan antarmuka Siakad UNJ, *goal* "akses mahasiswa" memiliki tiga *expectation* yang harus dilakukan oleh mahasiswa sebagai *user* agar *goal* "akses mahasiswa" dapat tercapai. *Expectation* tersebut, yaitu: (1) Terdaftar sebagai mahasiswa aktif UNJ; (2) Mengisi *form login*; (3) Menekan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



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

4.1.4.2. Hasil Analisis *Expectation* dan *Obstacle* Manajemen Akun

Berdasarkan tampilan antarmuka Siakad UNJ, *goal* "manajemen akun" memiliki tiga *expectation* yang harus dilakukan oleh mahasiswa sebagai *user* agar *goal* "manajemen akun" dapat tercapai. *Expectation* tersebut, yaitu: (1) Masuk ke dalam sistem dengan mengisi *form login*; (2) Mengisi *form* ubah *password*; (3) Mengisi *form unggah* file foto melalui *form* unggah file foto.

Expectation "masuk ke dalam sistem dengan mengisi *form login*" merupakan sebuah skenario atau kondisi berupa fakta yang harus dilakukan oleh jenis *user* mahasiswa untuk memenuhi *goal* "manajemen akun". *Expectation* "Mengisi *form* ubah *password*" merupakan sebuah skenario atau kondisi berupa fakta yang harus dilakukan oleh jenis *user* mahasiswa untuk memenuhi *goal* "ubah *password*". *Expectation* "Unggah *file* foto melalui form unggah *file* foto" merupakan sebuah skenario atau kondisi berupa fakta yang harus dilakukan oleh jenis *user* mahasiswa untuk memenuhi *goal* "unggah file foto".

Goal "unggah *file* foto" memiliki *obstacle* atau kondisi yang menghalangi pencapaian *goal* pada *goal* "Hanya file gambar bertipe png, jpg dan gif berukuran tidak lebih dari 75 KB yang dapat di unggah", yaitu: pengguna tidak memiliki *file* foto dengan ukuran yang telah ditentukan. *Parralelogram graph* hasil dari analisis *expectation* dan *obstacle* dari *goal* "manajemen akun" dicantumkan di dalam lampiran.

4.1.4.3. Hasil Analisis *Expectation* dan *Obstacl*e Informasi Panduan

Berdasarkan tampilan antarmuka Siakad UNJ, *goal* "informasi panduan" memiliki tiga *expectation* yang harus dilakukan oleh mahasiswa sebagai *user* agar *goal* "informasi panduan" dapat tercapai. *Expectation* tersebut, yaitu:

(1) Mengunduh dokumen panduan siakad melalui tautan; (2) Mengunduh *form* administrasi sitem melalui tautan; (3) Mengunduh dokumen panduan akademik melalui tautan.

Expectation "Mengunduh dokumen panduan siakad melalui tautan" merupakan sebuah skenario atau kondisi berupa fakta yang harus dilakukan oleh jenis *user* mahasiswa untuk memenuhi *goal* "dokumen panduan sistem". *Expectation* "Mengunduh *form* administrasi sitem melalui tautan" merupakan sebuah skenario atau kondisi berupa fakta yang harus dilakukan oleh jenis *user* mahasiswa untuk memenuhi *goal* "*form* administrasi sitem". *Expectation* "Mengunduh dokumen panduan akademik melalui tautan" merupakan sebuah skenario atau kondisi berupa fakta yang harus dilakukan oleh jenis *user* mahasiswa untuk memenuhi *goal* "dokumen panduan akademik".

Goal "informasi panduan" tidak memiliki *obstacle* atau kondisi yang menghalangi pencapaian *goal* dan *subgoal* turunannya. *Parralelogram graph* hasil dari analisis *expectation* dan *obstacle* dari *goal* "informasi panduan" dicantumkan di dalam lampiran.

4.1.4.4. Hasil Analisis *Expectation* dan *Obstacle* Informasi Pengunjung

Berdasarkan tampilan antarmuka Siakad UNJ, *goal* "informasi pengunjung" memiliki satu *expectation* yang harus dilakukan oleh mahasiswa sebagai *user* agar *goal* "informasi pengunjung" dapat tercapai. *Expectation* tersebut, yaitu: Masuk ke dalam sistem dengan mengisi *form login*.

Expectation "masuk ke dalam sistem dengan mengisi *form login*" merupakan sebuah skenario atau kondisi berupa fakta yang harus dilakukan oleh jenis *user* mahasiswa untuk memenuhi *goal* "data pengunjung mahasiswa".

Goal “informasi pengunjung” tidak memiliki *obstacle* atau kondisi yang menghalangi pencapaian *goal* dan *subgoal* turunannya. *Parralelogram graph* hasil dari analisis *expectation* dan *obstacle* dari *goal* “informasi pengunjung” dicantumkan di dalam lampiran.

4.1.4.5. Hasil Analisis *Expectation* dan *Obstacle* Manajemen Data Pribadi Mahasiswa

Berdasarkan tampilan antarmuka Siakad UNJ, *goal* “manajemen data pribadi mahasiswa” memiliki lima *expectation* yang harus dilakukan oleh mahasiswa sebagai *user* agar *goal* “manajemen data pribadi dapat tercapai. *Expectation* tersebut, yaitu: (1) Masuk ke dalam sistem dengan mengisi *form login*; (2) Mengisi *form* ubah data pribadi; (3) Mengisi *form* ubah data asal sekolah; (4) Mengisi *form* ubah data kontak; (5) Mengisi *form* ubah data orangtua.

Expectation “masuk ke dalam sistem dengan mengisi *form login*” merupakan sebuah skenario atau kondisi berupa fakta yang harus dilakukan oleh jenis *user* mahasiswa untuk memenuhi *goal* “manajemen data pribadi mahasiswa”. *Expectation* “mengisi *form* ubah data pribadi” merupakan sebuah skenario atau kondisi berupa fakta yang harus dilakukan oleh jenis *user* mahasiswa untuk memenuhi *goal* “ubah data pribadi”. *Expectation* “mengisi *form* ubah data asal sekolah” merupakan sebuah skenario atau kondisi berupa fakta yang harus dilakukan oleh jenis *user* mahasiswa untuk memenuhi *goal* “ubah data asal sekolah”. *Expectation* “mengisi *form* ubah data kontak” merupakan sebuah skenario atau kondisi berupa fakta yang harus dilakukan oleh jenis *user* mahasiswa untuk memenuhi *goal* “ubah data kontak”. *Expectation* “mengisi *form* ubah data

orangtua" merupakan sebuah skenario atau kondisi berupa fakta yang harus dilakukan oleh jenis *user* mahasiswa untuk memenuhi *goal* "ubah data orangtua".

Goal "manajemen data pribadi mahasiswa" tidak memiliki *obstacle* atau kondisi yang menghalangi pencapaian *goal* dan *subgoal* turunannya. *Parralelogram graph* hasil dari analisis *expectation* dan *obstacle* dari *goal* "manajemen data pribadi" dicantumkan di dalam lampiran.

4.1.4.6. Hasil Analisis *Expectation* dan *Obstacl*e Informasi Perkuliahan

Berdasarkan tampilan antarmuka Siakad UNJ, *goal* "manajemen data pribadi mahasiswa" memiliki dua *expectation* yang harus dilakukan oleh mahasiswa sebagai *user* agar *goal* "informasi perkuliahan" dapat tercapai. *Expectation* tersebut, yaitu: (1) Masuk ke dalam sistem dengan mengisi *form login*; (2) Memilih semester untuk melihat jadwal kuliah melalui *form* jadwal mata kuliah.

Goal "informasi perkuliahan" tidak memiliki *obstacle* atau kondisi yang menghalangi pencapaian *goal* dan *subgoal* turunannya. *Parralelogram graph* hasil dari analisis *expectation* dan *obstacle* dari *goal* "informasi perkuliahan" tertera dicantumkna di dalam lampiran.

4.1.4.7. Hasil Analisis *Expectation* dan *Obstacle* Informasi Hasil Studi Mahasiswa

Berdasarkan tampilan antarmuka Siakad UNJ, *goal* "informasi hasil studi" memiliki empat *expectation* yang harus dilakukan oleh mahasiswa sebagai *user* agar *goal* "informasi hasil studi". *Expectation* tersebut, yaitu: (1) Masuk ke dalam sistem dengan mengisi *form login*; (2) Mengisi evaluasi mata kuliah; (3) Pilih semester kartu hasil melalui *form* pilih semester; (4) Unduh KRS dan KHS melalui tautan.

Expectation “masuk ke dalam sistem dengan mengisi *form login*” merupakan sebuah skenario atau kondisi berupa fakta yang harus dilakukan oleh jenis *user* mahasiswa untuk memenuhi *goal* “informasi hasil studi”. *Expectation* “Mengisi evaluasi mata kuliah” merupakan sebuah skenario atau kondisi berupa fakta yang harus dilakukan oleh jenis *user* mahasiswa untuk memenuhi *goal* “kartu hasil studi (KHS)”. *Expectation* “pilih semester kartu hasil melalui *form* pilih semester” merupakan sebuah skenario atau kondisi berupa fakta yang harus dilakukan oleh jenis *user* mahasiswa untuk memenuhi *goal* “kartu hasil studi (KHS)”. *Expectation* “unduh KRS dan KHS melalui tautan” merupakan sebuah skenario atau kondisi berupa fakta yang harus dilakukan oleh jenis *user* mahasiswa untuk memenuhi *goal* “unduh KRS dan KHS.

Goal “informasi hasil studi” tidak memiliki *obstacle* atau kondisi yang menghalangi pencapaian *goal* dan *subgoal* turunannya. *Parralelogram graph* hasil dari analisis *expectation* dan *obstacle* dari *goal* “informasi hasil studi” dicantumkan di dalam lampiran.

4.1.4.8. Hasil Analisis *Expectation* dan *Obstacle* Pengisian KRS

Berdasarkan tampilan antarmuka Siakad UNJ, *goal* “informasi hasil studi” memiliki lima *expectation* yang harus dilakukan oleh mahasiswa sebagai *user* agar *goal* “informasi hasil studi”. *Expectation* tersebut, yaitu: (1) Masuk ke dalam sistem dengan mengisi *form login*; (2) Mengisi evaluasi mata kuliah; (3) Pilih semester kartu hasil melalui *form* pilih semester; (4) Menggunakan *button* batal; (5) Unduh KRS dan KHS melalui tautan.

Expectation “masuk ke dalam sistem dengan mengisi *form login*” merupakan sebuah skenario atau kondisi berupa fakta yang harus dilakukan oleh jenis *user*

mahasiswa untuk memenuhi *goal* "pengisian krs". *Expectation* "mengisi evaluasi mata kuliah" merupakan sebuah skenario atau kondisi berupa fakta yang harus dilakukan oleh jenis *user* mahasiswa untuk memenuhi *goal* "kartu hasil studi (KHS)". *Expectation* "pilih semester kartu hasil melalui *form* pilih semester" merupakan sebuah skenario atau kondisi berupa fakta yang harus dilakukan oleh jenis *user* mahasiswa untuk memenuhi *goal* "kartu hasil studi (KHS)". *Expectation* "menggunakan *button logout*" merupakan sebuah skenario atau kondisi berupa fakta yang harus dilakukan oleh jenis *user* mahasiswa untuk memenuhi *goal* "*form* pengisian KRS". *Expectation* "unduh KRS dan KHS melalui tautan" merupakan sebuah skenario atau kondisi berupa fakta yang harus dilakukan oleh jenis *user* mahasiswa untuk memenuhi *goal* "unduh KRS dan KHS".

Goal "informasi hasil studi" tidak memiliki *obstacle* atau kondisi yang menghalangi pencapaian *goal* dan *subgoal* turunannya. *Parralelogram graph* hasil dari analisis *expectation* dan *obstacle* dari *goal* "informasi hasil studi" dicantumkan di dalam lampiran.

4.1.4.9. Hasil Analisis *Expectation* dan *Obstacle* Pendaftaran PKM

Berdasarkan tampilan antarmuka Siakad UNJ, *goal* "pendaftaran PKM" memiliki tiga *expectation* yang harus dilakukan oleh mahasiswa sebagai *user* agar *goal* "pendaftaran PKM" dapat tercapai. *Expectation* tersebut, yaitu: (1) Masuk ke dalam sistem dengan mengisi *form login*; (2) Mengisi *form* pendaftaran PKM. *Goal* "pendaftaran PKM" memiliki satu *obstacle, yaitu:* batal melakukan pendaftaran. *Parralelogram graph* hasil dari analisis *expectation* dan *obstacle* dari *goal* "verifikasi pembayaran" dicantumkan di dalam lampiran.

4.1.4.10.Hasil Analisis *Expectation* dan *Obstacle* Evaluasi Mata Kuliah

Berdasarkan tampilan antarmuka Siakad UNJ, *goal* "evaluasi perkuliahan" memiliki tiga *expectation* yang harus dilakukan oleh mahasiswa sebagai *user* agar *goal* "evaluasi perkuliahan" dapat tercapai. *Expectation* tersebut, yaitu: (1) Terdaftar sebagai peserta kuliah; (2) Masuk ke dalam sistem dengan mengisi *form login*; (3) Mengisi *form* instrumen evaluasi perkuliahan.

