PERANCANGAN MODEL *GOAL ORIENTED REQUIREMENTS ENGINEERING* (GORE) UNTUK PROSES *REVERSE ENGINEERING*

FAUZIAH RIZQY
5235128591

Skripsi ini Ditulis untuk Memenuhi Sebagian Persyaratan dalam Memperoleh Gelar Sarjana

PROGRAM STUDI PENDIDIKAN TEKNIK INFORMATIKA DAN KOMPUTER
JURUSAN TEKNIK ELEKTRO
FAKULTAS TEKNIK
UNIVERSITAS NEGERI JAKARTA
2016

PERANCANGAN MODEL *GOAL ORIENTED REQUIREMENTS ENGINEERING* (GORE) UNTUK PROSES *REVERSE ENGINEERING*

FAUZIAH RIZQY

ABSTRAK

Penelitian ini berjudul Perancangan Model *Goal Oriented Requirements Engineering* (GORE) untuk Proses *Reverse Engineering*. Penelitian dilatar belakangi oleh tuntutan dalam dunia *software development* mengenai teknologi *reverse engineering*, model konvensional dinilai kurang efektif dalam melakukan proses *reverse engineering* terutama untuk mengatasi sistem *software* yang lebih kompleks dan juga pendekatan konvensional cenderung lebih menekankan pemodelan *requirements* dalam bentuk *low-level data*. Penelitian ini bertujuan untuk membangun model pendekatan sebagai alat bantu dalam proses *reverse engineering* yaitu GORE dengan metode *Goal-Skill-Preferences* (GSP). Metode penelitian yang digunakan adalah metode kualitatif dengan melakukan *reverse engineering* pada sistem aplikasi siap pakai untuk merancang model *reverse engineering*. Penelitian dilakukan di laboraturium komputer Jurusan Teknik Elektro Fakultas Teknik Universitas Negeri Jakarta dari bulan Agustus 2015 sampai Desember 2015. Adapun hasil penelitian merupakan model GORE dengan metode GSP yang direpresentasikan dalam bentuk diagram yang menjelaskan langkah-langkah untuk melakukan *reverse*. Berikut langkah-langkah model GORE dengan metode GSP untuk proses *reverse engineering*, yaitu: (1) Mengambil *main goal* dari tampilan sistem aplikasi; (2) Merepresentasikan *goal* dalam bentuk *goal graph*; (3) Mengembangkan *goal* menjadi *subgoal*; (4) Membuat *set of alternatives* dari *subgoal*; dan (5) Mengambil *requirements* dari setiap *alternatives*. Setelah dibandingkan dengan hasil *reverse* konvensional, maka dapat dibuktikan bahwa model GORE dengan metode GSP dapat digunakan sebagai alat bantu untuk melakukan *reverse engineering*.

Kata kunci: GORE, GSP, *reverse engineering*, dan *requirements*.

DESIGN OF GOAL ORIENTED REQUIREMENTS ENGINEERING (GORE) MODEL FOR REVERSE ENGINEERING PROCESS

FAUZIAH RIZQY

ABSTRACT

This research entitled Design Goal Oriented Requirements Engineering (GORE) Model for Reverse Engineering Process. The research is motivated by the demands of the world of software development about reverse engineering technology, the conventional model is considered less effictive in the process of reverse engineering, especially to cope with more complex software systems and also a conventional approach is more tend to more emphasis on requirements modeling in the form of low-level data. This research aims to develop an approach model as a tool in the process of reverse engineering that is GORE with Goal-Skill-Preferences (GSP) method. The research method that use is qualitative method by doing reverse engineering on the already use application for designing the reverse engineering model. The research was conducted in a computer lab Department of Electrical Engineering, Faculty of Engineering, State University of Jakarta from August 2015 till December 2015. The research results are GORE model with GSP method which represented in the form of a diagram that explains steps to do the reverse. Here are the steps of GORE model with GSP method for reverse engineering process, namely: (1) Take the main goal from the system application interface; (2) Represents a goal in the form of a goal graph; (3) Develop goal into subgoal; (4) Makes set of alternatives from the subgoal; and (5) Take the requirements from each of alternatives. After compared with results of the conventional reverse, it can be proved that the GORE with GSP method model can be used as a tool for reverse engineering.

Keywords: GORE, GSP, reverse engineering, and requirements.

HALAMAN PENGESAHAN

NAMA DOSEN	TANDA TANGAN	TANGGAL
Widodo, M.Kom (Dosen Pembimbing I)	.....	15/1-'16
Hamidillah Ajie, S.Si., M.T (Dosen Pembimbing II)	.....	18-01-2016

PENGESAHAN PANITIA UJIAN SKRIPSI

NAMA DOSEN	TANDA TANGAN	TANGGAL
Prasetyo Wibowo Yunanto, M.Eng (Ketua Penguji)	.....	15-1-2016
M. Ficky Duskarnaen, M.Sc (Sekretaris Penguji)	.....	12-01-2016
Bambang Prasetya Adhi, M.Kom (Dosen Ahli)	.....	12-01-2016

Tanggal Lulus: 05-01-2016

HALAMAN PERNYATAAN

Dengan ini saya menyatakan:

1. Karya tulis skripsi saya ini adalah asli dan belum pernah diajukan untuk mendapatkan gelar akademik sarjana, baik di Universitas Negeri Jakarta maupun di perguruan tinggi lain.
2. Skripsi ini adalah murni gagasan, rumusan, dan penelitian saya sendiri dengan arahan dosen pembimbing.
3. Dalam skripsi ini tidak terdapat karya atau pendapat yang telah ditulis atau dipublikasikan orang lain, kecuali secara tertulis dengan jelas dicantumkan sebagai acuan dalam naskah dengan disebutkan nama pengarang dan dicantumkan dalam daftar pustaka.
4. Pernyataan ini saya buat dengan sesungguhnya dan apabila di kemudian hari terdapat penyimpangan dan ketidakbenaran dalam pernyataan ini, maka saya akan 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, 5 Januari 2016

Yang membuat pernyataan

Fauziah Rizqy

5235128591

KATA PENGANTAR

Puji syukur penulis panjatkan kehadirat Allah SWT atas karunia dan rahmat-Nya, sehingga penulis dapat menyelesaikan skripsi dengan judul **"Perancangan Model *Goal Oriented Requirements Engineering* (GORE) untuk Proses *Reverse Engineering*"** ini tepat pada waktunya, sebagai salah satu syarat untuk meraih gelar Sarjana Pendidikan Teknik Informatika dan Komputer pada Jurusan Teknik Elektro, Fakultas Teknik, Universitas Negeri Jakarta.

Penulis menyadari keterbatasan kemampuan dan pengetahuan dalam penyusunan skripsi ini. Oleh karena itu, tidak sedikit kesulitan dan kendala yang dialami penulis selama proses penyusunan skripsi berlangsung. Namun berkat adanya bimbingan, dorongan, saran-saran, dan bantuan dari berbagai pihak, akhirnya skripsi ini dapat diselesaikan. Sehubungan dengan hal tersebut, pada kesempatan ini penulis ingin menyampaikan ucapan terima kasih kepada:

1. Ibu Dr. Yuliatri Sastrawijaya, M.Pd, selaku Ketua Program Studi Pendidikan Teknik Informatika dan Komputer, Fakultas Teknik, Universitas Negeri Jakarta.
2. Bapak Widodo, M.Kom dan Bapak Hamidillah Ajie, S.Si., M.T, selaku dosen pembimbing yang penuh kesabaran memberi penguatan, serta dukungan sehingga skripsi ini dapat terselesaikan.
3. Bapak M. Ficky Duskarnaen, M.Sc, selaku penasehat akademik.
4. Pihak Dinas Pendidikan Provinsi Kepulauan Riau yang sudah memberikan kesempatan berkuliah dengan beasiswa penuh selama tiga setengah tahun.
5. Bapak Kadarisman dan Ibu Asnah, selaku kedua orang tua yang tiada henti menghantarkan doa untuk anak perempuan mereka satu-satunya yang berjuang di rantauan. Perjuangan ini membuktikan bahwa doa orang tua adalah motivasi terhebat yang pernah ada. Semoga mereka senantiasa diberikan kesehatan dan kebahagiaan serta selalu dalam lindungan-Nya.

6. Kakak (Firdaus Bagus Prayogi) dan adik-adik (Muhammad Faiz Rhamdany, Fahmi Arisandy) yang selalu mendukung dan menjadi motivasi dalam perjuangan hidup.
7. Sahabat yang selalu saling menjaga satu sama lain, Anne Lestari, Dinda Nurrahma, Nida Aulia Hasanah, dan Rahma Qonita. Semua suka, duka, susah, dan senang yang pernah dilewati semasa kuliah akan menjadi kenangan indah. Dan juga teman-teman seperjuangan Program Studi Pendidikan Teknik Informatika dan Komputer tahun angkatan 2012 yang memberikan semangat dan dukungan dalam penulisan skripsi ini.
8. Seluruh pihak yang telah mendukung dan tidak bisa disebutkan satu persatu demi terselesaikannya skripsi ini dengan baik dan lancar. Semoga Yang Maha Kuasa membalas semua kebaikan yang telah diberikan.

Penulis menyadari bahwa skripsi ini jauh dari kata sempurna. Oleh karena itu, penulis mohon maaf apabila terdapat kekurangan dan kesalahan. Adapun demi pengembangan ilmu pengetahuan dan perbaikan kesalahan di kemudian hari, penulis mengharapkan kritik dan saran yang bermanfaat dan bersifat membangun. Semoga skripsi ini dapat diterima dan bermanfaat bagi penulis khususnya dan bagi para pembaca pada umumnya.

Penulis

Fauziah Rizqy
5235128591

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I

PENDAHULUAN

1.1. Latar Belakang Masalah

Kemajuan teknologi rekayasa perangkat lunak menuntut perkembangan dan perbaikan sistem perangkat lunak dari waktu ke waktu. Pesatnya tingkat persaingan berbagai instansi dan organisasi membuat kepuasan *user* merupakan hal mutlak yang harus dipenuhi dalam membangun sistem perangkat lunak. Hal ini membuat berbagai model pendekatan digunakan untuk memudahkan proses *reverse engineering* pada sebuah sistem perangkat lunak yang siap pakai untuk memperoleh inovasi dan sebagai bentuk evaluasi dari sistem perangkat lunak tersebut. Keberhasilan suatu sistem perangkat lunak ditentukan dari seberapa besar sebuah sistem dalam memenuhi atau sesuai dengan tujuan pembuatannya. *Requirements Engineering* adalah proses menemukan tujuan sistem perangkat lunak dengan mengidentifikasi *stakeholder* (orang atau organisasi yang akan dipengaruhi oleh sistem dan yang memiliki pengaruh langsung atau tidak langsung pada *requirements system*), dokumentasi dilakukan berdasarkan persetujuan analisis, komunikasi, dan implementasi selanjutnya (Kotonya dan Sommerville. 1998, diacu dalam Lapouchnian, 2005: 1). Proses menemukan *requirements* pada sistem perangkat lunak yang sudah siap pakai dapat dilakukan dengan melakukan *reverse engineering*. Menurut Chikofsky (1993) diacu dalam Santosa (1994: 12) 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.

Kelemahan utama pada proses *reverse engineering* perangkat lunak adalah terdapat batas-batas secara praktisi untuk sejauh mana sebuah sistem dapat diperbaiki melalui *reverse engineering* (Singhal dan Gandhi, 2014: 3). Hal ini mengakibatkan munculnya banyak cara untuk memaksimalkan hasil dari proses *reverse engineering* tersebut, salah satunya yaitu dengan membuat model pendekatan seefektif mungkin.

Dalam bidang pengembangan perangkat lunak, inovasi yang membangun sangat diperlukan demi menghasilkan perangkat lunak yang sesuai dengan tujuan pembuatannya. Namun, dalam realisasinya tidak semua model pendekatan mampu me-*reverse* sebuah sistem perangkat lunak sesuai dengan tujuan utamanya yaitu untuk memenuhi *user requirements* dan *system requirements*. Sistem perangkat lunak memiliki tingkat kompleksitas yang berbeda-beda. Suatu sistem dikatakan kompleks apabila memiliki tujuan pembuatan sistem yang bermacam-macam. Dengan kata lain, makin banyak fungsi yang dimiliki oleh sistem, maka sistem tersebut dikategorikan dalam kategori sistem yang kompleks. Dalam dunia rekayasa perangkat lunak, sistem yang lebih kompleks akan kurang efektif apabila dianalisis dengan model pendekatan konvensional. Karena model pendekatan konvensional menghasilkan *requirements* dengan cara langsung mengambil dari tampilan tanpa adanya konsep yang terstruktur, sehingga *requirements* yang dihasilkan mengalami *missing requirements* atau terjadi pengulangan *requirements* yang sama serta *requirements* yang dihasilkan tidak rinci. Selain itu, model pendekatan konvensional cenderung lebih menekankan pemodelan

requirements dalam bentuk *low-level data*, operasi, dan lainnya yang lebih banyak dipahami oleh *programmer* dan *developer internal*, sedangkan *stakeholder*, *user*, dan *customer* cenderung kurang peduli dengan pemodelan dalam bentuk *low-level data*. Pada *requirements engineering*, orientasi *goal* dan *actor* dinilai lebih menjanjikan dibanding dengan model pendekatan lainnya, hal ini dikarenakan model pendekatan melibatkan tujuan pembuatan sistem dan kebutuhan *user* dalam sistem perangkat lunak tersebut. Selain itu, model ini juga dinilai lebih presisi, karena *requirements* yang dihasilkan akan jelas mengarah pada suatu tujuan pembuatan sistem. Pada dasarnya GORE merupakan model pendekatan yang umum digunakan pada proses *forward engineering*. Akan tetapi, berpatokan pada beberapa keunggulan tersebut, GORE dijadikan sebagai alat bantu dalam proses *reverse engineering*. Oleh karena itu, skripsi ini akan membahas perancangan suatu model pendekatan *reverse engineering* menggunakan *Goal Oriented Requirements Engineering* (GORE) dengan metode *Goals-Skills-Preferences* (GSP) yang diasumsikan mampu me-*reverse* sebuah sistem perangkat lunak yang siap pakai dan menghasilkan *requirements* dengan benar.

1.2. Identifikasi Masalah

Dari latar belakang yang telah diuraikan di atas, maka dapat diambil beberapa permasalahan, diantaranya:

1. Model pendekatan yang konvensional kurang efektif dalam melakukan *reverse engineering.*
2. Sistem perangkat lunak yang kompleks tidak efektif apabila dianalisis dengan model pendekatan konvensional.

3. *Reverse engineering* dengan pendekatan konvensional tidak menggunakan konsep yang terstruktur sehingga *requirements* yang dihasilkan mengalami *missing requirements* atau terjadi pengulangan *requirements* yang sama serta *requirements* yang dihasilkan tidak rinci.
4. Pendekatan konvensional lebih menekankan pemodelan *requirements* dalam bentuk *low-level data*, operasi, dan lainnya yang lebih banyak dipahami oleh *programmer* dan *developer internal*, sedangkan *stakeholder*, *user*, dan *customer* cenderung kurang peduli dengan pemodelan dalam bentuk *low-level data*.

1.3. Batasan Masalah

Pembatasan masalah pada skripsi ini, yaitu:

1. Model dibangun untuk mengetahui bagaimana GORE melakukan *Reverse Engineering* dengan menggunakan metode GSP.
2. Sistem perangkat lunak yang akan dijadikan sampel merupakan sistem perangkat lunak yang umum digunakan dan memiliki tingkat kompleksitas yang menengah.
3. *Requirements* yang diperlukan pada hasil *reverse engineering* adalah *functional requirements*.
4. Sistem perangkat lunak yang digunakan untuk menemukan model GORE merupakan sistem manajemen Penmaba UNJ 2015 Modul Admin.

1.4. Perumusan Masalah

Permasalahan yang akan dibahas dalam skripsi ini adalah Bagaimana merancang model GORE dengan menggunakan metode GSP untuk melakukan proses *Reverse Engineering*?

1.5. Tujuan Penelitian

Tujuan dari penelitian ini adalah untuk merancang model GORE dengan menggunakan metode GSP untuk melakukan proses *Reverse Engineering.*

1.6. Manfaat Penelitian

Manfaat dari penelitian ini adalah untuk merancang model *reverse engineering* yang sederhana namun lebih efektif terutama untuk me-*reverse* sebuah sistem perangkat lunak yang lebih kompleks. Model yang dihasilkan diharapkan dapat membantu proses *reverse engineering* dengan menemukan *requirements* yang lebih presisi dan mengatasi adanya pengulangan *requirements* atau *missing requirements* yang biasanya terjadi pada *reverse* konvensional. Model pendekatan yang berorientasi *goal* dan *actor*, dengan metode GSP ini diharapkan dapat dimengerti oleh semua tingkatan *user*, bukan hanya *programmer* dan *developer internal* saja, akan tetapi *stakeholder*, *user*, dan *customer* juga diharapkan bisa memberikan pendapat, asumsi, dan evaluasi dari sebuah sistem perangkat lunak tersebut.

BAB II

KERANGKA TEORETIK DAN KERANGKA BERPIKIR

2.1. Kerangka Teoretik

2.1.1. Model

Menurut Gerald V. Post dan David L. Anderson 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 subyektif (Gaol, 2008: 102).

Definisi lain yaitu, model adalah suatu tiruan dari alam nyata. Kendala yang sering dihadapi dalam merancang model adalah bahwa model yang dirancang tidak mampu mencerminkan seluruh variabel alam nyata, sehingga keputusan yang diambil tidak sesuai dengan kebutuhan. Oleh karena itu, dalam menyimpan berbagai model harus diperhatikan dan harus dijaga fleksibilitasnya. Hal lain yang harus diperhatikan adalah pada setiap model yang disimpan hendaknya ditambahkan rincian keterangan dan penjelasan yang komprehensif mengenai model yang dibuat (Nofriansyah, 2015: 4).

Penjelasan model digambarkan oleh Raymond McLeod ialah sebagai berikut (Gaol, 2008: 105):

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. Model fisik bermanfaat untuk

tujuan, tetapi tidak dapat dipenuhi oleh hal nyata. Contohnya, penanam modal dan pembuat mobil dapat mengubah dengan sedikit lebih murah pada rancangan model fisik mereka ketimbang hasil akhirnya. Dari keempat jenis model yang ada, model fisik mungkin memiliki nilai yang kecil untuk *manager* usaha/bisnis. Biasanya tidak penting untuk seorang *manager* melihat sesuatu pada bentuk tiga dimensi dalam memahami atau menggunakannya dalam memecahkan masalah.

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 lambing atau bentuk-bentuk yang abstrak.

4. Model Matematika

   Model merupakan rumus matematika atau persamaan matematika. Keuntungan terbesar dari model matematika adalah tingkat ketelitian dalam menggambarkan hubungan di antara bagian suatu obyek. Matematika dapat menangani pengutaraan hubungan lebih dari dua ukuran model grafik atau tiga ukuran model fisik.

Model memiliki beberapa kegunaan, berikut merupakan kegunaan model, yaitu (Gaol, 2008: 106-107):

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. Ke semua jenis model sebenarnya mengomunikasikan informasi secara cepat dan akurat kepada orang-orang yang memahami maksud bentuk, kata-kata, grafik, dan matematika. Semuanya itu tergantung pada kecakapan personil sistem informasi itu sendiri.

Jadi, model adalah representasi dari suatu obyek yang bersifat sederhana dan menyeluruh. Dengan kata lain, model merupakan kerangka sebuah obyek yang menjadi acuan untuk memaparkan obyek secara keseluruhan. Model memiliki kegunaan untuk memudahkan pengertian dan memudahkan komunikasi antar personal.

2.1.2. *Reverse Engineering*

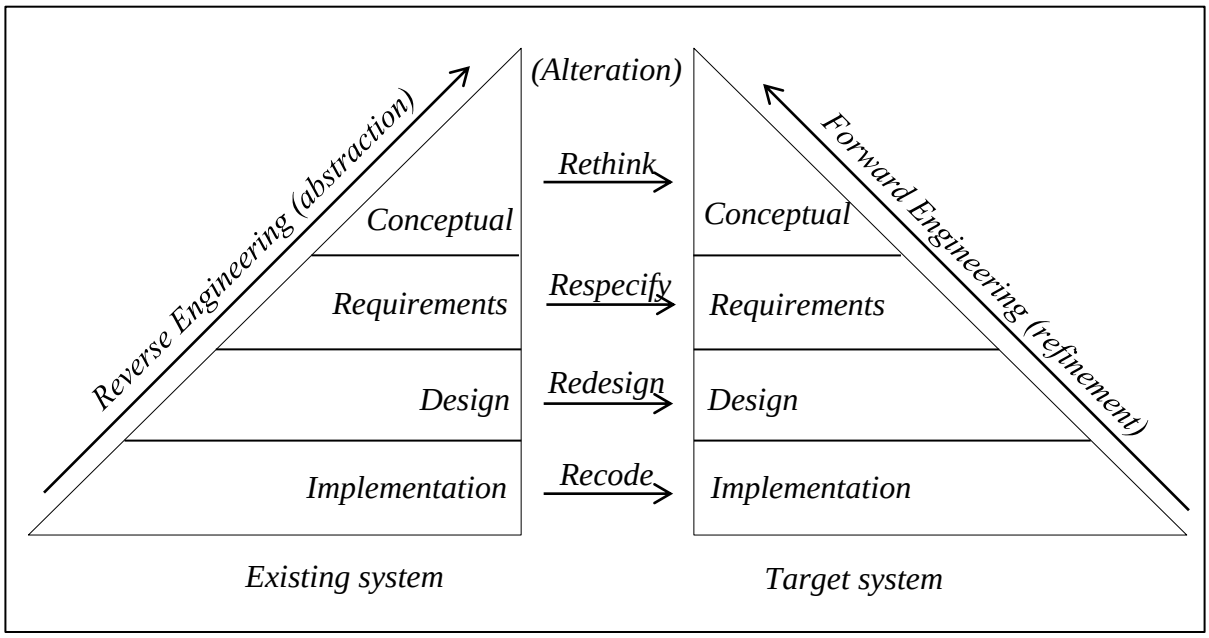
Menurut Chikofsky (1993) diacu dalam Santosa (1994: 12) dalam dunia perangkat lunak, *reverse engineering* digunakan untuk melakukan proses analisis sistem agar diperoleh identifikasi komponen sistem, hubungan antar komponen, dan membuat representasi sistem dalam bentuk lain atau melakukan abstraksi pada tingkat yang lebih tinggi.

Reverse engineering 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 (Simarmata, 2010: 404).

Pengertian lain dari *reverse engineering* adalah suatu pendekatan kerekayasaan dalam membanding-bandingkan produk, termasuk membongkar barang hasil perusahaan lain dan mengevaluasinya secara teknik. Dalam generasi pertama ini, perbandingan difokuskan pada produknya. Oleh karena itu, disebut *product oriented approach* (Indrajit dan Djokopranoto, 2011:79).

Menurut McClure (1992) diacu dalam Santosa (1994: 16) *reverse engineering* 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.

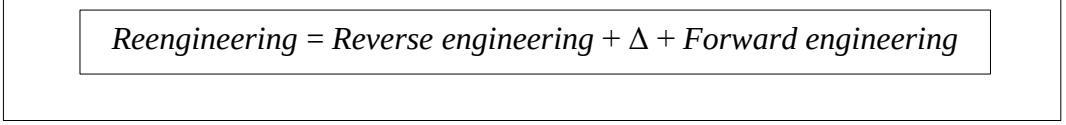
Gambar 2.1. menggambarkan proses untuk semua tingkat abstraksi dari *reengineering*. Gambar tersebut menunjukkan bahwa *reengineering* adalah urutan ketiga dari kegiatan *reverse engineering*, *redesain*, dan *forward engineering*. *Reengineering* juga didukung oleh tiga prinsip, yaitu, abstraksi, perubahan, dan perbaikan. Berikut merupakan Gambar 2.1. (Tripathy dan Naik, 2015: 138):



Gambar 2.1. Model Umum dari *Reengineering* Perangkat Lunak

Reengineering menyiratkan satu siklus pengambilan sistem yang ada dan menghasilkan sistem baru, sedangkan evolusi pada perangkat lunak dapat terjadi secara terus-menerus. Secara keseluruhan, evolusi perangkat lunak dapat dilihat sebagai perangkat lunak dengan proses *reengineering* berulang. *Reengineering* dilakukan untuk mengubah sistem siap pakai yang "lebih rendah atau sederhana" ke dalam sistem baru yang "lebih baik". Chikofsky dan Palang II mendefinisikan *reengineering* sebagai "pemeriksaan dan perubahan subyek sistem untuk menyusun kembali dalam bentuk baru dan pengimplementasian sistem dalam bentuk baru".

Oleh karena itu, *reengineering* mencakup beberapa jenis dari kegiatan *reverse engineering* untuk merancang pandangan abstrak dari sistem yang diberikan. Pandangan abstrak yang baru direstrukturisasi, dan kegiatan *forward engineering* dilakukan untuk menerapkan sistem dalam bentuk baru. Proses tersebut ditangkap oleh Jacobson dan Lindstrom dengan persamaan yang ditunjukkan oleh Gambar 2.2. berikut ini (Tripathy dan Naik, 2015: 10):



Gambar 2.2. Persamaan *Reengineering*, *Reverse Engineering*, dan *Forward Engineering*

Persamaan pada Gambar 2.2. menunjukkan bahwa elemen pertama "*reverse engineering*" adalah kegiatan mendefinisikan sebuah abstrak dan lebih mudah untuk memahami representasi dari sistem. Sebagai contoh, *input* ke proses *reverse engineering* adalah *source code* dari sistem, dan *output*-nya adalah arsitektur sistem. Inti dari *reverse engineering* adalah proses pemeriksaan sistem, dan bukan merupakan proses perubahan. Oleh karena itu, *reverse engineering* tidak melibatkan pengubahan perangkat lunak di bawah pemeriksaan. Unsur ketiga yaitu "*forward engineering*" adalah proses konvensional yang bergerak dari abstraksi tingkat tinggi dan logis, desain implementasi yang independen untuk pelaksanaan fisik dari sistem. Elemen kedua "Δ" menangkap perubahan yang terjadi pada sistem awal (Tripathy dan Naik, 2015: 10).

Saat melakukan *reverse engineering* pada sistem yang kompleks, alat dan metodologi umumnya tidak stabil. Oleh karena itu, paradigma organisasi tingkat tinggi memungkinkan pengulangan proses sehingga *maintenance engineer*

mempelajari tentang sistem. Benedusi dkk. telah mengusulkan paradigma berulang, disebut *Goals*/*Model*/*Tools*, yang menggambarkan *reverse engineering* dalam tiga tahap berturut-turut, yaitu, *Goal*, Model, dan *Tools*. Berikut penjelasan tentang paradigma tersebut (Tripathy dan Naik, 2015: 10):

1. *Goals*

   Pada fase ini, alasan untuk mengatur proses *reverse engineering* diidentifikasi dan dianalisa. Analisis dilakukan untuk mengidentifikasi kebutuhan informasi dari proses dan abstraksi yang akan dibuat oleh proses. Sebuah tim mengatur proses pengembangan pertama sebuah pemahaman yang baik pada aktivitas *forward engineering* dan lingkungan di mana produk-produk dari proses *reverse engineering* akan digunakan. Hasil dari pemahaman tersebut digunakan secara akurat untuk mengidentifikasi informasi yang akan dihasilkan dan formalisme yang akan digunakan untuk mewakili informasi. Misalnya, dokumen desain yang akan dihasilkan dari *source code* adalah sebagai berikut (Tripathy dan Naik, 2015: 156):

   a. *Low-level* dokumen, memberikan baik gambaran dan deskripsi rinci dari modul individu; deskripsi rinci meliputi struktur modul dalam hal aliran kontrol dan struktur data.
   b. *High-Level* dokumen, 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 diidentifikasi dalam tahap *goals* yang dianalisis untuk membuat representasi model. Representasi model termasuk informasi yang diperlukan untuk abstraksi berikutnya. Kegiatan pada fase ini adalah (Tripathy dan Naik, 2015: 156):

   a. Mengidentifikasi jenis dokumen yang akan dihasilkan.
   b. Untuk menghasilkan dokumen-dokumen, mengidentifikasi informasi, dan hubungan mereka berasal dari *source code*.
   c. Menentukan model yang akan digunakan untuk mewakili informasi dan hubungan mereka diekstrak dari *source code*.
   d. Untuk menghasilkan dokumen yang diinginkan dari model-model dan menentukan algoritma abstraksi untuk *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 *reverse engineering* diidentifikasi, diperoleh, dan/atau dikembangkan sendiri. *Tools* tersebut dikelompokkan menjadi dua kategori, yaitu: (i) alat untuk mengekstrak informasi dan menghasilkan representasi program sesuai dengan identifikasi model dan (ii) alat untuk mengekstrak informasi dan menghasilkan dokumen yang diperlukan. Ekstraksi *tools* umumnya bekerja pada *source code* untuk merekonstruksi dokumen

> desain. Oleh karena itu, alat-alat yang efektif dalam memproduksi masukan bagi proses abstraksi bertujuan untuk menghasilkan dokumen desain tingkat tinggi (Tripathy dan Naik, 2015: 156).

