BAB II

LANDASAN TEORI

2.1 Telepon Genggam

Pada jaman yang telah maju dimana semua informasi dapat diperoleh dan disampaikan dengan cepat melalui media internet, akan merubah gaya orang bertukar informasi. Jika pada awalnya informasi disampaikan dan diperoleh dari perbincangan mulut ke mulut, kemudian berkembang menjadi melalui media cetak dan elektronik, saat ini informasi dapat disampaikan dan diperoleh melalui media yang terbilang relatif kecil, ringan dan hampir dimiliki oleh semua orang, yaitu telepon genggam.

Pada perkembangannya telepon genggam tidak lagi hanya berfungsi untuk melakukan atau menerima panggilan, namun juga dapat digunakan untuk mengirim dan menerima pesan singkat dalam bentuk SMS. Ketika kebutuhan manusia akan telepon genggam semakin berkembang, maka berkembang dan bertambah pula kemampuan yang dimiliki perangkat ini. Selain dapat memproses suara dan text, telepon genggam juga dapat memproses data yang dikirim dan diterima melalui media internet. Pada beberapa telepon genggam juga dibenamkan fungsi-fungsi yang lebih lengkap dan hampir menyerupai sebuah perangkat computer. Sebut saja fitur *Global Positioning System* atau lebih dikenal dengan GPS. Dengan fitur ini sebuah perangkat telepon genggam dapat mengetahui koordinat dimana perangkat ini berada dan biasanya fitur ini dipadukan dengan sebuah peta. Selain daripada itu, sebuah telepon genggam juga

memiliki sebuah sistem operasi untuk menunjang kegiatan pada perangkat tersebut.

Dengan semakin mutakhirnya sebuah perangkat telepon genggam, tentu semakin tinggi juga harapan pengguna terhadap perangkat tersebut. Pengguna mengharapkan aplikasi-aplikasi yang tersedia dapat merubah gaya hidup mereka sehari-hari menjadi lebih mudah.

Salah satu sistem operasi yang tengah berkembang adalah Android. Android merupakan sistem operasi yang bersifat terbuka, dapat dikembangkan oleh siapa saja dan disponsori oleh Google. Dengan Android para pengembang aplikasi telepon genggam dapat menggunakan kemampuan telepon genggamnya dengan maksimal.

2.1 Android

Aplikasi Android dibuat dengan bahasa pemrograman Java dan di *install* menggunakan ekstensi *.apk* (Meier, 2012).

2.1.1 Activity

Activity mewakili tampilan yang dilihat oleh pengguna dan hanya memiliki satu fungsi. Satu aplikasi dapat memiliki satu atau banyak Activity tergantung pada jenis aplikasinya. Satu aplikasi dapat membuat Activity baru atau menggunakan Activity yang sudah ada dari aplikasi lain (Meier, 2012).

2.1.2 Service

Service tidak memiliki tampilan, tapi berjalan sebagai *background process* untuk waktu yang tidak ditentukan. Sebagai contoh, aplikasi yang memainkan musik akan memiliki Service yang terus berjalan walaupun pengguna sedang membuka aplikasi lain. Pengguna kemudian dapat kembali ke aplikasi pemutar musik untuk mematikan Service nya. Walaupun tidak memiliki tampilan, Service dapat memanggil Activity untuk menerima masukan dari pengguna.

2.1.3 Broadcast Receiver

Merupakan komponen yang berfungsi untuk menerima dan berreaksi terdapat pengumuman yang bersifat tersiar. Contohnya, terjadi perubahan zona waktu, baterai dalam kondisi kritis, telah terjadi pengambilan gambar melalui kamera atau pengguna merubah pilihan bahasa. Aplikasi juga dapat memulai siaran, misalnya telah selesai melakukan pengunduhan file.

2.1.4 Content Provider

Content Provider memungkinkan data yang digunakan pada satu aplikasi tersedia untuk aplikasi yang lain. Data dapat disimpan dalam sistem file, basis data SQL atau cara apapun yang memungkinkan.

2.1.5 Intent

Satu komponen dapat mengaktifkan komponen yang lain melalui Intent yang berisi informasi-informasi yang dibutuhkan untuk mengaktifkan komponen lain dan informasi yang dibutuhkan oleh komponen tersebut.

2.2 Global Positioning System

GPS (*Global Positioning System*) adalah system navigasi satelit milik Amerika Serikat yang menyediakan layanan pemetaan posisi, navigasi dan waktu kepada pengguna di seluruh dunia (Garmin, 2008).

2.2.1 Bagian-bagian GPS

GPS terdiri dari 3 bagian, yaitu:

a. Space

Terdiri dari 24 sampai 32 satelit pada orbit medium termasuk pendorong yang digunakan untuk menjaga orbitnya

b. Control

Terdiri dari *Master Control Station*, *Alternate Master Control Station* dan kumpulan antena serta stasiun pengamat

c. User

Adalah perangkat yang menerima sinyal dari bagian *Space* dan menyediakan lokasi secara 3 dimensi (latitude, longitude dan altitude)

2.2.2 Cara Kerja GPS

GPS memiliki 24 sampai 32 satelit pada orbit medium, sekitar 19.300 km, di atas permukaan laut. Setiap harinya satelit ini melakukan 2 rotasi mengelilingi bumi. Orbitnya diatur sedemikian sehingga setiap saat dimanapun di bumi terdapat minimal 4 satelit yang terlihat di langit.