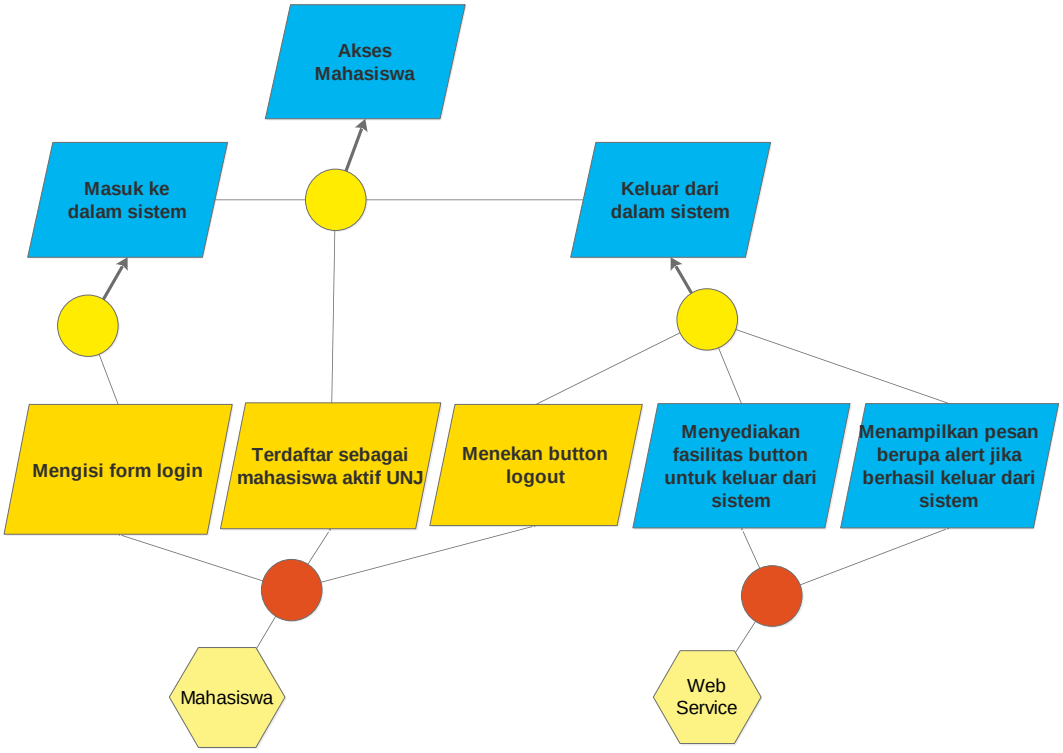
Goal "verifikasi pembayaran" dan *subgoal* turunannya tidak memiliki *obstacle* atau kondisi yang menghalangi pencapaian *goal*. *Parralelogram graph* hasil dari analisis *expectation* dan *obstacle* dari *goal* "evaluasi perkuliahan" dacnatumkan di dalam lampiran.

4.1.4.11.Hasil Analisis *Expectation* dan *Obstacle* Verifikasi Pembayaran

Berdasarkan tampilan antarmuka Siakad UNJ, *goal* "verifikasi pembayaran" memiliki dua *expectation* yang harus dilakukan oleh mahasiswa sebagai *user* agar *goal* verifikasi pembayaran dapat tercapai. *Expectation* tersebut, yaitu: (1) Melakukan pembayaran kuliah; (2) Masuk ke dalam sistem dengan mengisi *form login*. *Goal* "verifikasi pembayaran" dan *subgoal* turunannya tidak memiliki *obstacle* atau kondisi yang menghalangi pencapaian *goal*. *Parralelogram graph* hasil dari analisis expectation dan obstacle dari *goal* "verifikasi pembayaran" dicantumkan di dalam lampiran.

4.1.5. Hasil Analisis *Agent*

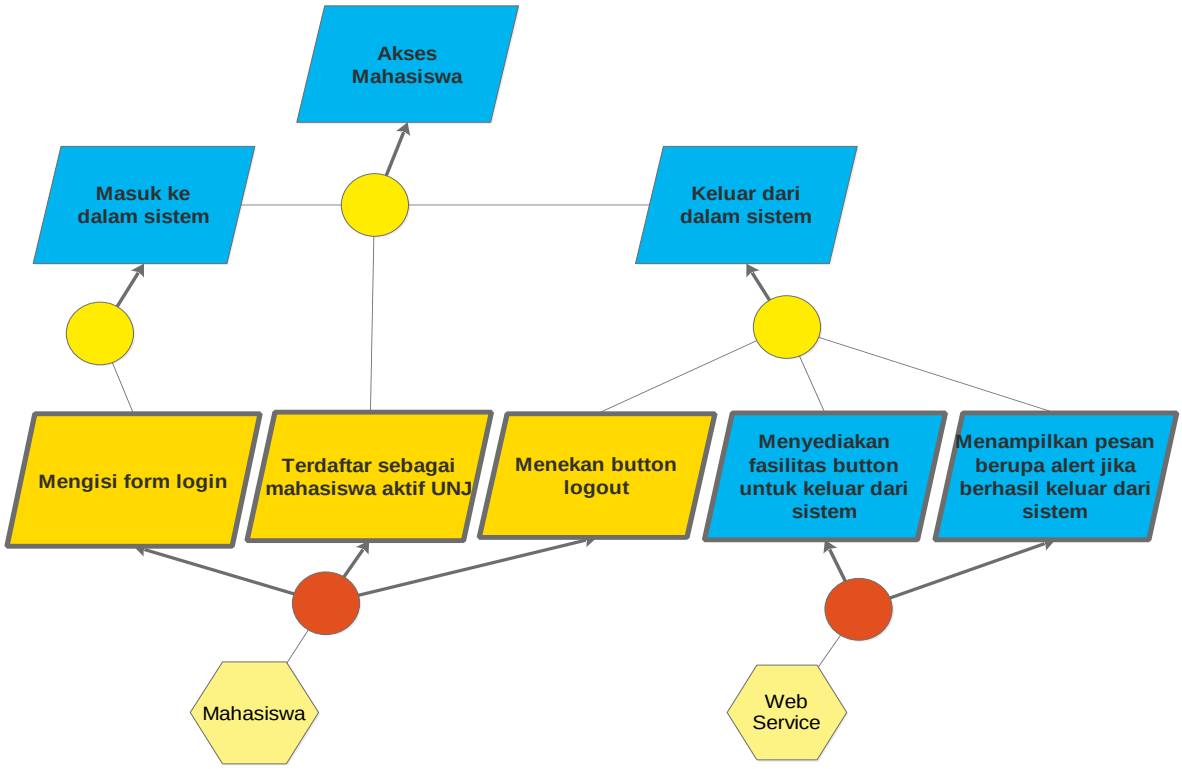
Berdasarkan tampilan anatarmuka sistem aplikasi Modul Mahasiswa Siakad UNJ dan proses analisis dengan menggunakan metode KAOS didapatkan hasil *agent* yang berperan dalam *functional requirement* sistem aplikasi siap pakai Modul Mahasiswa Siakad UNJ adalah *web service* dan mahasiswa. *Agent* di dalam 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 4.5 berikut ini adalah contoh *parralellogram graph* hasil analisis *agent*:



Gambar 4.5 Hasil Analisis *Agent* pada *Goal* Akses Mahasiswa

4.1.6. Hasil Representasi *Requirement*

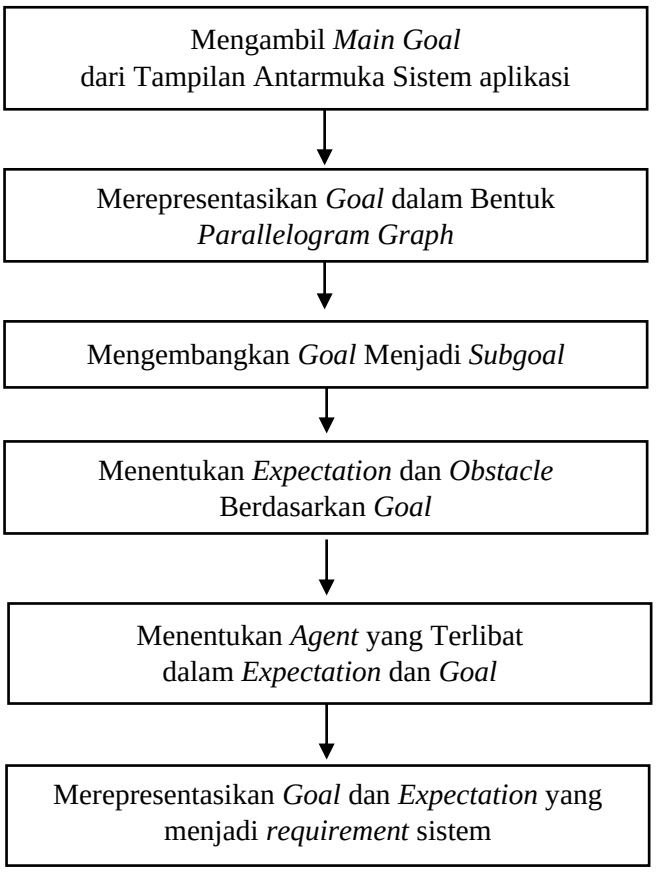
Requirement di dalam *parralelogram graph* merupakan sebuah *goal* yang berupa tugas yang harus dilakukan oleh *agent* untuk memenuhi pencapaian suatu *goal*. *Requirement* di dalam *parralelogram graph* digambarkan dengan menggunakan garis border tebal pada *goal* yang menjadi suatu *requirement* sistem. 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

4.1.7. Hasil Model *Reverse Engineering* dengan Metode KAOS

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 Model *Reverse Engineering* Menggunakan Model GORE dan Metode KAOS

4.2. Analisis Data Penelitian

Berdasarkan tampilan antarmuka dari sistem ampilkasi Siakad UNJ dan proses analisis serta penelusuran *requirement* dengan menggunakan metode KAOS, diperoleh *functional requirement* dengan jumlah 125. *Requirements* terebut kemudian dianalisis kembali dengan menggunakan *Requirement Traceability Matrix* untuk melihat keterkaitan antar *requirement*. Tabel 4.1 merupakan contoh dari rangkuman hasil analisis keterkaitan dari sebuah *requirement* yang ada pada Lampiran 24. Analisis keterkaitan untuk setiap *requirement* yang diperoleh dari analisis dengan menggunakan metode KAOS dicantumkan pada Lampiran 24 .

Tabel 4.1 Contoh Hasil Rangkuman Tabel Requirement Traceability Matrix Modul Mahasiswa Siakad UNJ

Kode *Requirement*	*Requirement*	Jumlah Relasi	Kode *Requirement* Relasi	*Requirement*
RSM015	Hanya file gambar bertipe png, jpg dan gif berukuran tidak lebih dari 75 KB yang dapat di unggah	3	RSM001	Menyediakan fasilitas *form login* berupa *username*, *password* dan *security code*
			RSM006	Menyediakan fasilitas *button* untuk keluar dari sistem
			RSM011	Menyediakan fasilitas *form* unggah *file* foto

Berdasarkan Tabel 4.1 dketahui bahwa *requirement* dengan kode RSM015 memiliki jumlah keterkaitan dengan tiga buah *requirement* lain, yaitu: RSM001, RSM006, dan RSM011. Informasi keterkaitan antar *requirement* akan membantu *developer* dalam mengembangkan sistem ketika mereka ingin memeriksa dampak dari perubahan satu *requirement* kepada *requirement* lainnya.

4.3. 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.

4.4. Aplikasi Hasil Penelitian

Model *reverse engineering* dengan menggunakan model GORE dan metode KAOS yang dihasilkan dapat diaplikasikan dan diterapkan pada proses analisis sistem perangkat lunak siap pakai untuk melakukan penelusuran sistem dan mendapatkan *functional requirement*.

BAB V

KESIMPULAN

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.

Kamus Besar Bahasa Indonesia. http://kbbi.web.id/ (diakses tanggal 10 Desember 2016 dan 27 Juli 2017).

IEEE STD 1220-1998. (1998). *Standard for Application and Management of the System Engineering Process*. New York: IEEE.

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

Pressman, R. S. (2010). *Software Engineering: A Practitioner's Approach, 7th Edition*. McGraw-Hill.

[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.

Santosa, Stefanus. 1994. *Reverse Engineering: Observasi – Struktur Berjenjang Terhadap Program Sumber Fortran [tesis]*. Jakarta: Program Pascasarjana, Universitas Indonesia.

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

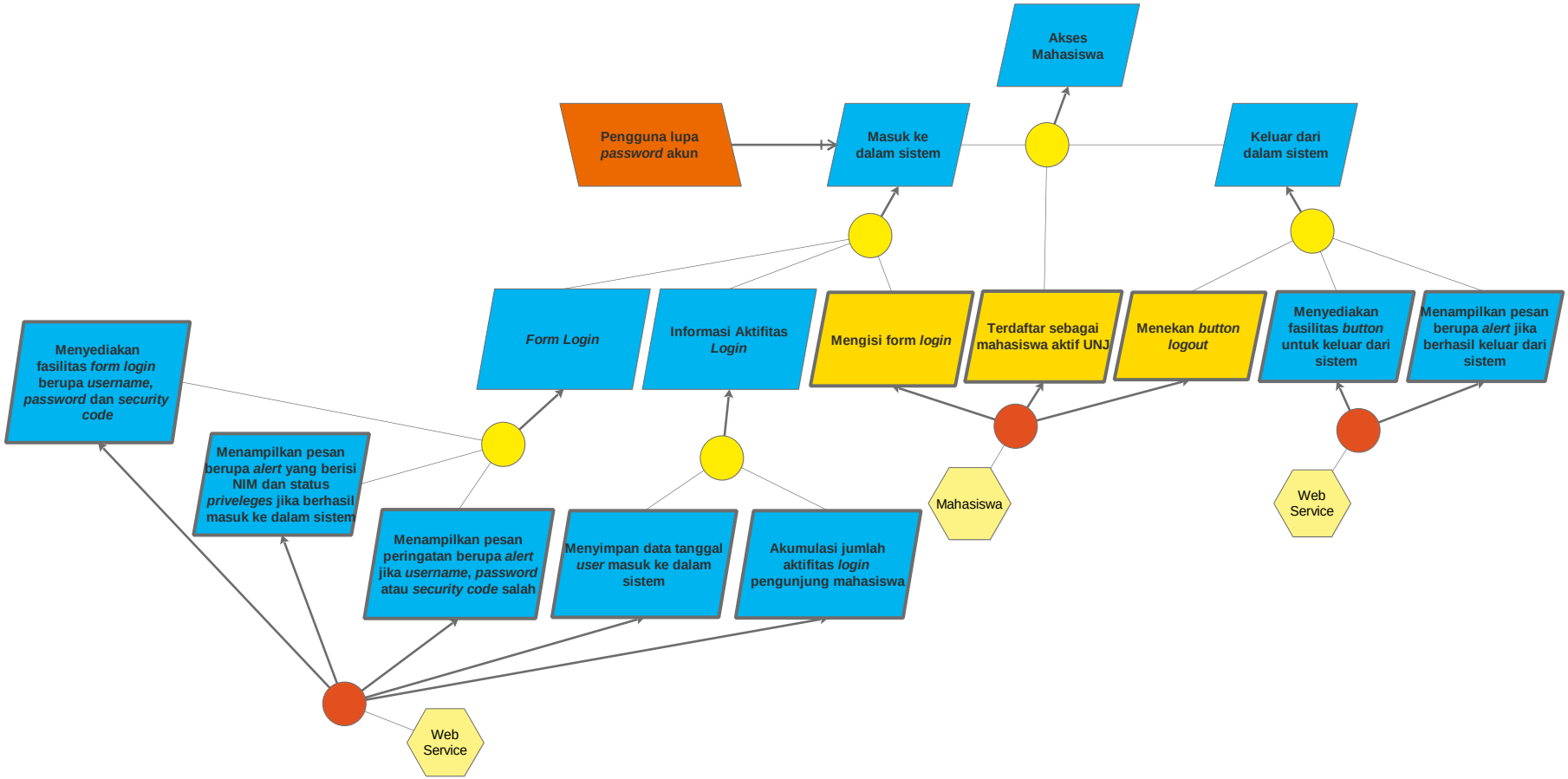
Sommerville, I. (2011). *Software Engineering: 9th Edition*. Boston: Addison-Wesley.