Hal terpenting adalah bahwa pengumpulan fakta dan informasi dari *source code* adalah kunci untuk paradigma *Goal*/*Model*/*Tools*. Dalam rangka untuk mengekstrak informasi yang tidak secara eksplisit tersedia dalam *source code*, teknik analisis otomatis, seperti analisis leksikal, analisis sintaksis, analisis aliran kontrol, analisis aliran data, dan program yang mengiris digunakan untuk memfasilitasi *reverse engineering* (Tripathy dan Naik, 2015: 10).

Reverse engineering merupakan topik yang hangat dibicarakan dalam bidang kerekayasaan, akan tetapi, saat ini beberapa literatur yang bersangkutan baru menjadi bahan eksperimen oleh beberapa pakar perangkat lunak untuk memperdalam pengetahuan tentang *reverse engineering*. Namun, suatu saat nanti *reverse engineering* akan menjadi kebutuhan mutlak dalam dunia rekayasa perangkat lunak.

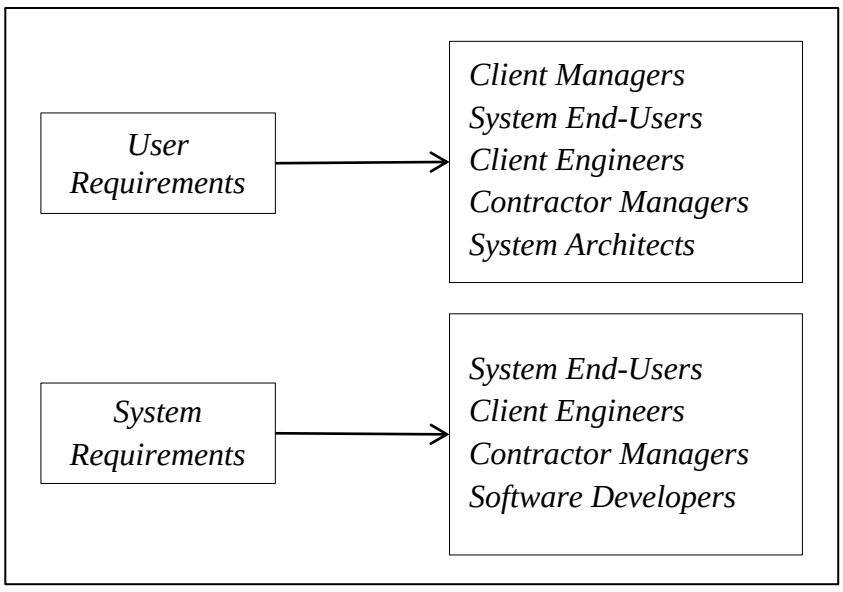
Jadi, *reverse engineering* adalah proses identifikasi sebuah sistem perangkat lunak dengan merekonstruksi komponen dan hubungan antar komponen pada suatu sistem, tujuannya ialah untuk mendokumentasikan ulang atau memperoleh informasi *design* tentang suatu sistem. *Reverse engineering* dilakukan pada sistem yang sudah siap pakai dan minim dokumentasi.

2.1.3. *Requirements Engineering*

Ukuran utama keberhasilan sistem perangkat lunak adalah untuk memenuhi tujuannya. Oleh karena itu, mengidentifikasi tujuan harus menjadi

salah satu kegiatan utama dalam pengembangan sistem perangkat lunak. Tidak memadainya suatu sistem, tidak lengkap, ambigu, atau tidak konsisten akan memberikan dampak yang signifikan pada kualitas perangkat lunak. Dengan demikian, *requirements engineering* merupakan salah satu cabang dari rekayasa perangkat lunak yang berhubungan dengan elisitasi, perbaikan, analisis, dan lain-lain dalam persyaratan sistem perangkat lunak (Lapouchnian, 2005: 1).

Functional requirements adalah pernyataan dari pelayanan 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. *Functional requirements* dibagi menjadi dua, yaitu *user requirements* dan *system requirements*. Berikut Gambar 2.3. merupakan perbedaan spesifikasi pada *requirements* (Sommerville, 2009: 84-85):



Gambar 2.3. Perbedaan Spesifikasi *Requirements*

Functional requirements pada sistem menggambarkan apa saja yang harus dilakukan oleh sistem. *requirements* ini tergantung pada jenis perangkat lunak

yang dikembangkan, harapan *user* pada perangkat lunak, dan pendekatan umum yang diambil oleh organisasi saat menulis *requirements*. Ketika dinyatakan sebagai *user requirements*, *functional requirements* biasanya digambarkan dengan cara yang abstrak yang dapat dipahami oleh *user*. Namun, *functional requirements* yang lebih spesifik menggambarkan fungsi sistem, *input*, *output*, pengecualian, dan lain-lain secara rinci (Sommerville, 2009: 85).

Functional requirements pada sistem bervariasi dari *general requirements* yaitu meliputi apa saja yang harus dilakukan oleh sistem untuk mencerminkan *requirements* yang sangat spesifik dari sebuah proyek yang sedang dilakukan atau sistem yang sudah ada. Berikut adalah contoh *functional requirements* untuk sistem MHC-PMS, digunakan untuk mempertahankan informasi tentang pasien yang menerima pengobatan untuk masalah kesehatan mental (Sommerville, 2009: 85-86):

1. *User* dapat mencari daftar janji untuk semua klinik.
2. Sistem harus menampilkan daftar pasien yang diharapkan untuk menghadiri janji hari itu setiap hari nya untuk setiap klinik,.
3. Setiap anggota staf yang menggunakan sistem harus diidentifikasi secara unik dengan nomor karyawan yang teridiri dari delapan angka.

User functional requirements ini menentukan fasilitas tertentu yang akan diberikan oleh sistem. *Requirements* tersebut diambil dari dokumen dan mereka menunjukkan bahwa *functional requirements* dapat ditulis di berbagai tingkat rincian (*requirements* kontras 1 dan 3) (Sommerville, 2009: 86).

Ketidaktepatan dalam spesifikasi *requirements* adalah penyebab banyak masalah rekayasa perangkat lunak. Hal ini wajar bagi pengembang sistem untuk

menafsirkan ambiguitas *requirements* dengan cara yang menyederhanakan pelaksanaannya. Namun, hal ini bukanlah sesuatu yang diinginkan oleh *customer*. *Requirements* baru harus dibentuk dan kemudian perubahan dilakukan pada sistem. Tentu saja hal ini menyebabkan penundaan pengiriman sistem dan peningkatan biaya (Sommerville, 2009: 86).

Sebagai contoh, misalnya *requirements* pertama untuk MHC-PMS menyatakan bahwa *user* akan dapat mencari daftar janji untuk semua klinik. Alasan untuk *requirements* ini adalah bahwa pasien dengan masalah kesehatan mental terkadang bingung. Mereka mungkin memiliki janji di salah satu klinik tapi benar-benar pergi ke klinik yang berbeda. Jika mereka memiliki janji, maka data mereka akan disimpan sebagai pasien yang telah hadir, terlepas dari urusan klinik (Sommerville, 2009: 86).

Dalam kasus ini anggota staf medis mengharapkan fungsi "pencarian", ketika diberi nama pasien, sistem akan mencari nama dan janjinya di semua klinik. Namun, hal ini tidak eksplisit dalam *requirements*. *System developer* dapat menafsirkan *requirements* dalam cara yang berbeda dan dapat mengimplementasikan pencarian sehingga *user* harus memilih klinik kemudian melakukan pencarian. Hal ini jelas akan lebih melibatkan *user input* dan memakan waktu lebih lama (Sommerville, 2009: 86).

Pada prinsipnya, spesifikasi *functional requirements* pada sistem harus lengkap dan konsisten. Kelengkapan berarti bahwa semua layanan yang dibutuhkan oleh *user* harus didefinisikan. Konsistensi berarti bahwa *requirements* tidak harus memiliki definisi yang kontradiktif. Dalam prakteknya, untuk sistem yang kompleks, secara praktis tidak mungkin untuk mencapai *requirements* yang

konsisten dan lengkap. Alasannya adalah bahwa akan mudah terjadi kesalahan dan kelalaian saat menulis spesifikasi untuk sistem yang kompleks. Alasan lain adalah bahwa ada banyak *stakeholder* dalam sistem kompleks itu sendiri. *Stakeholder* adalah orang atau peran yang dipengaruhi oleh sistem dalam beberapa cara. *Stakeholder* memiliki kebutuhan yang berbeda dan biasanya tidak konsisten. Hal ini kemungkinan tidak dijelaskan ketika *requirements* yang pertama ditentukan, sehingga *requirements* yang tidak konsisten termasuk dalam spesifikasi. Masalah hanya muncul setelah analisis lebih dalam atau setelah sistem disampaikan pada *customer*. (Sommerville, 2009: 87)

Functional requirements adalah kemampuan sistem yang harus diperlihatkan untuk memecahkan masalah. *Functional requirements* dapat disubklasifikasikan dalam beberapa kebutuhan berikut (Simarmata, 2010: 216-217):

1. Kebutuhan data, juga dikenal sebagai kebutuhan konseptual, kebutuhan konten, atau kebutuhan penyimpanan. Kebutuhan-kebutuhan ini menetapkan bagaimana informasi disimpan dan diadministrasikan oleh aplikasi.
2. Kebutuhan antarmuka (kepada *user*), juga dikenal sebagai kebutuhan interaksi. Mereka memberikan suatu jawaban bagaimana *user* akan berinteraksi dengan aplikasi.
3. Kebutuhan navigasional, mewakili kebutuhan navigasional *user* melalui *hyperspace*.
4. Kebutuhan personalisasi, juga dikenal sebagai kebutuhan kustomisasi atau adaptasi. Mereka menggambarkan bagaimana aplikasi harus

(secara dinamis) beradaptasi sendiri, tergantung pada profil lingkungan dan *user*.

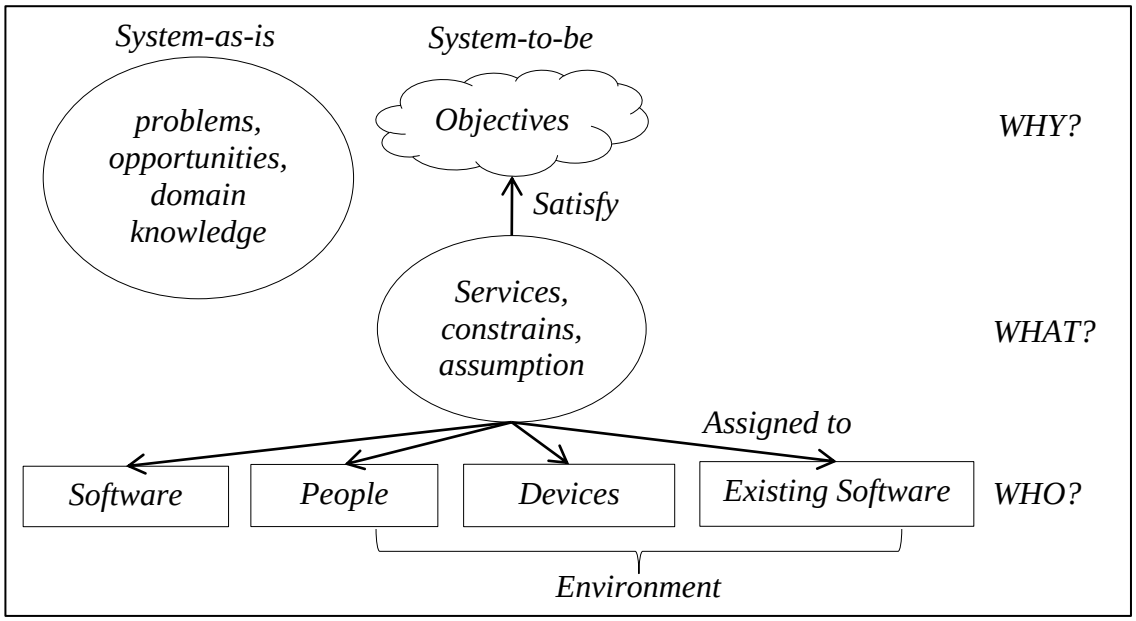
5. Kebutuhan transaksional, juga dikenal sebagai kebutuhan transaksional atau kebutuhan layanan, yang menegaskan apakah aplikasi harus diperhitungkan secara internal, tanpa mempertimbangkan aspek-aspek antarmuka dan interaksi.

Salah satu definisi tertua *requirements engineering* mengatakan bahwa "definisi *requirements* adalah penilaian yang teliti dari sebuah kebutuhan sistem untuk dipenuhi. Hal ini yang menyebabkan mengapa sistem sangat dibutuhkan, baik dalam kondisi sekarang atau yang akan datang, yang memungkinkannya menjadi pengoperasian internal atau pemasaran eksternal. *Requirements engineering* juga menentukan apakah fitur sistem akan melayani dan memuaskan *user* dengan baik dan bagaimana sistem akan dibangun" (Ross dan Schoman, 1997, diacu dalam Lapouchnian, 2005: 1). Dengan demikian, *requirements engineering* harus mengatasi alasan mengapa sistem perangkat lunak diperlukan, fungsi yang harus dimiliki untuk mencapai tujuan, dan kendala bagaimana perangkat lunak harus dirancang dan diimplementasikan. (Lapouchnian, 2005: 1).

Definisi lain dari *requirements engineering* adalah sebagai proses menemukan tujuan sistem perangkat lunak dengan mengidentifikasi *stakeholder* (orang atau organisasi yang akan dipengaruhi oleh sistem dan yang memiliki pengaruh langsung atau tidak langsung pada sistem *requirements*) dokumentasi dilakukan berdasarkan persetujuan analisis, komunikasi, dan implementasi selanjutnya (Kotonya dan Sommerville. 1998, diacu dalam Lapouchnian, 2005: 1).

Selain itu *requirements engineering* juga didefinisikan sebagai cabang rekayasa perangkat lunak yang bersangkutan dengan tujuan dunia nyata, fungsi, dan kendala pada sistem perangkat lunak. Hal ini berkaitan pada hubungan faktor-faktor dengan spesifikasi yang tepat dari perangkat lunak, dan evolusi mereka dari waktu ke waktu dalam keluarga perangkat lunak (Zave, 1997, diacu dalam Lapouchnian, 2005: 1).

Requirements 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 bertanggungjawab untuk menyelesaiakan masalah tersebut (Van Lamsweerde, 2009, diacu dalam Shofi dan Budiarjo, 2011: 2). *Why*, *what*, dan *who* dikenal sebagai tiga dimensi dalam *requirements engineering*. Berikut merupakan Gambar 2.4. yang menjelaskan tentang tiga dimensi dalam *requirements engineering*:



Gambar 2.4. Tiga Dimensi *Requirements Engineering*

Gambar 2.4. menjelaskan bahwa dimensi *why* fokus pada alasan kontekstual terhadap usulan versi baru dari sistem yang secara eksplisit merupakan obyektif (*objectives*) yang harus dipenuhi berdasarkan keterbatasan-keterbatasan (*limitations*) dari *system-as-is* dan peluang-peluang eksploitasi dari *system-as-is* tersebut (Van Lamsweerde, 2009, diacu dalam Shofi dan Budiarjo, 2011: 2).

Dimensi *what* fokus pada layanan fungsional (*functional services*) *system-to-be* yang harus disediakan untuk memenuhi obyektif yang diidentifikasi pada dimensi *why* (Van Lamsweerde, 2009, diacu dalam Shofi dan Budiarjo, 2011: 2). Diantara layanan-layanan tersebut akan diimplementasikan oleh *system to-be*, dan sebagian yang lain akan direalisasikan melalui prosedur manual maupun *device*. Layanan-layanan sistem, konstrain, dan asumsi dapat diidentifikasi dari obyektif sistem yang telah disetujui dan diformulasikan dengan istilah/bahasa yang tepat/cermat dimana semua pihak yang berkepentingan memahami (*understand*) untuk memvalidasi dan merealisasikannya (Shofi dan Budiarjo, 2011: 2).

Sedangkan dimensi *who* mengarah pada penugasan (*assignment*) tanggung jawab untuk mencapai obyektif, layanan, dan konstrain diantara komponen-komponen pada *system-to-be*, manusia, alat (*devices*), atau perangkat lunak (Van Lamsweerde, 2009 diacu dalam Shofi dan Budiarjo, 2011: 2). Penugasan tanggung jawab ini mungkin saja membutuhkan evaluasi terhadap pilihan, satu tanggung jawab (tanggung jawab yang sama) dapat di-*assign* pada komponen sistem yang berbeda dimana masing-masing penugasannya tersebut mempunyai pro dan kontranya masing-masing.

Proses *requirements engineering* dapat terdiri dari (Van Lamsweerde, 2009, diacu dalam Shofi dan Budiarjo, 2011: 2):

1. *Domain understanding*, merupakan aktifitas yang terdiri dari studi *system-as-is* diantara organisasionalnya dan konteks teknikal dengan tujuan untuk mendapatkan pemahaman yang baik terhadap domain dimana permasalahan tersebut berakar/bersumber dan apa akar/sumber permasalahan tersebut. Pada tahap ini akan ditemukan permasalahan yang kemudian akan di proses ke tahap *elicitation*.
2. *Requirements elicitation*, merupakan aktifitas yang terdiri dari penemuan (*discovering*) kandidat kebutuhan (*requirements*) dan asumsi yang akan membentuk *system-to-be* berdasarkan kekurangan-kekurangan dari *systemas-is* yang muncul dari *domain understanding*.
3. *Evaluation & agreement*, bertujuan untuk membuat keputusan informasi tentang isu-isu yang muncul selama proses *elicitation*. Keputusan dan persetujuan tentunya didasarkan pada keadaan lingkungan, tingkat kebutuhan, dan lain sebagainya.
4. *Specification & documentation*, merupakan aktifitas yang terdiri dari perincian (*detailing*), pengstrukturan (*structuring*), dan pendokumentasian (*documenting*) karakteristik *system-to-be* yang telah disetujui selama proses evaluasi.
5. *Requirements consolidation*, merupakan aktifitas yang bertujuan untuk menjamin kualitas (*quality assurance*).

Jadi, *requirements engineering* adalah salah satu cabang rekayasa perangkat lunak yang terdiri dari beberapa proses (*domain understanding*,

requirements elicitation, *evaluation & agreement*, *specification & documentation*, dan *requirements consolidation*) dengan maksud untuk menemukan tujuan dari sistem perangkat lunak. *Requirements engineering* mencari alasan mengapa sistem perangkat lunak diperlukan, fungsi apa saja yang harus dimiliki untuk mencapai tujuan sebuah sistem, dan kendala bagaimana merancang perangkat lunak serta iimplementasi dari perangkat lunak itu sendiri.

2.1.4. *Goal Oriented Requirements Engineering* (GORE)

Proses *reverse engineering* bisa dilakukan secara langsung atau dengan metode konvensional, namun untuk program yang terdiri dari ratusan bahkan ribuan baris hanya akan memakan banyak biaya dan waktu yang lebih lama. Oleh karena itu, diperlukan suatu alat bantu yang dapat me-*reverse* suatu program menjadi abstrak tingkat tinggi yang lebih mendekati pola pemahaman *stakeholder*. GORE merupakan salah satu pendekatan di dalam *Requirements Engineering* berorientasi *goal* dan *actor* yang akhir-akhir ini memiliki peningkatan sangat pesat. Alasan utamanya adalah metode konvensional yang tidak memadai dalam melakukan pendekatan analisis ketika berhadapan dengan sistem perangkat lunak yang lebih kompleks (Lapouchnian, 2005: 4). Selain itu, model pendekatan konvensional cenderung lebih menekankan pemodelan *requirements* dalam bentuk *low-level data*, operasi, dan lainnya yang lebih banyak dipahami oleh *programmer* dan *developer internal*, sedangkan *stakeholder*, *user*, dan *customer* cenderung kurang peduli dengan pemodelan dalam bentuk *low-level data*. (Lapouchnian, 2005: 4).

Ada tiga tingkatan atau level dalam abstraksi data, yaitu:

1. Level Fisik (*Physical Level*)

   Level abstraksi data yang paling rendah, yang menggambarkan bagaimana (*how*) data disimpan dalam kondisi sebenarnya. Level ini sangat kompleks karena struktur data dijelaskan secara rinci. (Yuhefizard, 2008: 9). Misalnya kita memiliki data mahasiswa. Pada level ini data mahasiswa dipandang dengan memperhatikan bahwa dalam data tersebut ada atribut Nama yang disimpan pada media penyimpanan (*disk*) sepanjang 20 byte (Kusrini, 2007: 15).

2. Level Konseptual/Logika

   Level ini menggambarkan data apa yang disimpan dalam *database* dan hubungan relasi yang terjadi antara data dari keseluruhan basis data. Level ini memperhatikan data apa sebenarnya (secara *functional*) disimpan dalam basis data dan hubungannya dengan data yang lain. pemakai tidak memperdulikan kerumitan dalam struktur level fisik lagi, penggambaran cukup dengan memakai kotak, garis, dan hubungan secukupnya. Pada level ini biasanya desainer basis data membuat rancangan dalam bentuk diagram-diagram/model.

3. Level Pandangan *User* (*User View Level*)/Eksternal

   Level ini merupakan level abstraksi data tertinggi yang menggambarkan hanya sebagian saja yang dilihat dan dipakai dari keseluruhan *database*. Hal ini disebabkan beberapa *user database* tidak membutuhkan semua isi *database*. Yang dimaksud dengan *user* di sini adalah *programmer*, *end user*, atau DBA (*Database*

Administrator). Setiap *user* mempunyai "bahasa" yang sesuai dengan kebutuhannya, antara lain:

a. *Programmer*: bahasa yang digunakan adalah bahasa pemrograman seperti C, CBOL, atau PL/I (*Programmer Language I*) (Kusrini, 2007: 15).
b. *End User*: bahasa yang digunakan adalah bahasa *query* atau menggunakan fasilitas yang tersedia pada program aplikasi. Pada level eksternal ini, *user* dibatasi pada kemampuan perangkat keras dan perangkat lunak yang digunakan aplikasi basis data (Kusrini, 2007: 16).

Abstraksi data dalam arsitektur sistem basis data memberikan pandangan terhadap data secara berbeda-beda tergantung dari levelnya. Sebagai contoh pada level *user view*, bayangan mengenai data tidak lagi memperhatikan bagaimana data disusun dan disimpan dalam *disk*, akan tetapi memperhatikan bagaimana data direpresentasikan sesuai dengan keadaan yang dihadapi sehari-hari oleh *user*. Hal ini dimaksudkan menghindari kerumitan teknik penyimpanan dan pengelolaan data dari pandangan awam (Kusrini, 2007: 16).

Goal adalah kondisi/keadaan yang diinginkan dari sebuah sistem berdasarkan beberapa pertimbangan. Dalam hal ini, *agent* bertanggung jawab atas pemenuhan *goal*. Terdapat beberapa konsep tentang *goal*, yaitu *goal type*, *belief*, *constraint*, level abstraksi *goal*, taxonomi *goal*, *agents*, *requirement*, asumsi, atribut *goal*, dan *goal link* (Anwer dan Ikram, 2006: 2). Berikut merupakan uraian dari beberapa konsep *goal* tersebut:

1. *Goal Type*

   Tipe *goal* 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).

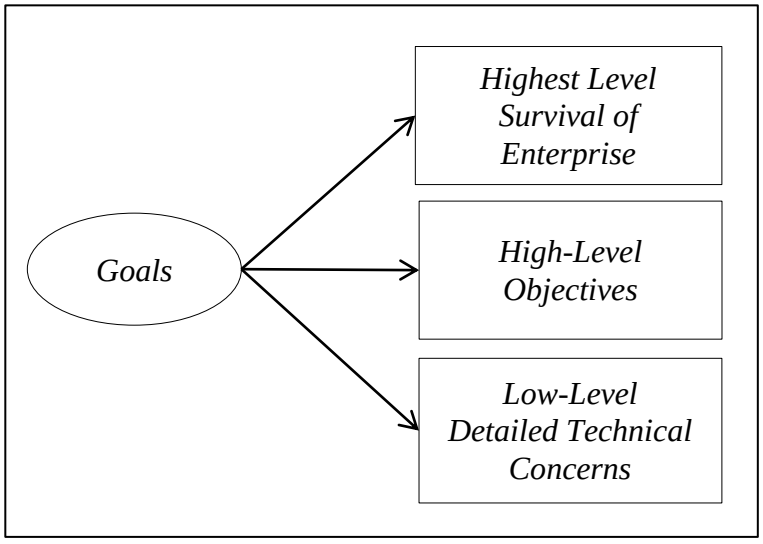
2. *Belief*

   Konsep *belief* sangat berkaitan dengan *agent*, merupakan cara pandang agen tentang statusnya sendiri maupun lingkungannya.

3. *Constraint*

   Constraint merupakan batasan pencapaian suatu *goal*.

4. Level Abstraksi *Goal*



Gambar 2.5. Level Abstraksi *Goal*

Gambar 2.5. yang menggambarkan level abstraksi atau tingkatan dari *goal*. Terdapat tiga level abstraksi atau tingkatan dari *goal* yaitu: *highest level* (berhubungan dengan *survival*/pertahanan suatu perusahaan atau organisasi), *high-level* (berhubungan dengan masalah

kestrategisan suatu perusahaan), dan *low-level* (berhubungan dengan teknik yang digunakan suatu perusahaan dalam menanggapi masalah).

5. *Taxonomi Goal*

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

6. *Agents*

   Agen adalah komponen aktif dari suatu sistem seperti manusia, alat, dan perangkat lunak. Agen 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*

   Goal di bawah pertanggung jawaban agen tunggal di dalam perangkat lunak yang akan dibuat menjadi sebuah *requirements*.

8. Asumsi

   Goal di bawah tanggung jawab agen tunggal di lingkungannya perangkat lunak yang akan dibuat menjadi asumsi.

9. Atribut *Goal*

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

10. *Goal Link*

    Link dimaksudkan 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 *goals*), *Coverage links* (berhubungan dengan *goal* dengan skenarionya), *Responsibility links* (yang merelasikan *goal* agen yang bertanggung jawab terhadapnya), dan *Wish links* (Link ini membantu agen menentukan *goal*. Sebagai contoh, seorang agen tidak boleh diberikan *goal* yang diinginkan untuk memenuhi *goal* lain yang dapat menyebabkan konflik antara *goal*).

GORE merupakan salah satu model pendekatan *Requirements Engineering* yang merasionalisasikan berbagai *requirements* yang diperlukan oleh sebuah sistem yang akan dibuat berdasarkan dari tujuan-tujuan yang dirumuskan sehingga diharapkan *requirements* yang didapatkan bukan hanya berdasarkan data dan proses bisnis manual (Adikara, dkk,. 2013: 229). Berikut merupakan penjelasan beberapa metode GORE, yaitu:

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

Metode *Goal-Based Requirements Analysis Method* (GBRAM) 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 (Lapouchnian, 2005: 11).

Serupa dengan pendekatan GORE lainnya, sistem dan lingkungan pada GBRAM direpresentasikan sebagai kumpulan agen. Di sini, agen didefinisikan sebagai entitas atau proses yang berusaha untuk mencapai tujuan dalam sebuah organisasi atau sistem berdasarkan tanggung jawab yang diasumsikan untuk *goal*. Pendekatan GBRAM membantu dalam *goal elicitation* dan perbaikan sistem dengan menggunakan *requirements* dan mewawancarai *programmer* dengan pertanyaan sederhana. Sebagai contoh, satu pertanyaan mungkin untuk mengetahui tujuan pemeliharaan sistem "Apakah keberhasilan terus menerus dari *goal* ini diperlukan?".