Tugas penerima GPS adalah menentukan lokasi dari 4 satelit atau lebih yang berdekatan dan menentukan jarak antara satelit-satelit tersebut. Informasi ini

digunakan untuk menentukan lokasi dari penerima itu sendiri. Operasi ini didasari pada prinsip matemática yang sederhana yang disebut Trilaterasi**.**

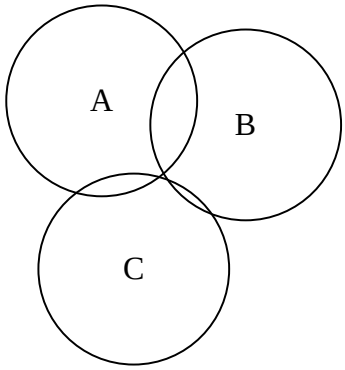
a. Trilaterasi 2 Dimensi

Jika kita berada pada suatu tempat yang kita tidak mengetahui dimana lokasinya, sebut saja lokasi X, kemudian kita bertanya pada seseorang dan mendapatkan informasi kalau kita berada 60 km dari kota A. Kita masih tidak akan mengetahui lokasi pastinya. Karena kita bisa saja berada pada radius 60 km dari kota A dan itu bisa berada dimana saja.

Kemudian kita bertanya pada orang kedua dan mendapat informasi bahwa kita berada 50 km dari kota B. Dengan menggabungkan kedua informasi tersebut, kita mengetahui bahwa kita berada pada 2 kemungkinan lokasi yang merupakan titik potong radius antara kota A dan kota B.

Kemudian kita kembali bertanya kepada orang ketiga dan mendapat informasi bahwa kita berada 70 km dari kota C. Dengan menggabungkan ketiga informasi yang telah kita dapat kita dapat mengetahui dengan pasti lokasi kita. Karena hanya ada satu titik potong radius antara 3 kota tersebut.

Gambarannya seperti gambar berikut:



Gambar 2.1 Trilaterasi 2 dimensi

b. Trilaterasi 3 Dimensi

Sama halnya seperti trilaterasi 2 dimensi yang menentukan posisi berdasarkan radius lingkaran, pada trilaterasi 3 dimensi penentuan posisi berdasarkan radius bola.

Jika penerima mengetahui jaraknya dengan satelit 1, maka penerima dapat berada dimana saja pada radius sebuah bola. Jika penerima kemudian mengetahui jaraknya dari satelit 2, maka perpotongan 2 bola tersebut akan berbentuk lingkaran, dan penerima dapat berada dimana saja di lingkaran tersebut. Jika penerima mengetahui jaraknya dari satelit 3, perpotongan radiusnya akan menghasilkan 2 titik, 1 di langit dan satu di bumi. Perpotongan yang berada di langit akan diabaikan dan dengan demikian penerima dapat mengetahui posisinya di bumi. Satelit ke-4 digunakan sebagai penunjang akurasi perhitungan.

2.2.3 *Assisted* GPS

GPS bekerja menggunakan sinyal radio dari satelit. Akan tetapi di kota-kota besar sinyal ini sangat sulit didapat karena adanya gedung-gedung tinggi dan menjadi sangat lemah karena harus melalui tembok-tembok tebal. *Assisted* GPS dapat mengatasi masalah ini dengan menggunakan data dari sebuah jaringan.

Server asistensi menyediakan data orbit satelit untuk penerima sehingga penerima dapat mengunci sinyal lebih cepat. Jaringan juga dapat menyediakan waktu yang tepat. Server asistensi juga mendapat sinyal GPS lebih baik

2.3. Siklus Pengembangan Sistem

Siklus hidup pengembangan sistem dalam rekayasa sistem dan rekayasa perangkat lunak, adalah proses pembuatan dan pengubahan sistem serta model dan metodologi yang digunakan untuk mengembangkan sistem-sistem tersebut. Konsep ini umumnya merujuk pada sistem komputer atau informasi.

Dalam rekayasa perangkat lunak, konsep ini mendasari berbagai jenis metodologi pengembangan perangkat lunak. Metodologi-metodologi ini membentuk suatu kerangka kerja untuk perencanaan dan pengendalian pembuatan sistem informasi, yaitu proses pengembangan perangkat lunak.

Tahapan pengembangan yang penting bagi pengembang adalah: perencanaan, analisa, perancangan dan implementasi.

Perencanaan merupakan tahapan kegiatan membuat rencana proyek dan dokumen-dokumen perencanaan lainnya. Tahapan ini menyediakan dasar untuk mendapatkan informasi-informasi yang dibutuhkan untuk mencapai solusi.

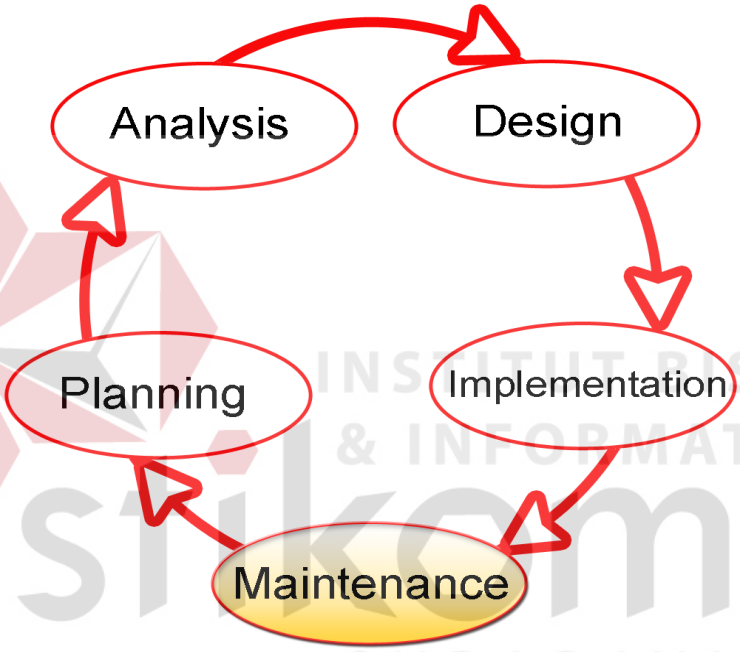
Tujuan dari tahap analisa adalah untuk menentukan masalah dalam usaha untuk menyelesaikannya. Tahapan ini melibatkan usaha untuk memecah sistem kedalam beberapa bagian untuk menganalisa situasi, tujuan proyek, memecah apa yang perlu dikembangkan dan bersama pengguna mendefinisikan kebutuhan.