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

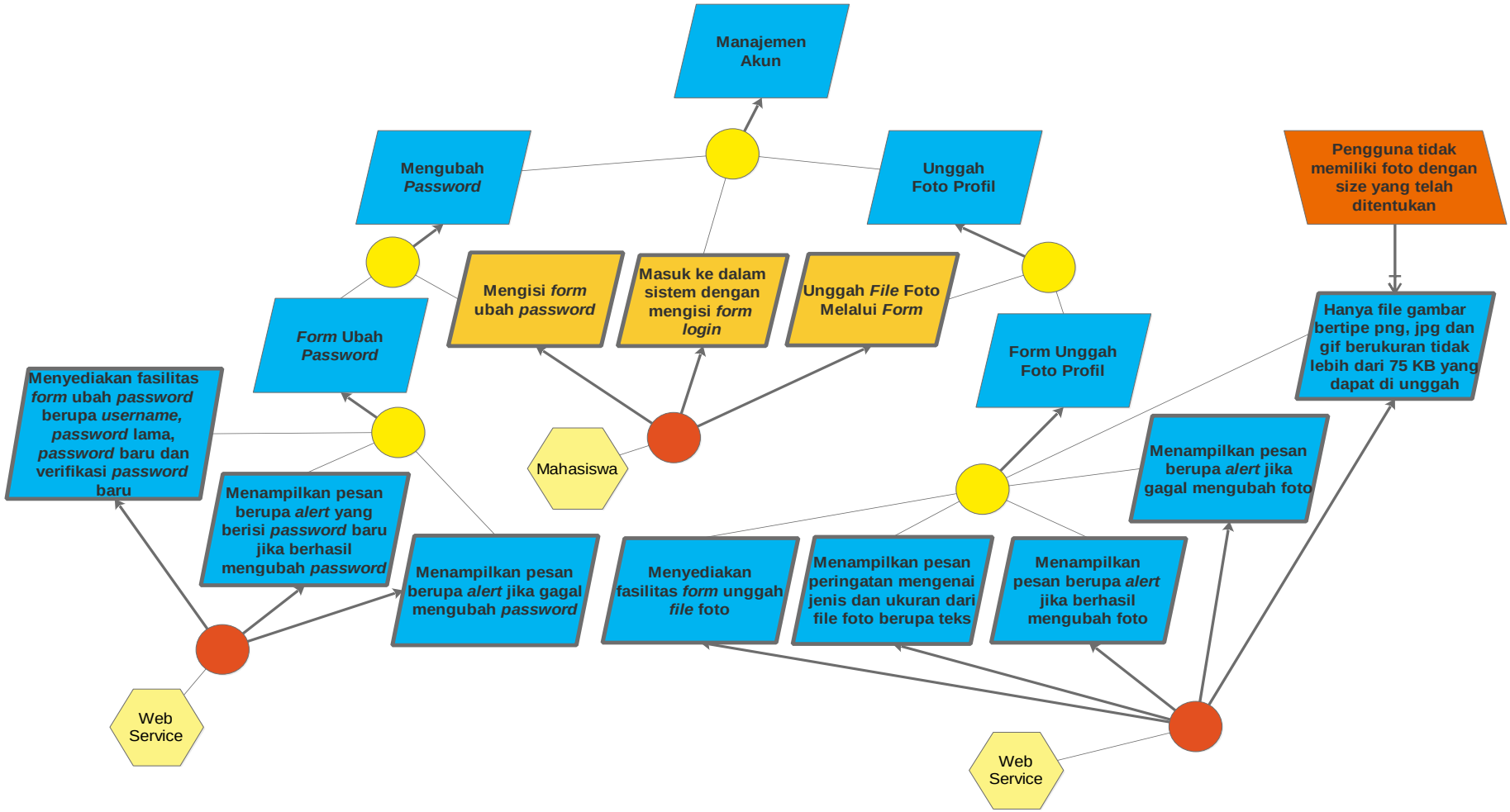
Zave, P. (1997). *Classification of Research Effortsin Requirements Engineering. ACM Computing Surveys*, 29(4), 315-321.

LAMPIRAN

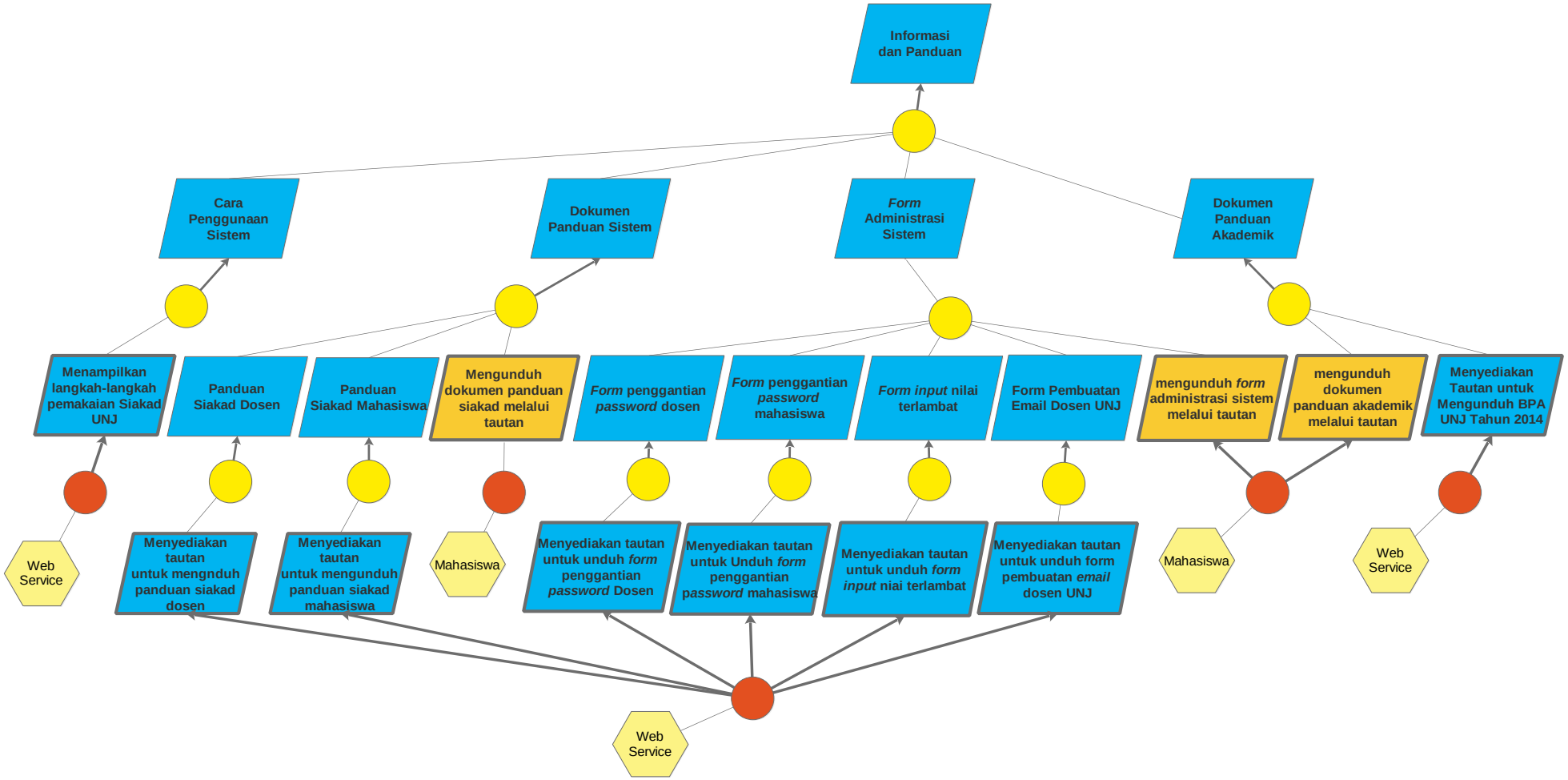
Lampiran 1. *Parralelogram graph* dari *goal* akses mahasiswa



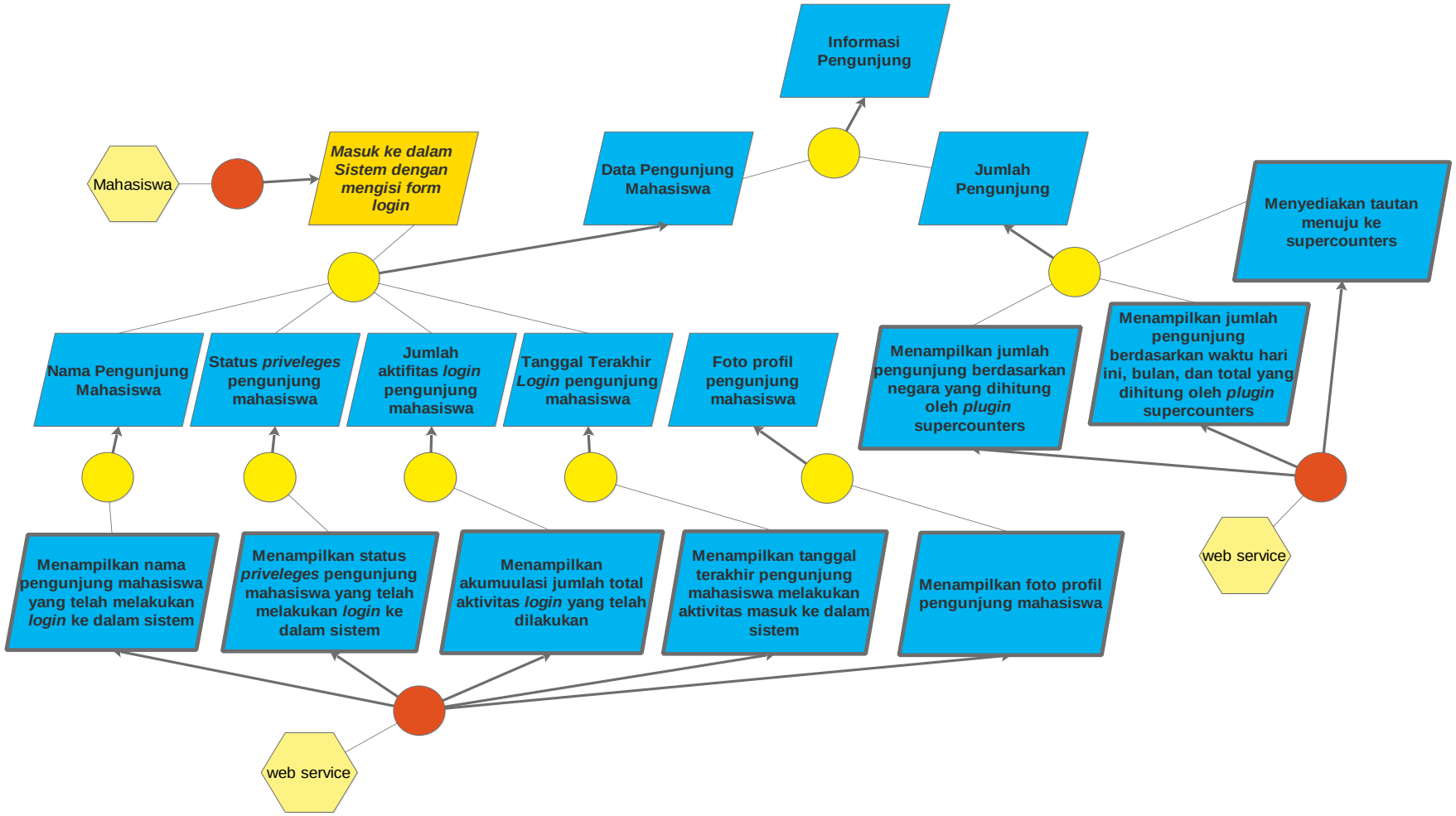
Lampiran 2. *Parralelogram graph* dari *goal* manajemen akun