Dalam GBRAM, tujuan, agen, *stakeholder*, dan lain-lain ditentukan dalam bentuk tekstual pada skema *goal*. Anehnya, metode ini tidak memberikan notasi grafis untuk mewakili tujuan, perbaikan tujuan, agen, dan lain-lain. Sementara secara metode hal tersebut dilibatkan, misalnya menciptakan hubungan antara prioritas *goal*, hubungan tersebut lebih mudah dimengerti ketika diwakili secara grafis (Lapouchnian, 2005: 12).

2.1.4.2. *Non-Functional Requirements* (NFR)

Metode *Non-Functional Requirements* (NFR) lebih berkonsentrasi pada pemodelan dan analisis *non-fuctional requirements*. Tujuan dari metode ini adalah untuk menempatkan *non-functional requirements* menjadi hal paling utama dalam pikiran *developer*. Metode ini bertujuan menangani dengan kegiatan utama

mengambil NFR untuk domain yang menarik, dekomposisi NFR, mengidentifikasi kemungkinan operasionalisasi NFR (desain alternatif untuk memenuhi NFR), berurusan dengan ambiguitas, prioritas, dan saling ketergantungan di antara NFR, memilih operasionalisasi, mendukung keputusan dengan desain rasional, dan mengevaluasi dampak dari keputusan. Gagasan utama dari pendekatan ini adalah untuk memodelkan secara sistematis dan memperbaiki *non-functional requirements* serta untuk memaparkan pengaruh positif dan negatif dari berbagai alternatif tentang *requirements* tersebut (Lapouchnian, 2005: 7).

Metode NFR didukung oleh tiga jenis *softgoals*, yaitu NFR *softgoal* yang merepresentasikan *requirements non-functional* sebagai bahan pertimbangan, operasionalisasi *softgoals* model dengan teknik *low-level* untuk memenuhi *softgoals* NFR itu sendiri, Klaim *softgoals* yang memungkinkan analis untuk mendokumentasikan desain pemikiran untuk perbaikan *softgoal*, prioritas softgoal, kontribusi *softgoal*, dan lain-lain. *Softgoals* dapat disempurnakan dengan menggunakan perbaikan *AND* atau *OR* dengan semantik yang jelas. Juga, saling ketergantungan *softgoal* dapat diambil sebagai kontribusi positif (“+”) atau negatif (“-”).

Hal yang utama dalam pemodelan ini adalah menyediakan grafik *softgoal* yang saling ketergantungan (*Softgoal Interdependency Goal*). Grafik tersebut akan mewakili *softgoals*, perbaikan *softgoal* (*AND*/*OR*), kontribusi *softgoal* (positif/negatif), operasionalisasi *softgoal*, dan klaim. Karena *softgoals* sedang dalam tahap perbaikan, biasanya *developer* pada akhirnya akan menemukan beberapa *softgoals* (pada kenyataannya, *softgoals* operasionalisasi) yang cukup rinci dan tidak dapat disempurnakan lebih lanjut. *Developer* dapat

menggunakannya atau tidak sama sekali, hal tersebut merupakan bagian dari target suatu sistem.

Dengan memilih kombinasi *alternative* dari *softgoals leaf-level* dan menggunakan algoritma penyebaran label yang disediakan, *developer* dapat melihat apakah *alternative* yang dipilih adalah cukup baik untuk memenuhi *high-level non-functional requirements* untuk sistem. Algoritma ini bekerja menjalar ke atas grafik mulai dari keputusan yang dibuat oleh *developer*. Prosedur label yang bekerja ke arah atas grafik menentukan dampak dari keputusan pada *goal-level* yang lebih tinggi. Ini mempertimbangkan label pada perbaikan *softgoal*. Sebagai contoh, jika *softgoal* menerima kontribusi dari sejumlah *softgoals* lainnya, maka hasil kontribusi dari masing-masing *child* digabungkan untuk mendapatkan kontribusi keseluruhan untuk *softgoal parents*. Dengan menganalisis operasionalisasi *alternative*, *developer* akan memilih salah satu yang paling memenuhi persyaratan kualitas *high-level* untuk sistem. *Set* yang dipilih *softgoals leaf* level/operasionalisasi dapat diimplementasikan dalam perangkat lunak.

Metode NFR mengelompokkan desain keilmuan katalognya menjadi tiga jenis, yaitu (Lapouchnian, 2005: 8):

1. Jenis katalog NFR mencakup konsep tentang jenis NFR tertentu, seperti kinerjanya.
2. Metode katalog mengkodekan pengetahuan yang membantu dalam perbaikan *softgoal* dan operasionalisasi. Katalog ini bisa memiliki metode generik yang menyatakan bahwa *softgoal* NFR diterapkan pada idem data yang dapat didekomposisi menjadi *softgoals* NFR untuk semua komponen dari barang tersebut.

3. Aturan hubungan katalog memiliki pengetahuan yang membantu dalam mendeteksi saling ketergantungan secara implisit antara *softgoals*. Sebagai contoh, katalog dapat mencakup fakta bahwa pengindeksan kontribusi positif untuk waktu respon.

Dalam sistem rekayasa perangkat lunak, sebuah *requirement* perangkat lunak tidak wajib menggambarkan hal yang perangkat lunak akan lakukan, akan tapi bagaimana perangkat lunak akan melakukannya, misalnya, persyaratan kinerja perangkat lunak, persyaratan perangkat lunak antarmuka eksternal, kendala desain, dan atribut kualitas perangkat lunak. *Non-functional requirements* sulit untuk melakukan tes terhadap beberapa hal di atas, oleh karena itu, biasanya dilakukan evaluasi secara subyektif.

2.1.4.3. i*/Tropos

i* adalah sebuah metode yang berorientasi agen dan dapat digunakan untuk *requirements engineering*, rekayasa ulang proses bisnis, analisis dampak organisasi, dan pemodelan proses perangkat lunak. Sejak i* mulai digunakan pada penerapan metode untuk pemodelan sistem *requirements*, deskripsi i* kemudian diarahkan pada proses *requirements engineering*. Metode ini memiliki dua komponen utama, yaitu: *the Strategic Dependency* (SD) model dan Strategic Rationale (SR) model (Lapouchnian, 2005: 8).

Metode pemodelan i* adalah dasar untuk tropos, yaitu metodologi pengembangan berbasis *requirements* dan berorientasi agen. Metodologi tropos memandu pengembangan sistem berbasis agen dari awal analisis *requirements* melalui desain arsitektur dan detail desain untuk tahap implementasi. Tropos

menggunakan metode pemodelan i* untuk merepresentasikan dan memberikan alasan tentang *requirements* serta pemilihan pada konfigurasi sistem. Tropos memiliki asosiasi resmi bahasa spesifikasi formal yang disebut tropos formal untuk menambahkan kendala dan *invariants* pada saat sebelum dan sesudah kondisi pada pemilihan domain subyek untuk model grafis di notasi i*. Model ini dapat divalidasi dengan model pengecekan (Lapouchnian, 2005: 10).

Pada awal fase *requirements*, metode i* digunakan untuk memodelkan lingkungan dari *sistem-to-be*. Analisis domain akan memungkinkan *modeler* dalam diagram mewakili *stakeholder* dari sistem, tujuan, dan hubungan mereka. Hal ini dapat memvisualisasikan proses dan memeriksa alasan di balik proses tersebut. i* model yang dikembangkan pada tahap ini membantu dalam memahami mengapa sistem baru diperlukan. Selama akhir fase *requirements*, model i* digunakan untuk mengusulkan konfigurasi sistem yang baru dan proses baru serta mengevaluasinya berdasarkan seberapa baik hal tersebut dalam memenuhi *functional requirements* dan non-*functional requirements* dari *user*.

Teknik pengembangan perangkat lunak konvensional telah terinspirasi dan didorong oleh paradigma pemrograman. Ini berarti bahwa konsep, metode, dan alat-alat yang digunakan selama fase pembangunan didasarkan pada apa yang ditawarkan oleh paradigma pemrograman unggulan. Jadi, selama era pemrograman terstruktur, analisis terstruktur, dan desain teknik diusulkan, pemrograman berorientasi obyek telah melahirkan yang lebih baru untuk *object-oriented* desain dan analisis. Untuk teknik pengembangan terstruktur, seluruh pengembangan perangkat lunak yang didapatkan *developer* merupakan konsep sistem perangkat lunak dalam hal fungsi, proses, *input*, dan *output*. Untuk

pengembangan berorientasi obyek, di sisi lain, *developer* memikirkan seluruh segi obyek, kelas, metode, warisan, dan sejenisnya.

Tropos merupakan kerangka pemodelan yang memandang perangkat lunak dari empat perspektif yang saling melengkapi, yaitu:

1. Sosial - merupakan aktor yang relevan, apa yang mereka inginkan? Apa kewajiban mereka? Apa kemampuan mereka?
2. Disengaja - apa tujuan yang relevan dan bagaimana mereka saling berhubungan? Bagaimana mereka terpenuhi dan oleh siapa?
3. Proses-berorientasi - apa proses bisnis/komputer yang relevan? Siapa yang bertanggung jawab dan untuk apa?
4. Berorientasi obyek - apa obyek dan kelas yang relevan, bersama dengan hubungan antar mereka?

2.1.4.4. *Knowledge Acquisition in autOmated Specification* atau *Keep All Objects Satisfied* (KAOS)

KAOS merupakan singkatan dari *Knowledge Acquisition in autOmated Specification* atau *Keep All Objects Satisfied*, yaitu pendekatan dalam GORE yang menggunakan sekumpulan teknik analisis formal (Lapouchnian, A., 2005: 10).

KAOS 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 (Adikara, dkk,. 2013: 230).

Elemen pada KAOS (Adikara, dkk,. 2013: 230) meliputi istilah berikut ini:

1. Tujuan (*goal*) didefinisikan sebagai kumpulan perilaku/keadaan yang harus dipenuhi atau dapat diterima oleh sistem dalam sebuah kondisi yang ditetapkan (Van Lamsweerde, 2001: 249). Definisi *goal* harus jelas sehingga dapat diverifikasi apakah sistem mampu memenuhi/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. Agen (*agents*) adalah sebuah jenis dari obyek yang bertindak sebagai pemroses kegiatan operasional. Agen merupakan komponen aktif bisa berupa manusia, perangkat keras, perangkat lunak, dan lainnya yang mempunyai peran spesifik dalam memuaskan sebuah tujuan.

Ada tiga jenis ketergantungan diantara *goal* pada KAOS (Adikara, dkk,. 2013: 230), yaitu:

1. *AND/OR-decomposition* yaitu 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*).

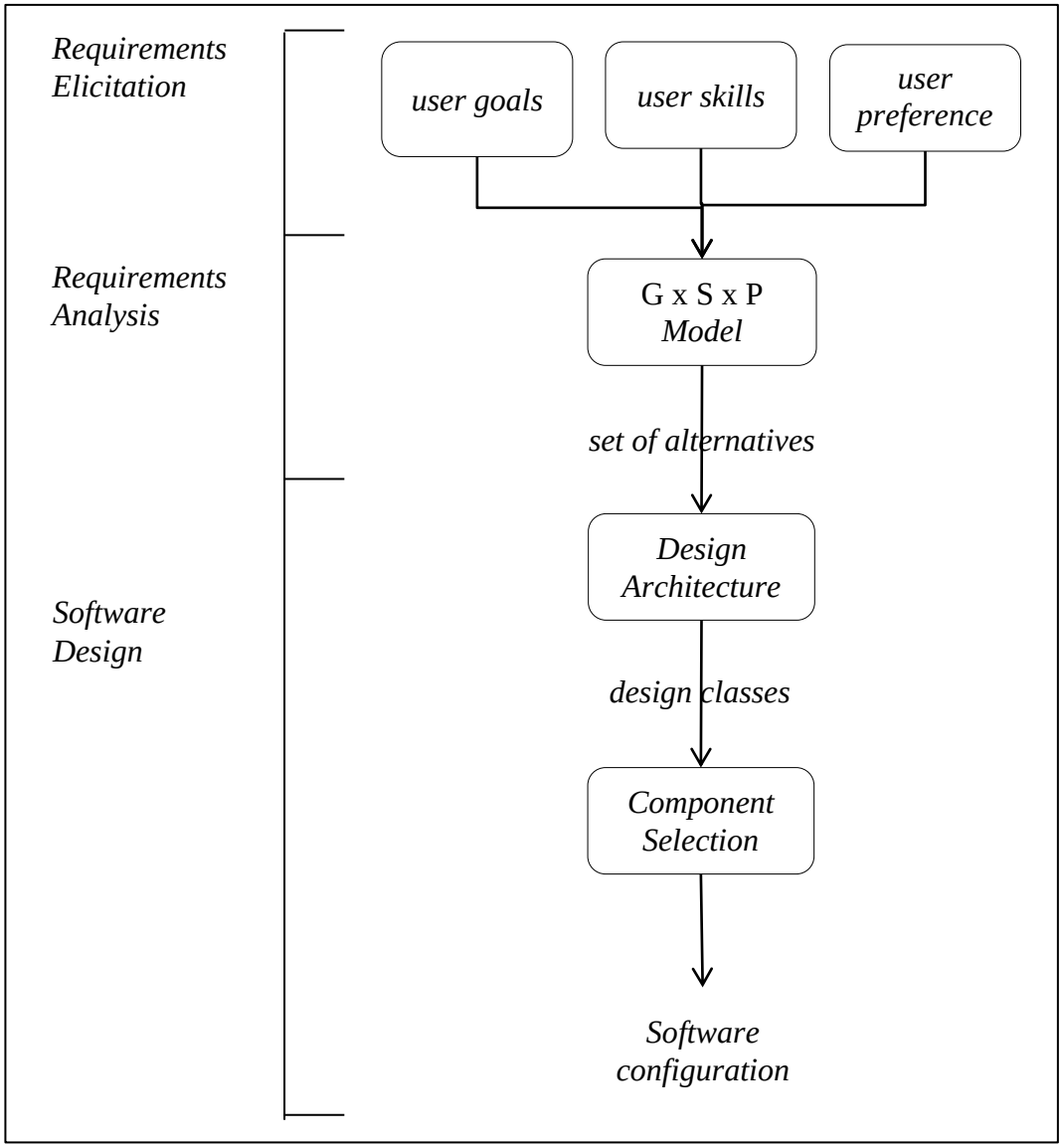
2. *Potential conflict* yaitu hubungan yang menggambarkan pada kondisi tertentu, jika sebuah *goal* terpenuhi ternyata dapat menyebabkan *goal* yang lainnya tidak terpenuhi. Konflik ini biasanya bisa muncul karena adanya perbedaan sudut pandang dan kepentingan dari entitas yang berhubungan.
3. *Responsibility assignment* yaitu hubungan antara agen dengan sebuah *goal*. Agen yang terhubung tersebut mempunyai tanggung jawab agar *goal* dapat dipenuhi/terpuaskan.

2.1.4.5. *Goal-Skill-Preferences* (GSP)

Salah satu metode yang ada dalam GORE adalah *Goal-Skill-Preferences* (GSP), metode ini memiliki tiga persyaratan utama, yaitu *user goal*, *user skill*, dan *user preference*. Sebuah desain perangkat lunak sangat penting apabila mengambil perpektif tujuan dari *user* yang menggunakannya, hal ini kemudian membentuk pertanyaan mendasar yang menjadi motivasi ketiga komponen tersebut, yaitu: (Hui, dkk., 2003: 2)

1. Apa yang harus ditampilkan sistem untuk membuat *user* puas dengan tujuan yang pasti?
2. Apa saja kemampuan yang harus dimiliki *user* untuk menjalankan sistem?
3. Bagaimana sistem perangkat lunak mengutamakan tujuan yang lebih diinginkan *user*?

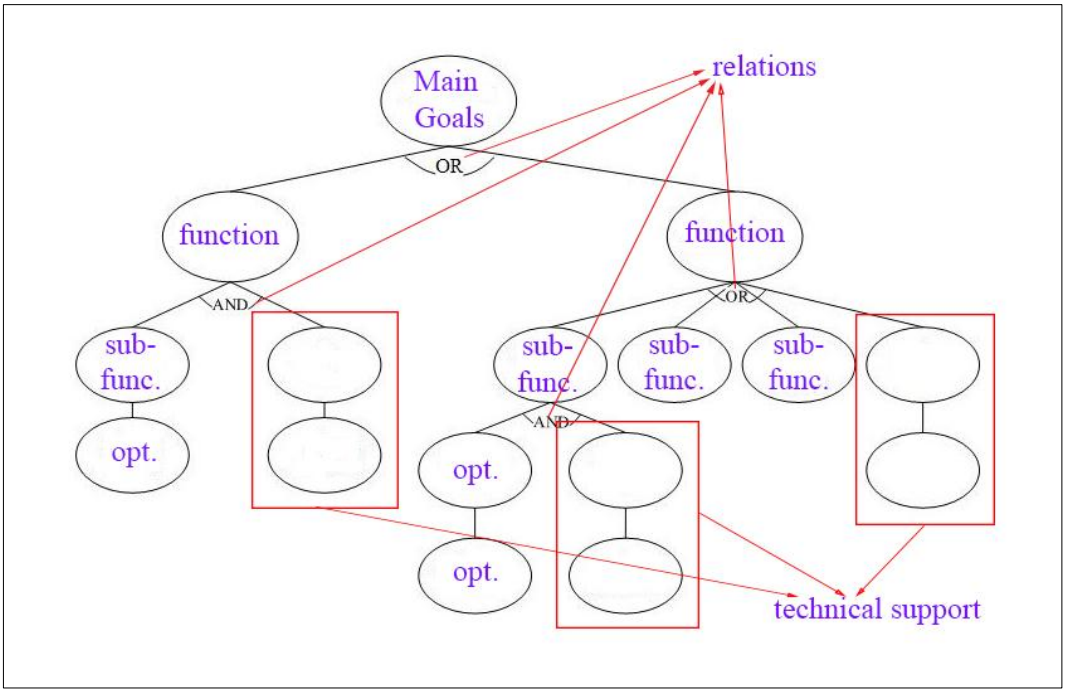
Secara umum, metodologi GSP dapat dilihat pada Gambar 2.6. berikut ini:



Gambar 2.6. Metodologi GSP

Berdasarkan Gambar 2.6., langkah pertama dalam memodelkan *user goal* adalah mengidentifikasikannya menggunakan teknik *elicitation*, seperti melakukan fokus *group discussion* maupun kuesioner. Masing-masing *goal* direpresentasikan sebagai *node* atau simpul dalam *goal graph* dan dapat didekomposisi menjadi *subgoal* sesuai kebutuhan. *Subgoal* dari *parent* yang sama dapat direlasikan dengan *AND* yang mengindikasikan bahwa masing-masing *subgoal* tersebut harus dipenuhi untuk mendukung *goal parent*. *Subgoal* juga

dapat direlasikan dengan *OR* yang mengindikasikan bahwa *goal parent* dapat terpenuhi dengan salah satu atau lebih *subgoal*-nya (Hui, dkk., 2003: 2). *Main goal* merupakan level tertinggi pada *goal graph*, *main goal* akan diturunkan menjadi beberapa *subgoal* yang berisi fungsi-fungsi dan sub fungsi, terdapat pula *option* dan *technical support* sebagai tambahan pendukung apabila tertera pada *interface* sistem aplikasi. Berikut Gambar 2.7. tentang *goal graph*:



Gambar 2.7. *Goal Graph*

Komponen kedua dalam pemodelannya adalah mengidentifikasi sekumpulan kebutuhan *skill* untuk masing-masing fungsi dan menggunakannya untuk membuat beberapa *alternative* desain (Hui, dkk., 2003: 3). *Skill* merupakan kemampuan seseorang yang akan menentukan kecenderungannya dalam menggunakan sistem. Dalam pembuatan sebuah sistem, penyesuaian fitur sistem dengan *user skill* sangatlah penting, hal ini akan mendukung keberhasilan sistem dalam mencapai tujuan pembuatannya.

Komponen ketiga adalah *user preference*. Selama *elicitaion* dari *user goal*, yang diidentifikasi tidak hanya *goals* penting (diperlukan), tetapi juga bagaimana *goal* tersebut dapat dicapai, kenapa hal tersebut penting dan skenario apa yang membuat hal itu penting (Hui, dkk., 2003: 5). Pada tahap *User Preference* akan terlihat *task* apa saja yang lebih dominan dilakukan *user* dengan suatu *skill* tertentu, contoh sederhananya adalah apabila seorang yang memiliki *skill* mengetik lebih cepat, maka orang tersebut akan lebih sering berkomunikasi dengan menggunakan fitur *texting* dibandingkan dengan menggunakan *voice note* atau *video call*. *Preferences* akan menjadi "*softgoal*" dengan memberi identifikasi kontribusi positif atau negatif pada *leaf goal*.

Tujuan utama pada metode ini adalah untuk menemukan dan mendokumentasikan *alternative* yang dapat menampung satu set *user goal* secara sistematis. Setelah tahap *elicitation*, satu set *user goal* yang dihasilkan tersebut akan dijadikan sebuah dimensi mendasar dalam merumuskan masalah di tahap *requirements analysis*. Metode akan mengambil *requirements* sebagai *input* dan menghasilkan satu set peringkat untuk kemudian ke tahap *design*. *Alternative* yang dimaksud merupakan satu set *tasks* yang memenuhi satu set *goal* dari sistem perangkat lunak. Setiap *task* yang sesuai dengan seperangkat komponen perangkat lunak akan membentuk arsitektur tertentu. Arsitektur dan pemilihan komponen pada akhirnya akan membentuk sebuah konfigurasi perangkat lunak. Akan ditemukan *requirements* dari hasil *reverse* sebuah sistem perangkat lunak yang siap pakai. Dan dari sinilah para *actor* seperti *programmer*, *developer*, *user*, *stakeholder*, dan *customer* dapat melakukan analisis perbaikan dan inovasi pengembangan sistem perangkat lunak.

Setelah diuraikan beberapa metode GORE di atas, dapat disimpulkan bahwa model GORE memiliki banyak keuntungan, diantaranya:

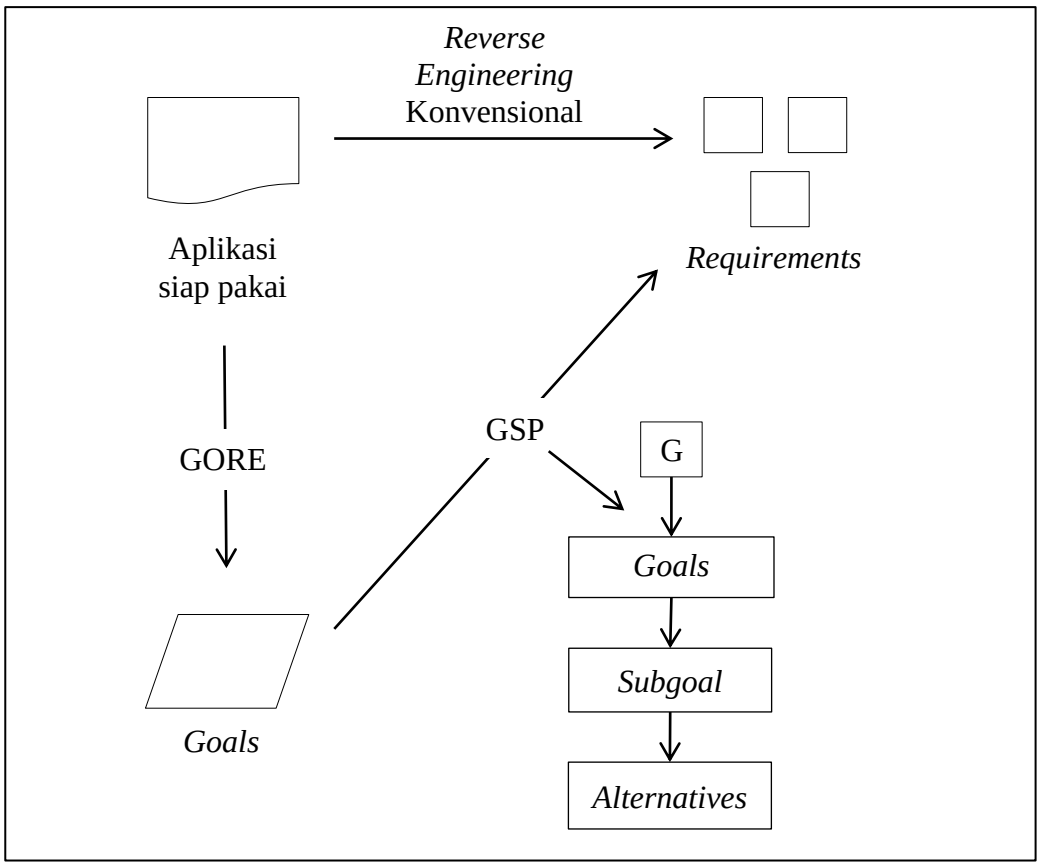
1. Model obyek dan *requirements* dapat diturunkan secara sistematis dari *goal*.
2. *Goal* memberikan dasar pemikiran bagi *requirements*.
3. Grafik *goal* menyediakan penelusuran vertikal dari *high-level strategic concerns* ke *low-level technical details*, hal ini memungkinkan adanya perkembangan versi dari sistem yang dipertimbangkan untuk kemudian diintegrasikan sebagai *alternative* ke dalam satu metode tunggal.
4. Grafik goal *AND*/*OR* memberikan tingkat abstraksi yang tepat di mana pembuat keputusan dapat terlibat untuk keputusan-keputusan penting.
5. Struktur penyempurnaan *goal* menyediakan struktur yang dapat dipahami untuk pembuatan dokumen *requirements*.
6. Perbaikan *goal alternative* dan tugas agen memungkinkan proposal sistem *alternative* untuk dieksplorasi.
7. Formalisasi *goal* memungkinkan perbaikan harus dibuktikan dengan benar dan lengkap.

Jadi, GORE adalah suatu model pendekatan *Requirements Engineering* yang merasionalisasikan *requirements* berdasarkan tujuan-tujuan yang ada pada suatu sistem perangkat lunak.

2.2. Kerangka Berpikir

Berdasarkan kajian teori yang telah diuraikan di atas, permasalahan kurang efektifnya proses *reverse engineering* masih belum sepenuhnya terselesaikan. Sistem aplikasi dapat di-*reverse* dengan pendekatan konvensional, namun terkadang *requirements* yang dihasilkan tidak sesuai, kerap kali terjadi *missing requirements* atau *redundant* (pengulangan *requirements* yang bermaksud sama). Oleh karena itu, diperlukan alat bantu yang dapat me-*reverse* sistem perangkat lunak bukan hanya untuk menemukan tujuan sebenarnya yaitu sesuai kebutuhan *user*, akan tetapi agar tidak hanya *programmer* dan *developer internal* saja yang terlibat dalam perbaikan dan pengembangan sistem perangkat lunak, diharapkan *actor* lain seperti *stakeholder*, *user*, dan *customer* juga ikut andil dalam memberikan evaluasi dan inovasi sebuah sistem perangkat lunak. Model yang dibangun merupakan salah satu alat bantu *reverse engineering* dengan menggunakan metode GSP, dimana model ini berorentasi *goal* dan *actor* sehingga proses *reverse* disesuaikan dengan tujuan pembuatan sistem perangkat lunak dan *actor* yang terlibat dalam sistem perangkat lunak tersebut. Langkah pertama yang dilakukan adalah melihat tampilan aplikasi dan mengumpulkan *main goal* dari aplikasi tersebut, kemudian setelah itu masuk pada tahapan metode yaitu membuat *goal graph* dengan memperhatikan relasi *AND* dan *OR*. *Subgoal* dari *parent* yang sama dapat direlasikan dengan *AND* yang mengindikasikan bahwa masing-masing *subgoal* tersebut harus dipenuhi untuk mendukung *goal parent*. *Subgoal* juga dapat direlasikan dengan *OR* yang mengindikasikan bahwa *goal parent* dapat terpenuhi dengan salah satu atau lebih *subgoal*-nya. Setelah menganalisis *goal* dan *subgoal* melalui *goal graph*, kemudian buat *alternative*

sebanyak kemungkinan *task* yang dapat dilakukan untuk mencapai *goals* tersebut. *Alternatives* tersebut juga direpresentasikan dalam bentuk *goal graph*. Kemudian, dari *alternative* itulah dapat diambil beberapa *requirements* yang menjadi dasar perancangan sebuah sistem. Adapun sistem perangkat lunak yang dijadikan sampel untuk membangun model GORE dalam penelitian ini adalah sistem manajemen Penmaba UNJ 2015 Modul Admin, sistem tersebut dipilih karena memenuhi syarat yaitu memliki tingkat kompleksitas yang menengah, sistem terdiri dari delapan fungsi utama, dan masing masing fungsi memiliki sub fungsi yang cukup banyak. Berikut merupakan gambar kerangka berpikir yang ditunjukkan oleh Gambar 2.8.:



Gambar 2.8. Kerangka Berpikir

BAB III

METODE PENELITIAN

3.1. Tempat dan Waktu Penelitian

Penelitian dilakukan di laboratorium komputer Jurusan Teknik Elektro Universitas Negeri Jakarta, dimulai dari bulan Agustus 2015 sampai bulan Desember 2015.