Dalam sistem, merancang fungsi dan operasi digambarkan secara detail. Hasil dari tahapan ini adalah sistem yang baru sebagai kumpulan modul dan subsistem. Tahap perancangan menggunakan dokumen kebutuhan sebagai input. Untuk setiap kebutuhan, satu atau lebih elemen perancangan akan dihasilkan. Elemen perancangan menggambarkan perangkat lunak yang diharapkan dalam

bentuk diagram, bisnis proses dan hubungan entitas dengan kamus data yang lengkap.

Pada tahap implementasi, modul dan subsistem akan diimplementasikan kedalam kode. Tes unit dan tes modul dilakukan pada tahap ini.

Tahapan tersebut dapat dilihat pada gambar berikut:



Gambar 2.2 Siklus Pengembangan Sistem

Salah satu metode dalam siklus pengembangan sistem adalah metode analisa dan perancangan berorientasi objek. Pada metode ini dilakukan beberapa penyesuaian agar tahapan-tahapan umum dapat dilakukan sesuai dengan paradigm berorientasi objek.

Pada metode analisa dan perancangan objek, tahap perencanaan dijadikan satu dengan tahap analisa menjadi tahap mendapatkan dan analisa kebutuhan.

Tahap perancangan dipecah menjadi dua yaitu perancangan sistem dan perancangan objek.

Tahap implementasi tetap digunakan untuk mengimplementasi hasil perancangan.

Penjelasan dari masing-masing tahapan tersebut dijelaskan pada bagian berikutnya.

2.4 Analisa Dan Perancangan Berorientasi Objek

2.4.1 Mendapatkat Kebutuhan

Mendapatkan kebutuhan berfokus pada kegiatan menjabarkan tujuan dari sistem dan apa yang dapat dilakukan oleh sistem. Hal ini dilihat dari sudut pandang pengguna. Antara lain, fungsi – fungsi sistem, interaksi antara pengguna dan sistem, kesalahan yang dapat dideteksi dan ditangani oleh sistem dan kondisi-kondisi sekitar dimana sistem tersebut ditempatkan.

Aktivitasnya adalah:

a. Identifikasi Aktor

Aktor mewakili entitas luar yang berinteraksi dengan sistem. Aktor dapat berupa manusia atau sistem yang lain. Aktor adalah peran secara abstrak dan tidak selalu merujuk kepada seseorang.

Pertanyaan-pertanyaan yang dapat diajukan untuk mengidentifikasi aktor adalah:

i. Kelompok pengguna mana yang mendapatkan dukungan sistem dalam menyelesaikan pekerjaannya?

ii. Kelompok pengguna mana yang menggunakan fungsi utama dari sistem?

iii. Kelompok pengguna mana yang melakukan fungsi kedua, seperti pemeliharaan dan administrasi?

iv. Dengan perangkat keras atau lunak apa sistem akan berinteraksi?

b. Identifikasi Skenario

Skenario adalah ”deskripsi naratif dari apa yang orang lakukan dan alami ketika mereka mencoba menggunakan sistem dan aplikasi komputer” (Carroll,1995). Skenario adalah deskripsi nyata, terfokus dan non formal dari sebuah fitur pada sistem dari sudut pandang aktor. Hal-hal yang perlu dideskripsikan dalam skenario:

i. nama skenario

ii. contoh aktor yang berpartisipasi

iii. alur kejadian

Skenario tidak dapat mengandung deskripsi dari suatu keputusan. Untuk mendeskripsikan keputusan, dua skenario harus dibuat untuk masing-masing keputusan.

Untuk mengidentifikasi skenario, digunakan pertanyaan sebagai berikut:

a. Tugas apa saja yang diinginkan aktor terhadap sistem

b. Informasi apa saja yang diakses aktor? Siapa yang membuat data? Apakah data dapat diubah atau dihapus? Oleh siapa?

c. Perubahan apa saja yang perlu diinformasikan aktor kepada sistem, seberapa sering dan kapan?

d. Kejadian apa saja yang perlu diinformasikan sistem kepada aktor?

c. Identifikasi Use Case

Skenario adalah contoh dari Use Case, sehingga Use Case adalah spesifikasi dari semua kemungkinan skenario untuk sebuah fungsionalitas.

Use Case dimulai oleh sebuah aktor. Setelah dimulai, sebuah Use Case dapat berinteraksi dengan aktor lainnya. Sebuah Use Case mewakili sebuah alur kejadian yang lengkap yang menggambarkan serangkaian interaksi yang terkait sebagai hasil dari inisiasi.