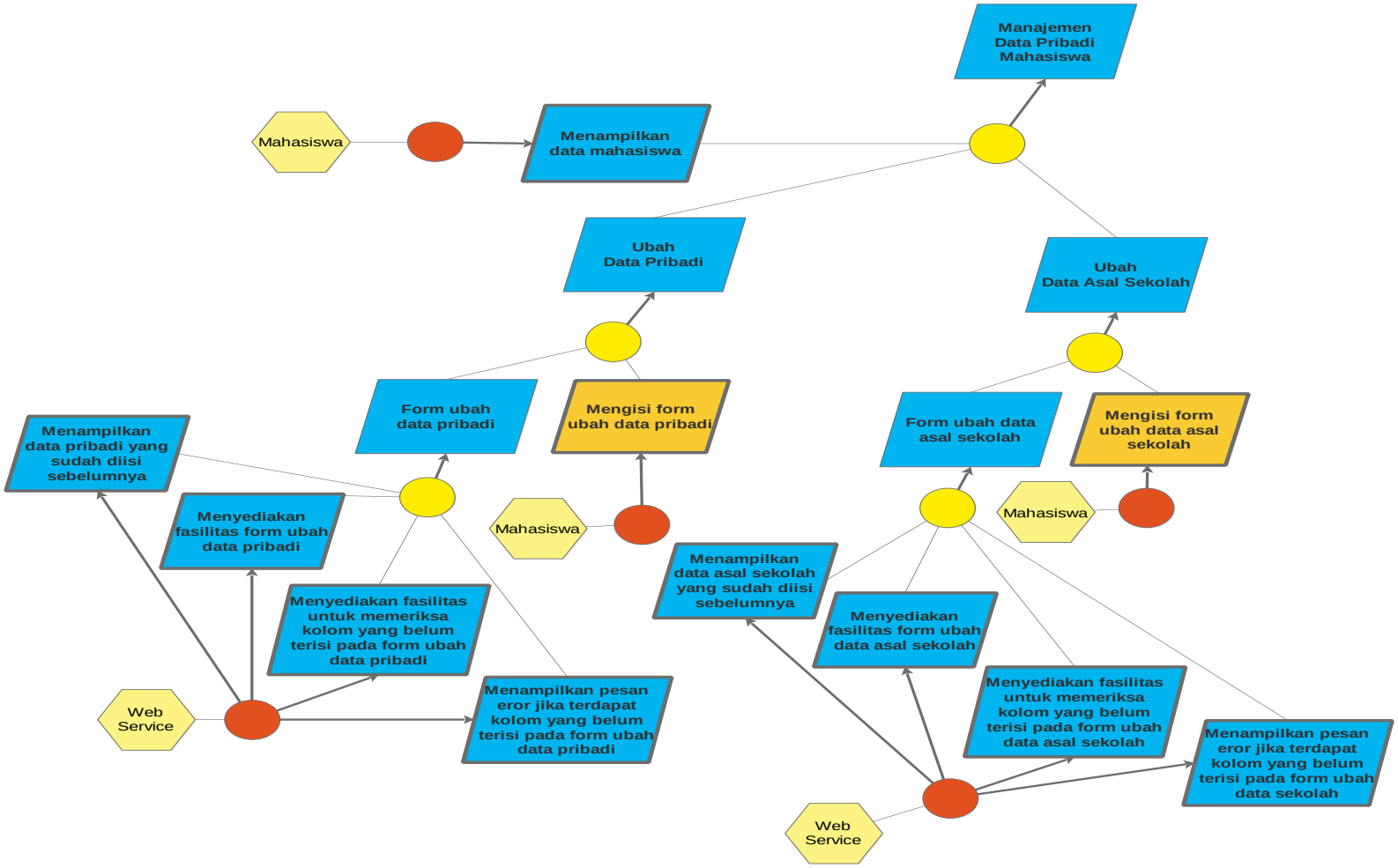
Lampiran 3. *Parralelogram graph* dari *goal* informasi panduan



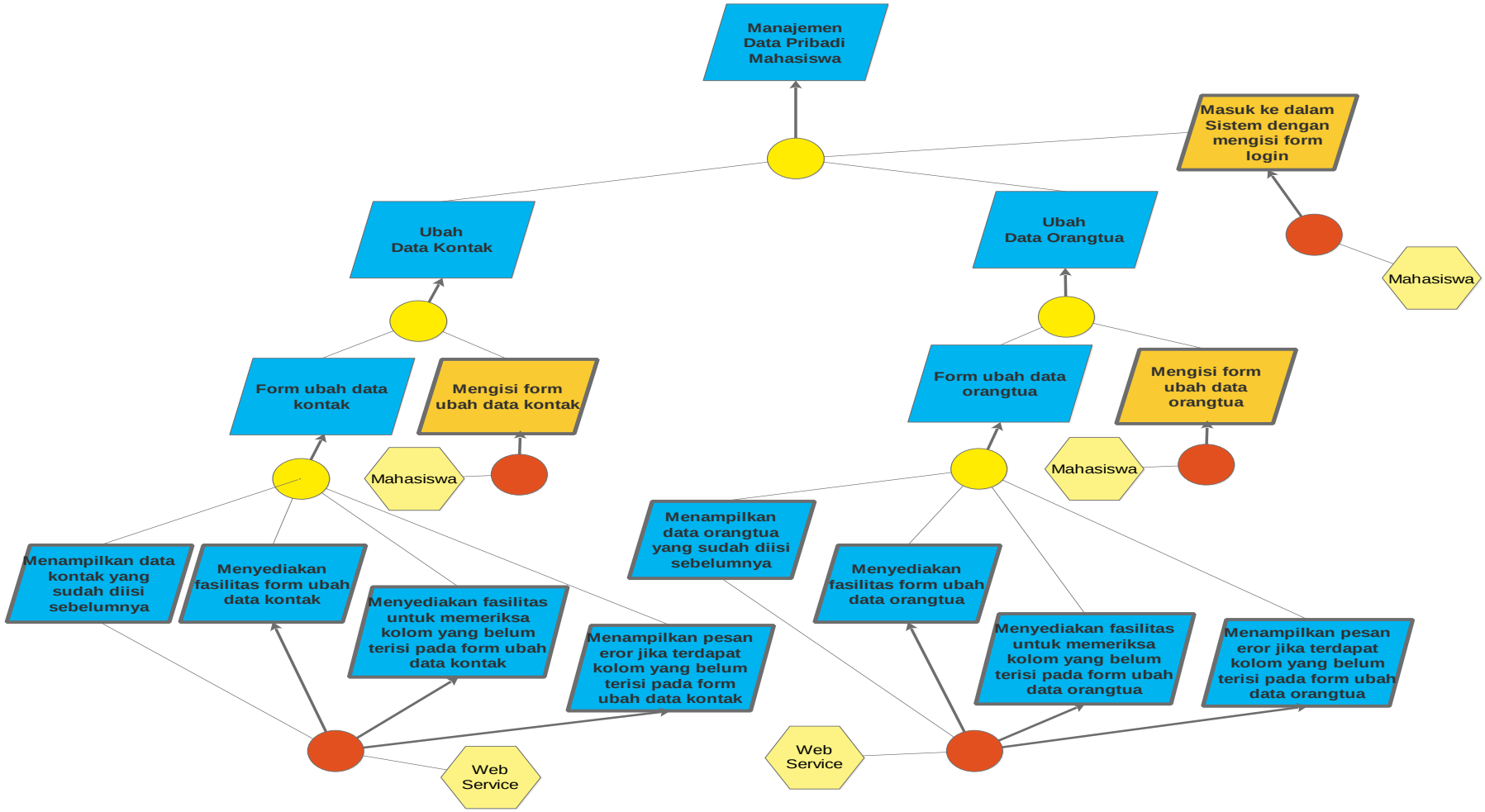
Lampiran 4. *Parralelogram graph* dari *goal* informasi pengunjung



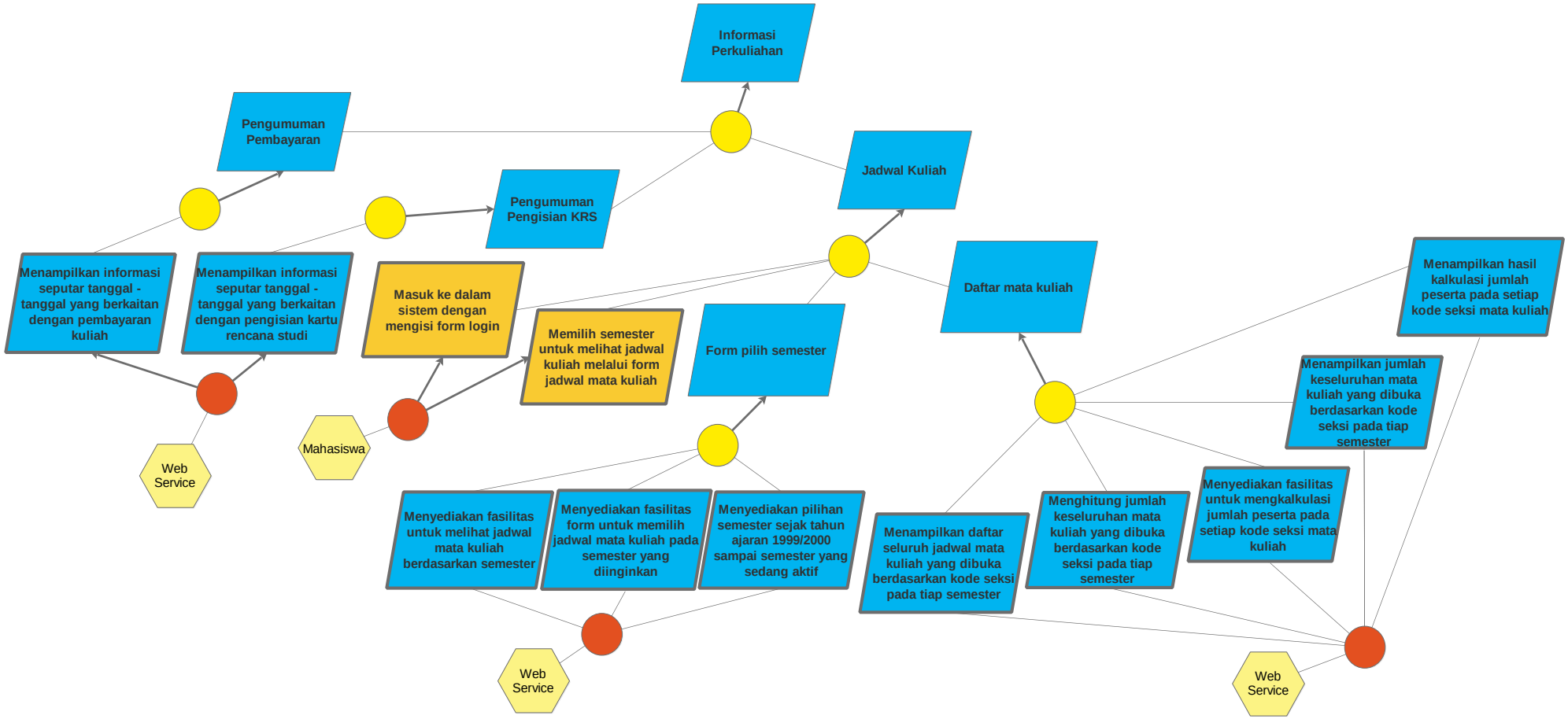
Lampiran 5. *Parralelogram graph* dari *goal* manajemen data pribadi mahasiswa bagian satu



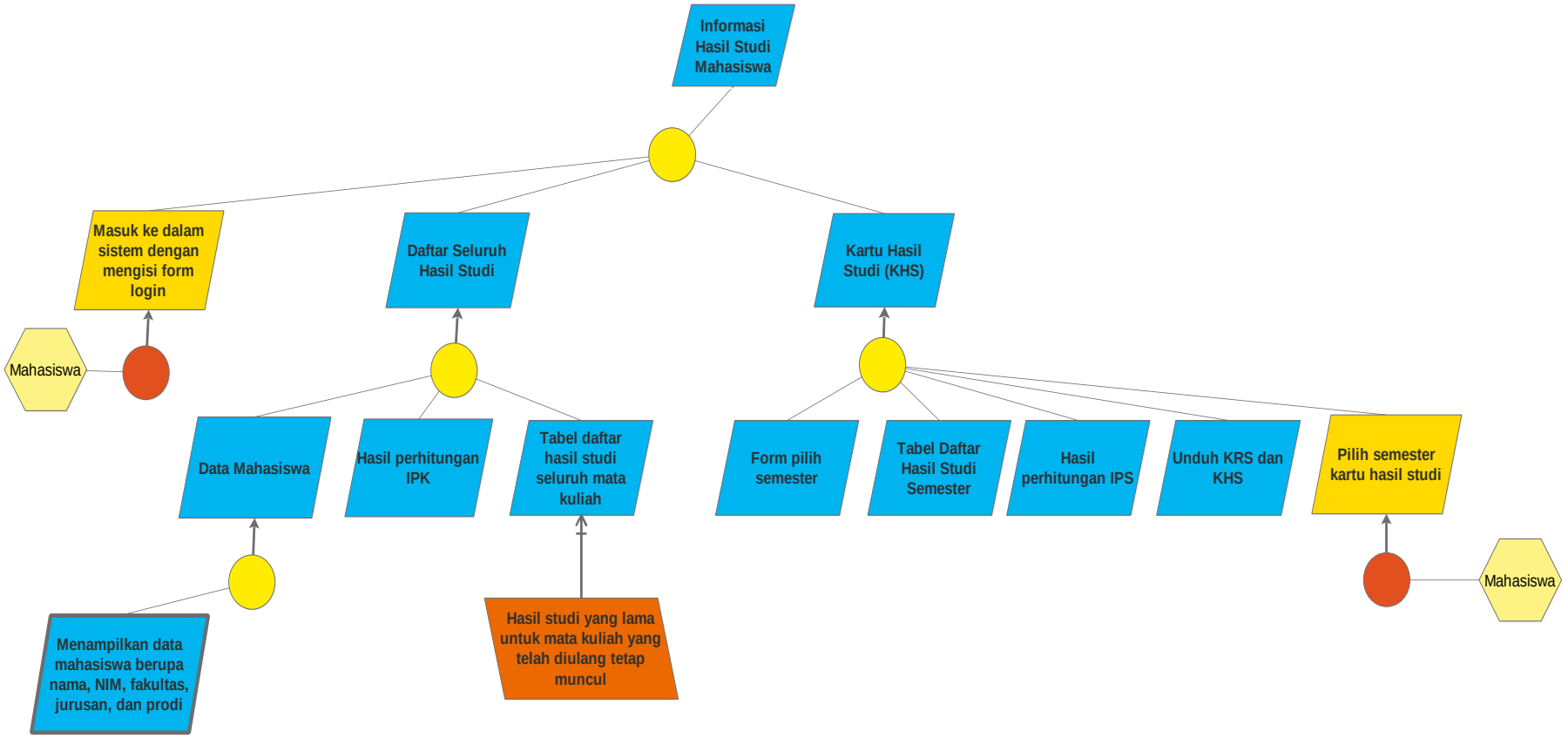
Lampiran 6. *Parralelogram graph* dari *goal* manajemen data pribadi mahasiswa bagian dua