3.2. Literatur Penelitian

Perancangan model pendekatan *Goal Oriented Requirements Engineering* (GORE) dengan menggunakan metode *Goal-Skill-Preferences* (GSP) ini untuk membantu *software developer* dalam melakukan proses *reverse engineering*. Adapun yang menjadi acuan dalam membangun model pendekatan adalah beberapa literatur yang berhubungan dengan teori *reverse engineering*, GORE, dan GSP sebagai berikut:

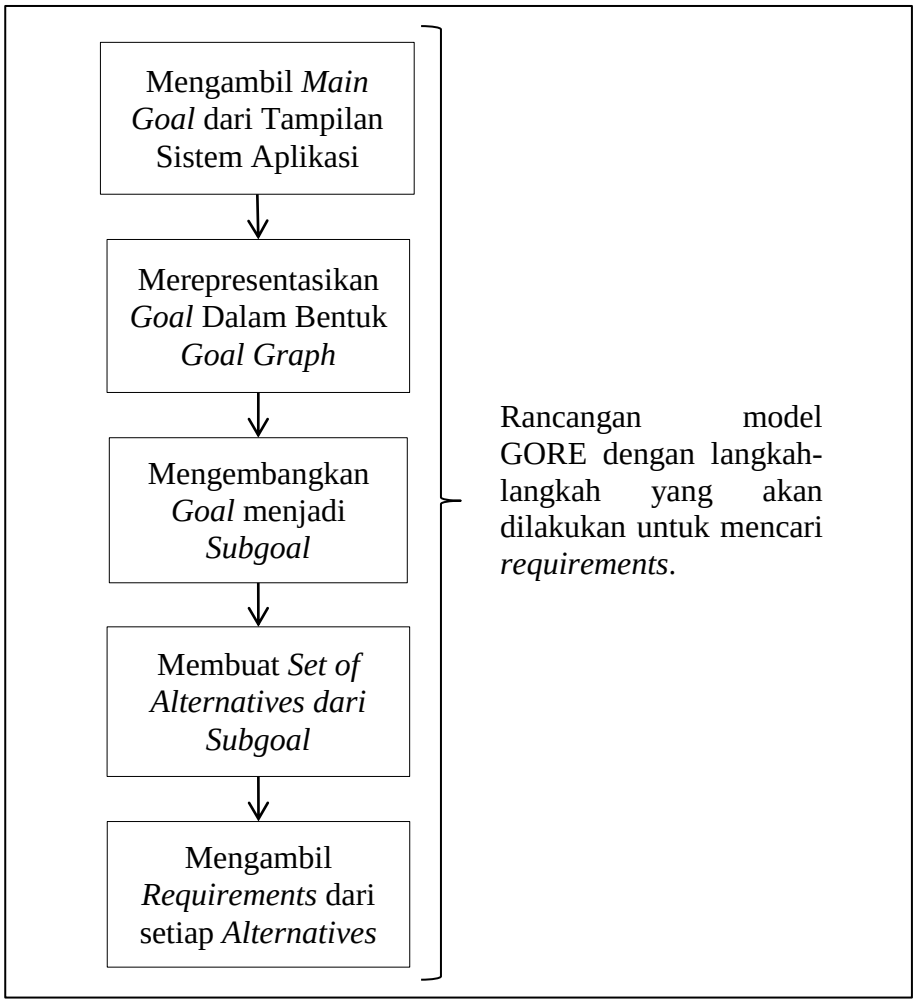
1. Adikara, F.; Sitohang, B.; Hendradjaya, B. 2013. *Penerapan Goal Oriented Requirements Engineering (GORE) Model (Studi Kasus: Penghygembangan Sistem Informasi Penjaminan Mutu Dosen (SIPMD) Pada Institusi Pendidikan Tinggi*, Seminar Nasional Informasi Indonesia.
2. Anwer, S. & Ikram, N. 2006. *Goal Oriented Requirement Engineering: A Critical Study of Techniques*, XIII ASIA PACIFIC SOFTWARE ENGINEERING CONFERENCE (APSEC'06).

3. Hui, B.; Liaskos, S.; & Mylopoulos, J. 2003. *Requirements Analysis for Customizable Software: A Goals-Skills-Preferences Framework*. Department of Computer Science, University of Toronto.
4. Lapouchnian, Alexei. 2005. *Goal-Oriented Requirements Engineering: An Overview of the Current Research*. Department of Computer Science, University of Toronto.
5. Santosa, Stefanus. 1994. *Reverse Engineering: Observasi – Struktur Berjenjang Terhadap Program Sumber Fortran [tesis]*. Jakarta: Program Pascasarjana, Universitas Indonesia.
6. Shofi, Imam M. & Budiarjo, Eko K. 2011. *Ontologi OWL untuk merepresentasikan Framework GSP pada GORE*. KNTIA 2011.
7. Singhal, Aabhas & Gandhi, Shlok. 2014. *Reverse Engineering*. India: International Journal of Computer Applications.
8. Van Lamsweerde, Axel. 2001. *Goal-Oriented Requirements Engineering: A Guided Tour*. De‘partement d’lngbnierie Informatique, Universite‘ catholique de Louvain.

3.3. Metode Penelitian

Metode penelitian yang digunakan pada penelitian ini adalah metode penelitian kualitatif. Menurut Creswell (2013: 140), metode penelitian didefinisikan sebagai suatu pendekatan atau penelusuran untuk mengeksplorasi dan memahami suatu gejala sentral. Ciri khas dari metode penelitian kualitatif adalah induktif. Cara induktif biasanya mulai mengobservasi sasaran penelitian secara rinci menuju generalisasi dan ide-ide yang abstrak. Tujuan dari cara indukti

yaitu untuk menemukan pola-pola atau tema-tema hasil analisa data yang diperoleh. Aspek lain yang mencrikkhaskan metode kualitatif adalah fleksibilitasnya. Fleksibilitas berarti terbuka terhadap keadaan yang selalu berubah dan memungkinkan perolehan pengertian yang mendalam. Hal lain yang penting dalam metode kualitatif yaitu bahwa datanya diperoleh dari tangan pertama dan berupa pengalaman langsung dari pasrtisipan (Raco, 2010: 59-60). Berikut merupakan gambar tahap metode penelitian:



Gambar 3.1. Tahap Metode Penelitian

Dalam penelitian ini, sebuah sistem aplikasi siap pakai akan digunakan sebagai sampel untuk menemukan model pendekatan yaitu sistem manajemen

Penmaba UNJ 2015 Modul Admin, sampel tersebut digunakan untuk merancang model *reverse engineering*. Sistem aplikasi tersebut akan di-*reverse* dengan menggunakan konsep model pendekatan GORE dengan metode GSP. Gambar 3.1. merepresentasikan tahap metode penelitian. Langkah awal adalah mengumpulkan *main goal* dengan mengamati tampilan sistem aplikasi tersebut. Setelah itu, melakukan *goal analysis* dengan merepresentasikannya ke dalam bentuk *goal graph*, kemudian *goal analysis* tersebut akan dikembangkan menjadi beberapa *subgoal*. Dari *subgoal* tersebut, muncul beberapa *alternative* yang menjadi acuan untuk menemukan *requirements*. Rancangan model GORE dibentuk dengan langkah-langkah yang akan dilakukan selama proses *reverse engineering*.

3.3.1. Mengambil *Main Goal* dari Tampilan Sistem Aplikasi

Metode GSP berpatokan pada tiga komponen penting pada sistem aplikasi yaitu, *goal*, *skill*, dan *preferences*. Akan tetapi dalam penelitian ini hanya mengambil komponen *goal* saja, hal ini dikarenakan pada dasarnya *functional requirements* dapat dihasilkan tanpa adanya peran dari komponen *skill* dan *preferences*, komponen *skill*, dan *preferences* lebih kepada menghasilkan *non-functional requirements*. Oleh karena itu, *goal analysis* menjadi tahap awal dalam me-*reverse* suatu sistem aplikasi menggunakan model GORE, *goal analysis* dibutuhkan untuk mengetahui *goal* apa saja yang disediakan oleh sistem aplikasi dalam memenuhi kebutuhan *stakeholder*. Untuk tahap awal, *goal* yang diambil dari tampilan sistem aplikasi hanya *main goal*-nya saja, *main goal* merupakan tujuan inti dari pembuatan sistem untuk memenuhi kebutuhan *stakeholder*. *Main*

goal dapat dilihat dari tampilan sistem aplikasi siap pakai berdasarkan fitur yang disediakan pada *interface* sistem aplikasi tersebut.

3.3.2. Merepresentasikan *Goal* Dalam Bentuk *Goal Graph*

Setelah mengambil *main goal*, representasikan *main goal* dalam bentuk *goal graph*. Hal ini bertujuan untuk mengetahui cakupan dari tujuan sistem. *Goal graph* merupakan grafik menyerupai bentuk pohon bercabang yang akan memetakan sejauh mana jangkauan sebuah *goal* pada suatu sistem aplikasi. *Graph* tersebut merupakan hasil dekomposisi dari *main goal*. *Goal* akan membentuk *node* dan *leaf* pada *goal graph* yang akan dirinci lagi menjadi *subgoal*.

3.3.3. Mengembangkan *Goal* Menjadi *Subgoal*

Langkah selanjutnya adalah mengembangkan *goal* tersebut menjadi beberapa *subgoal*. Dalam satu *main goal* dapat menghasilkan beberapa *subgoal* yang merupakan cabang dari *graph* yang dimiliki oleh *goal* tertentu. *Subgoal analysis* dihasilkan dari perluasan *main goal* yang sebelumnya sudah dijelaskan, *subgoal analysis* secara sederhana memperluas jangkauan *goal* hingga sampai kepada rincian *task* yang dapat dilakukan oleh *stakeholder*. *Subgoal* dari *parent* yang sama akan membentuk hubungan *AND* apabila semua *subgoal* harus terpenuhi agar tujuan dari *goal parent* juga terpenuhi, sedangkan *subgoal* akan membentuk hubungan *OR* apabila salah satu atau lebih dari *subgoal* tersebut harus terpenuhi apabila *goal parent* ingin terpenuhi. Proses dekomposisi pada *subgoal* akan berhenti apabila sudah mendapat tujuan yang dapat didelegasikan kepada komponen menjadi sebuah sistem aplikasi.

3.3.4. Membuat *Set of Alternatives* dari *Subgoal*

Dari beberapa *subgoal analysis* yang sudah diuraikan sebelumnya, buat *set alternatives* berdasarkan banyaknya kemungkinan *task* yang dapat dilakukan oleh sistem aplikasi dalam satu *subgoal*. Jumlah *alternatives* pada tiap *subgoal* tidak selalu sama, karena jumlah cabang pada setiap *subgoal* cenderung berbeda. Pada satu *subgoal* dapat menghasilkan lebih dari satu *alternative*. *Alternatives* tersebut masih dalam bentuk *graph*, yang mana nantinya akan menghasilkan *requirements* dari *alternatives* tersebut.

3.3.5. Mengambil *Requirements* dari setiap *Alternatives*

Langkah terakhir yaitu dari *alternatives* tersebut tarik *requirements* berdasarkan alur *goal graph* yang sudah dibuat. Dari satu *alternatives* dapat diambil satu *requirements*, sehingga banyaknya *requirements* yang dihasilkan bergantung pada banyaknya *alternatives* yang muncul pada tahap sebelumnya. *Requirements* yang dihasilkan akan lebih lengkap dan terstruktur karena diambil langsung dari alur *goal graph* yang sudah ada.

BAB IV

HASIL PENELITIAN DAN PEMBAHASAN

4.1. Hasil Penelitian

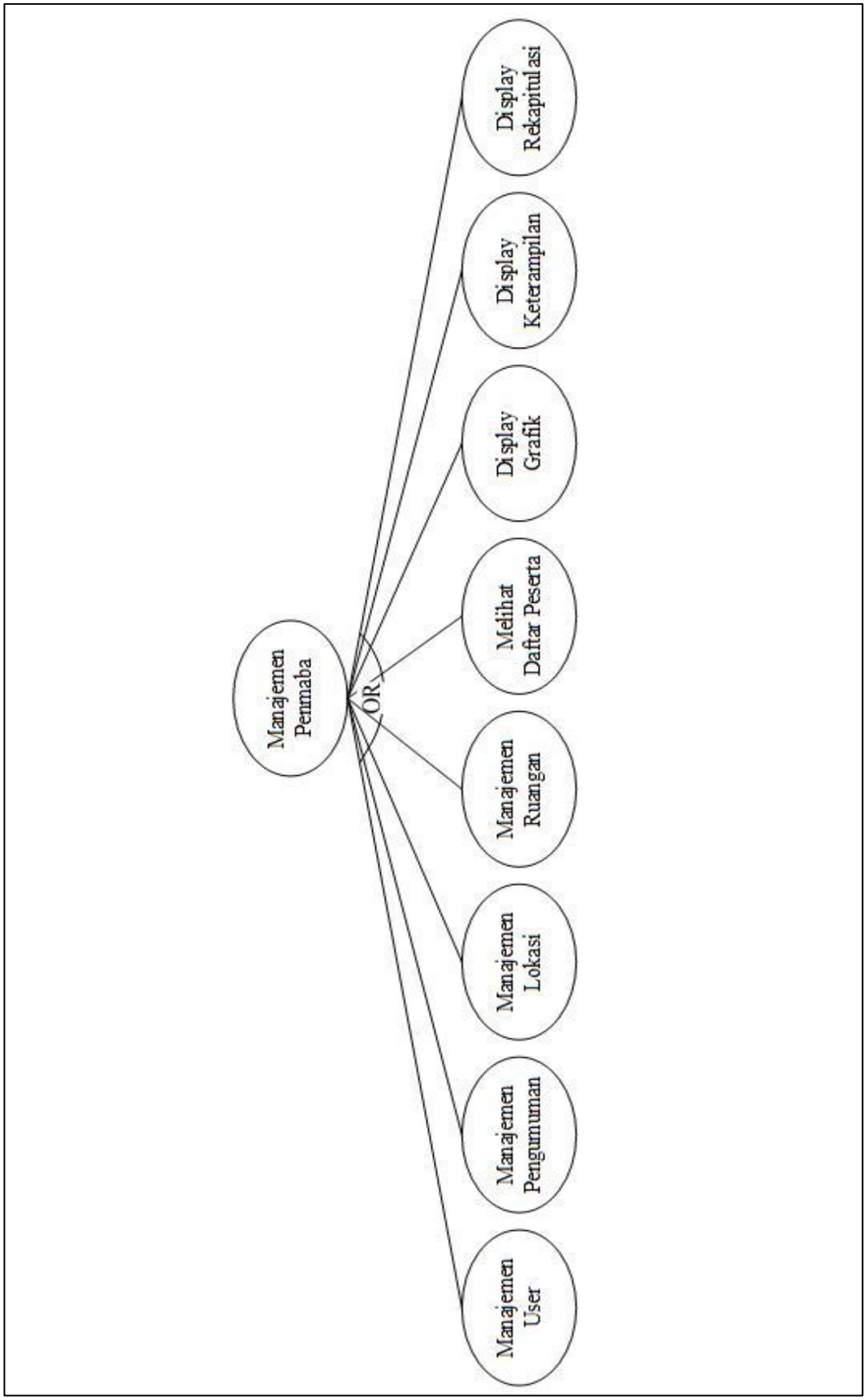
4.1.1. Hasil Pengambilan *Main Goal*

Berdasarkan tampilan aplikasi, dapat diambil beberapa *main goal* dari sistem manajemen Penmaba UNJ 2015 Modul Admin, yaitu sistem dapat melakukan:

1. Manajemen *User*
2. Manajemen Pengumuman
3. Manajemen Lokasi
4. Manajemen Ruangan
5. Melihat Daftar Peserta
6. *Display* Grafik
7. *Display* Rekapitulasi
8. *Display* Keterampilan

4.1.2. Hasil Representasi *Goal* Dalam Bentuk *Goal Graph*

Main goal yang sudah ada dibuat dalam bentuk *goal graph*, berikut merupakan hasil representasi *main goal* dalam bentuk *goal graph*:

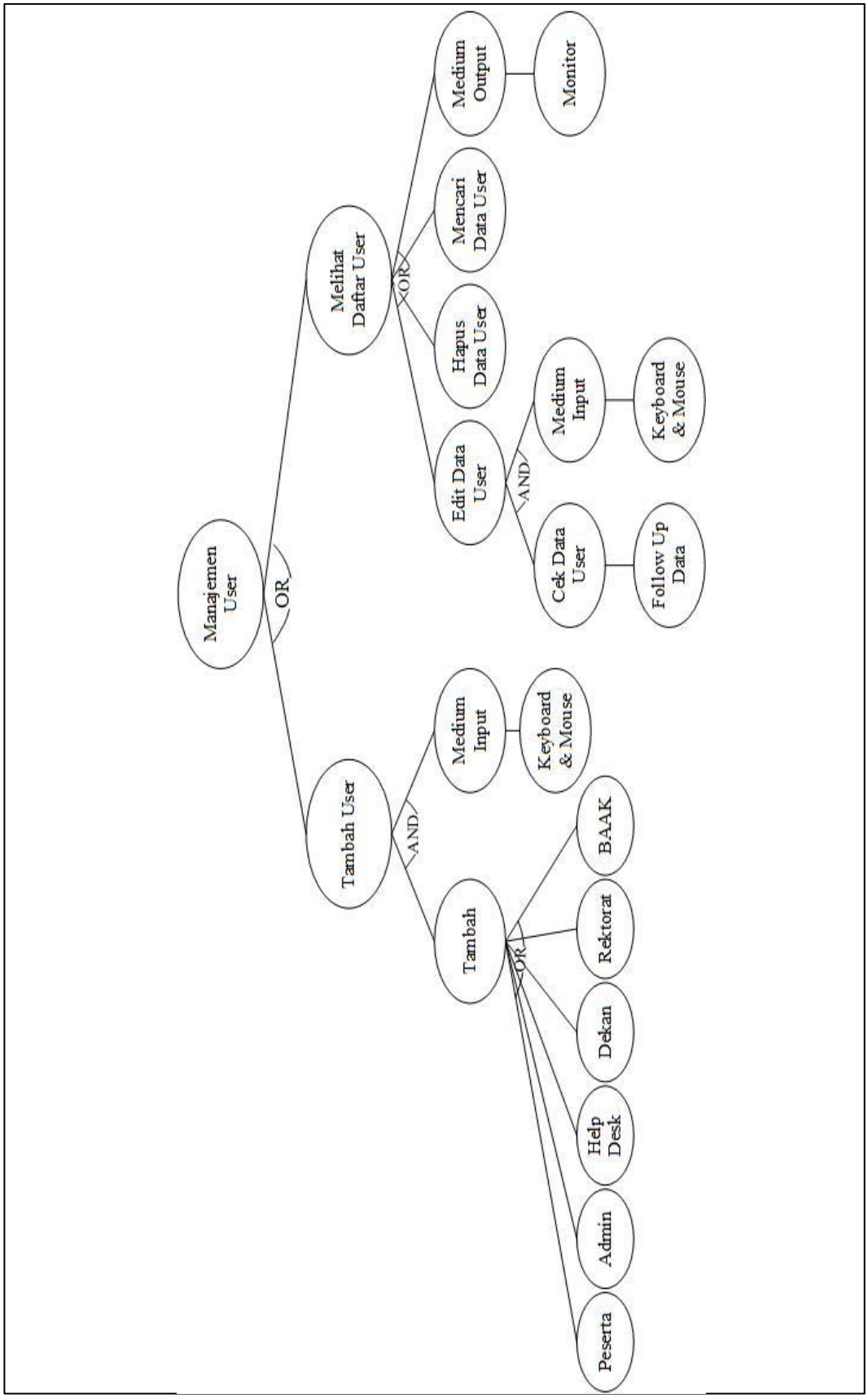


Gambar 4.1. *Goal* Analysis Manajemen Penmaba UNJ 2015 Modul Admin

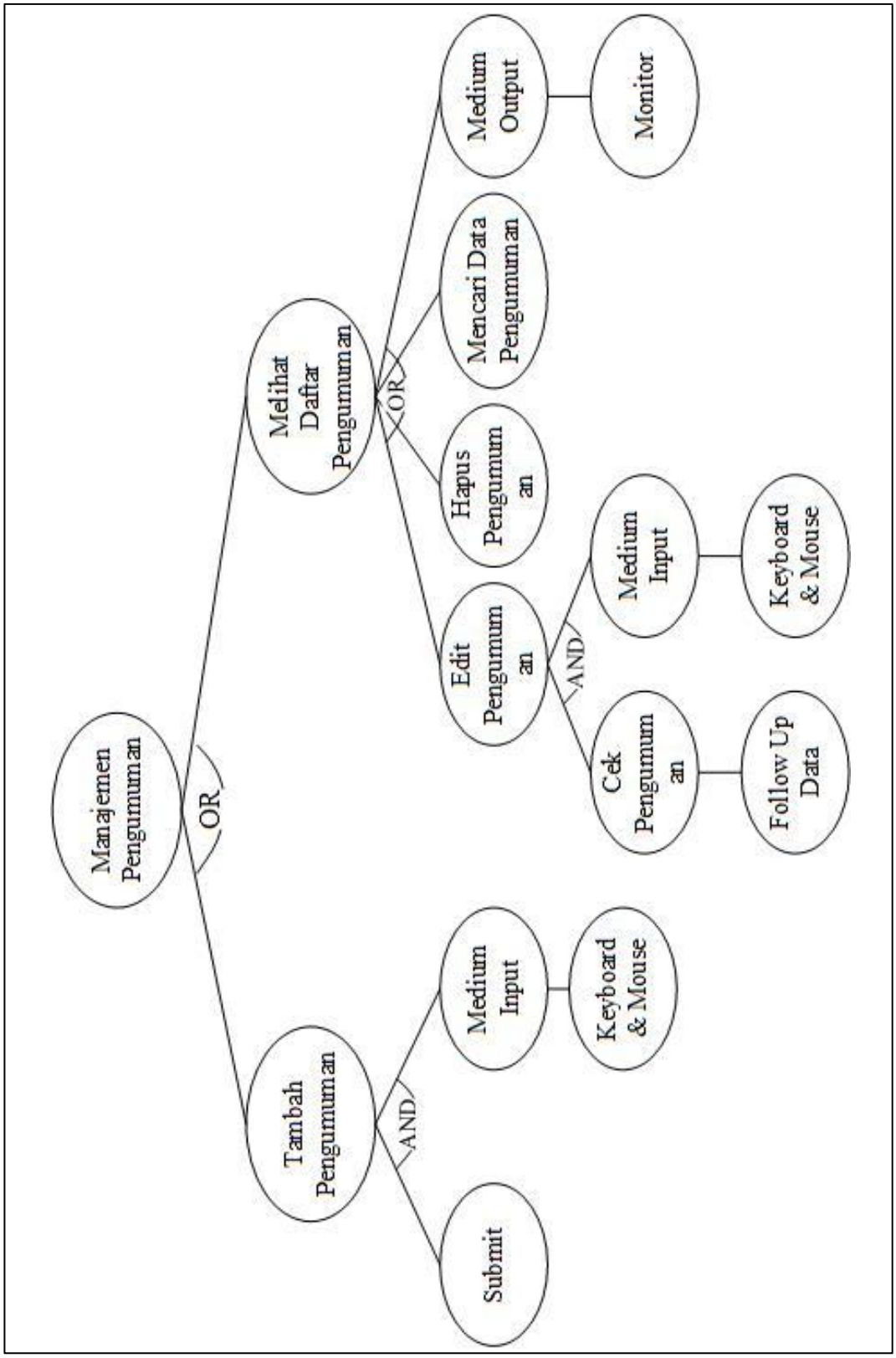
Goal graph pada Gambar 4.1. merupakan hasil *requirements elicitation* yang terlihat pada tampilan aplikasi siap pakai. Dari gambar tersebut dapat diketahui bahwa dalam sistem aplikasi Manajemen Penmaba menyediakan beberapa fitur, di antaranya adalah Manajemen *User*, Manajemen Pengumuman, Manajemen Lokasi, Manajemen Ruangan, Melihat Daftar Peserta, *Display* Grafik, *Display* Rekapitulasi, dan *Display* Keterampilan.

4.1.3. Hasil *Subgoal Analysis*

Goal graph pada sistem aplikasi Penmaba UNJ 2015 Modul Admin menghasilkan beberapa *subgoal*. Gambar 4.2. merupakan salah satu *subgoal* yang dihasilkan dari *goal* Manjemen *User*. Pada *subgoal* ini, terdapat relasi *OR* antara tambah *user* dan melihat daftar *user*, menunjukkan bahwa admin dapat melakukan tambah *user* atau melihat data *user*. Jadi ketika sistem berjalan, admin dapat melakukan salah satu atau kedua fitur tersebut. Admin menambahkan *user* dengan medium *input keyboard* dan *mouse* yang mana memiliki relasi *AND* dengan fitur tambah, sehingga keduanya harus dipenuhi untuk melengkapi syarat dalam melakukan tambah *user*, adapun *user* yang dimaksud yaitu seperti peserta, admin, *help desk*, dekan, rektorat, atau BAAK. Admin juga dapat melihat daftar *user*, yang memungkinkan admin melakukan beberapa fungsi sistem seperti *edit* data *user* dengan mengecek data *user* terlebih dahulu ketika akan meng-*edit* dan data akan di-*follow up* ketika sudah di-*edit*, dalam hal ini medium *input* yang digunakan adalah *keyboard* dan *mouse*. Selain itu admin juga dapat menghapus atau mencari data *user* yang sudah di-*input* sebelumnya dengan medium *output* layar monitor.