Hal-hal yang perlu digambarkan dalam sebuah Use Case adalah:

i. Nama Use Case : nama dari Use Case sebaiknya merupakan frase kata kerja yang mengindikasikan kegiatan yang ingin diselesaikan Aktor.
ii. Aktor yang terlibat : Aktor yang memulai dan aktor-aktor yang berinteraksi. Sebaiknya merupakan kata benda
iii. Alur Kejadian
iv. Kondisi Awal : Kondisi yang dibutuhkan untuk memulai Use Case
v. Kondisi Akhir : Kondisi ketika Use Case telah selesai

Dalam membuat Use Case, sebaiknya tidak menggambarkan antarmuka pengguna agar pembuatan Use Case tidak terpengaruh.

d. Identifikasi Hubungan Aktor dan Use Case

Hubungan antara Aktor dan Use Case memungkinkan pembuat dan pengguna untuk mengurangi kompleksitas model dan menambah tingkat pengertiannya.

Kita menggunakan hubungan komunikasi antara aktor dan Use Case untuk menggambarkan sistem dalam lapisan fungsionalitas.

Kita menggunakan hubungan ”mengembangkan” untuk memisahkan alur kejadian ”pengecualian” dan sama.

Kita menggunakan hubungan ”menyertakan” untuk mengurangi redundan diantara Use Case

e. Identifikasi Objek Analisa Awal

Salah satu rintangan awal ketika pembuat dan pengguna mulai berkolaborasi adalah kesalahan dalam perbedaan terminologi. Kesalahan dapat terjadi karena istilah yang sama digunakan pada area yang berbeda dan memiliki arti yang berbeda.

Untuk membangun sebuah terminologi yang jelas, pembuat mengidentifikasi objek-objek yang terlibat untuk masing-masing Use Case. Pembuat harus mengidentifkasi, memberi nama dan menggambarkan serta mendokumentasikan kedalam kamus data. Membuat kamus data merupakan langkah awal menuju tahap analisa.

Panduan untuk mengidentifikasikan Objek Analisa Awal adalah:

i. Istilah-istilah yang harus dijelaskan oleh pembuat dan pengguna untuk mengerti Use Case
ii. Kata benda yang tampil berkali-kali dalam Use Case
iii. Entitas dunia nyata yang harus dilacak oleh sistem
iv. Proses dunia nyata yang harus dilacak oleh sistem
v. Use Case itu sendiri
vi. Sumber data
vii. Benda-benda yang berinteraksi dengan pengguna
viii. Selalu gunakan istilah dalam area aplikasi

Pada tahap mencari kebutuhan, objek-objek yang berpartisipasi dihasilkan untuk setiap Use Case. Jika dua Use Case mereferensi pada objek yang sama, definisi objek tersebut harus sama. Jika dua Use Case menggunakan nama yang sama untuk konsep yang berbeda, maka salah satu atau kedua nama harus diganti untuk memperjelas perbedaan mereka. Konsolidasi ini menghilangkan kerancuan dalam terminologi yang digunakan.

2.4.2 Analisa Kebutuhan

a. Identifikasi Objek Entitas

Objek Entitas adalah objek yang memiliki data dan disimpan secara permanen oleh aplikasi. Objek jenis ini, keberadaannya dimonitor oleh aplikasi. Cara mengidentifikasi objek ini adalah dengan mengenali objek-objek mana yang merupakan kata benda, sering digunakan pada Use Case dan memiliki data-data.

Secara konvensi, penulisan objek entitas menggunakan kata benda. Untuk membedakan, juga digunakan *Stereotype* <<*entity*>>.

b. Identifikasi Objek Batas

Objek batasan adalah objek yang menjadi tempat interaksi antara aktor dengan aplikasi. Objek ini dikenali sebagai tempat aktor mengisi data, menerima konfirmasi dari aplikasi dan berinteraksi dengan aplikasi. Tombol, Formulir, Kotak Dialog dan Kotak Pemberitahuan, adalah contoh dari objek Batasan.

Secara konvensi, penulisan objek batasan diawali dengan jenis batasannya, seperti TombolMulai, FormulirPendaftaran, DialogSimpan. Menggunakan *Stereotype* <<*boundary*>>.

c. Identifikasi Objek Kendali

Objek kendali merupakan objek yang mengendalikan berjalannya sebuah Use Case. 1 Use Case biasanya memiliki 1 objek kendali.

Dalam penulisannya menggunakan prefiks “Kendali”, seperti Kendali Pendaftaran Pengguna, Kendali Menambah Lokasi. Menggunakan *Stereotype* *<<control>>*.

d. Pemetaan Use Case ke Objek dengan Diagram Urutan

Sebuah diagram urutan, menghubungkan use case–use case dengan objek-objek. Diagram ini menunjukan bagaimana tingkah laku sebuah use case didistribusikan ke objek-objek yang berpartisipasi. Diagram ini juga menunjukan interaksi antara objek-objek.

Dalam pembuatan sebuah diagram urutan, digunakan petunjuk sebagai berikut:

i. kolom pertama harus mewakili aktor yang memulai use case
ii. kolom kedua harus mewakili objek batas yang digunakan aktor untuk memulai use case
iii. kolom ketiga harus mewakili objek kendali yang mengatur keseluruhan use case.
iv. Objek kendali diinisiasi oleh objek batas yang memulai use case.
v. Objek batas diinisiasi oleh objek kendali.
vi. Objek entitas digunakan oleh objek kendali dan batas.
vii. Objek entitas tidak diperbolehkan untuk menggunakan objek batas atau kendali.

e. Identifikasi Asosiasi

Diagram Kelas digunakan untuk menggambarkan asosiasi diantara objek-objek dan juga atributnya.