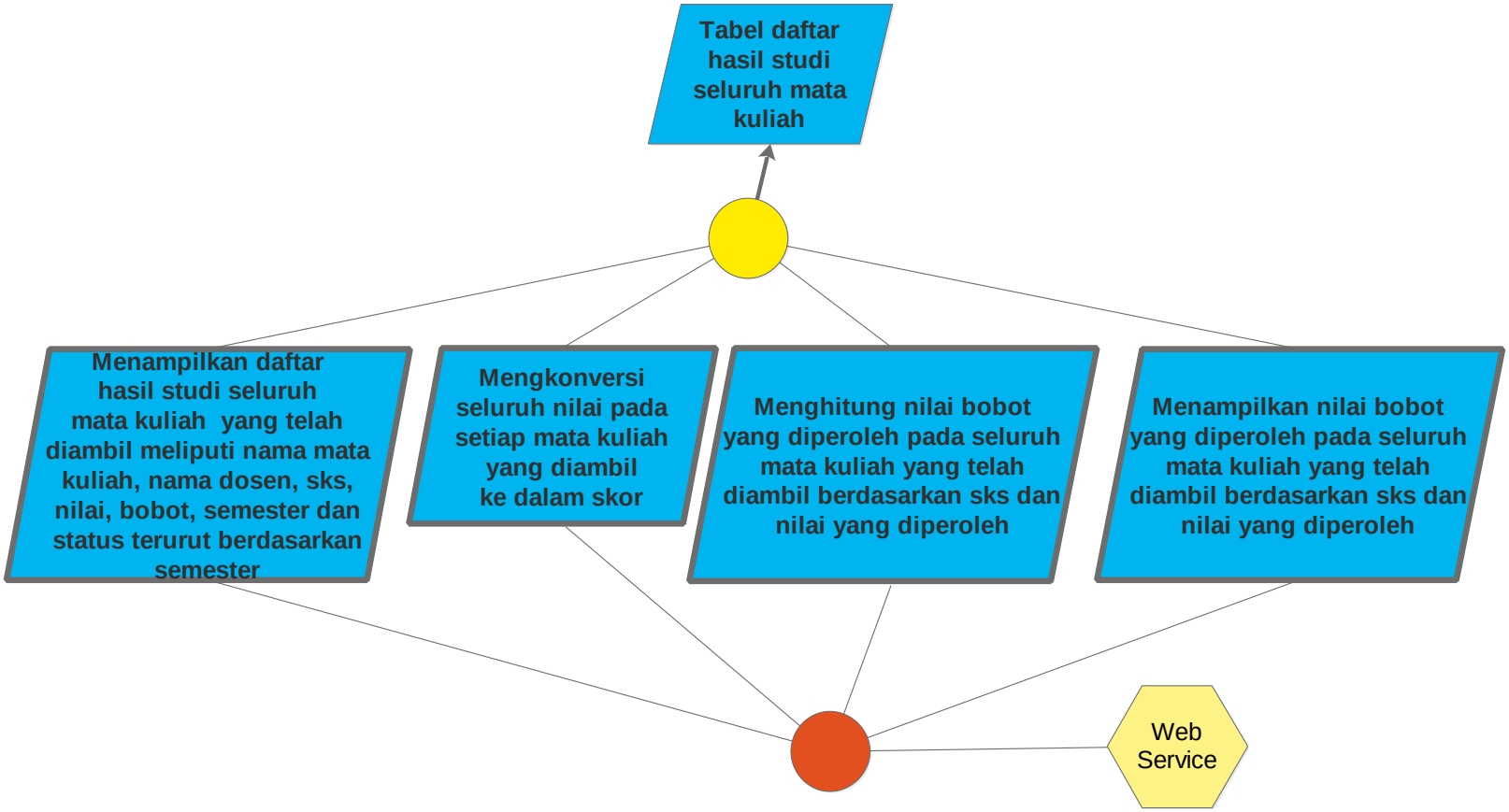
Lampiran 7. *Parralelogram graph* dari *goal* informasi perkuliahan



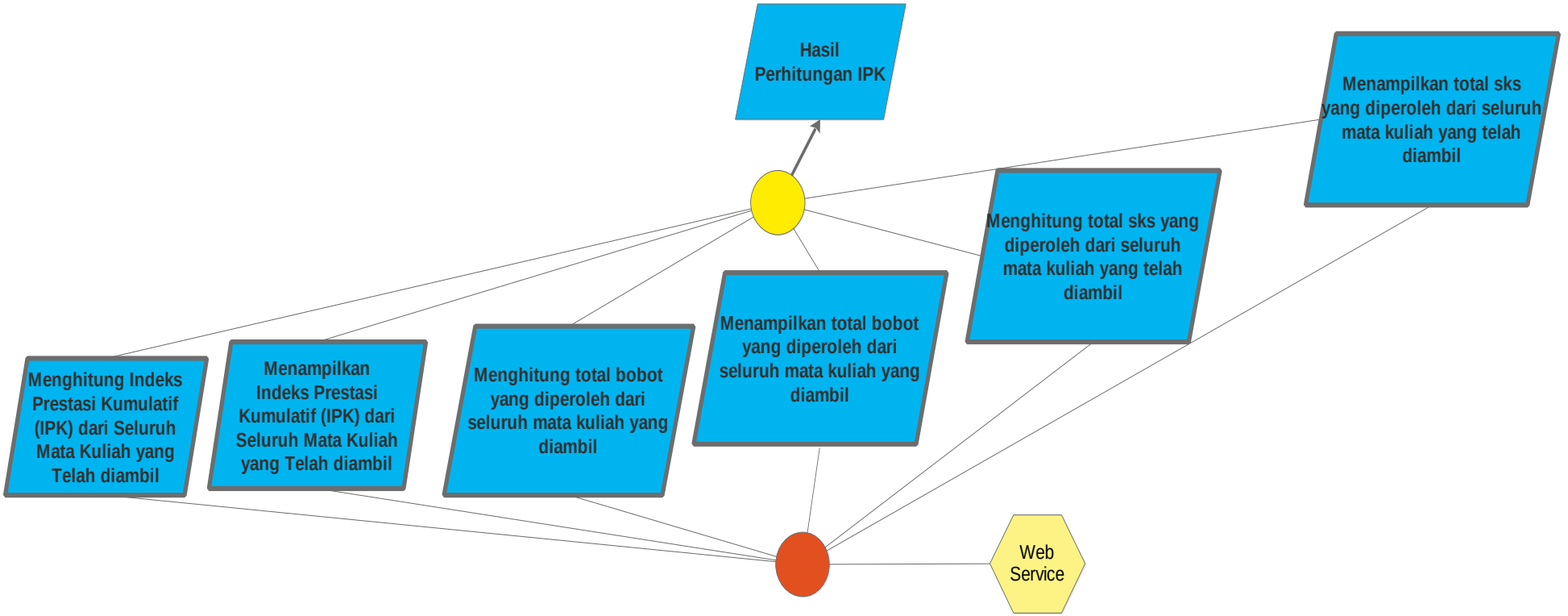
Lampiran 8. *Parralelogram graph* dari *goal* informasi hasil studi mahasiswa bagian satu



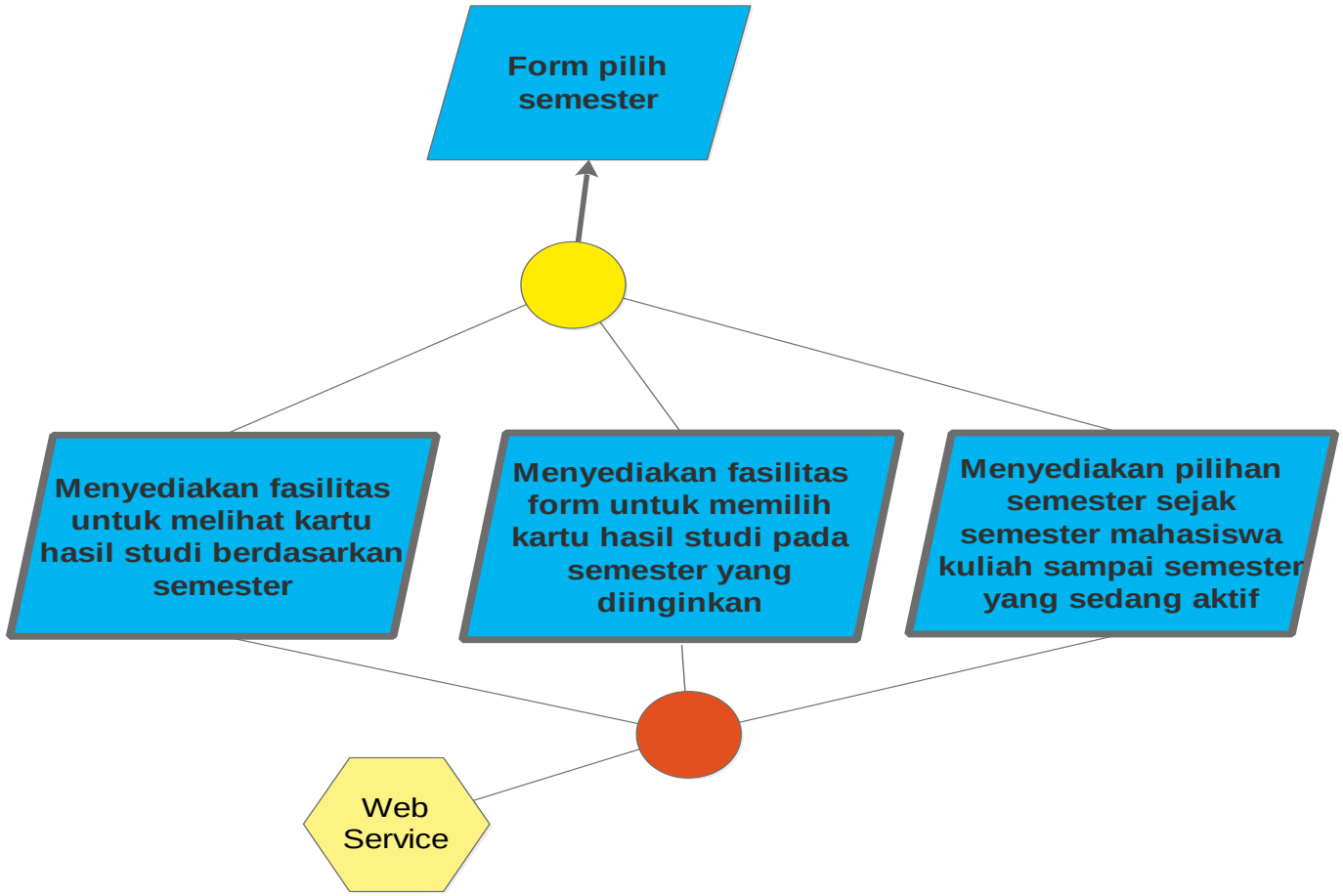
Lampiran 9. *Parralelogram graph* dari *goal* informasi hasil studi mahasiswa bagian dua



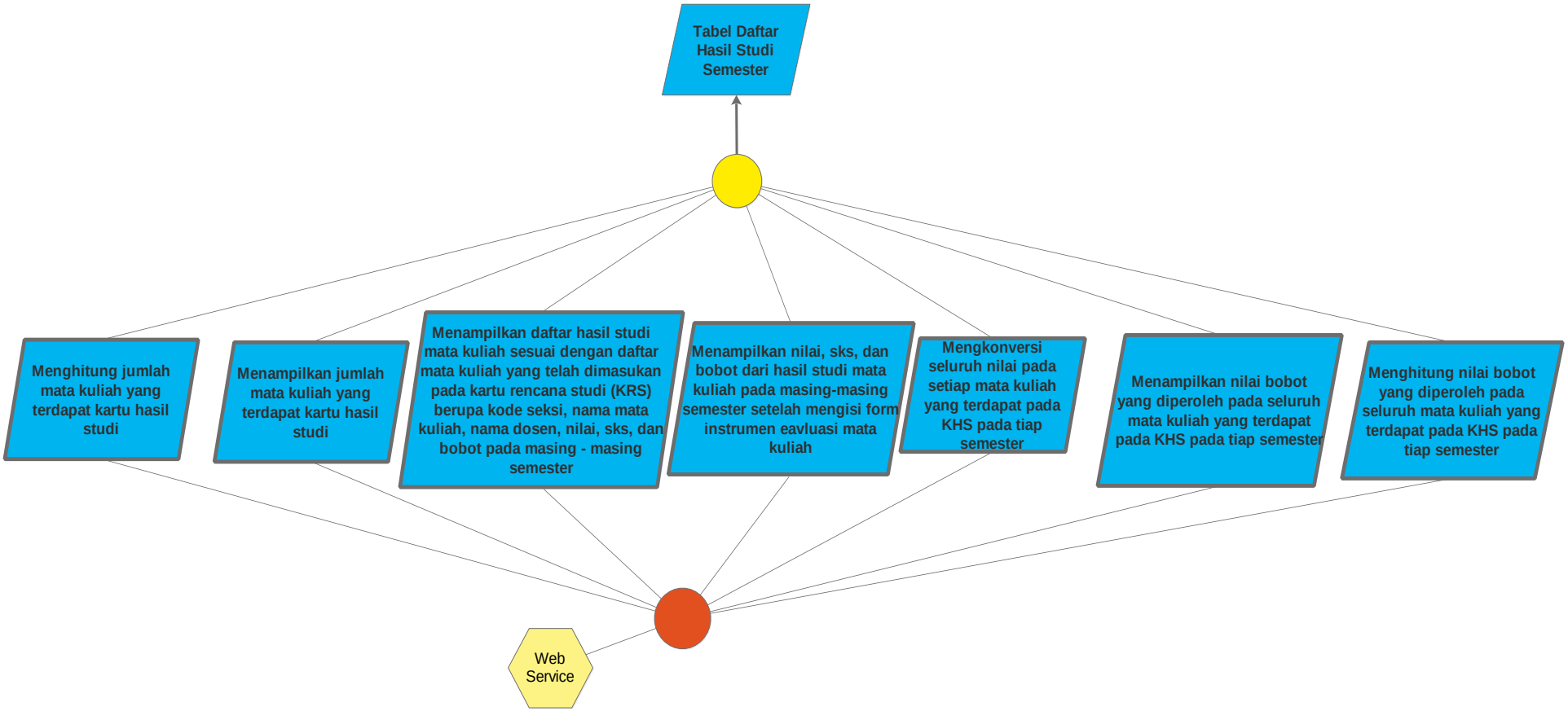
Lampiran 10. *Parralelogram graph* dari *goa*l informasi hasil studi mahasiswa bagian tiga



