BAB 2 METODE REKAYASA PERANGKAT LUNAK

(Sumber: Clip Art Microsoft Office 2007)

Gambar 2.1. Bekerja dengan komputer.

Ketika kita bekerja dengan komputer seperti pada Gambar 2.1., kita membutuhkan serangkaian tahapan dan cara-cara tertentu agar dapat menghasilkan sesuatu yang menjadi harapan kita. Demikian juga dalam rekayasa perangkat lunak, diperlukan tahapan-tahapan kerja yang harus dilalui. Rekayasa perangkat lunak yang sukses tidak hanya membutuhkan kemampuan komputasi seperti algoritma, pemrograman, dan basis data yang kuat, namun juga perlu penentuan tujuan yang baik, identifikasi cara penyelesaian, metode pengembangan, urutan aktifitas, identifikasi kebutuhan sumberdaya, dan faktor-faktor lain. Hal-hal seperti ini terkait dengan apa yang disebut dengan metode rekayasa perangkat lunak.

Isi dari bab ini tidak termasuk dalam standar kompetensi bidang keahlian RPL. Namun penulis memandang perlu disampaikan agar kalian dapat mengetahui bagaimana sebenarnya rekayasa perangkat lunak dilakukan dan metode-metode apa saja yang biasa digunakan. Beberapa bagian dari bab ini mungkin agak sulit dipahami, sehingga peran guru dalam membantu menjelaskan akan sangat diperlukan. Rangkuman bab disampaikan di bagian akhir dari uraian isi.

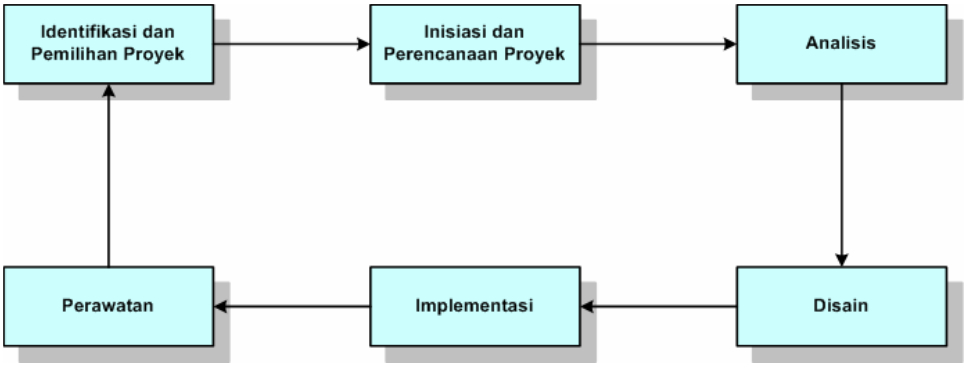
TUJUAN

Setelah mempelajari bab ini diharapkan kalian akan mampu :

- Memahami karakteristik umum model proses dalam rekayasa perangkat lunak.
- Menyebutkan beberapa model rekayasa perangkat lunak .
- Mengetahui prinsip-prinsip dari metode *waterfall*, *prototyping*, dan *unified process*.
- Memahami tahapan-tahapan dalam rekayasa perangkat lunak.

2.1. MODEL PROSES REKAYASA PERANGKAT LUNAK

Pada rekayasa perangkat lunak, banyak model yang telah dikembangkan untuk membantu proses pengembangan perangkat lunak. Model-model ini pada umumnya mengacu pada model proses pengembangan sistem yang disebut ***System Development Life Cycle (SDLC)*** seperti terlihat pada Gambar 2.2.



Gambar 2.2. System Development Life Cycle (SDLC).

Setiap model yang dikembangkan mempunyai karakteristik sendiri-sendiri. Namun secara umum ada persamaan dari model-model ini, yaitu:

- *Kebutuhan terhadap definisi masalah yang jelas*. Input utama dari setiap model pengembangan perangkat lunak adalah pendefinisian masalah yang jelas. Semakin jelas akan semakin baik karena akan memudahkan dalam penyelesaian masalah. Oleh karena itu pemahaman masalah seperti dijelaskan pada Bab 1, merupakan bagian penting dari model pengembangan perangkat lunak.
- *Tahapan-tahapan pengembangan yang teratur*. Meskipun model-model pengembangan perangkat lunak memiliki pola yang berbeda-beda, biasanya model-model tersebut mengikuti pola umum *analysis – design – coding – testing - maintenance*.
- *Stakeholder berperan sangat penting dalam keseluruhan tahapan pengembangan*. Stakeholder dalam rekayasa perangkat lunak dapat berupa pengguna, pemilik, pengembang, pemrogram dan orang-orang yang terlibat dalam rekayasa perangkat lunak tersebut.
- *Dokumentasi merupakan bagian penting dari pengembangan perangkat lunak*. Masing-masing tahapan dalam model biasanya menghasilkan sejumlah tulisan, diagram, gambar atau bentuk-bentuk lain yang harus didokumentasi dan merupakan bagian tak terpisahkan dari perangkat lunak yang dihasilkan.
- *Keluaran dari proses pengembangan perangkat lunak harus bernilai ekonomis*. Nilai dari sebuah perangkat lunak sebenarnya agak susah di-rupiah-kan. Namun efek dari penggunaan perangkat lunak yang telah dikembangkan haruslah memberi nilai tambah bagi organisasi. Hal ini dapat