Sebuah asosiasi menunjukan sebuah hubungan antara dua atau lebih kelas. Mengidentifikasi asosiasi mempunyai dua keuntungan. Pertama, memperjelas model analisa dengan membuat hubungan antara objek secara eksplisit. Kedua, memunkinkan pembuat aplikasi untuk menemukan kasus-kasus batasan yang terkait dengan hubungan. Misalnya apakah kelas tertentu dapat diakses oleh satu atau lebih kelas.

Asosiasi mempunyai beberapa properti:

i. Nama, untuk menggambarkan asosiasi antara dua kelas.
ii. Peran, pada setiap titik kelas, untuk mengidentifikasi fungsi dari masing-masing kelas dalam setiap asosiasi.
iii. *Multiplicity*, pada setiap titik kelas, untuk mengidentifikasi jumlah nyata kelas yang diperbolehkan.

f. Identifikasi Agregat

Agregat adalah bentuk khusus dari asosiasi yang menggambarkan hubungan *whole-part*. Jika satu objek merupakan bagian dari dari objek lain, maka hubungannya adalah agregat

g. Identifikasi Atribut

Atribut adalah properti dari objek individual yang berisi kondisi dari objek tersebut. Dalam mengidentifikasi atribut, hanya atribut-atribut yang terkait dengan aplikasi saja yang dipertimbangkan.

Atribut mempunyai beberapa properti:

i. Nama, mengidentifikasi atribut pada objek tertentu.

ii. Deskripsi mengenai atribut tersebut.

iii. Tipe, menunjukan nilai-nilai apa saja yang dapat diterima oleh atribut tersebut.

h. Pemodelan Tingkah Laku berdasarkan Status Objek

Diagram status menggambarkan tingkah laku dilihat dari sudut pandang satu objek. Melihat tingkah laku satu objek memungkinkan pembuat aplikasi untuk membangun deskripsi yang lebih formal dari tingkah laku objek dan akibatnya dapat mengidentifikasi use case yang belum ada. Tidak semua objek membutuhkan diagram status. Hanya objek yang memiliki waktu hidup lebih lama dan tingkah laku yang bergantung pada status yang perlu dibuatkan diagram status. Itu berarti, semua objek kendali, sebagian objek entitas dan tidak berlaku bagi objek batas.

i. Pemodelan Hubungan Pewarisan antara objek

Generalisasi digunakan untuk menghilangkan redundan pada model analisa. Jika dua atau lebih kelas memiliki atribut atau tingkah laku yang sama, maka kesamaan tersebut dikonsolidasi kedalam sebuah kelas super.

2.4.3 Perancangan Sistem

Perancangan sistem adalah proses transformasi sebuah model analisa ke dalam sebuah model perancangan sistem. Selama perancangan sistem, pembuat mendefinisikan tujuan perancangan aplikasi dan memecah sistem menjadi sub sistem yang lebih kecil yang dapat direalisasikan oleh tim-tim secara individu.

Pembuat juga memilih strategi untuk membangun sistem seperti strategi perangkat keras dan lunak, strategi penyimpanan data, aliran data secara umum, kebijakan kendali akses dan mengatasi masalah kondisi batasan. Hasil dari perancangan sistem adalah sebuah model yang memuat sebuah dekomposisi subsistem dan deskripsi yang jelas dari masing-masing strategi

a. Identifikasi Tujuan Rancangan

Definisi tujuan perancangan adalah langkah pertama dari perancangan sistem. Hal ini mengidentifikasi kualitas yang harus menjadi fokus pembuatan sistem. Banyak tujuan perancangan dapat di ambil dari kebutuhan non fungsional atau domain aplikasi.

Secara umum, tujuan perancangan dapat dibagi menjadi kriteria-kriteria sebagai berikut:

i. Kriteria *performance*, termasuk kecepatan dan kebutuhan kapasitas.
ii. Kriteria keandalan, menentukan berapa besar usaha yang perlu dilakukan untuk meminimalkan kesalahan dan akibatnya.
iii. Kriteria biaya, termasuk biaya untuk pembuatan sistem, penempatannya dan pengaturannya.Jika sistem menggantikan sistem yang telah ada, berapa biaya untuk memastikan bahwa sistem baru dapat berjalan menggunakan data-data lama.
iv. Kriteria pemeliharaan, menentukan seberapa sulit untuk merubah sistem setelah sistem dipasang. Seberapa mudah untuk menambah fungsi baru.
v. Kriteria pengguna akhir, termasuk kualitas yang diharapkan oleh pengguna namun belum diakomodir oleh kriteria *performance* dan

keandalan. Apakah aplikasi sulit untuk digunakan dan dipelajari. Apakah pengguna dapat menyelesaikan tugasnya menggunakan sistem.

b. Identifikasi Subsistem

Menemukan subsistem selama perancangan sistem hampir sama dengan menemukan objek selama analisa.

Langkah awal pemecahan subsistem diambil dari kebutuhan fungsional. Langkah lainnya adalah dengan menjadikan satu objek-objek yang terkait secara fungsional. Objek-objek dalam satu use case dimasukan kedalam subsistem sendiri. Jika ada objek-objek yang digunakan pada use case lain, maka dibuatkan subsistem sendiri atau menempatkan objek tersebut pada subsistem yang membuat mereka.

c. Pemetaan Subsistem ke *Processor* dan Komponen

Pemetaan subsistem ke *processor* dan komponen memiliki 2 tahap. Tahapannya adalah: memilih konfigurasi perangkat keras dan *platform*, kemudian mengalokasi objek-objek dan subsistem ke *node-node*.