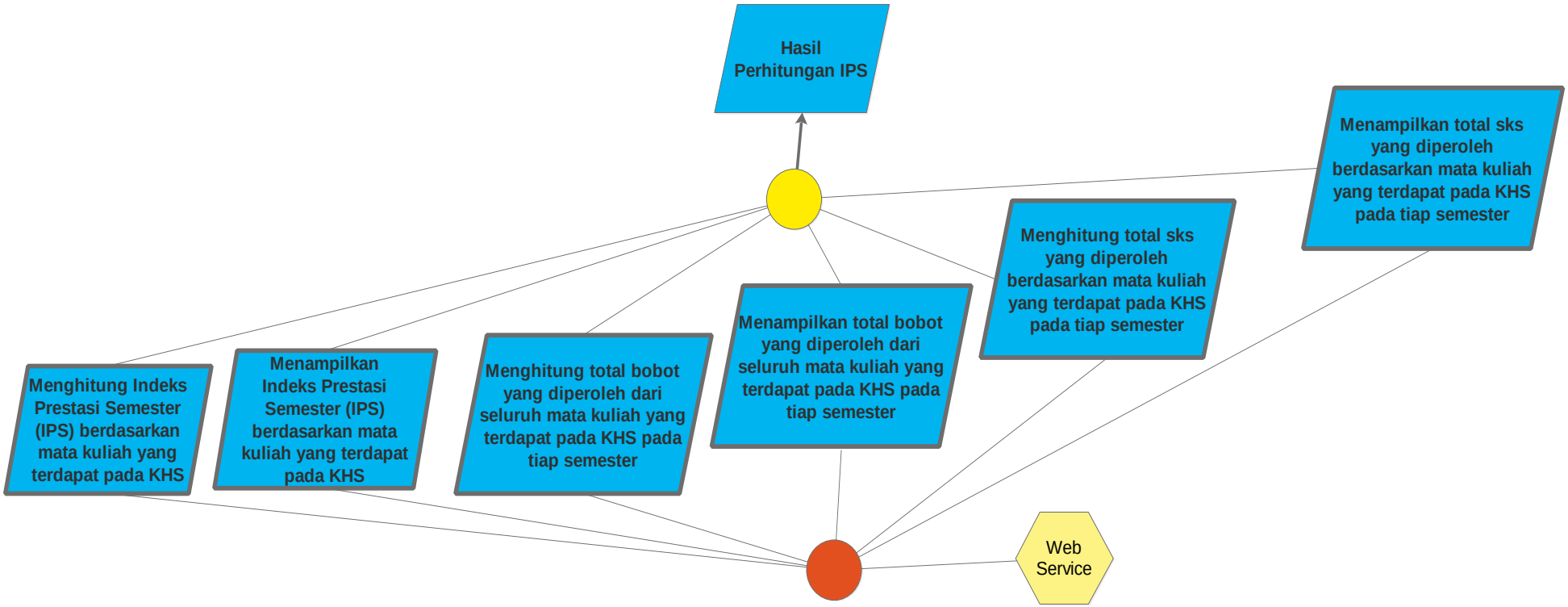
Lampiran 11: *Parralelogram graph* dari *goal* informasi hasil studi mahasiswa bagian empat



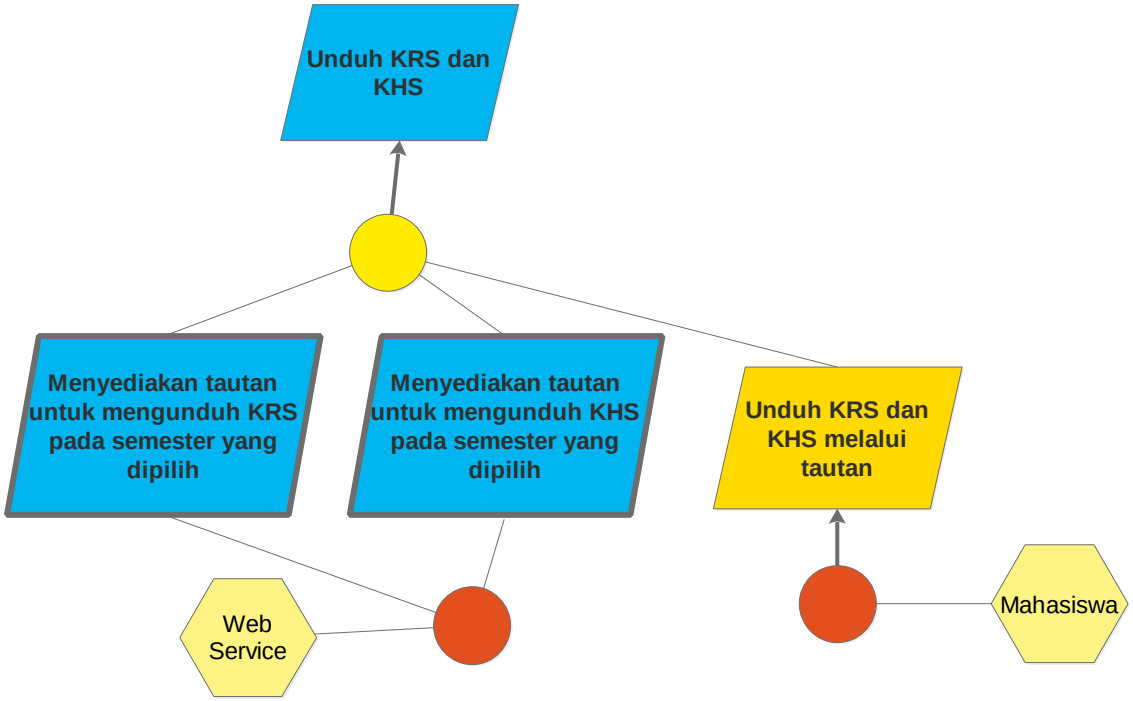
Lampiran 12: *Parralelogram graph* dari *goal* informasi hasil studi mahasiswa bagian lima



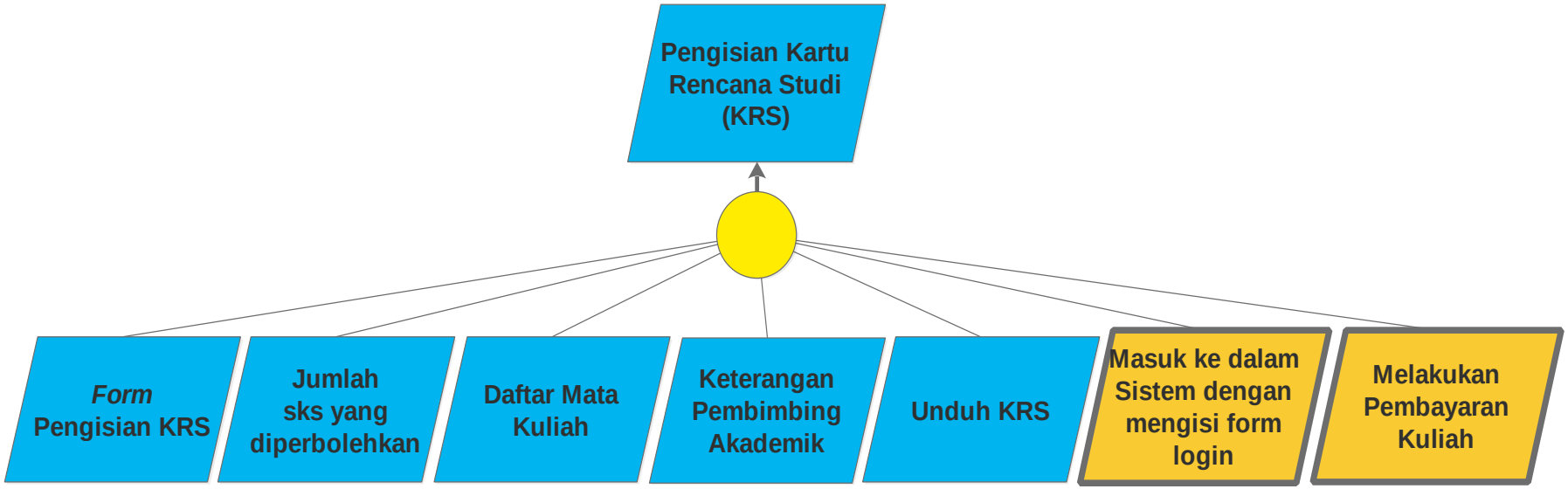
Lampiran 13: *Parralelogram graph* dari *goal* informasi hasil studi mahasiswa bagian enam



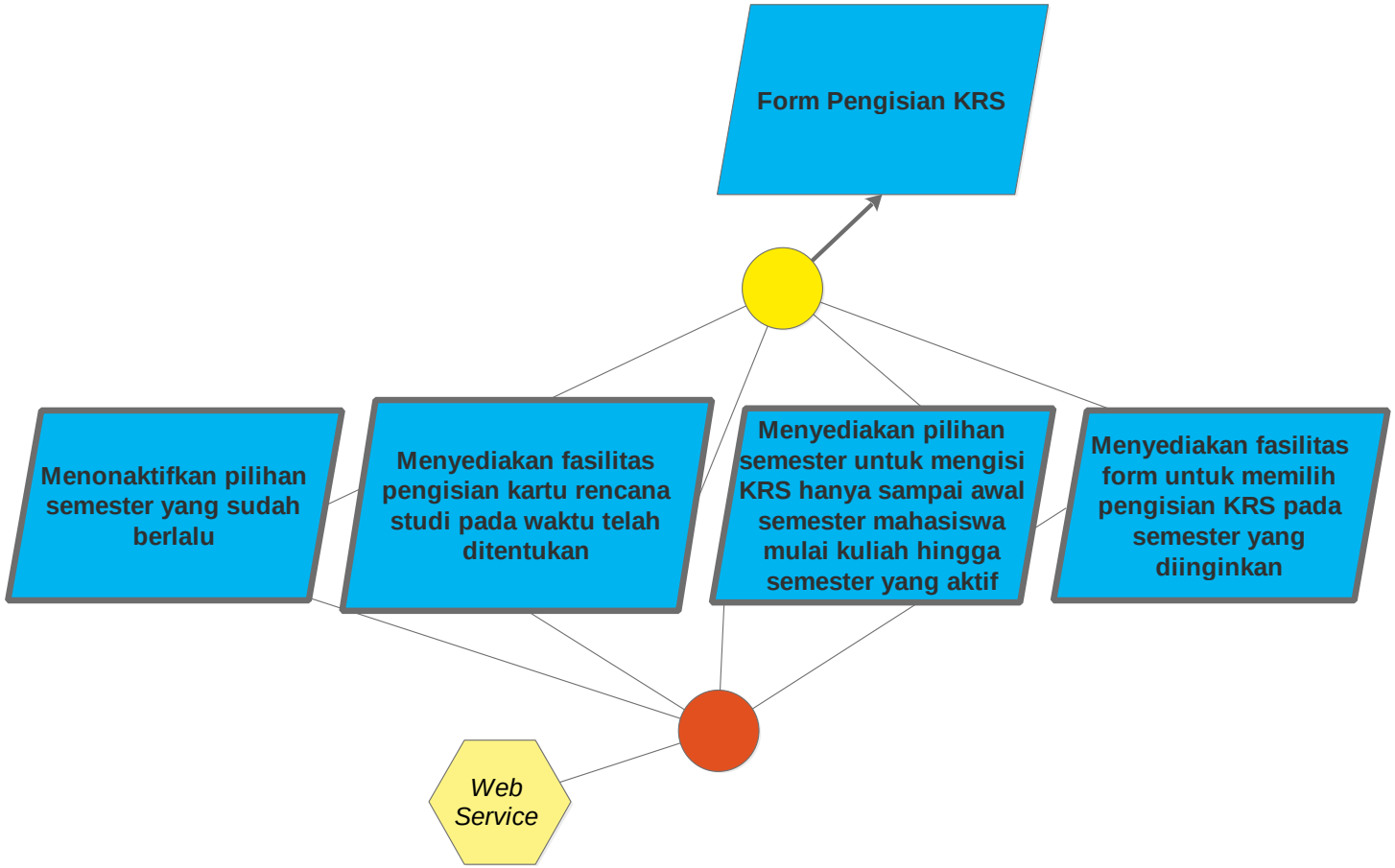
Lampiran 14: *Parralelogram graph* dari *goal* informasi hasil studi mahasiswa bagian tujuh



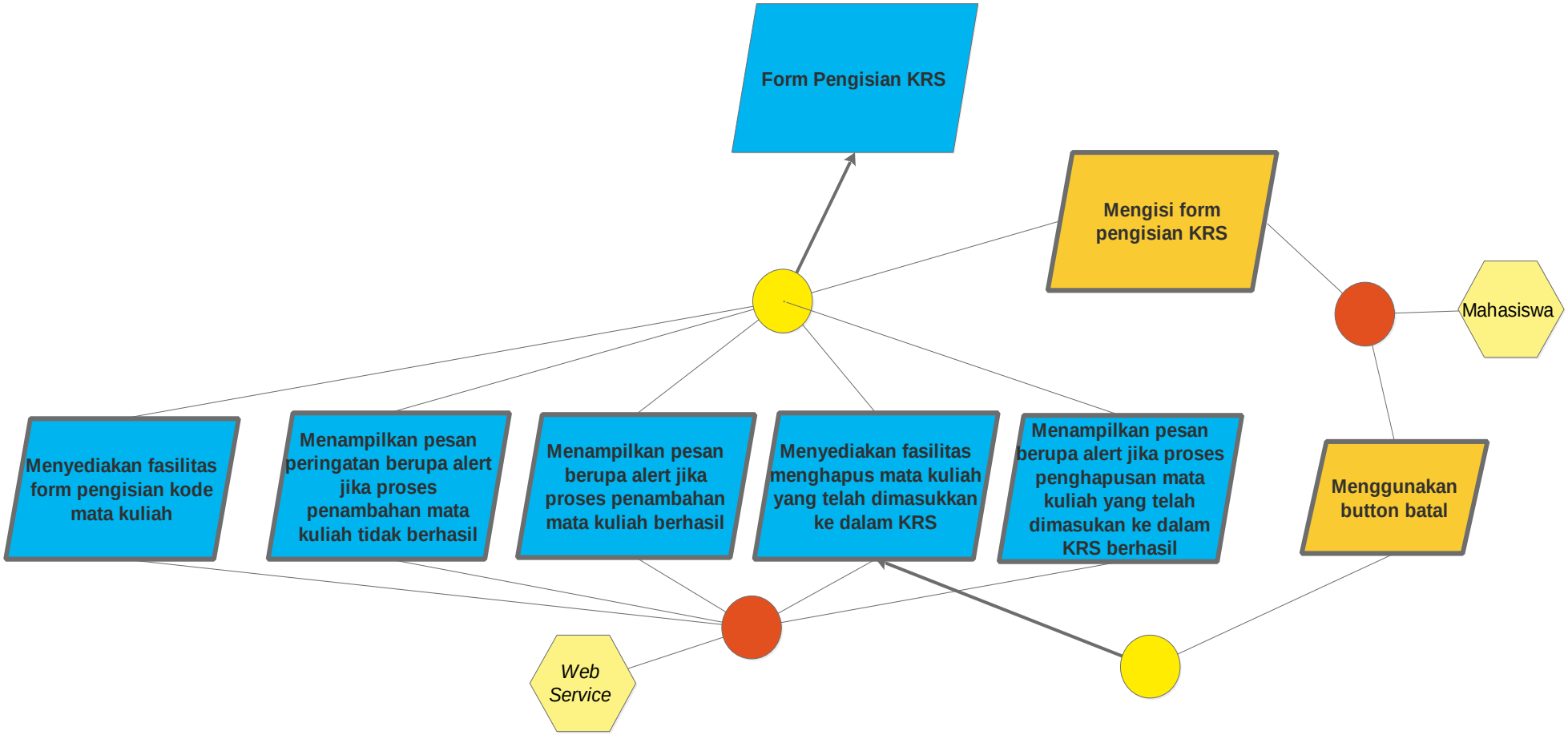
Lampiran 15: *Parralelogram graph* dari *goal* pengisian KRS bagian satu