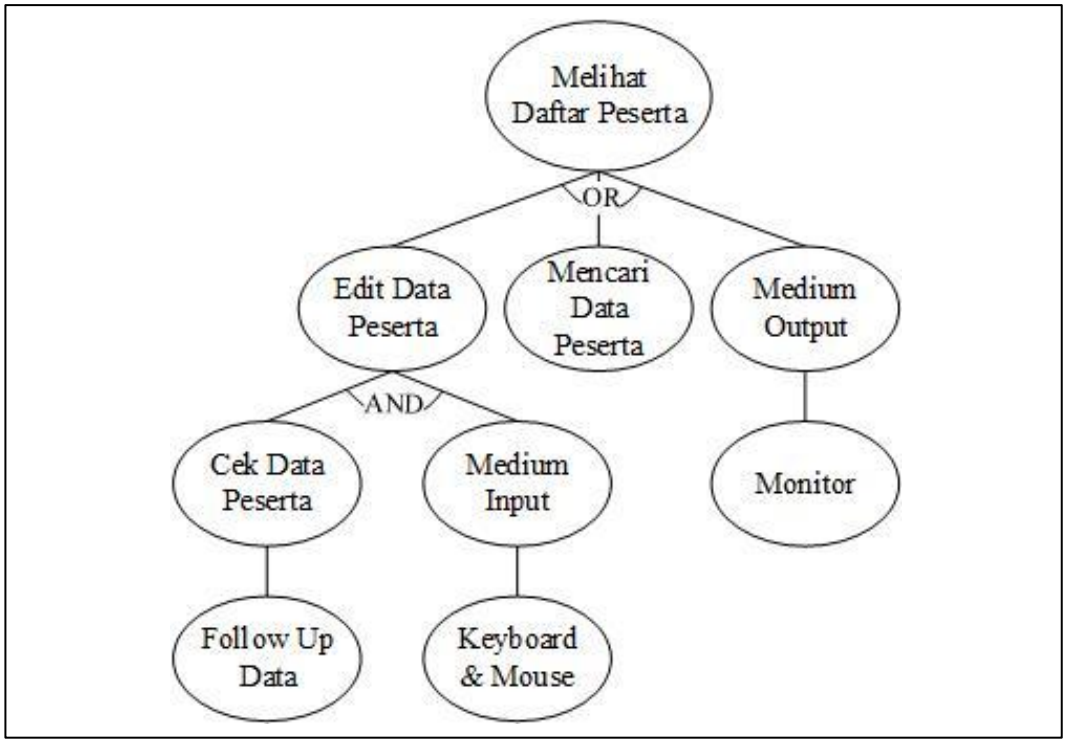


Gambar 4.2. *Subgoal Analysis* Manajemen *User*



Gambar 4.3. *Subgoal Analysis* Manajemen Pengumuman

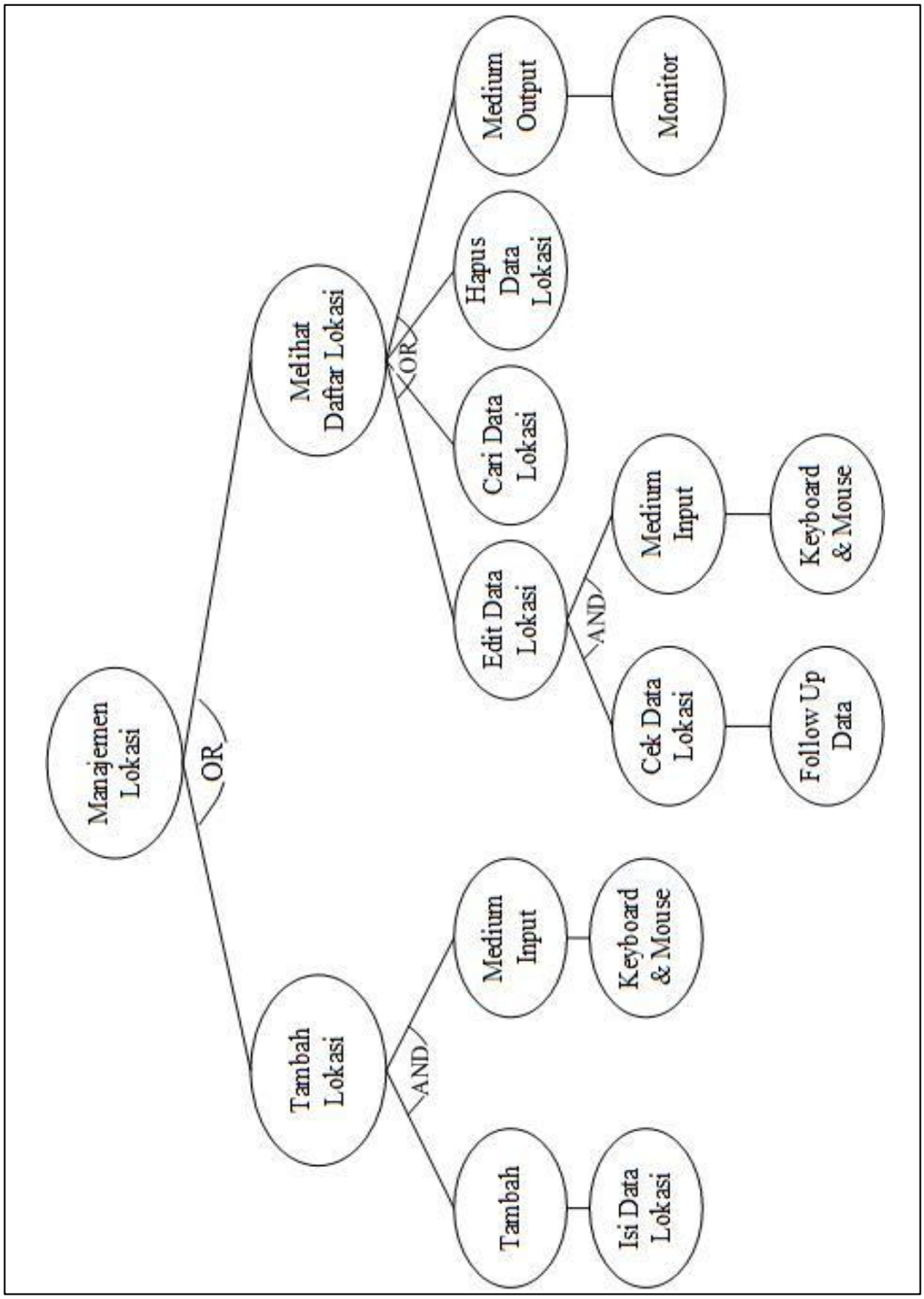
Selain Manajemen *User*, admin juga dapat melakukan Manajemen Pengumuman. Pada Gambar 4.3. dapat dilihat bahwa sama halnya dengan Manajemen *User*, admin memiliki hak akses untuk menambah atau melihat daftar pengumuman. Adapun medium *input* yang digunakan untuk menambah pengumuman adalah *keyboard* dan *mouse*. Pada hak akses melihat daftar pengumuman, admin dapat meng-*edit* pengumuman, menghapus, atau mencari data pengumuman yang sebelumnya sudah di-*input* dengan medium *output* layar monitor. Dalam meng-*edit* pengumuman, admin melakukan evaluasi terlebih dahulu dengan mengecek data pengumuman yang sudah ada, kemudian meng-*edit* data yang perlu di-*edit* dengan medium *input keyboard* dan *mouse*.



Gambar 4.4. *Subgoal Analysis* Melihat Daftar Peserta

Subgoal yang selanjutnya adalah Melihat Daftar Peserta, pada Gambar 4.4. dapat dilihat bahwa admin memiliki hak akses untuk meng-*edit* atau mencari data peserta yang sudah di-*input* dengan medium *output* layar monitor, ketika admin

meng-*edit* data peserta, admin akan melakukan evaluasi dengan mengecek data yang sudah di-*input* dan kemudian data yang sudah di-*edit* akan di-*follow up* oleh sistem, medium *input* yang digunakan adalah *keyboard* dan *mouse*.

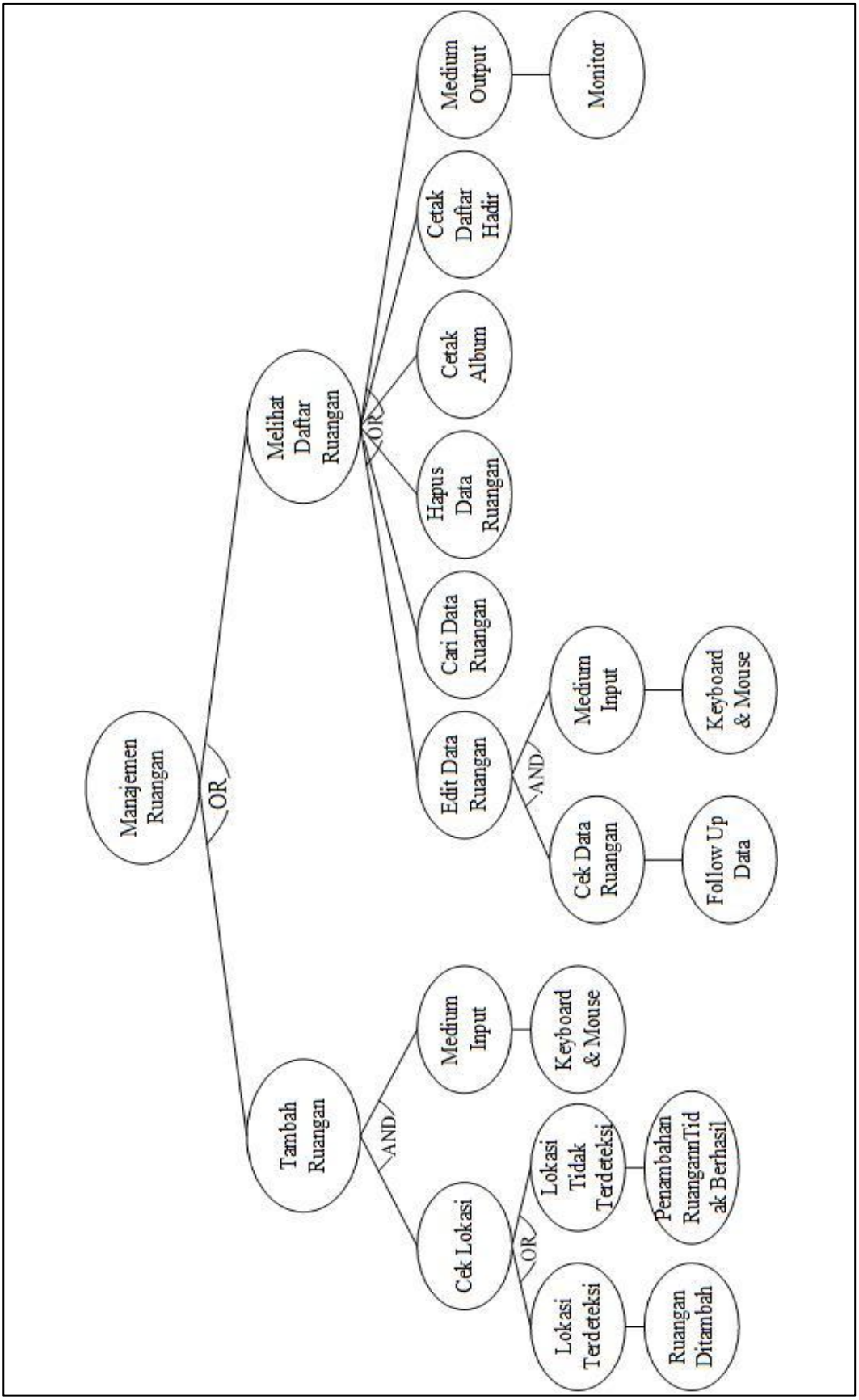


Gambar 4.5. *Subgoal Analysis* Manajemen Lokasi

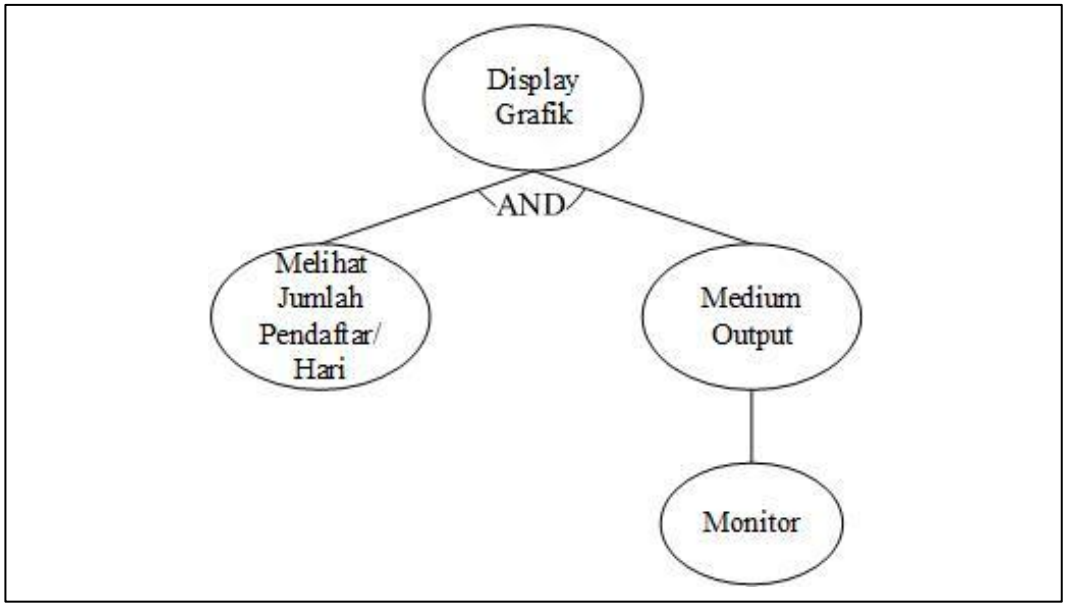
Pada Gambar 4.5. menunjukkan bahwa admin dapat melakukan tambah lokasi atau melihat daftar lokasi dalam lingkup *goal* Manajemen Lokasi. Admin dapat menambahkan lokasi dengan medium *input keyboard* dan *mouse*. sedangkan pada *subgoal* melihat daftar lokasi, admin dapat meng-*edit*, mencari, atau menghapus data lokasi dengan medium *output* layar monitor. Ketika meng-*edit* data lokasi, admin dapat mengecek data lokasi yang sebelumnya sudah di-*input* untuk kemudian di-*follow up* oleh sistem ketika dilakukan pengeditan, adapun medium *input* yang digunakan adalah *keyboard* dan *mouse*.

Untuk *goal* Manajemen Ruangan, terdapat dua *subgoal* yang dihasilkan. Berdasarkan Gambar 4.6., terdapat *subgoal* tambah ruangan dan melihat daftar ruangan yang memiliki relasi *OR*. Pada *subgoal* tambah ruangan, sistem akan mengecek lokasi yang ruangannya akan ditambah, apabila lokasi terdeteksi, maka ruangan berhasil ditambahkan, begitu juga sebaliknya. Medium *input* yang digunakan ketika melakukan tambah ruangan adalah *keyboard* dan *mouse*. Selain tambah ruangan, admin juga dapat melihat daftar ruangan, yang kemudian juga memungkinkan admin meng-*edit*, mencari, serta menghapus data ruangan serta mencetak album dan daftar hadir peserta di ruangan tertentu dengan medium *output* layar monitor.

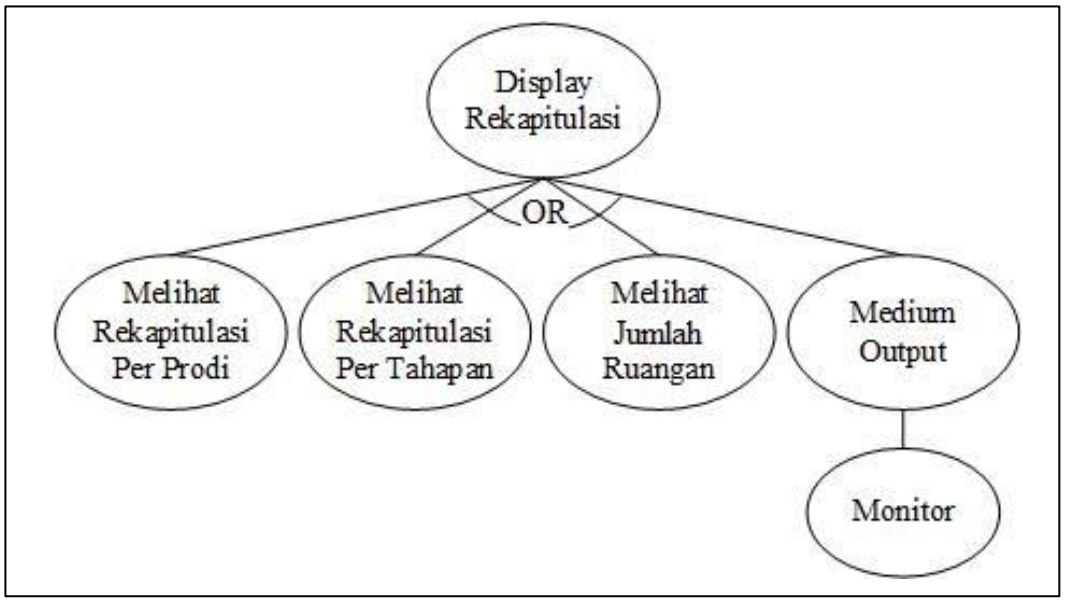
Gambar 4.7. merupakan *subgoal Display* Grafik, pada *subgoal* ini admin hanya dapat melihat jumlah daftar hadir setiap harinya dalam bentuk grafik dengan medium *output* layar monitor.



Gambar 4.6. *Subgoal Analysis* Manajemen Ruangan

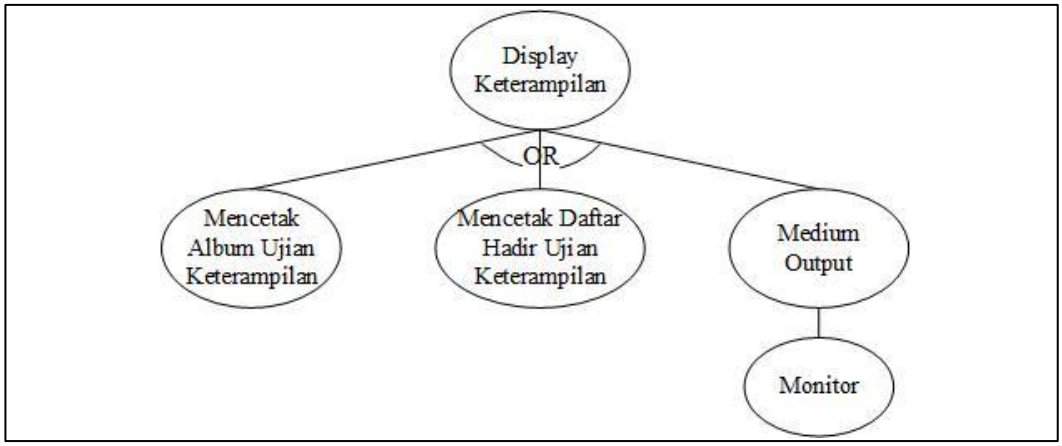


Gambar 4.7. *Subgoal Analysis Display* Grafik



Gambar 4.8. *Subgoal Analysis Display* Rekapitulasi

Subgoal Display Rekapitulasi digambarkan pada Gambar 4.8., dapat dilihat bahwa admin dapat melakukan beberapa hal seperti melihat rekapitulasi per prodi, rekapitulasi per tahapan, atau melihat jumlah ruangan. Kesemuanya dilakukan dengan medium *output* layar monitor.

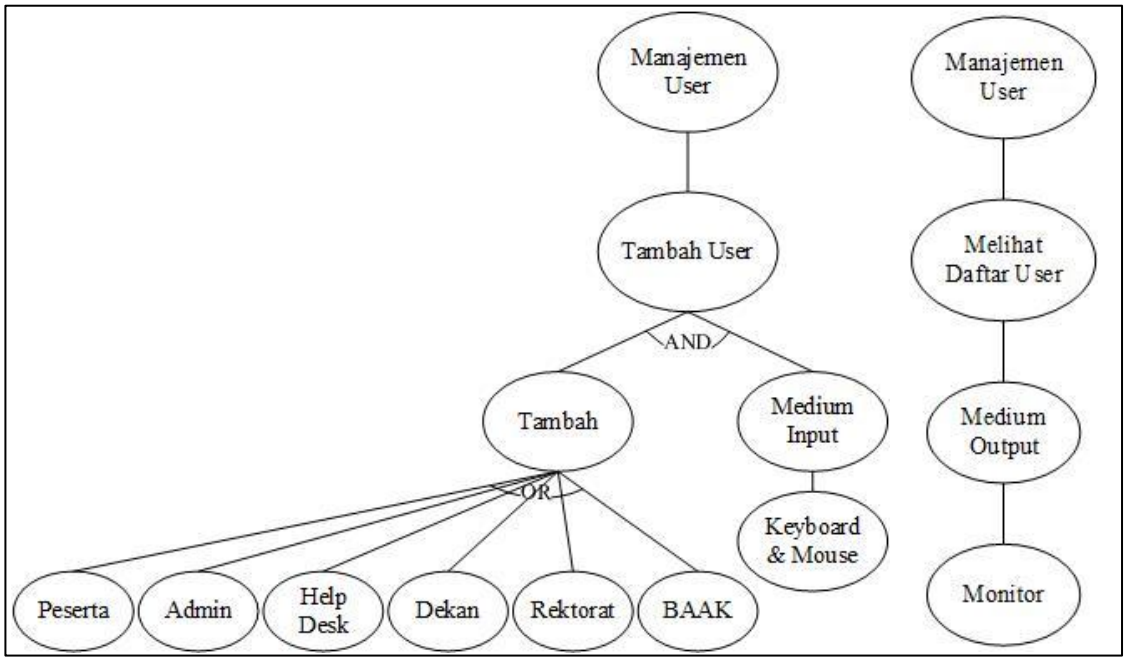


Gambar 4.9. *Subgoal Analysis Display* Keterampilan

Pada Gambar 4.9., *goal Display* Keterampilan menyediakan beberapa *subgoal* yaitu mencetak album ujian keterampilan dan mencetak daftar hadir keterampilan dengan medium *output* layar monitor.

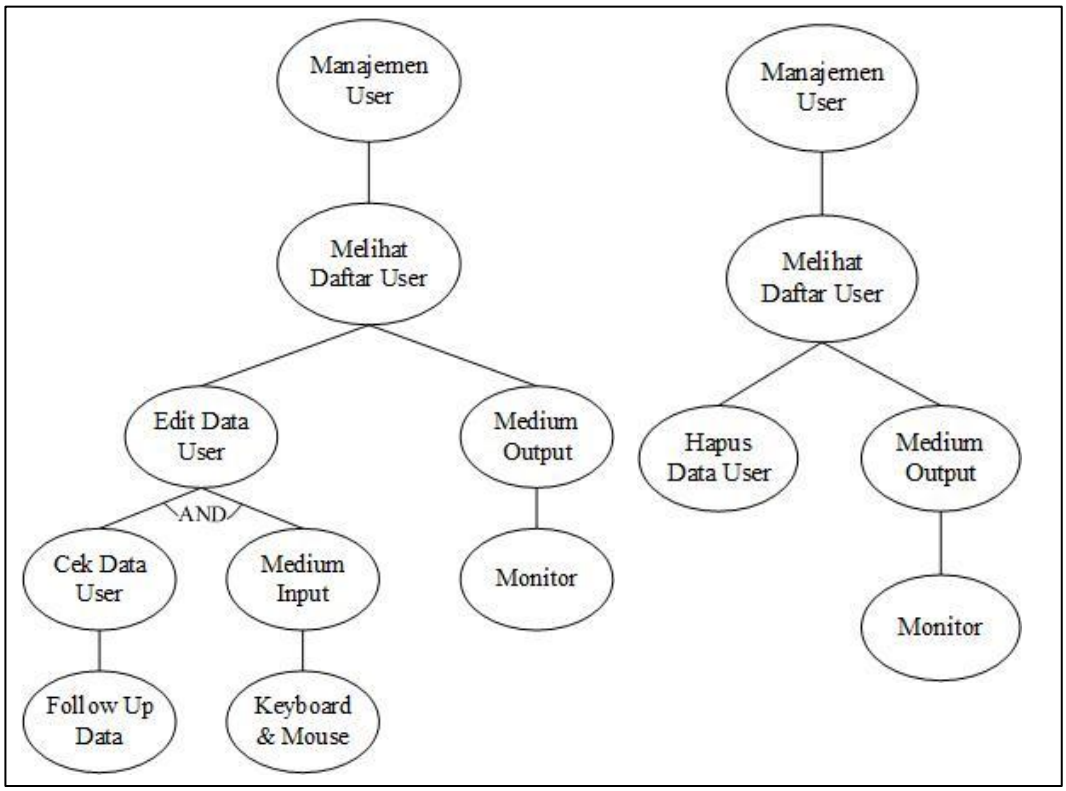
4.1.4. Hasil *Set of Alternatives*

Berikut merupakan beberapa set *alternatives* yang dihasilkan dari *subgoal* yang sudah dibuat pada tahap sebelumnya:



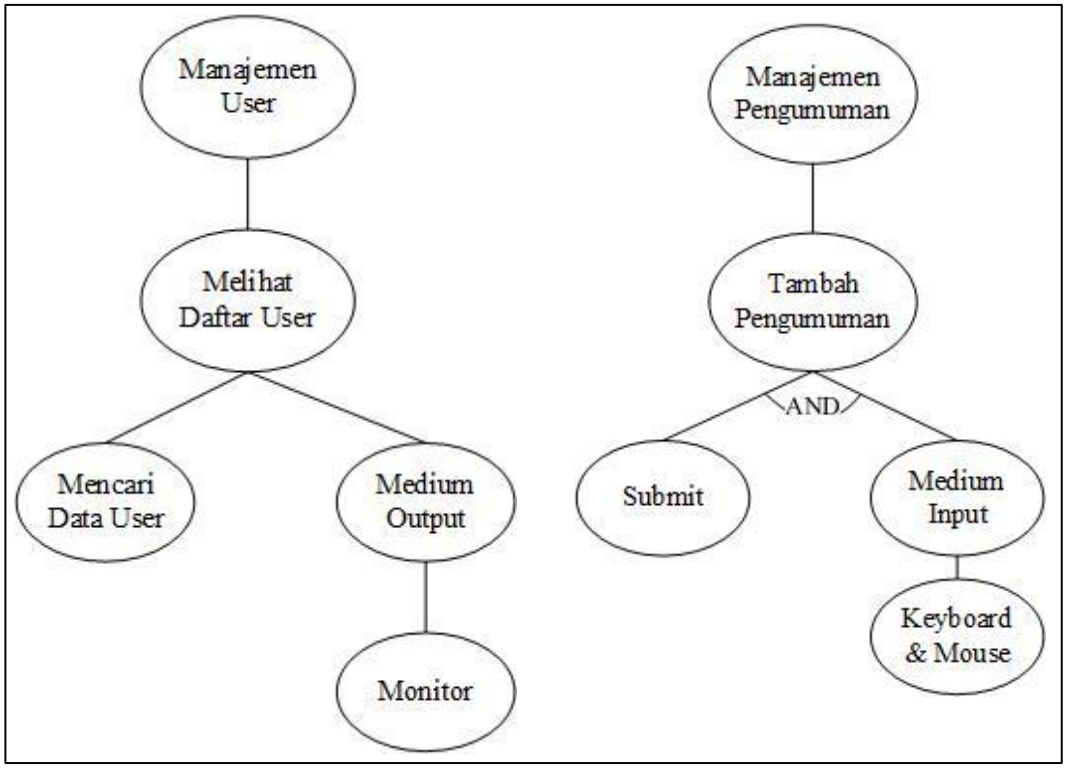
Gambar 4.10. *Alternatives*: 1 (kiri) dan 2 (kanan)

Alternative pertama dan kedua dihasilkan dari *subgoal* Manajemen *User*. Berdasarkan Gambar 4.10., *alternative* 1 (kiri) menunjukkan kemungkinan admin melakukan penambahan *user* dengan medium *input keyboard* dan *mouse*, terdapat relasi *AND* sehingga keduanya harus terpenuhi untuk mencapai *subgoal* tambah *user*. Sedangkan *alternative* 2 (kanan) merupakan kemungkinan *task* dimana admin melihat daftar *user* dengan medium *output* layar monitor.



Gambar 4.11. *Alternatives*: 3 (Kiri) dan 4 (Kanan)

Gambar 4.11. menunjukkan *alternatives* 3 dan 4 yang dihasilkan dari *subgoal* Manajemen *User*, *alternative* 3 (kiri) merupakan *task* dimana admin dapat meng-*edit* data *user* dengan medium *input keyboard* dan *mouse* serta medium *output* layar monitor. Dan *alternative* 4 (kanan) menggambarkan bahwa admin dapat menghapus data admin yang sudah di-*input* sebelumnya dengan medium *output* layar monitor.