Banyak sistem yang berjalan pada lebih dari satu sistem dan bergantung pada akses intranet atau internet. Penggunaan banyak komputer dapat mewujudkan kebutuhan akan kinerja yang tinggi dan keterhubungan beberapa pengguna yang terdistribusi. Konsekuensinya, kita perlu melakukan alokasi subsistem kedalam komputer dan rancangan infrastruktur untuk mendukung komunikasi antara subsistem tersebut secara hati-hati. Komputer-komputer ini dimodelkan sebagai node di dalam diagram *deployment* UML.

Memilih konfigurasi perangkat keras juga melakukan pemilihan mesin virtual tempat sistem akan dibangun. Mesin virtual termasuk sistem operasi dan

perangkat lunak apa saja yang dibutuhkan, seperti sistem basis data atau sistem komunikasi. Pemilihan sebuah mesin virtual mengurangi jarak antara sistem dan *platform* perangkat keras tempat sistem akan dijalankan. Semakin banyak fungsi yang disediakan oleh komponen mesin virtual, semakin sedikit pekerjaan pengembangan yang harus dilakukan.

Setelah konfigurasi perangkat keras telah ditentukan dan mesin-mesin virtual telah dipilih, objek-objek dan subsistem-subsistem ditempatkan pada *node-node* nya. Kadang-kadang hal ini memicu identifikasi objek-objek dan subsistem-subsistem baru untuk memindahkan data antara *node*.

d. Identifikasi Penyimpanan Data

Penyimpanan dibutuhkan agar data dapat tetap diakses walaupun setelah sistem dimatikan atau mengalami kegagalan.

Dalam mengidentifikasikan data yang perlu disimpan, kita dapat mengambil sebagian atau seluruh objek-objek entitas. Namun tidak hanya objek entitas yang perlu disimpan, data yang terkait dengan keadaan sistem juga perlu disimpan. Data seperti alamat *webserver*, parameter koneksi hubungan basis data dan data-data lain yang dibutuhkan ketika sistem pertama kali diaktifkan.

Setelah diidentifikasi data yang perlu disimpan, kita perlu memilih strategi penyimpanan datanya. Ada beberapa strategi yang dapat dipilih seperti pada tabel berikut:

Tabel 2.1 Strategi Penyimpanan Data

No	Strategi	Deskripsi
1	Sistem Berkas / *File*	Aplikasi menyimpan data sebagai urutan biner dan menentukan bagaimana dan bila data diambil. Memiliki kecepatan yang tinggi namun membutuhkan pengaturan lebih lanjut jika ada akses secara bersamaan
2	Sistem basis data relasional	Data disimpan dalam tabel yang telah ditentukan dalam skema. Setiap kolom mewakili atribut. Setiap baris mewakili data tunggal sebagai gabungan dari beberapa atribut. Pemetaan model objek yang kompleks menjadi tantangan tersendiri. Sistem basis data menyediakan fungsi untuk mengatur akses secara bersamaan dan mengembalikan data setelah terjadi kegagalan.
3	Sistem basis data berorientasi objek	Sistem ini menyediakan fungsi hampir sama dengan basis data relasional. Yang membedakannya adalah data disimpan sebagai objek-objek beserta hubungannya dengan objek lain. Sistem ini mengurangi waktu yang dibutuhkan oleh pengembang untuk

		pengembangan basis data awal. Akan tetapi sistem ini lebih lambat daripada sistem relasional untuk pencarian yang sama dan lebih sulit untuk di optimasi.

e. Penyediaan Kendali Akses

Pada sistem dengan banyak pengguna, aktor yang berbeda memiliki akses ke fungsi dan data yang berbeda. Selama proses analisa, kita memodelkan kondisi ini dengan mengasosiasikan use case yang berbeda untuk aktor yang berbeda. Pada proses perancangan sistem, kita memodelkan akses dengan menentukan objek mana yang digunakan bersama oleh para aktor dan menentukan bagaimana aktor dapat mengatur akses.

Secara umum, kita perlu mendefinisikan untuk setiap aktor, operasi mana yang dapat di akses untuk objek yang digunakan secara bersamaan. Kita memodelkan akses terhadap kelas-kelas dengan sebuah matriks akses. Baris dari matriks mewakili aktor dari sistem. Kolom-kolom mewakili kelas-kelas yang kita atur aksesnya. Pertemuan baris dan kolom disebut hak akses dan didaftarkan operasi-operasi yang dapat dijalankan oleh aktor terhadap kelas tertentu.

Matriks akses dapat digambarkan menggunakan salah satu dari 3 pendekatan.

i. tabel akses umum, mewakili setiap pertemuan dalam matriks sebagai suatu data. Untuk menentukan apakah sebuah aktor memiliki akses terhadap sebuah objek spesifik membutuhkan pencarian terhadap data. Jika terdapat data, maka akses ditolak.

ii. daftar akses kontrol, mengasosiasikan daftar aktor dan operasi terhadap kelas yang diakses. Setiap waktu sebuah objek diakses, daftar aksesnya diperiksa terhadap aktor dan operasi yang memanggilnya.

iii. Kapabilitas, mengasosiasikan kelas dan operasi dengan aktor. Setiap waktu sebuah aktor mencoba untuk mengakses kelas dan operasi tertentu dilakukan pengecekan apakah aktor tersebut memiliki kapabilitas tersebut, jika tidak, maka akses ditolak.