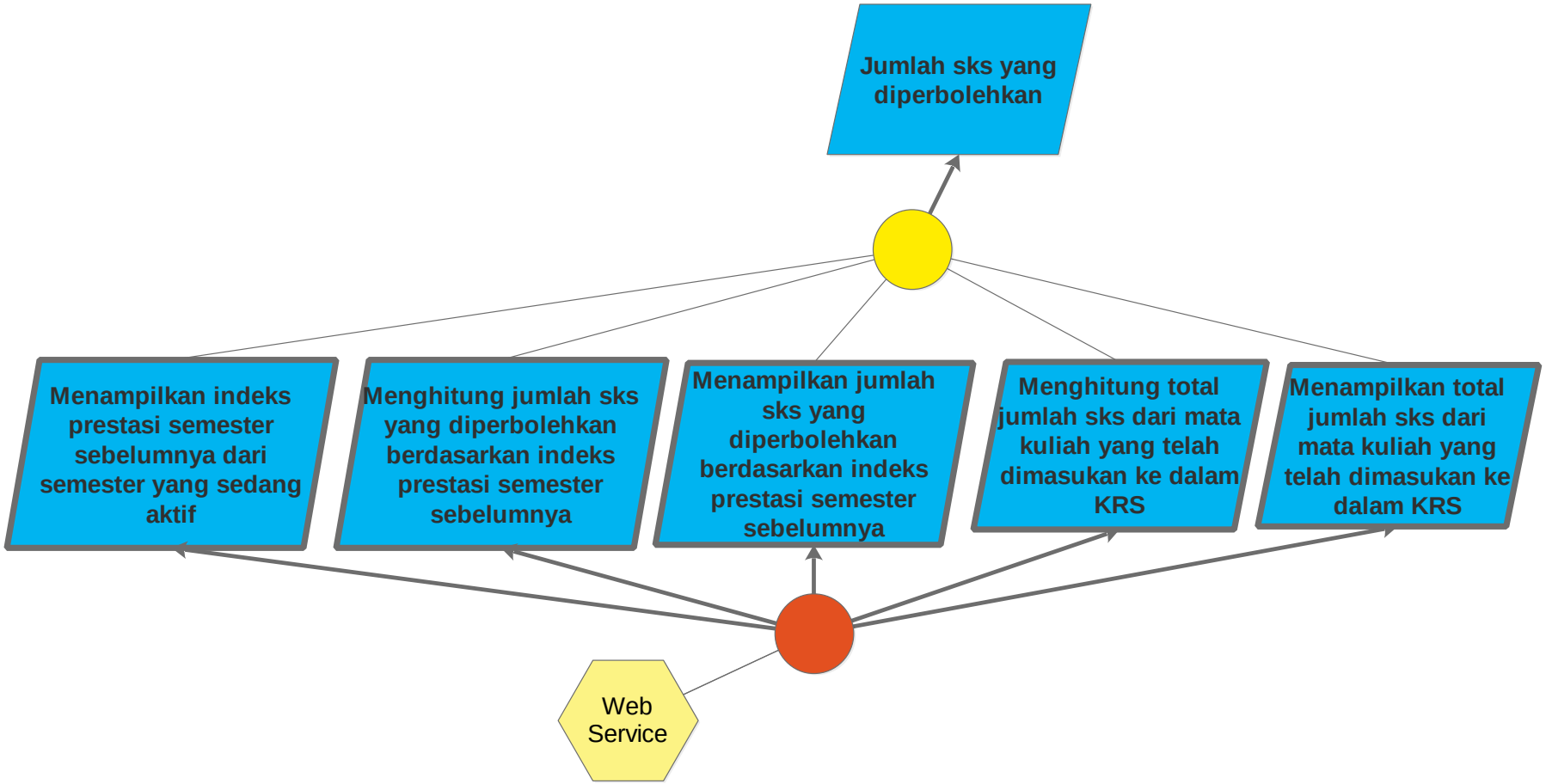
Lampiran 16: *Parralelogram graph* dari *goal* pengisian KRS bagian dua



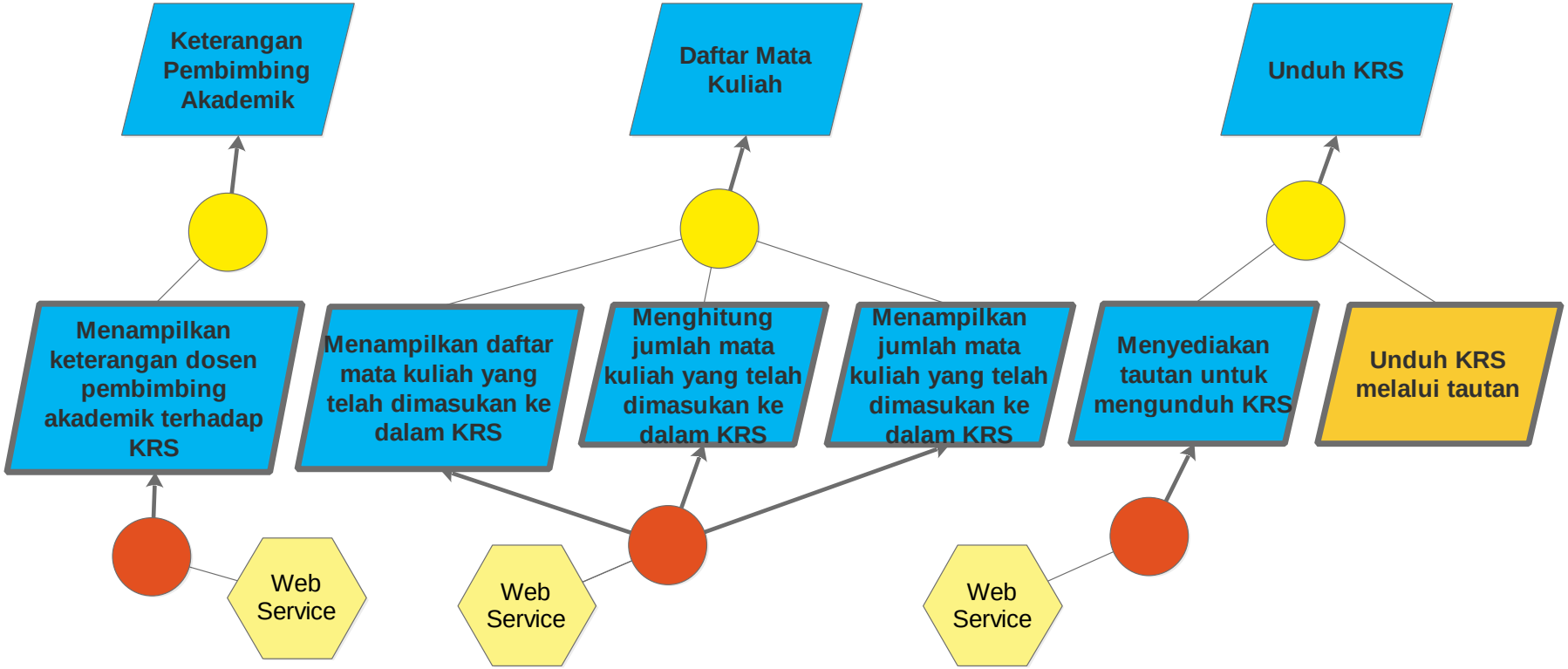
Lampiran 17: *Parralelogram graph* dari *goal* pengisian KRS bagian tiga



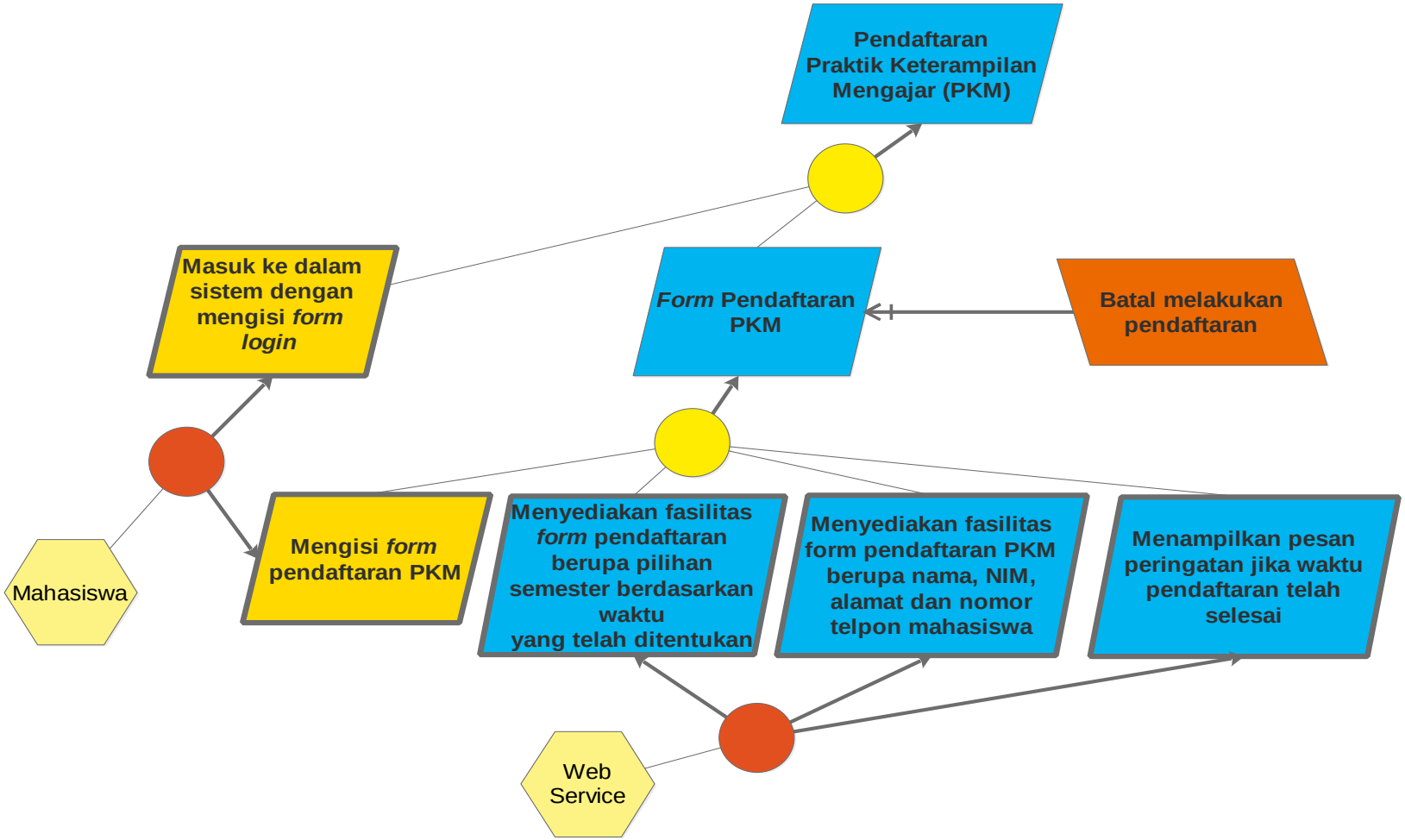
Lampiran 18: *Parralelogram graph* dari *goal* pengisian KRS bagian empat