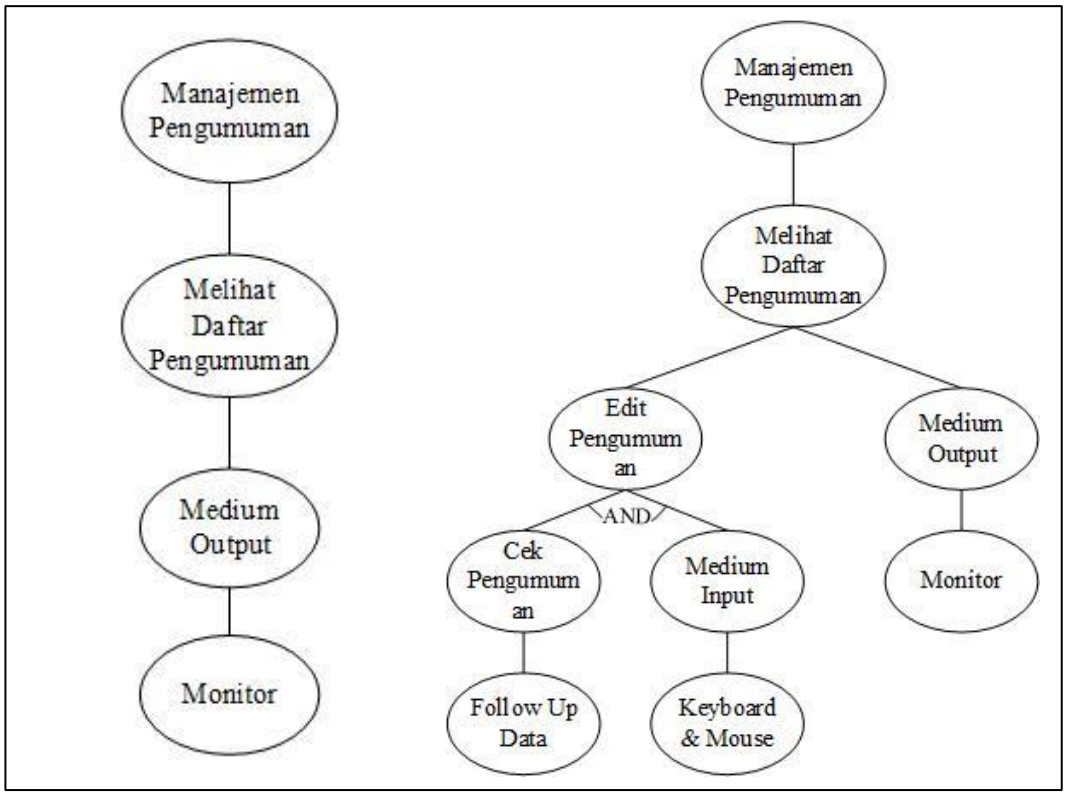


Gambar 4.12. *Alternatives*: 5 (Kiri) dan 6 (Kanan)

Selanjutnya ialah *alternatives* 5 dan 6 yang ditunjukkan pada Gambar 4.12., *alternative* 5 dihasilkan dari *subgoal* Manajemen *User*, sedangkan *alternative* 6 dihasilkan dari *subgoal* Manajemen Pengumuman. Pada *alternative* 5 (kiri) menunjukkan *task* yang dapat dilakukan admin berupa mencari data *user* dengan medium *output* layar monitor. Adapun *alternative* 6 (kanan) menunjukkan admin dapat menambah pengumuman dengan melakukan *submit* pengumuman dengan medium *input keyboard* dan *mouse*.

Alternatives 7 dan 8 ditunjukkan oleh Gambar 4.13., admin dapat melihat daftar pengumuman dengan medium *output* layar monitor dijelaskan pada gambar *alternative* 7 (kiri), sedangkan *alternative* 8 (kanan) menunjukkan admin dapat meng-*edit* pengumuman dengan melakukan evaluasi pengecekan dan kemudian data di-*follow up* ketika sudah melakukan peng-*edit*-an, dan adapun medium

output yang digunakan adalah monitor serta medium *input* untuk melakukan peng-*edit*-an adalah *keyboard* dan *mouse*.

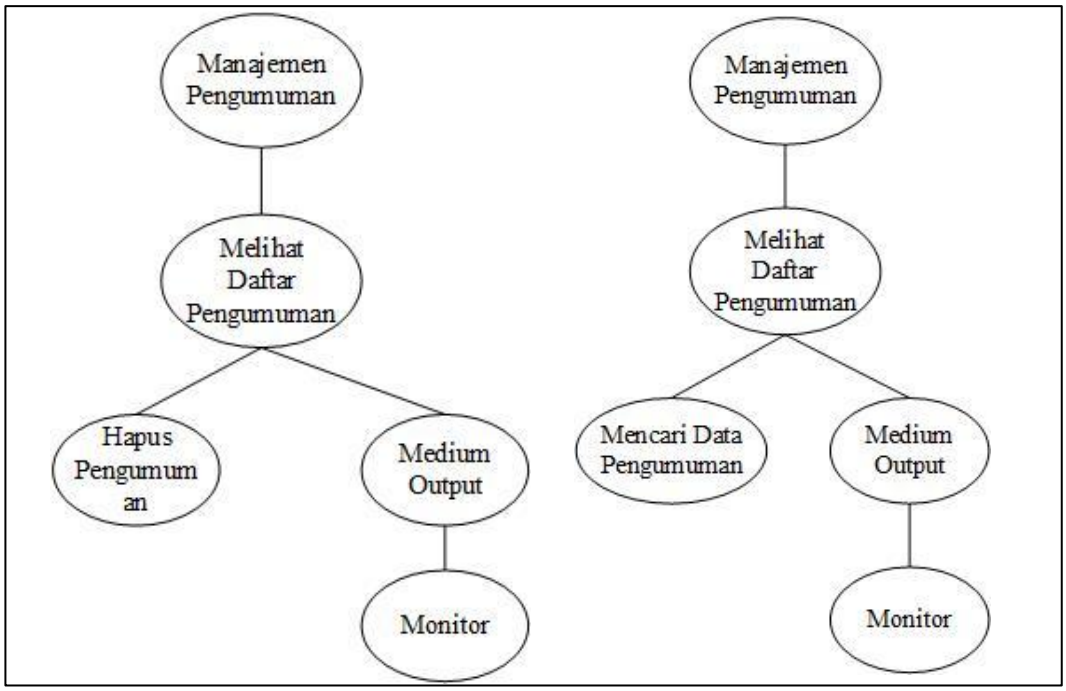


Gambar 4.13. *Alternatives*: 7 (Kiri) dan 8 (Kanan)

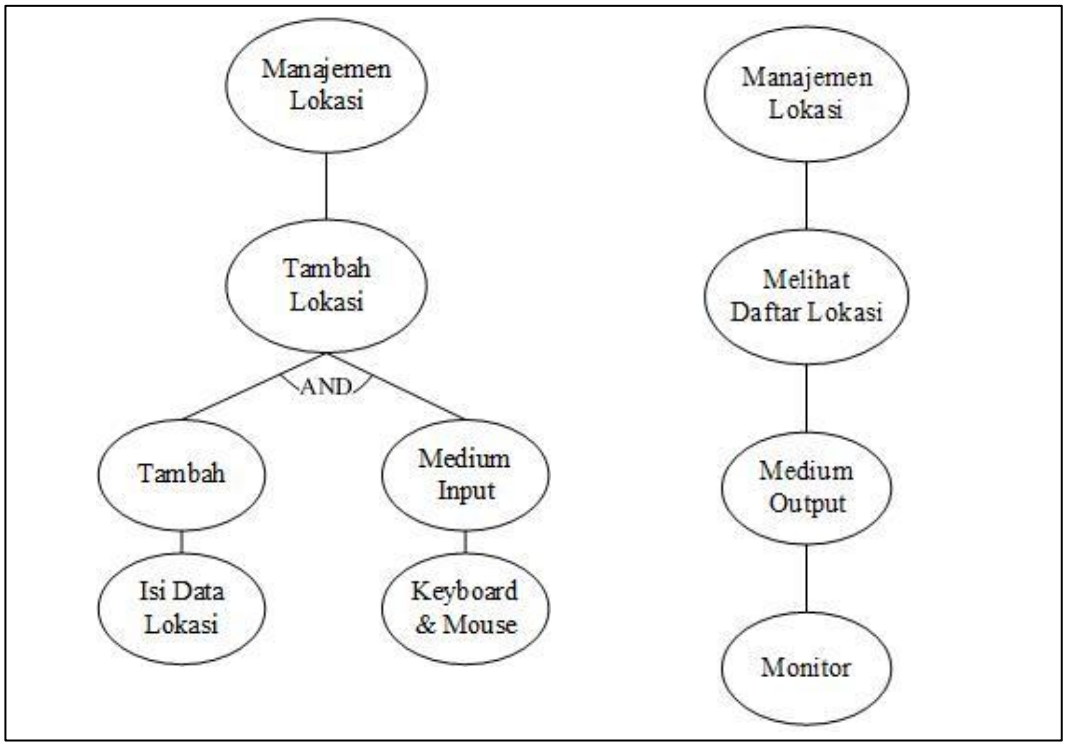
Alternatives 9 dan 10 merupakan kemungkinan *task* yang dihasilkan dari *subgoal* Manajemen Pengumuman. Pada Gambar 4.14. terdapat *alternative* 9 (kiri) yang menunjukkan admin dapat menghapus pengumuman dengan medium *output* layar monitor. Sedangkan *alternative* 10 (kanan) menunjukkan admin dapat melakukan pencarian data pengumuman dengan medium *output* layar monitor.

Gambar 4.15. menunjukkan *alternatives* 11 dan 12 yang dihasilkan dari *subgoal* Manajemen Lokasi, *alternative* 11 (kiri) merupakan kemungkinan *task* dimana admin dapat menambah lokasi dengan mengisi data lokasi terlebih dahulu

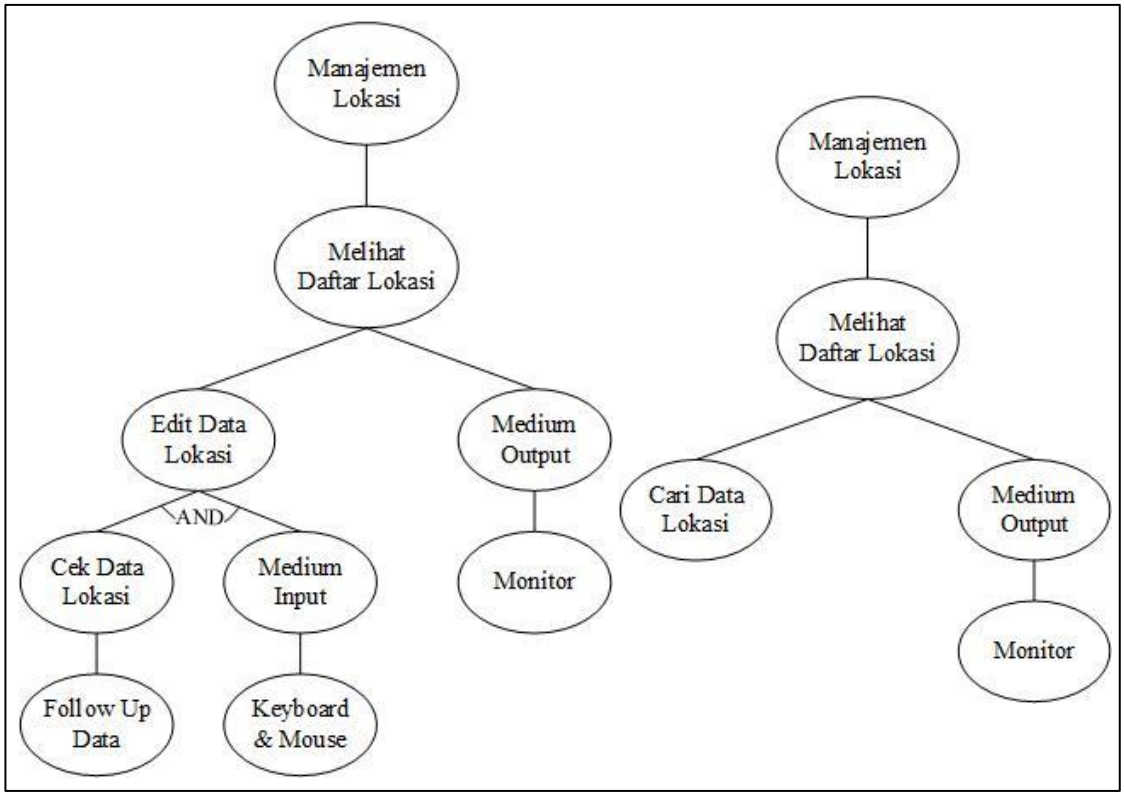
dengan medium *input keyboard* dan *mouse*. *alternative* 12 (kanan) menunjukkan admin dapat melihat daftar lokasi dengan medium *output* layar monitor.



Gambar 4.14. *Alternatives*: 9 (Kiri) dan 10 (Kanan)



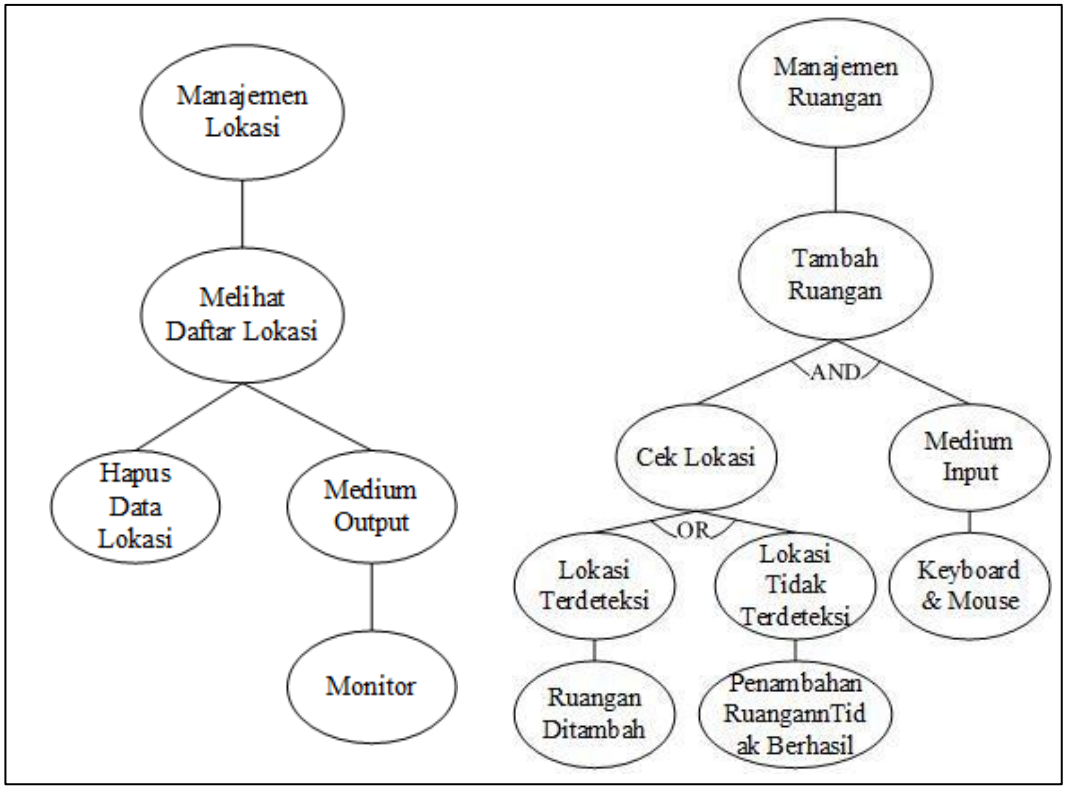
Gambar 4.15. *Alternatives*: 11 (Kiri) dan 12 (Kanan)



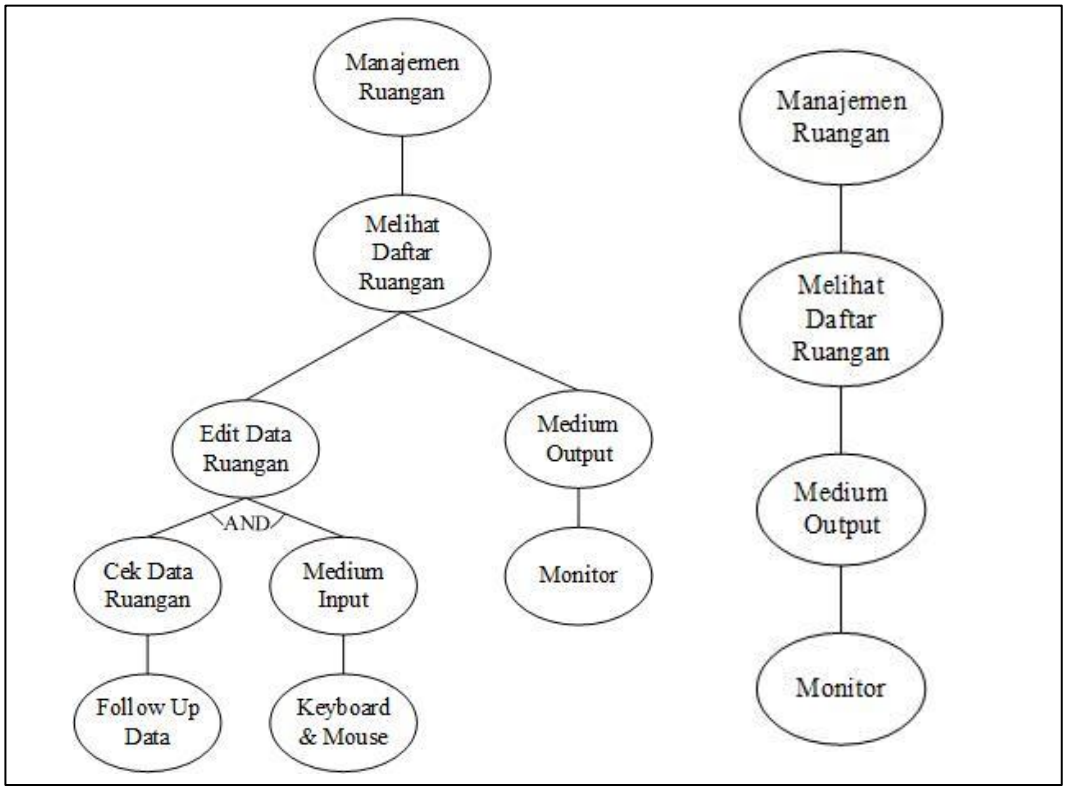
Gambar 4.16. *Alternatives*: 13 (Kiri) dan 14 (Kanan)

Gambar 4.16. merupakan *alternatives* yang dihasilkan dari *subgoal* Manajemen Lokasi, *alternative* 13 (kiri) menunjukkan bahwa admin dapat meng-*edit* data lokasi dengan mengecek data lokasi yang akan di-*edit* terlebih dahulu, kemudian data akan di-*follow up* apabila sudah di-*edit*. Adapun medium *input* yang digunakan adalah *keyboard* dan *mouse*, sedangkan medium *output* yang digunakan adalah monitor. Pada *alternative* 14 (kanan) menunjukkan bahwa admin dapat mencari data lokasi dengan medium *output* layar monitor.

Berdasarkan Gambar 4.17., *alternatives* 15 (kiri) menunjukkan bahwa admin dapat menghapus data dengan medium *output* layar monitor. Sedangkan pada *alternative* 16 (kanan) menunjukkan bahwa admin dapat menambahkan ruangan dengan mengecek lokasi terlebih dahulu, medium *input* yang digunakan adalah *keyboard* dan *mouse*.

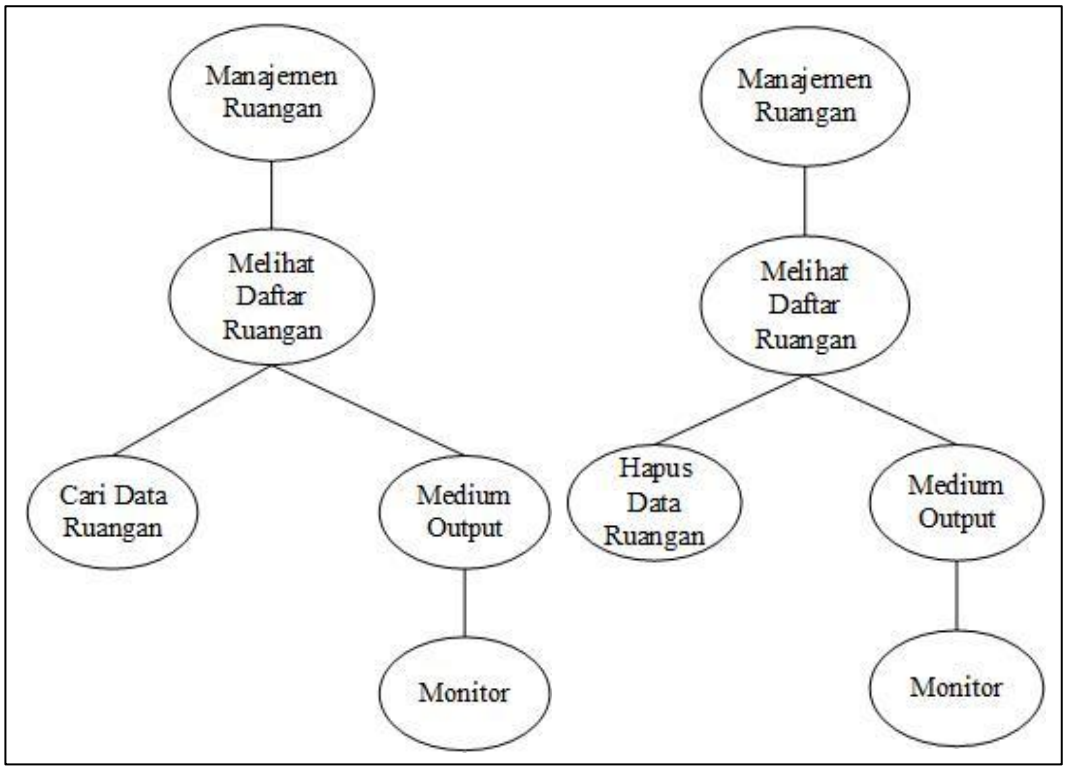


Gambar 4.17. *Alternatives*: 15 (Kiri) dan 16 (Kanan)



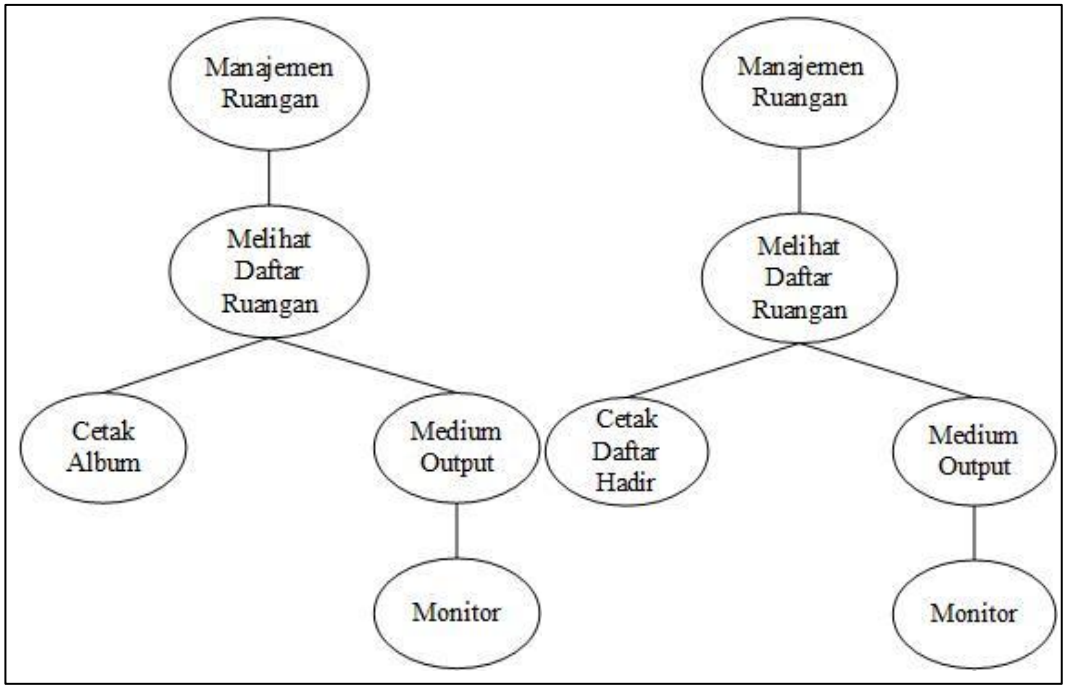
Gambar 4.18. *Alternatives*: 17 (Kiri) dan 18 (Kanan)

Gambar 4.18. menunjukkan *alternatives* yang dihasilkan dari *subgoal* Manajemen Ruangan. *Alternative* 17 (kiri) menunjukkan admin dapat meng-*edit* data ruangan dengan mengecek data ruangan terlebih dahulu untuk kemudian di-*follow up* apabila sudah di-*edit*, adapun medium *input* yang digunakan adalah *mouse* dan *keyboard*. Kemudian *alternative* 18 (kanan) menunjukkan bahwa pada *subgoal* Manajemen Ruangan, admin dapat melihat daftar ruangan dengan medium *output* layar monitor.



Gambar 4.19. *Alternatives*: 19 (Kiri) dan 20 (Kanan)

Pada Gambar 4.19. menunjukkan *alternatives* 19 dan 20 yang merupakan kemungkinan *task* yang dihasilkan dari *subgoal* Manajemen Lokasi. *Alternatives* 19 (kiri) menunjukkan admin dapat melakukan pencarian data ruangan dengan medium *output* layar monitor. Sedangkan *alternatives* 20 (kanan) menunjukkan admin dapat menghapus data ruangan dengan medium *output* layar monitor.



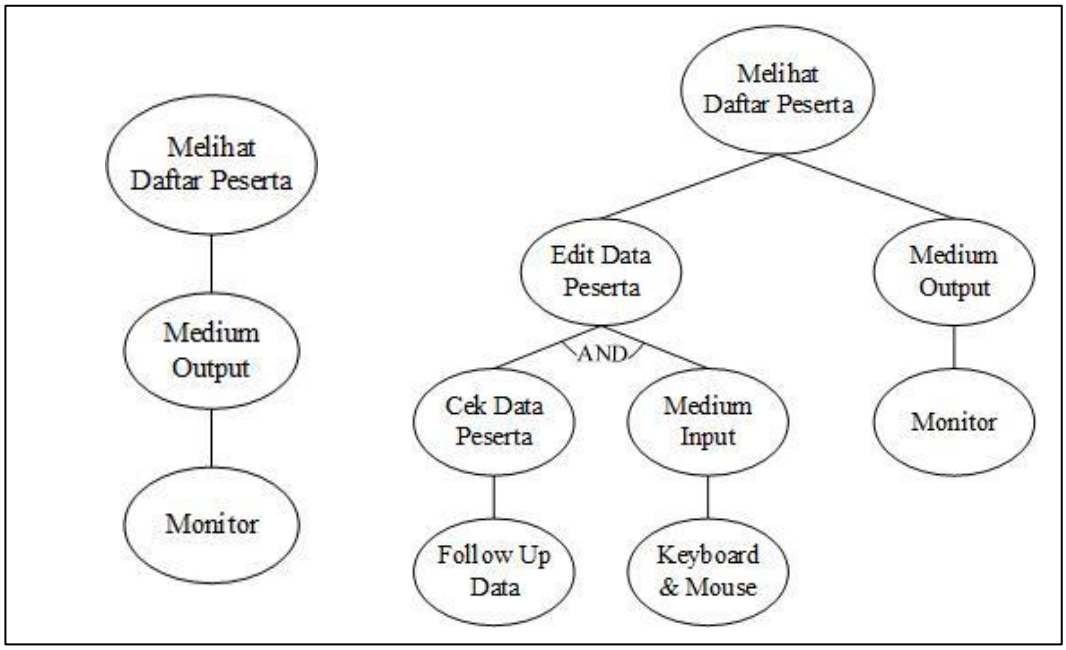
Gambar 4.20. *Alternatives*: 21 (Kiri) dan 22 (Kanan)

Gambar 4.20. merupakan *alternatives* yang dihasilkan dari *subgoal* Manajemen Ruangan. *Alternatives* 21 (kiri) menunjukkan bahwa admin dapat mencetak album dengan medium *output* layar monitor. Sedangkan *alternatives* 22 (kanan) menunjukkan bahwa admin dapat mencetak daftar hadir dengan medium *output* layar monitor.

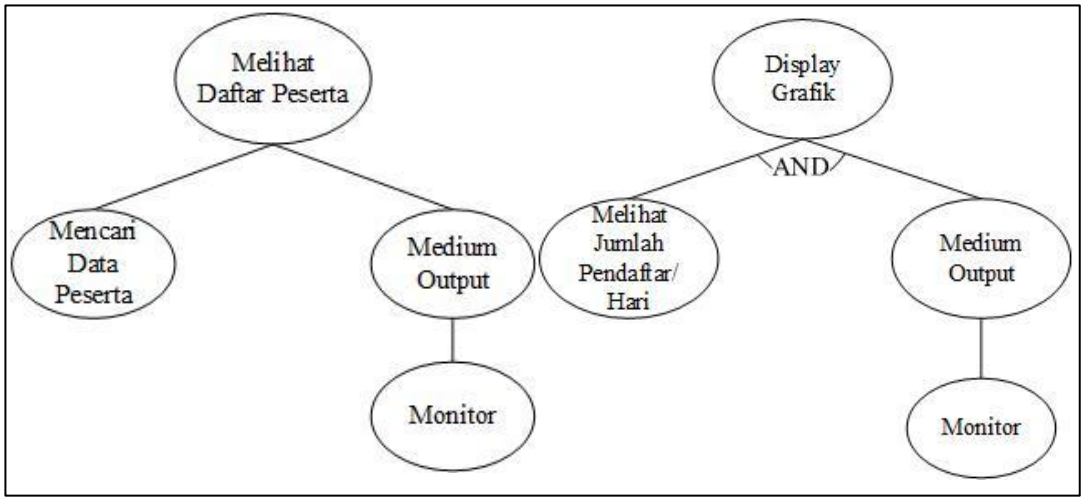
Pada Gambar 4.21. menunjukkan *alternatives* yang dihasilkan dari *subgoal* Melihat Daftar Peserta. *Alternative* 23 (kiri) menunjukkan admin dapat melihat daftar peserta dengan medium *output* layar monitor. Sedangkan *alternative* 24 (kanan) merupakan kemungkinan *task* yang memberikan hak akses pada admin untuk meng-*edit* data peserta dengan medium *input keyboard* dan *mouse,* data akan di-*follow up* apabila sudah di-*edit*. Adapun medium *output* yang digunakan adalah monitor.