Pemilihan pendekatan mempengaruhi kinerja. Tabel akses umum membutuhkan ruang yang besar namun dapat diketahui hak akses secara keseluruhan. Daftar akses kontrol dan kapabilitas tidak membutuhkan ruang yang besar akan tetapi hanya diketahui hak dari aktor atau kelas tertentu.

Sebuah matriks akses hanya mewakili kendali akses statis. Artinya hak akses dapat dimodelkan sebagai atribut dari objek pada sistem. Hak akses bersifat umum untuk semua objek.

Jika kita menginginkan agar objek tertentu memiliki hak akses yang berbeda dengan objek kelas yang sama, maka kita membutuhkan kendali akses dinamis. Pada kendali akses dinamis, dibutuhkan satu subsistem untuk melakukan pengecekan apakah objek dengan identitas tertentu memiliki akses ke objek lain dengan identittas tertentu.

f. Merancang Kendali Aliran Umum

Aliran data adalah urutan aksi di dalam sebuah sistem. Pada sistem berorientasi objek, mengurutkan aksi termasuk menentukan operasi mana yang

harus dijalankan dan bagaimana urutannya. Pilihan ini didasarkan atas kejadian yang dilakukan aktor atau berdasarkan berjalannya waktu.

Terdapat tiga mekanisme kendali aliran:

i. kendali oleh prosedur, operasi menunggu masukan ketika membutuhkan data dari aktor.

ii. kendali oleh kejadian, sebuah perulangan utama menunggu untuk sebuah kejadian. Ketika sebuah kejadian terjadi, informasi disalurkan ke objek terkait.

iii. *Thread*, adalah variasi konkuren dari kendali oleh prosedur. Sistem dapat membuat beberapa *thread* masing-masing untuk kejadian yang berbeda. Jika sebuah *thread* membutuhkan data tambahan, dia akan menunggu masukan dari aktor tertentu. Jenis kendali ini adalah yang paling intuitif.

g. Identifikasi Kondisi Batasan

Mengidentifikasi kondisi batasan berarti memutuskan bagaimana sistem dimulai, diinisialisasi dan dimatikan. Kita juga perlu mendefinisikan bagaimana menangani kesalahan seperti kerusakan data dan tidak adanya jaringan, baik itu disebabkan oleh kesalahan piranti lunak atau kekurangan daya listrik. Use case yang digunakan disebut use case batasan.

Secara umum, kita mengidentifikasi use case batasan dengan mengamati setiap subsistem dan setiap objek persisten.

i. konfigurasi, untuk setiap objek, kita amati pada use case mana objek tersebut dibuat atau dihancurkan. Jika objek tidak dibuat atau dihancurkan

pada use case manapun, maka kita perlu menambah use case yang diakses oleh administrator sistem.

ii. Start-up dan shutdown, untuk setiap komponen, ktia tambahkan 3 use case untuk memulai, mematikan dan mengatur komponen.

iii. Penanganan pengecualian, untuk setiap kesalahan komponen, kita memutuskan bagaimana sistem harus bereaksi.

2.4.4 Perancangan Objek

Secara konsep, pengembangan sistem perangkat lunak mengisi jurang antara permasalahan dan kondisi perangkat keras yang ada.

Perancangan objek adalah tahap terakhir untuk menutup jurang tersebut. Tahap ini termasuk mengidentifikasi objek-objek solusi baru, menyesuaikan dengan komponen yang telah ada dan menentukan interface dan kelas subsistem dengan jelas. Sebagai hasilnya, model perancangan objek dapat dibagi menjadi kumpulan kelas-kelas yang dapat diimplementasikan oleh pengembang secara terpisah.

a. *Reuse* (Penggunaan Kembali)

Komponen yang telah ada dan teridentifikasi pada tahap perancangan sistem digunakan untuk membantu realisasi dari masing-masing subsistem. Pustaka kelas dan komponen tambahan dipilih untuk struktur data dasar dan layanan-layanan. Pola perancangan dipilih untuk menyelesaikan permasalahan yang hampir sama dan untuk melindungi kelas tertentu terhadap perubahan dimasa datang. Kadang-kadang komponen dan pola perancangan perlu disesuaikan sebelum digunakan.

Ini dilakukan dengan membungkus komponen yang telah ada dengan objek buatan atau dengan memperbaiki menggunakan pewarisan.

i. Pewarisan spesifikasi dan implementasi

Pewarisan merupakan salah satu bentuk reuse dimana ditentukan kelas orang tua dari sekelompok kelas. Fokus dari pewarisan selama perancangan objek adalah untuk mengurangi redundan. Dengan memindahkan tingkah laku redundan ke dalam 1 kelas orang tua, kita mengurangi resiko inkonsistensi jika ada perubahan. Penggunaan pewarisan yang semata-mata hanya untuk penggunaan kembali disebut pewarisan implementasi. Dengan pewarisan implementasi, pengembang menggunakan kembali kode secara cepat dengan membuat kelas anak dan memperjelas tingkah lakunya. Sebaliknya, klasifikasi konsep kedalam hirarki tipe disebut pewarisan spesifikasi.

ii. Delegasi

Delegasi adalah alternatif lain dari pewarisan implementasi yang harus digunakan ketika menginginkan penggunaan kembali. Sebuah kelas dikatakan mendelegasi kelas lain jika mengimplementasi operasi dengan mengirim kembali pesan ke kelas lain. Delegasi memperjelas ketergantungan antara kelas yang digunakan kembali dengan kelas yang baru. Pengguna kelas baru tidak perlu mengetahui jika terjadi perubahan delegasi. Sementara pengguna kelas yang digunakan kembali tidak mengalami perubahan tingkah laku.