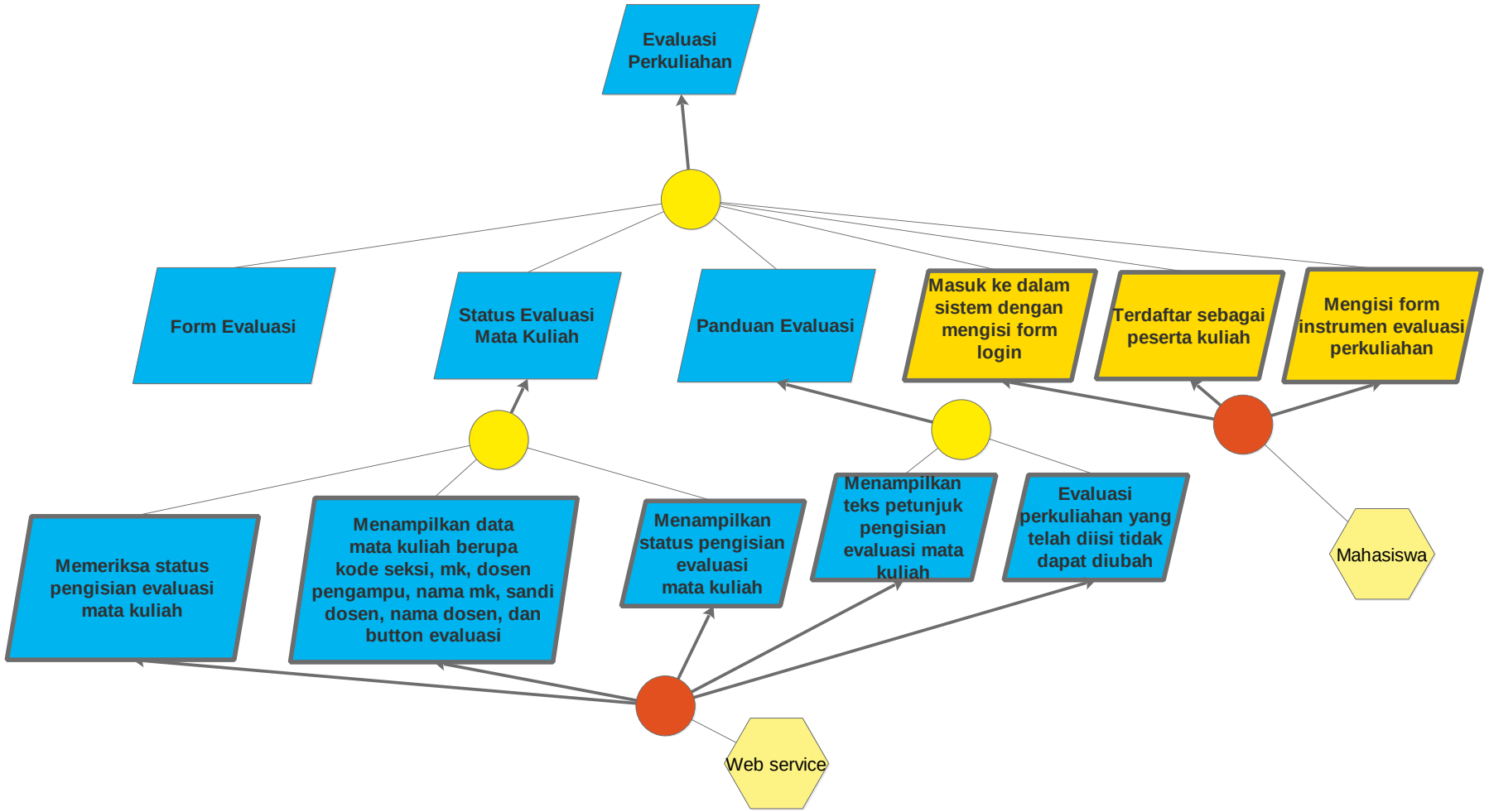
Lampiran 19: *Parralelogram graph* dari *goal* pengisian KRS bagian lima



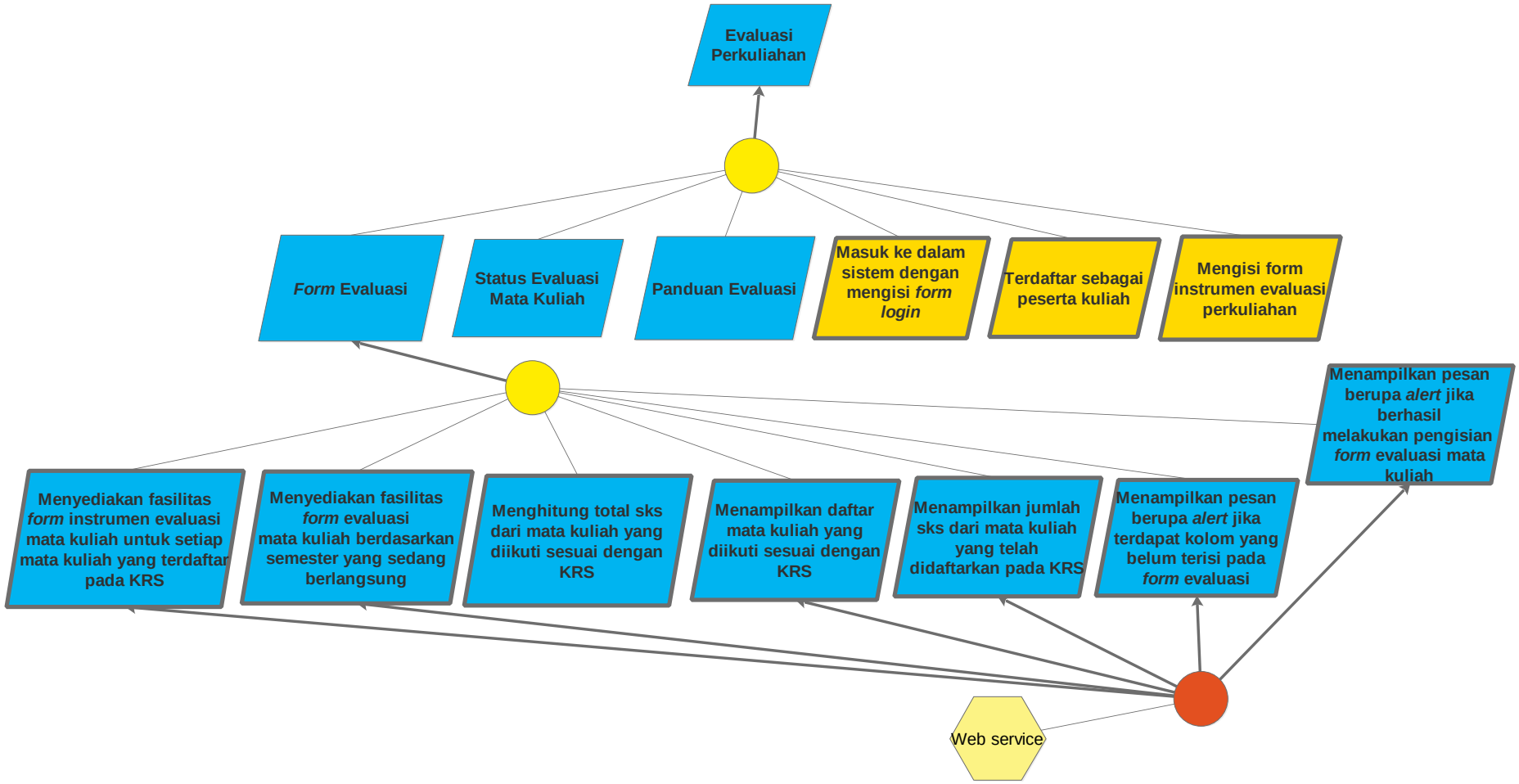
Lampiran 20: *Parralelogram graph* dari *goal* pendaftaran PKM



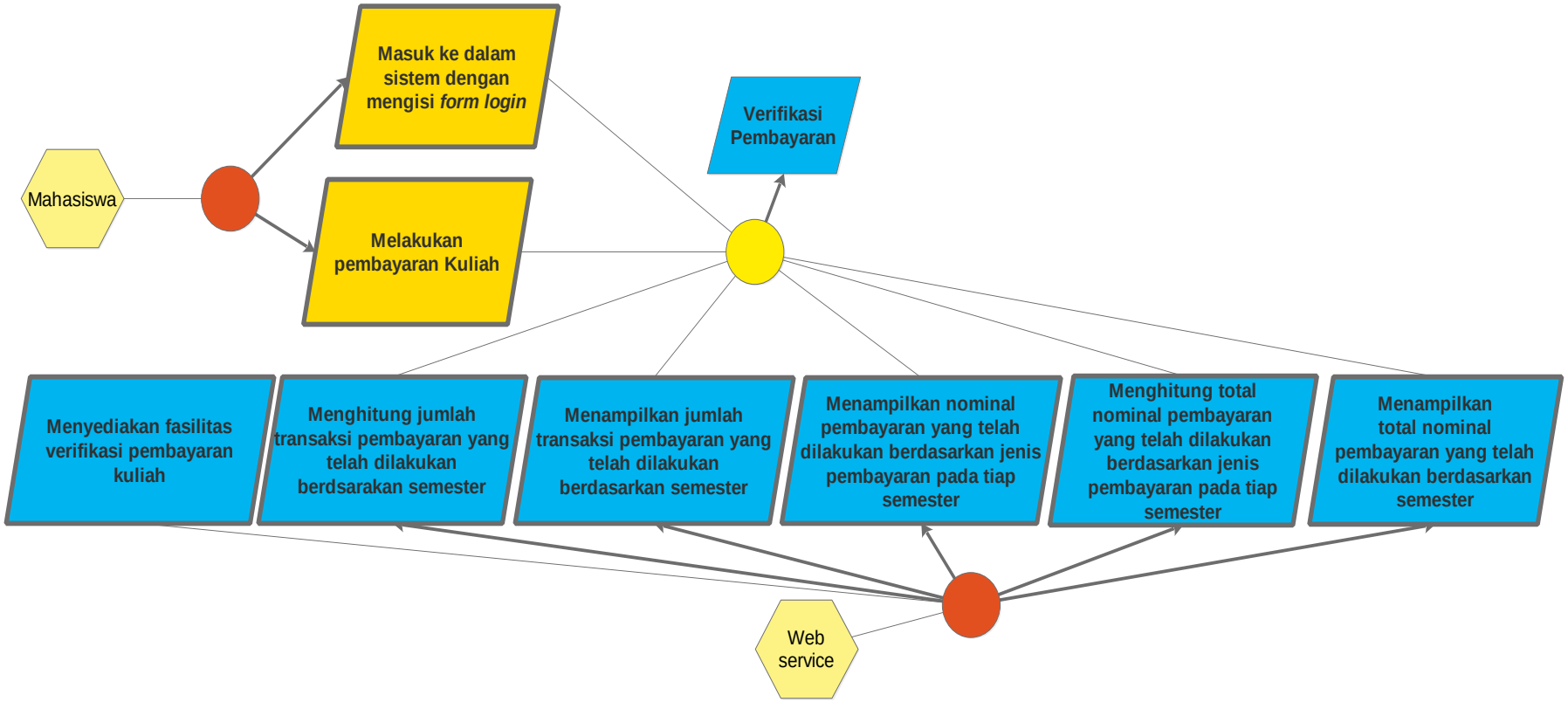
Lampiran 21: *Parralelogram graph* dari *goal* evaluasi perkuliahan bagian satu



Lampiran 22: *Parralelogram graph* dari *goal* evaluasi perkuliahan bagian dua



Lampian 23. *Parralelogram graph* dari *goal* verifikasi pembayaran



Lampiran 26. Jumlah Keterkaitan antar *requirements* Modul Mahasiswa Siakad UNJ

Kode *Requirement*	Jumlah Keterkaitan Antar *Requirement*
RSM001	0
RSM002	1
RSM003	1
RSM004	1
RSM005	1
RSM006	1
RSM007	2
RSM008	2
RSM009	3
RSM010	3
RSM011	2
RSM012	2
RSM013	4
RSM014	4
RSM015	3
RSM016	0
RSM017	0
RSM018	0
RSM019	0
RSM020	0
RSM021	0
RSM022	0
RSM023	0
RSM024	0
RSM025	2
RSM026	2
RSM027	3
RSM028	3
RSM029	0
RSM030	0
RSM031	0
RSM032	0
RSM033	0
RSM034	9
RSM035	4
RSM036	4
RSM037	5
RSM038	6
RSM039	7
RSM040	6
RSM041	7
RSM042	2
RSM043	3
RSM044	5

Kode *Requirement*	Jumlah Keterkaitan Antar *Requirement*
RSM045	4
RSM046	4
RSM047	3
RSM048	5
RSM049	4
RSM050	4
RSM051	3
RSM052	5
RSM053	4
RSM054	4
RSM055	3
RSM056	5
RSM057	4
RSM058	4
RSM059	2
RSM060	7
RSM061	8
RSM062	5
RSM063	3
RSM064	4
RSM065	5
RSM066	6
RSM067	7
RSM068	3
RSM069	4
RSM070	17
RSM071	4
RSM072	4
RSM073	5
RSM074	6
RSM075	7
RSM076	8
RSM077	9
RSM078	7
RSM079	8
RSM080	9
RSM081	6
RSM082	7
RSM083	6
RSM084	5
RSM085	5
RSM086	20
RSM087	3
RSM088	4
RSM089	5
RSM090	6
RSM091	7

Kode *Requirement*	Jumlah Keterkaitan Antar *Requirement*
RSM092	7
RSM093	6
RSM094	5
RSM095	7
RSM096	7
RSM097	7
RSM098	7
RSM099	8
RSM100	8
RSM101	9
RSM102	10
RSM103	6
RSM104	6
RSM105	4
RSM106	3
RSM107	3
RSM108	2
RSM109	3
RSM110	4
RSM111	5
RSM112	5
RSM113	6
RSM114	6
RSM115	4
RSM116	6
RSM117	7
RSM118	5
RSM119	6
RSM120	8
RSM121	3
RSM122	4
RSM123	3
RSM124	4
RSM125	5

v																		
	v																	
	v	v																
	v	v	v															
	v	v						v										
	v	v			v			v										
	v	v			v			v										
	v	v																
	v	v			v			v										
	v	v			v			v	v									
	v	v			v			v										
	v	v			v				v									
	v	v			v			v										
														v	v	v	v	v
													v					
													v	v				
													v					
													v			v		
													v			v	v	

Mengetahui
Developer UPT-TIK

Irfansyah, S.Pd.
19790909 200012 1002

TENTANG PENULIS

Hafiez Arief Raharjo lahir di Jakarta 27 Juli 1995. Penulis merupakan anak ke dua dari dua bersaudara. Sejak lahir hingga saat ini penulis tinggal dan di besarkan di Jakarta Timur.

Penulis telah menempuh pendidikan sejak usia dini, yakni penulis pernah bersekolah di TK Fahmi (2000-2001), kemudian SDN Pulogebang 08 Pagi (2001-2007), pada tingkat menengah di MTs. Negeri 24 Jakarta (2007-2010), serta tingkat atas di SMA Negeri 11 Jakarta (2010-2013), dan terakhir di S1 Pendidikan Teknik Informatika dan Komputer Universitas Negeri Jakarta (2013-2017).

Penulis menyukai pelajaran sains dan komputer sejak sekolah tingkat menengah dan mulai menekuni dunia komputer ketika kuliah s1 di Universitas Negeri Jakarta. Penulis memiliki minat khususnya di bidang pemrograman *website*. Penulis dapat dihubungi di hafiezraharjo@gmail.com atau akun facebook bernama "Hafiez Arief" atau untuk melihat foto hasil jepretan penulis dapat dilihat pada instagram "hafiezar".