Gambar 4.22. merupakan *alternatives* dari *subgoal* melihat daftar peserta dan *Display* Grafik. Pada *alternative* 25 (kiri) menunjukkan bahwa admin dapat

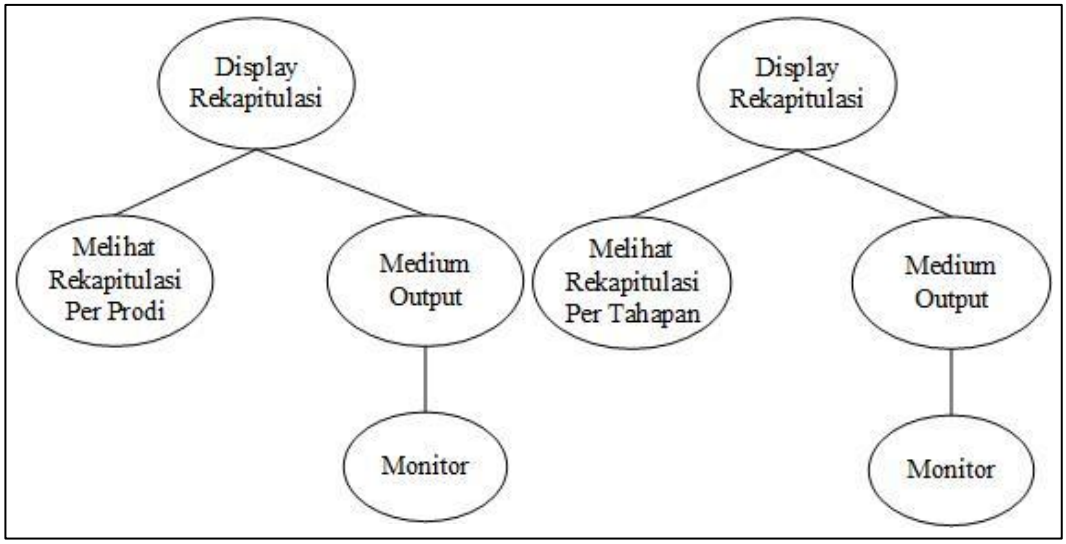
mencari data peserta dengan medium *output* layar monitor. Sedangkan *alternative* 26 (kanan) menunjukkan pada *display* grafik admin dapat melihat jumlah pendaftar yang dikelompokkan per hari dengan medium *output* layar monitor.



Gambar 4.21. *Alternatives*: 23 (Kiri) dan 24 (Kanan)

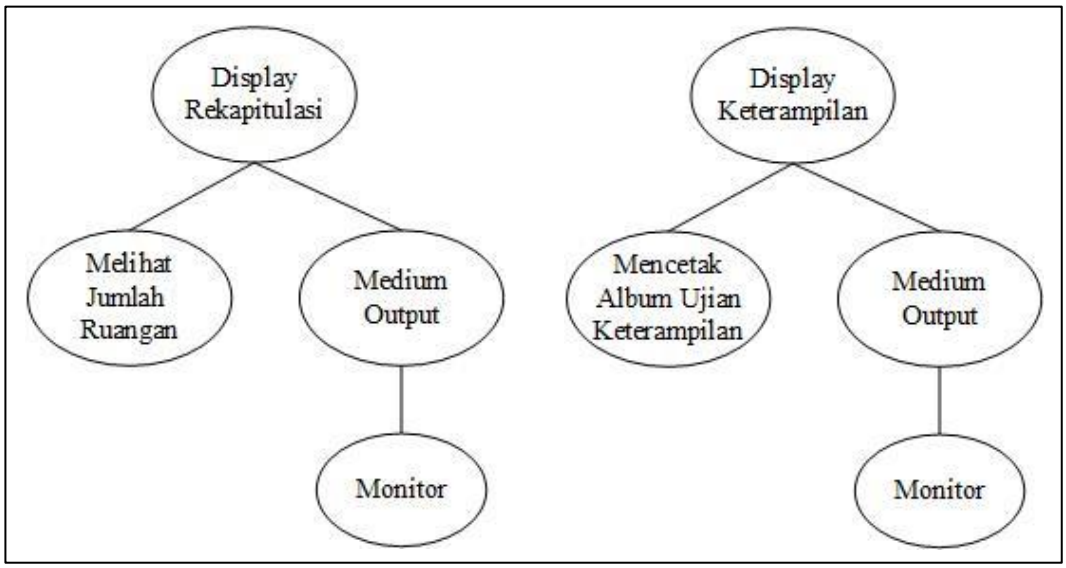


Gambar 4.22. *Alternatives*: 25 (Kiri) dan 26 (Kanan)



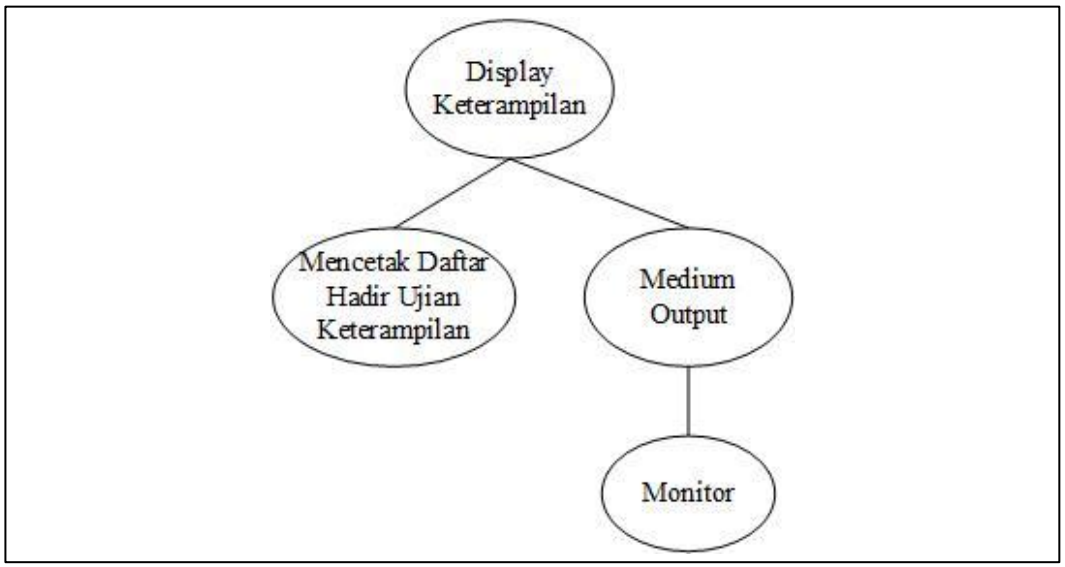
Gambar 4.23. *Alternatives*: 27 (Kiri) dan 28 (Kanan)

Alternatives selanjutnya adalah dari *subgoal Display* Rekapitulasi yang ditunjukkan oleh Gambar 4.23., *alternative* 27 (kiri) menunjukkan admin dapat melihat rekapitulasi per prodi dengan medium *output* layar monitor, sedangkan *alternative* 28 (kanan) menunjukkan admin dapat melihat rekapitulasi peserta per tahapan dengan medium *output* layar monitor.



Gambar 4.24. *Alternatives*: 29 (Kiri) dan 30 (Kanan)

Alternative 29 dan 30 ditunjukkan oleh Gambar 4.24., *alternative* 29 (kiri) masih merupakan *alternative* yang dihasilkan dari *subgoal Display* Rekapitulasi, menunjukkan bahwa admin dapat melihat jumlah ruangan dengan medium *output* layar monitor. Sedangkan *alternative* 30 (kanan) merupakan *alternative* yang dihasilkan dari *subgoal Display* Keterampilan, menunjukkan bahwa admin dapat mencetak album ujian keterampilan dengan medium *output* layar monitor.



Gambar 4.25. *Alternative* 31

Gambar 4.25. merupakan *alternative* yang dihasilkan dari *subgoal Display* Keterampilan. *Alternative* 31 menunjukkan admin dapat mencetak daftar hadir ujian keterampilan dengan medium *output* layar monitor.

Dari beberapa *alternative* di atas dapat kita lihat bahwa dari delapan *subgoal* menghasilkan 31 *alternatives*, adapun *alternatives* yang dihasilkan setiap *subgoal* jumlahnya tidak selalu sama, tergantung pada kemungkinan *task* yang dapat dilakukan oleh *user* pada sebuah sistem yang dihasilkan dari sebuah *subgoal*. Untuk memudahkan proses pembuktian dalam penelitian, perlu adanya

pengelompokkan *alternatives* berdasarkan *subgoal* yang membentuknya. Berikut merupakan tabel pengelompokan *alternatives* berdasarkan *subgoal*-nya:

Tabel 4.1. Pengelompokan *Alternatives* Berdasarkan *Subgoal*

Subgoal	***Alternatives***
Manajemen *User*	1, 2, 3, 4, dan 5.
Manajemen Pengumuman	6, 7, 8, 9, dan 10.
Manajemen Lokasi	11, 12, 13, 14, dan 15.
Manajemen Ruangan	16, 17, 18, 19, 20, 21, dan 22.
Melihat Daftar Peserta	23, 24, dan 25.
Display Grafik	26.
Display Rekapitulasi	27, 28, dan 29.
Display Keterampilan	30 dan 31.

Berdasarkan Tabel 4.1., dapat dilihat bahwa *alternatives* terbanyak merupakan *alternatives* pada *subgoal* Manajemen Ruangan, sedangkan *alternatives* paling sedikit adalah *alternatives* pada *subgoal Display* Grafik. Ini menunjukkan bahwa pada *subgoal* Manajemen Ruangan, *user* dapat melakukan lebih banyak *task* dibandingkan pada *subgoal Display* Grafik. Setiap *alternatives* sudah pasti masuk dalam salah satu pengelompokan *subgoal*.

4.1.5. Hasil *Reverse Engineering* Menggunakan GORE dengan Metode GSP

Setelah dilakukan proses *reverse engineering* menggunakan model GORE dengan metode GSP, ditarik beberapa *requirements* yang memenuhi syarat relasi *AND* dan *OR* dari beberapa *alternative* yang sudah diuraikan pada tahap sebelumnya. Berikut merupakan hasil *reverse engineering* dengan menggunakan GORE dengan metode GSP:

Tabel 4.2. Hasil *Reverse Engineering* menggunakan GORE dengan metode GSP

No. *Alternatives*	*Requirements*
1	Pada submenu *User*, Admin dapat menambahkan *user* sebagai peserta, admin, *help desk*, dekan, rektorat, atau BAAK dengan medium *input keyboard* dan *mouse*.
2	Pada submenu *User*, Admin dapat melihat daftar *user* dengan medium *output* layar monitor.
3	Pada submenu *User*, Admin dapat meng-*edit* data *user* dengan medium *input keyboard* dan *mouse*, sistem akan mem-*follow up* data ketika dilakukan peng-*edit*-an.
4	Pada submenu *User*, admin dapat menghapus data *user* dengan medium *output* layar monitor.
5	Pada submenu *User*, admin dapat melakukan pencarian data *user* dengan medium *output* layar monitor.
6	Pada submenu Pengumuman, Admin dapat menambah pengumuman dengan melakukan *submit* pengumuman melalui media *input keyboard* dan *mouse*.
7	Pada submenu Pengumuman, Admin dapat melihat daftar pengumuman dengan medium *output* layar monitor.
8	Pada submenu Pengumuman, Admin dapat meng-*edit* pengumuman dengan medium *input keyboard* dan *mouse*, data akan di-*follow up* secara otomatis apabila melakukan peng-*edit*-an.
9	Pada submenu Pengumuman, Admin dapat menghapus pengumuman dengan medium *output* layar monitor.
10	Pada submenu Pengumuman, Admin dapat melakukan pencarian data pengumuman dengan medium *output* layar monitor.
11	Pada submenu Lokasi, Admin dapat menambah lokasi dengan mengisi data lokasi melalui medium *input keyboard* dan *mouse*.
12	Pada submenu Lokasi, Admin dapat melihat daftar lokasi dengan medium *output* layar monitor.
13	Pada submenu Lokasi, Admin dapat meng-*edit* data lokasi dengan medium *input keyboard* dan *mouse*, data akan di-*follow up* secara otomatis apabila melakukan peng-*edit*-an.
14	Pada submenu Lokasi, Admin dapat mencari data lokasi dengan medium *output* layar monitor.
15	Pada submenu Lokasi, Admin dapat menghapus data lokasi dengan medium *output* layar monitor.
16	Pada submenu Ruangan, Admin dapat menambah ruangan dengan mengecek lokasi terlebih dahulu, apabila lokasi sudah terdaftar maka penambahan lokasi berhasil, dan apabila lokasi belum terdaftar maka ruangan gagal

No. *Alternatives*	*Requirements*
	ditambahkan, sehingga harus melakukan penambahan lokasi terlebih dahulu di submenu Manajemen Lokasi.
17	Pada submenu Ruangan, Admin dapat meng-*edit* data ruangan yang sudah di-*input* dengan medium *input keyboard* dan *mouse*, data akan di-*follow up* secara otomatis apabila melakukan peng-*edit*-an.
18	Pada submenu Ruangan, Admin dapat melihat daftar ruangan yang sudah di-*input*.
19	Pada submenu Ruangan, Admin dapat melakukan pencarian data ruangan dengan medium *output* layar monitor.
20	Pada submenu Ruangan, Admin dapat menghapus data ruangan yang sudah di-*input* melalui medium *output* layar monitor.
21	Pada submenu Ruangan, Admin dapat mencetak lembar album dengan medium *output* layar monitor.
22	Pada submenu Ruangan, Admin dapat mencetak lembar daftar hadir dengan medium *output* layar monitor.
23	Pada submenu Pencarian Peserta, Admin dapat melihat daftar peserta medium *output* layar monitor.
24	Pada submenu Pencarian Peserta, Admin dapat meng-*edit* data peserta kecuali *username*, melalui medium *input keyboard* dan *mouse*, data akan di-*follow up* secara otomatis apabila melakukan peng-*edit*-an.
25	Pada submenu Pencarian Peserta, Admin dapat melakukan pencarian data peserta melalui medium *output* layar monitor.
26	Pada submenu Grafik, Admin dapat melihat jumlah pendaftar yang dikelompokkan per hari melalui medium *output* layar monitor.
27	Pada submenu Rekapitulasi, Admin dapat melihat rekapitulasi pendaftar per prodi melalui medium *output* layar monitor.
28	Pada submenu Rekapitulasi, Admin dapat melihat rekapitulasi pendaftar per tahapan melalui medium *output* layar monitor.
29	Pada submenu Rekapitulasi, Admin dapat melihat jumlah ruangan beserta jumlah pendaftar yang ada di dalamnya melalui medium *output* layar monitor.
30	Pada submenu Keterampilan, Admin dapat mencetak lembar album ujian keterampilan melalui medium *output* layar monitor.
31	Pada submenu Keterampilan, Admin dapat mencetak lembar daftar hadir ujian keterampilan melalui medium *output* layar monitor.

Berdasarkan Tabel 4.2. yaitu hasil dari penarikan *requirements* di atas, dilakukan perbandingan hasil *requirements* tersebut dengan hasil *reverse* konvensional yang dilakukan oleh pihak analis dari aplikasi yang di-*reverse*. Perbandingan hasil *reverse* dilakukan berdasarkan pengelompokkan *alternatives*. Pembahasan pertama, yaitu perbandingan *requirements* yang dihasilkan dari *subgoal* Manajemen *User*, dapat dilihat pada Tabel 4.3., *alternatives* 1, 2, 3, 4, dan 5 dikelompokkan pada *subgoal* Manajemen *User*. Berikut merupakan perbandingan antara *requirements* yang dihasilkan dari proses *reverse engineering* GORE dengan hasil *reverse* konvensional pada *subgoal* Manajemen *User*:

Tabel 4.3. Perbandingan Hasil *Reverse Engineering* Menggunakan GORE dengan Metode GSP dan *Reverse* Konvensional pada *Subgoal* Manajemen *User*

***Reverse* dengan GORE**	***Reverse* Konvensional**
1: Pada submenu *User*, Admin dapat menambahkan *user* sebagai peserta, admin, *help desk*, dekan, rektorat, atau BAAK dengan medium *input keyboard* dan *mouse*. 2: Pada submenu *User*, Admin dapat melihat daftar *user* dengan medium *output* layar monitor. 3: Pada submenu *User*, Admin dapat meng-*edit* data *user* dengan medium *input keyboard* dan *mouse*, sistem akan mem-*follow up* data ketika dilakukan peng-*edit*-an. 4: Pada submenu *User*, Admin dapat menghapus data *user* dengan medium *output* layar monitor. 5: Pada submenu *User*, Admin dapat melakukan pencarian data *user* dengan medium *output* layar monitor.	1. Admin dapat menambahkan *user* sebagai admin, *help desk*, dekan, rektorat, atau BAAK. 2. Admin dapat melihat daftar *user*. 3. Admin dapat meng-*edit* data *user*. 4. Admin dapat menghapus data *user*. 5. Admin dapat melakukan pencarian data *user*.

Berdasarkan Tabel 4.3. dapat dilihat perbedaan hasil *requirements* yang dihasilkan melalui proses *reverse engineering* GORE dengan proses *reverse*

engineering konvensional. Pada dasarnya konten dari masing-masing poin *requirements* bermaksud sama, hanya saja pada hasil *reverse engineering* dengan GORE hasilnya menjadi lebih rinci dan terstruktur, sehingga ketika *stakeholder* akan melakukan pengembangan dan perbaikan, maka prosesnya akan lebih mudah. *Stakeholder* hanya perlu mempelajari dari hasil *requirements* yang sudah ada.

Namun, pada poin pertama terdapat perbedaan yaitu pada hasil *reverse* GORE dikatakan bahwa "Pada submenu *User*, Admin dapat menambahkan *user* sebagai peserta, admin, *help desk*, dekan, rektorat, atau BAAK dengan medium *input keyboard dan mouse*.", sedangkan pada hasil *reverse* konvensional "Admin dapat menambahkan *user* admin, *help desk*, dekan, rektorat, atau BAAK.". Pada hasil *reverse* GORE dapat menambahkan Peserta sedangkan pada hasil *reverse* konvensional tidak. Hal ini terjadi dikarenakan konsep GORE yang murni mengandalkan tampilan aplikasi yang sudah siap pakai ketika akan di *reverse*. Pada tampilan aplikasi, terdapat fungsi untuk menambahkan Peserta, namun ternyata ketika melakukan wawancara mengenai hasil *reverse* konvensional dari pihak analis, analis mengatakan bahwa Admin tidak dapat menambahkan peserta, karena peserta bertambah secara otomatis berdasarkan banyaknya peserta yang mendaftar melalui portal yang sudah disediakan. Pada tampilan aplikasi dibuat fungsi untuk menambah Peserta dikarenakan sinkronisasi ke *database*, pada *database* terdapat tabel Peserta sehingga harus juga ditampilkan fungsi penambahan Peserta yang sebenarnya pihak Admin tidak memiliki hak untuk menambah Peserta.

Untuk *alternatives* 2, 3, 4, dan 5 hasilnya adalah sama dengan hasil *reverse* konvensional. Pada submenu *User*, Admin diberikan hak akses untuk melihat daftar *user* dengan medium *output* layar monitor. Selain itu, Admin juga dapat meng-*edit*, menghapus, dan melakukan pencarian data *user* yang sudah di-*input* sebelumnya. Selanjutnya adalah analisis dari hasil *reverse engineering* pada *subgoal* Manajemen Pengumuman, berikut hasilnya:

Tabel 4.4. Perbandingan Hasil *Reverse Engineering* Menggunakan GORE dengan Metode GSP dan *Reverse* Konvensional pada *Subgoal* Manajemen Pengumuman

***Reverse* dengan GORE**	***Reverse* Konvensional**
6: Pada submenu Pengumuman, Admin dapat menambah pengumuman dengan melakukan *submit* pengumuman melalui media *input keyboard dan mouse*. 7: Pada submenu Pengumuman, Admin dapat melihat daftar pengumuman dengan medium *output* layar monitor. 8: Pada submenu Pengumuman, Admin dapat meng-*edit* pengumuman dengan medium *input keyboard dan mouse*, data akan di-*follow up* secara otomatis apabila melakukan peng-*edit*-an. 9: Pada submenu Pengumuman, Admin dapat menghapus pengumuman dengan medium *output* layar monitor. 10: Pada submenu Pengumuman, Admin dapat melakukan pencarian data pengumuman dengan medium *output* layar monitor.	Submenu Pengumuman tidak berfungsi.

Pada *subgoal* Manajemen Pengumuman, berdasarkan hasil *reverse* GORE pada Tabel 4.4. menghasilkan lima *requirements* yang sesuai dengan *interface* aplikasi. Namun, pada hasil *reverse* konvensional yang dipaparkan oleh analis mengatakan bahwa submenu Pengumuman tidak memiliki fungsi sama sekali.

Interface yang disediakan belum bisa digunakan, sehingga fungsi yang disediakan tidak dapat berjalan. Dikarenakan prinsip GORE yang mengambil semua *requirements* pada *interface*, maka proses *reverse* menghasilkan beberapa *requirements* yang seharusnya ada dalam submenu Manajemen Pengumuman.

Selanjutnya ialah analisis perbandingan dari hasil *reverse subgoal* Manajemen Lokasi:

Tabel 4.5. Perbandingan Hasil *Reverse Engineering* Menggunakan GORE dengan Metode GSP dan *Reverse* Konvensional pada *Subgoal* Manajemen Lokasi

***Reverse* dengan GORE**	***Reverse* Konvensional**
11: Pada submenu Lokasi, Admin dapat menambah lokasi dengan mengisi data lokasi melalui medium *input keyboard dan mouse*. 12: Pada submenu Lokasi, Admin dapat melihat daftar lokasi dengan medium *output* layar monitor. 13: Pada submenu Lokasi, Admin dapat meng-*edit* data lokasi dengan medium *input keyboard dan mouse*, data akan di-*follow up* secara otomatis apabila melakukan peng-*edit*-an. 14: Pada submenu Lokasi, Admin dapat mencari data lokasi dengan medium *output* layar monitor. 15: Pada submenu Lokasi, Admin dapat menghapus data lokasi dengan medium *output* layar monitor.	1. Admin dapat menambahkan lokasi. 2. Admin dapat melihat daftar lokasi. 3. Admin dapat meng-*edit* data lokasi. 4. Admin dapat melakukan pencarian data lokasi. 5. Admin dapat menghapus data lokasi.

Berdasarkan hasil *reverse* pada Tabel 4.5., dapat dilihat bahwa jumlah *requirements* yang dihasilkan pada *reverse* menggunakan GORE dan *reverse* konvensional adalah sama, dan fungsi yang dihasilkan dari *requirements*-nya juga sama, hanya saja hasil *requirements* dari *reverse* GORE lebih lengkap dan terstruktur sehingga mudah dimengerti oleh *stakeholder* ketika melakukan

pengembangan dan perbaikan sistem. Adapun hak akses Admin pada sistem yaitu pada submenu Lokasi, Admin dapat menambah lokasi dengan mengisi data lokasi melalui medium *input keyboard dan mouse*. Jadi, ketika Admin ingin menambahkan lokasi, Admin akan diminta untuk mengisi data lokasi terlebih dahulu, adapun data lokasi yang diminta berdasarkan *form* yang ditampilkan pada sistem aplikasi. Selain itu, Admin juga dapat melihat daftar lokasi yang sudah di-*input* sebelumnya dengan medium *output* layar monitor. Admin juga memiliki hak akses untuk meng-*edit* data lokasi tersebut, setelah di-*edit*, data akan di-*follow up* secara otomatis. Dan apabila dibutuhkan, Admin juga memiliki hak akses untuk menghapus data lokasi dengan melakukan evaluasi terlebih dahulu, serta pada daftar lokasi yang sudah di-*input*, Admin juga dapat melakukan pencarian data lokasi jika dibutuhkan.

Berikut ini merupakan hasil *reverse* pada *subgoal* Manajemen Ruangan:

Tabel 4.6. Perbandingan Hasil *Reverse Engineering* Menggunakan GORE dengan Metode GSP dan *Reverse* Konvensional pada *Subgoal* Manajemen Ruangan

***Reverse* dengan GORE**	***Reverse* Konvensional**
16: Pada submenu Ruangan, Admin dapat menambah ruangan dengan mengecek lokasi terlebih dahulu, apabila lokasi sudah terdaftar maka penambahan lokasi berhasil, dan apabila lokasi belum terdaftar maka ruangan gagal ditambahkan, sehingga harus melakukan penambahan lokasi terlebih dahulu di submenu Manajemen Lokasi. 17: Pada submenu Ruangan, Admin dapat meng-*edit* data ruangan yang sudah di-*input* dengan medium *input keyboard dan mouse*, data akan di-*follow up* secara otomatis apabila melakukan peng-*edit*-an.	1. Admin dapat menambahkan ruangan, jika lokasi sudah terdaftar terlebih dahulu. 2. Admin dapat meng-*edit* data ruangan. 3. Admin dapat melihat daftar ruangan. 4. Admin dapat melakukan pencarian data ruangan. 5. Admin dapat menghapus data ruangan. 6. Admin dapat mencetak album. 7. Admin dapat mencetak daftar hadir.

***Reverse* dengan GORE**	***Reverse* Konvensional**
18: Pada submenu Ruangan, Admin dapat melihat daftar ruangan yang sudah di-*input*. 19: Pada submenu Ruangan, Admin dapat melakukan pencarian data ruangan dengan medium *output* layar monitor. 20: Pada submenu Ruangan, Admin dapat menghapus data ruangan yang sudah di-*input* melalui medium *output* layar monitor. 21: Pada submenu Ruangan, Admin dapat mencetak lembar album dengan medium *output* layar monitor. 22: Pada submenu Ruangan, Admin dapat mencetak lembar daftar hadir dengan medium *output* layar monitor.	

Secara keseluruhan, *requirements* yang dihasilkan dari *reverse* GORE dan *reverse* konvensional adalah sama, hanya saja lebih lengkap dan tertata jika menggunakan GORE karena prinsip GORE yang menarik *requirements* dari *goal graph* yang sudah dielisitasi terlebih dahulu. Berdasarkan Tabel 4.6., hak akses yang dapat dilakukan Admin pada submenu Pencarian Peserta yaitu Admin dapat menambah ruangan dengan mengecek lokasi terlebih dahulu, apabila lokasi sudah terdaftar maka penambahan lokasi berhasil, dan apabila lokasi belum terdaftar maka ruangan gagal ditambahkan, sehingga harus melakukan penambahan lokasi terlebih dahulu di submenu Manajemen Lokasi. Apabila lokasi dari ruangan yang akan ditambahkan belum terdaftar, maka akan muncul notifikasi bahwa lokasi belum terdaftar. Hak akses juga diberikan pada Admin untuk meng-*edit* data ruangan yang sudah di-*input* dengan medium *input keyboard dan mouse*, data akan di-*follow up* secara otomatis apabila melakukan peng-*edit*-an. Selain itu, Admin juga dapat melihat daftar ruangan yang sudah di-*input*, melakukan

pencarian, dan menghapus data ruangan jika diperlukan dengan melakukan evaluasi terlebih dahulu. Kemudian, Admin juga diberi akses untuk mencetak lembar album dengan medium *output* layar monitor. Lembar album merupakan data nama peserta beserta foto peserta yang akan mengikuti ujian. *Requirements* di atas dilengkapi dengan hak akses Admin yang dapat mencetak lembar daftar hadir. Sehingga, cakupan dari Manajemen Ruangan bukan hanya tentang data ruangan saja, akan tetapi menyangkut peserta, lokasi, dan daftar hadir yang akan digunakan sebagai lembar presensi ketika peserta mengikuti ujian.