iii. Pola Perancangan

Ada beberapa pola perancangan yang menggunakan teknik penggunaan kembali seperti dapat dilihat pada tabel berikut:

Tabel 2.2 Pola Perancangan

No	Pola	Deskripsi
1	*Bridge*	Pola ini memisahkan *interface* dengan implementasinya. Kelas pengguna hanya perlu mengetahui *interface* nya saja
2	*Adapter*	Pola ini membungkus komponen lama sehingga masih dapat digunakan untuk sistem yang baru.
3	*Strategy*	Pola ini memisahkan antara algoritma dengan implementasi
4	*Abstract Factory*	Pola ini membungkus pembuatan dari objek-objek terkait sehingga pengguna tidak dapat menggunakan objek dari hirarki yang berbeda
5	*Command*	Pola ini memisahkan tanggung jawab objek untuk memproses perintah dari perintah itu sendiri.
6	*Composite*	Pola ini membungkus hirarki dengan menyediakan kelas orang tua yang umum untuk agregat.

b. Spesifikasi Interface

Selama tahap ini, layanan subsistem yang diidentifikasi pada perancangan sistem dispesifikasikan dalam istilah interface kelas, termasuk operasi, argumen, tipe dan eksepsi. Operasi dan objek tambahan yang dibutuhkan untuk mengirim data antara subsistem juga diidentifikasikan. Hasil dari spesifikasi layanan adalah sebuah spesifikasi interface yang lengkap dari masing-masing subsistem.

i. Identifikasi atribut dan operasi tambahan

Pada tahap ini, kita mengevaluasi deskripsi layanan dari subsistem dan mengidentifikasi atribut dan operasi tambahan. Selama proses analisa, kita mungkin melewatkan banyak atribut dikarenakan kita terfokus pada fungsionalitas sistem. Kita menggambarkan fungsionalitas sistem menggunakan model use case

bukan operasi didalam model objek. Pada saat membuat model objek, kita berfokus pada wilayah aplikasi dan mengabaikan hal-hal detail terkait dengan sistem yang tidak bergantung pada wilayah aplikasi.

ii. Menentukan tipe, tanda dan keterlihatan

Pada langkah ini, kita menspesifikasikan tipe dari atribut, tanda dari operasi dan keterlihatan atribut dan operasi. Menspesifikasi tipe memperbaiki model perancangan objek dalam 2 hal. Pertama kita menambahkan hal rinci pada model dengan menspesifikasi jangkauan pada setiap atribut. Kedua, kita memetakan kelas dan atribut dari model objek kepada tipe-tipe yang telah sediakan oleh lingkungan pengembangan.

Kita juga mempertimbangkan hubungan antara kelas-kelas yang kita identifikasikan dengan kelas dari komponen yang telah ada.

Kemudian kita menentukan keterlihatan masing-masing atribut dan operasi. Dengan demikian, kita menentukan atribut mana yang hanya dapat diakses oleh operasi di dalam kelas dan atribut mana yang dapat diakses oleh kelas lain secara langsung. Hal yang sama juga berlaku pada operasi. Pada kelas abstract dan kelas-kelas yang memang ditujukan untuk diperbaiki, kita juga mendefinisikan atribut dan operasi terlindungi (*protected*) yang hanya dapat diakses oleh kelas anaknya.

iii. Menentukan Kondisi Sebelum dan Sesudah

Pada tahap ini, kita menentukan perjanjian untuk masing-masing operasi milik umum pada masing-masing kelas. Perjanjian adalah antara kelas pengguna

dengan kelas yang mengimplementasi. Kondisi sebelum dari sebuah operasi menggambarkan bagian dari perjanjian yang harus dipatuhi oleh kelas pengguna. Kondisi sesudah menggambarkan apa yang dijamin oleh kelas yang mengimplementasi jika kelas pengguna memenuhi persyaratan. Ketika memperbaiki sebuah kelas, kelas yang memperbaiki mendapat perjanjian dari kelas yang mengimplementasi diawal.

Kondisi sebelum dan sesudah juga dapat digunakan untuk menentukan ketergantungan antara operasi didalam kelas yang sama.

iv. Menentukan Kondisi Konstan

Kondisi konstan merupakan sebuah perjanjian permanen yang mengganti perjanjian yang spesifik pada operasi. Kegiatan mengidentifikasi konstan sama dengan menemukan kelas abstrak selama proses analisa. Alasan membuat konstan adalah untuk memperjelas asumsi yang dibuat oleh kelas yang mengimplementasi kepada kelas penggunanya.

v. Pewarisan Perjanjian

Dalam bahasa yang mendukung mekanisme banyak bentuk seperti Java, sebuah kelas dapat diganti oleh kelas manapun yang merupakan turunannya. Sehingga dapat terjadi kemungkinan kelas pengguna yang memanggil operasi pada sebuah kelas sebenarnya menjalankan kelas anaknya. Kelas pengguna mengharapkan perjanjian yang berlangsung dengan kelas orang tua masih tetap berlaku dengan kelas anaknya. Kondisi ini disebut pewarisan perjanjian.

Perjanjian diwariskan dengan ketentuan sebagai berikut:

- operasi pada kelas anak diperbolehkan untuk melemahkan kondisi sebelum. Dengan kata lain, operasi pada kelas anak dapat menangani lebih banyak kondisi dari pada kelas orang tuanya.
- Kondisi sesudah pada kelas anak harus sama atau lebih ketat daripada kelas orang tua.
- Kondisi konstan pada kelas anak harus sama dengan kelas orang tua.