Di bawah ini merupakan hasil *reverse* untuk *subgoal* Pencarian Peserta:

Tabel 4.7. Perbandingan Hasil *Reverse Engineering* Menggunakan GORE dengan Metode GSP dan *Reverse* Konvensional pada *Subgoal* Pencarian Peserta

***Reverse* dengan GORE**	***Reverse* Konvensional**
23: Pada submenu Pencarian Peserta, Admin dapat melihat daftar peserta medium *output* layar monitor. 24: Pada submenu Pencarian Peserta, Admin dapat meng-*edit* data peserta kecuali *username*, melalui medium *input keyboard dan mouse*, data akan di-*follow up* secara otomatis apabila melakukan peng-*edit*-an. 25: Pada submenu Pencarian Peserta, Admin dapat melakukan pencarian data peserta melalui medium *output* layar monitor.	1. Admin dapat melihat daftar peserta. 2. Admin dapat meng-*edit* data peserta, kecuali *user*name. 3. Admin dapat mencari data peserta.

Tabel 4.7. menunjukkan bahwa pada submenu Pencarian Peserta, Admin hanya memiliki tiga hak untuk melakukan *task* yang ada pada submenu tersebut. Pada dasarnya, Admin tidak dapat melakukan penambahan peserta, karena jumlah peserta akan bertambah berdasarkan banyaknya peserta yang mendaftar pada portal khusus yang sudah disediakan. Oleh karena itu, Admin hanya bisa melihat

daftar peserta yang sudah terdaftar di sistem, kemudian Admin juga diberikan hak akses untuk meng-*edit* data peserta jika dibutuhkan, namun tidak untuk *username*, karena biasanya proses peng-*edit*-an ini dilakukan apabila pada saat peserta mengisi *form* pendaftaran, internet yang digunakan kurang mendukung, sehingga biasanya foto yang di-*upload* sebagai salah satu identitas peserta menjadi rusak atau komponennya tidak lengkap. Maka dibutuhkan Admin yang bertugas meng-*edit* data peserta, sehingga saat proses percetakan album tidak terjadi kekurangan data. Kemudian, tentunya dalam hal ini Admin dapat melakukan pencarian, hal ini diperlukan apabila Admin ingin mencari data peserta tertentu, sehingga memudahkan Admin dalam proses pencarian.

Berikutnya adalah *subgoal Display* Grafik, yang mana submenu ini cenderung sebagai bentuk evaluasi data untuk melihat jumlah pendaftar. Fungsi yang disediakan dalam submenu ini juga tidak banyak, di bawah ini merupakan hasil *reverse* dari *subgoal Display* Grafik:

Tabel 4.8. Perbandingan Hasil *Reverse Engineering* Menggunakan GORE dengan Metode GSP dan *Reverse* Konvensional pada *Subgoal Display* Grafik

***Reverse* dengan GORE**	***Reverse* Konvensional**
26: Pada submenu Grafik, Admin dapat melihat jumlah pendaftar yang dikelompokkan per hari melalui medium *output* layar monitor.	1. Admin dapat melihat jumlah pendaftar (yang sudah pilih prodi) yang dikelompokkan per harinya.

Berdasarkan Tabel 4.8., pada submenu *Display* Grafik, kedua hasil *reverse* baik dengan GORE maupun konvensional menghasilkan poin *requirements* yang sama. Dalam submenu ini Admin hanya dapat melihat jumlah pendaftar yang dikelompokkan setiap hari. Jadi, grafik akan menunjukkan jumlah pendaftar pada

hari tertentu, naik turunnya grafik berdasarkan banyak-sedikitnya pendaftar dari hari ke hari.

Selanjutnya ialah perbandingan hasil *reverse* pada *subgoal Display* Rekapitulasi, yaitu sebagai berikut:

Tabel 4.9. Perbandingan Hasil *Reverse Engineering* Menggunakan GORE dengan Metode GSP dan *Reverse* Konvensional pada *Subgoal Display* Rekapitulasi

***Reverse* dengan GORE**	***Reverse* Konvensional**
27: Pada submenu Rekapitulasi, Admin dapat melihat rekapitulasi pendaftar per prodi melalui medium *output* layar monitor. 28: Pada submenu Rekapitulasi, Admin dapat melihat rekapitulasi pendaftar per tahapan melalui medium *output* layar monitor. 29: Pada submenu Rekapitulasi, Admin dapat melihat jumlah ruangan beserta jumlah pendaftar yang ada di dalamnya melalui medium *output* layar monitor.	1. Admin dapat melihat rekapitulasi per prodi. 2. Admin dapat melihat rekapitulasi per tahapan. 3. Admin dapat melihat daftar ruangan.

Tabel 4.9. menunjukkan bahwa submenu Rekapitulasi merupakan submenu yang menampilkan data secara keseluruhan, dan dalam hal ini data dikelompokkan berdasarkan ruangan, prodi, dan tahapan. Hasil *requirements* pertama yaitu Admin dapat melihat rekapitulasi pendaftar per prodi melalui medium *output* layar monitor. Jumlah pendaftar akan dikelompokkan berdasarkan prodi dan pilihan ke berapa prodi tersebut dipilih, sehingga Admin dapat mengetahui jumlah peserta yang memilih untuk pilihan pertama, kedua, dan ketiga pada prodi tertentu. Selain itu, Admin juga dapat melihat rekapitulasi pendaftar per tahapan. Ada lima tahap yaitu, buat akun, pilih prodi, bayar tagihan, isi biodata, dan selesai (melakukan keempat tahap di atasnya). Tujuan dari

rekapitulasi per tahapan adalah memudahkan Admin untuk me-*monitoring* pendaftar dan mengevaluasi sudah sejauh mana prosedur yang sudah dilakukan oleh pendaftar. Bukan hanya mengenai rekapitulasi prodi dan tahapan, pada submenu ini, Admin juga dapat melihat jumlah ruangan beserta jumlah pendaftar yang ada di dalamnya melalui medium *output* layar monitor. Ini berguna untuk mengantisipasi apabila ruangan sudah memasuki batas *overload* pendaftar, maka Admin berhak menambahkan ruangan baru pada submenu Ruangan.

Yang terakhir adalah hasil *reverse* pada *subgoal Display* Keterampilan. Berikut merupakan uraiannya:

Tabel 4.10. Perbandingan Hasil *Reverse Engineering* Menggunakan GORE dengan Metode GSP dan *Reverse* Konvensional pada *Subgoal Display* Keterampilan

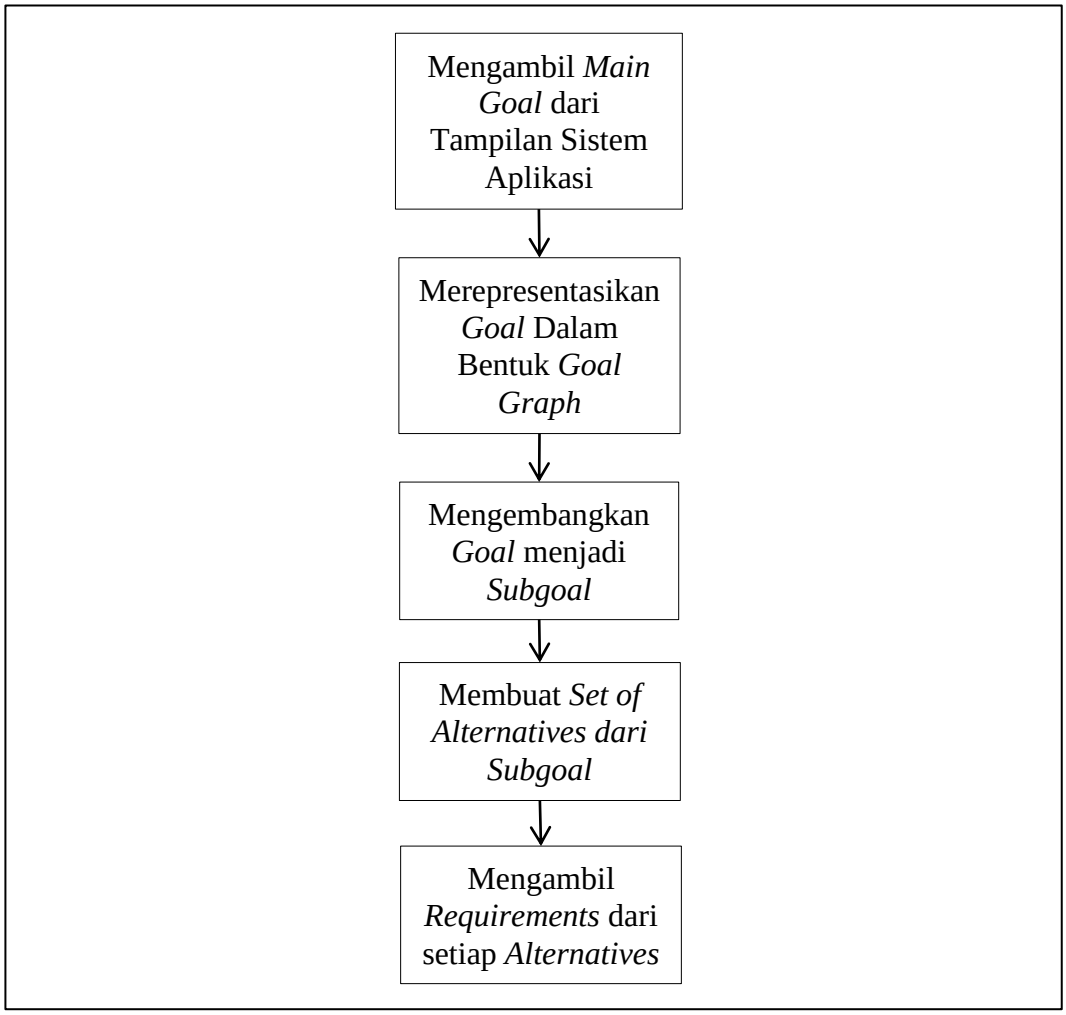
***Reverse* dengan GORE**	***Reverse* Konvensional**
30: Pada submenu Keterampilan, Admin dapat mencetak lembar album ujian keterampilan melalui medium *output* layar monitor. 31: Pada submenu Keterampilan, Admin dapat mencetak lembar daftar hadir ujian keterampilan melalui medium *output* layar monitor.	1. Admin dapat mencetak lembar album ujian keterampilan. 2. Admin dapat mencetak daftar hadir ujian keterampilan.

Tabel 4.10. menunjukkan bahwa submenu Keterampilan ini berfungsi untuk me-*monitoring* pendaftar dalam mengikuti ujian keterampilan, ujian keterampilan diberikan kepada pendaftar yang memilih program studi tertentu, seperti Pendidikan Seni Rupa, Pendidikan Seni Tari, Pendidikan Seni Musik, Pendidikan Jasmani, dan lain sebagainya. Dalam submenu ini, hasil *reverse* menghasilkan *requirements* yang sama intinya baik pada *reverse* menggunakan GORE maupun *reverse* konvensional. Pada submenu Keterampilan, Admin dapat

mencetak lembar album ujian keterampilan dan lembar daftar hadir ujian keterampilan melalui medium *output* layar monitor. Begitu juga hasil *reverse* konvensional menghasilkan *requirements* yang sama.

4.1.6. Hasil Model GORE dengan Metode GSP

Berdasarkan langkah-langkah yang sudah dilakukan, maka muncullah hasil penelitian berupa model *Goal Oriented Requirements Engineering* (GORE) untuk proses *Reverse Engineering* dengan metode GSP. Berikut merupakan model GORE dengan metode GSP yang direpresentasikan dalam bentuk diagram:



Gambar 4.26. Model GORE dengan metode GSP untuk Proses *Reverse Engineering*

Berdasarkan Gambar 4.26., langkah-langkah untuk menggunakan GORE dalam proses *reverse engineering* berdasarkan Gambar 4.26. adalah sebagai berikut:

1. Mengambil *main goal* dari tampilan sistem aplikasi

   Konsep GORE adalah berorientasi *goal*, sehingga kita cukup fokus hanya pada *goal* saja. *Main goal* merupakan tujuan dari pembuatan sistem untuk memenuhi kebutuhan *stakeholder*. *Main goal* dapat dilihat dari tampilan aplikasi siap pakai berdasarkan fitur yang tertera pada *interface* aplikasi tersebut.

2. Merepresentasikan *goal* dalam bentuk *goal graph*

   Setelah itu, representasikan *main goal* dalam bentuk *goal graph*. Ini bertujuan untuk mengetahui cakupan dari tujuan sistem. *Goal graph* merupakan grafik menyerupai bentuk pohon bercabang yang akan memetakan sejauh mana jangkauan sebuah *goal* pada suatu sistem aplikasi.

3. Mengembangkan *goal* menjadi *subgoal*

   Langkah selanjutnya adalah mengembangkan *goal* tersebut menjadi beberapa *subgoal*. Dalam satu *main goal* dapat memiliki beberapa *subgoal* yang merupakan cabang dari *graph* yang dimiliki oleh *goal* tertentu, *subgoal* dari *parent* yang sama akan membentuk hubungan *AND* apabila semua *subgoal* harus terpenuhi agar tujuan dari *goal parent* juga terpenuhi, sedangkan *subgoal* akan membentuk hubungan *OR* apabila salah satu atau lebih dari *subgoal* tersebut harus terpenuhi apabila *goal parent* ingin terpenuhi.

4. Membuat *set of alternatives* dari *subgoal*

   Dari *subgoal* tersebut, buat *set alternatives* berdasarkan banyaknya kemungkinan *task* yang dapat dilakukan dalam satu *subgoal*. Jumlah *alternatives* pada tiap *subgoal* tidak selalu sama, karena jumlah cabang pada *subgoal* cenderung berbeda.

5. Mengambil *requirements* dari setiap *alternatives*

   Langkah terakhir yaitu dari *alternatives* tersebut tarik *requirements* berdasarkan alur *goal graph* yang sudah dibuat. Dari satu *alternatives* dapat diambil satu *requirements*, sehingga banyaknya *requirements* yang dihasilkan bergantung pada banyaknya *alternatives* yang muncul pada tahap sebelumnya. *Requirements* yang dihasilkan akan lebih lengkap dan terstruktur karena diambil langsung dari alur *goal graph* yang sudah ada.

Sebagai bentuk kontribusi terhadap ilmu pengembangan rekayasa perangkat lunak, dilakukan beberapa renovasi pada konsep GORE dan GSP itu sendiri, yaitu sebagai berikut:

1. Pemilihan konsep *user goal* sebagai dasar menemukan *requirements*

   GSP pada awalnya memiliki konsep tiga hal penting yang perlu diperhatikan dalam menggunakannya, yaitu *User Goal*, *User Skill*, dan *User Preferences*. Akan tetapi, ditemukan bahwa ternyata dengan konsep *User Goal* saja sudah bisa mencapai tujuan *reverse engineering* yaitu menemukan *requirements*. Sedangkan konsep *User Skill* dan *User Preferences* membuat proses menjadi lebih rumit dan juga konsep keduanya diperlukan untuk menghasilkan *non-functional*

requirements. Perlu diketahui bahwa *requirements* yang diutamakan dalam *reverse engineering* adalah *functional requirements*, oleh karena itu, terdapat perubahan konsep GSP yaitu dengan menggunakan konsep *Goal* sebagai metode yang dipilih.

2. Teknik analisa *interface* sebagai cara menemukan *goal* pada sistem

   Kontribusi lain juga dilakukan pada tahap awal GSP, untuk menemukan *main goal* pada metode GSP seharusnya dengan melakukan teknik wawancara kepada *programmer* atau *developer internal*, akan tetapi penelitian ini memandang jauh pada permasalahan ketika analis akan melakukan *reverse* namun yang tersedia hanya aplikasi siap pakai, maka terdapat perubahan konsep yaitu pada proses menemukan *main goal*, analis hanya perlu mengamati tampilan dari aplikasi tersebut tanpa harus melakukan wawancara dengan *programmer* atau *developer internal*. Beberapa kontribusi tersebut pada akhirnya menghasilkan *requirements* yang sesuai dan tidak menyimpang jauh dengan hasil *reverse* konvensionalnya.

3. Pembatasan jumlah *alternative* yang dipertimbangkan berdasarkan fungsi yang tertera pada tampilan aplikasi

   Pada saat tahap membuat *alternative*, jumlah *alternative* yang dihasilkan pada metode GSP berdasarkan jumlah *OR* yang ada pada *subgoal*-nya. Namun, dalam hal ini terdapat faktor tampilan sistem aplikasi yang menjadi bahan pertimbangan dan akan mempengaruhi jumlah *alternative* yang ada. Berdasarkan hasil penelitian,

requirements yang dihasilkan tetap sesuai dengan fungsi sistem dan justru akan lebih menyesuaikan dengan tampilan sistem aplikasi.

3.2. Pembahasan

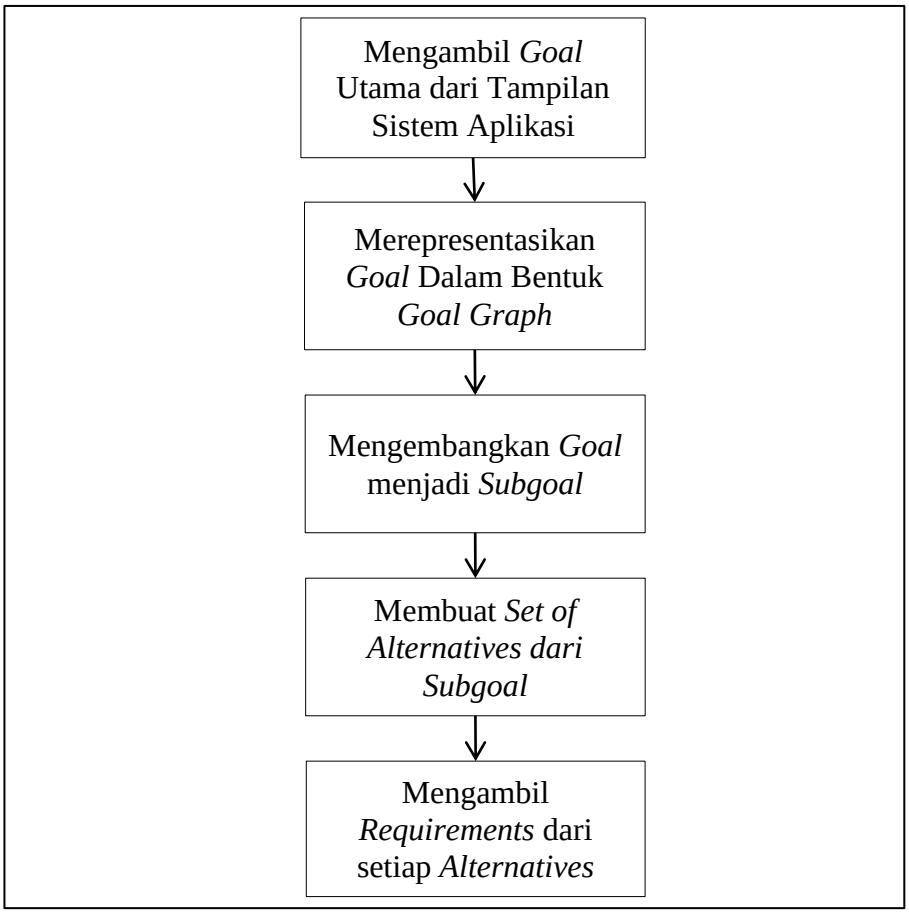
Setelah melalui proses *reverse* dengan menggunakan konsep GORE dan metode GSP, didapatkan hasil *requirements* seperti yang tertera pada Tabel 4.2., hasil tersebut didapat melalui penarikan *requirements* yang diambil dari *set of alternatives*. Setelah dibandingkan dengan hasil *reverse* konvensional, maka dapat dibuktikan bahwa hasilnya tidak berbeda jauh. Poin-poin penting *requirements* yang dihasilkan masih sejalan dengan *requirements* aslinya. Yang menjadi hasil utama dari penelitian ini adalah model GORE yang didapat setelah melakukan penerapan konsep GORE pada sebuah sistem yang sudah siap pakai. Model yang dihasilkan dapat digambarkan dalam bentuk diagram yaitu seperti pada Gambar 4.26., model GORE slebih efektif untuk melakukan *reverse engineering* pada sistem yang lebih kompleks, karena konsep dari GORE itu sendiri ialah merangkum semua *Goal* yang tercakup dalam sistem, untuk kemudian dikembangkan menjadi beberapa *subgoal* dalam bentuk *goal graph*. Selain itu, model GORE menjadikan *requirements* lebih presisi, karena *requirements* yang dihasilkan memiliki tujuan untuk memenuhi *goals* pada sistem. Model ini juga memudahkan analis maupun *stakeholder* lainnya dalam melakukan *reverse*. Konsep GORE juga menyediakan proses analisis dalam bentuk *high-level data*, sehingga para *stakeholder* dapat ikut berkontribusi dalam pengembangan dan perbaikan sistem.

BAB V

KESIMPULAN DAN SARAN

5.1. Kesimpulan

Berdasarkan hasil penelitian pada BAB IV dengan sampel sistem manajemen Penmaba UNJ 2015 Modul Admin, maka pada penelitian ini dihasilkan sebuah model *reverse engineering*. Adapun model *reverse* yang dihasilkan ialah model *Goal Oriented Requirements Engineering* (GORE) dengan metode GSP. Berikut merupakan model GORE dengan metode GSP pada Gambar 5.1.:



Gambar 5.1. Model GORE dengan metode GSP untuk Proses *Reverse Engineering*

Berdasarkan gambar 5.1., berikut merupakan langkah-langkah untuk menggunakan GORE dalam proses *reverse engineering*:

1. Mengambil *main goal* dari tampilan sistem aplikasi
2. Merepresentasikan *goal* dalam bentuk *goal graph*
3. Mengembangkan *goal* menjadi *subgoal*
4. Membuat *set of alternatives* dari *subgoal*
5. Mengambil *requirements* dari setiap *alternatives*

Adapun keunggulan dari model GORE dengan metode GSP ini adalah:

1. Model GORE menjadikan proses *reverse engineering* menjadi lebih efektif jika diproses pada sistem yang lebih kompleks, karena konsep GSP yang memetakan main *goal* pada awal proses, sehingga untuk mendapatkan *requirements* cukup dengan mengikuti alur *goal graph* pada *alternatives* yang sudah ada.
2. Requirements yang dihasilkan lebih rinci dan terstruktur, karena *requirements* diambil langsung dari *set of alternatives* yang direpresentasikan dalam bentuk *goal graph* melalui penelusuran vertikal mulai dari tingkat *main goal* sampai pada tingkat *technical support*, sehingga ketika *stakeholder* akan melakukan pengembangan dan perbaikan, maka prosesnya akan lebih mudah. *Stakeholder* hanya perlu mempelajari dari hasil *requirements* yang sudah ada.
3. Model ini juga menyediakan proses *reverse* yang berbasis *high-level-data*, sehingga *stakeholder* dapat lebih mudah mengerti dan ikut andil pada proses *reverse* dalam upaya perbaikan dan pengembangan sistem.

4. Requirements yang dihasilkan lebih presisi, karena *requirements* akan mengarah pada suatu *goal* yang memenuhi fungsi sistem.

Adapun sebagai bentuk kontribusi, dilakukan beberapa renovasi pada konsep GORE dan GSP itu sendiri, yaitu sebagai berikut:

1. Pemilihan konsep *user goal* sebagai dasar menemukan *requirements*.
2. Teknik analisa *interface* sebagai cara menemukan *goal* pada sistem.
3. Pembatasan jumlah *alternative* yang dipertimbangkan berdasarkan fungsi yang tertera pada tampilan aplikasi.

5.2. Saran

Atas dasar kesimpulan di atas maka disarankan:

1. Melakukan penelitian dalam bidang *reverse engineering*, namun dengan menggunakan konsep model berbeda.
2. Melakukan perbandingan kinerja model GORE dengan kinerja model lainnya dalam melakukan proses *reverse engineering*.
3. Melakukan penelitian model GORE dengan metode GSP menggunakan unsur GSP yang lain yaitu *skill* dan *preferences*.
4. Mengembangkan model GORE yang dapat menghasilkan *requirements* yang lebih baik.
5. Membuat perumusan dalam mencari jalur *alternatives* dalam *goal graph*, sehingga memudahkan proses *reverse engineering* ketika berhadapan dengan sistem yang memiliki *subgoal* lebih banyak.

DAFTAR PUSTAKA

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

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

Al Fatta, Hanif. 2007. *Analisis dan Perancangan Sistem Informasi Untuk Keunggulan Bersaing Perusahaan dan Organisasi Modern*. Yogyakarta: Andi Offset.

Anwer, S. & Ikram, N. 2006. *Goal Oriented Requirement Engineering: A Critical Study of Techniques*, XIII ASIA PACIFIC SOFTWARE ENGINEERING CONFERENCE (APSEC'06).

Creswell, John W. 2013. *Research Design: Qualitative, Quantitative, and Mixed Methods Approaches*. Los Angeles: SAGE Publications.

Gaol, J.L. 2008. *Sistem Informasi Manajemen*. Jakarta: Grasindo.

Hui, B.; Liaskos, S.; & Mylopoulos, J. 2003. *Requirements Analysis for Customizable Software: A Goals-Skills-Preferences Framework*. Department of Computer Science, University of Toronto.

Indrajit, Richardus E. & Djokopranoto, Richardus. 2011. *Proses Bisnis Outsourcing*. Jakarta: Grasindo.

Kusrini. 2007. *Strategi Perancangan dan Pengelolaan Basis Data*. Yogyakarta: Andi Offset.

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

Nofriansyah, Dicky. 2015. *Konsep Data Mining Vs Sistem Pendukung Keputusan*. Yogyakarta: Deepublish.

Raco, J.R. 2010. *Metode Penelitian Kualitatif (Jenis, Metode, dan Keunggulannya)*. Jakarta Grasindo.

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

Shofi, Imam M. & Budiarjo, Eko K. 2011. *Ontologi OWL untuk merepresentasikan Framework GSP pada GORE*. KNTIA 2011.

Simarmata, Janner. 2010. *Rekayasa Perangkat Lunak*. Yogyakarta: Andi Offset.

Simarmata, Janner. 2010. *Rekayasa Web*. Yogyakarta: Andi Offset.

Singhal, Aabhas & Gandhi, Shlok. 2014. *Reverse Engineering*. India: International Journal of Computer Applications.

Sommerville, Ian. 2009. *Software Engineering*. Boston: Pearson Education.

Sunarto. 2005. *Teknologi Informasi dan Komunikasi (untuk kelas X)*. Jakarta: Grasindo.

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

Van Lamsweerde, Axel. 2001. *Goal-Oriented Requirements Engineering: A Guided Tour*. De‘partement d’lngbnierie Informatique, Universite‘ catholique de Louvain.

Yuhefizard. 2008. *Database Management Menggunakan Ms. Access 2003*. Jakarta: Elex Media Komputindo.

TENTANG PENULIS

Fauziah Rizqy, merupakan anak perempuan kelahiran Batam, 15 Agustus 1994 dan juga merupakan anak dari Bapak Kadarisman dan Ibu Asnah. Sejarah pendidikan yang ditempuh oleh penulis, yaitu: (1) Tahun 2000 menempuh pendidikan dasar di SD Negeri 010 Batam; (2) Tahun 2000-2006 menempuh pendidikan dasar di SD Negeri 011 Tanjungpinang; (3) Tahun 2006-2009 menempuh pendidikan menengah pertama di SMP Negeri 7 Tanjungpinang; dan (4) 2009-2012 menempuh pendidikan menengah atas di SMA Negeri 2 Tanjungpinang.

Penulis memiliki minat di bidang perangkat lunak dan pemrograman komputer. Penulis memiliki prestasi di bidang komputer sejak tahun 2010 yaitu meraih juara harapan 2 Olimpiade Sains Nasional (OSN) Komputer Tingkat Kota Tanjungpinang pada tahun 2010 dan 2011. Penulis mulai mengenali tentang perangkat lunak pada saat semester lima ketika menempuh pendidikan di Universitas Negeri Jakarta.