berupa penurunan biaya operasi, efisiensi penggunaan sumberdaya, peningkatan keuntungan organisasi, peningkatan "image" organisasi dan lain-lain.

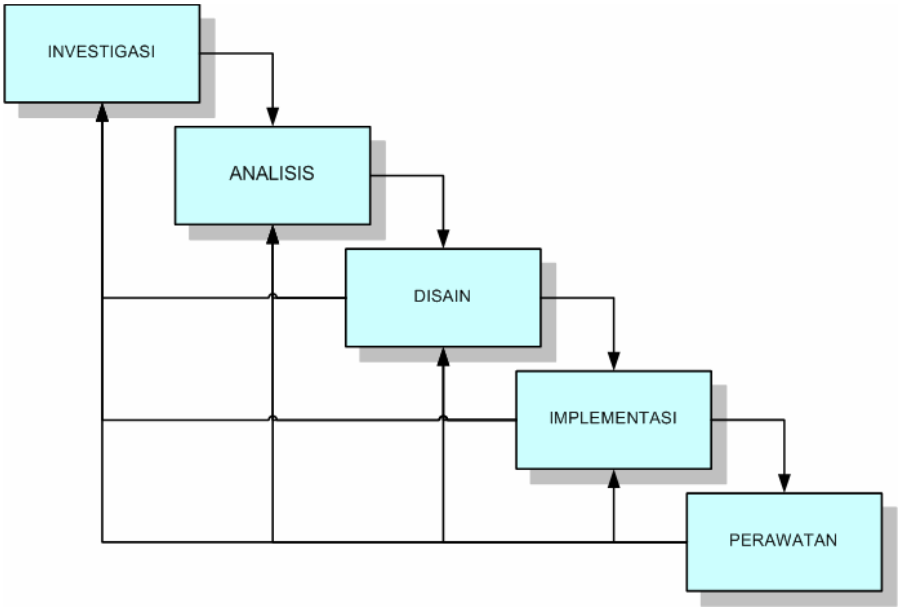
Ada banyak model pengembangan perangkat lunak, antara lain *The Waterfall Model, Joint Application Development (JAD), Information Engineering (IE), Rapid Application Development (RAD)* termasuk di dalamnya *Prototyping, Unified Process (UP), Structural Analysis and Design (SAD) dan Framework for the Application of System thinking (FAST).* Pada buku ini akan dibahas tiga model pengembangan yaitu *The Waterfall Model, Prototyping, dan Unified Processs (UP).*

2.1.1. The waterfall model

Model siklus hidup (*life cycle model*) adalah model utama dan dasar dari banyak model. Salah satu model yang cukup dikenal dalam dunia rekayasa perangkat lunak adalah *The Waterfall Model*. Ada 5 tahapan utama dalam *The Waterfall Model* seperti terlihat pada Gambar 2.3. Disebut *waterfall* (berarti air terjun) karena memang diagram tahapan prosesnya mirip dengan air terjun yang bertingkat.

Tahapan-tahapan dalam The Waterfall Model secara ringkas adalah sebagai berikut:

- Tahap investigasi dilakukan untuk menentukan apakah terjadi suatu masalah atau adakah peluang suatu sistem informasi dikembangkan. Pada tahapan ini studi kelayakan perlu dilakukan untuk menentukan apakah sistem informasi yang akan dikembangkan merupakan solusi yang layak
- Tahap analisis bertujuan untuk mencari kebutuhan pengguna dan organisasi serta menganalisa kondisi yang ada (sebelum diterapkan sistem informasi yang baru).
- Tahap disain bertujuan menentukan spesifikasi detil dari komponen-komponen sistem informasi (manusia, *hardware, software, network* dan data) dan produk-produk informasi yang sesuai dengan hasil tahap analisis.
- Tahap implementasi merupakan tahapan untuk mendapatkan atau mengembangkan *hardware* dan *software* (pengkodean program), melakukan pengujian, pelatihan dan perpindahan ke sistem baru.
- Tahapan perawatan (maintenance) dilakukan ketika sistem informasi sudah dioperasikan. Pada tahapan ini dilakukan monitoring proses, evaluasi dan perubahan (perbaikan) bila diperlukan.

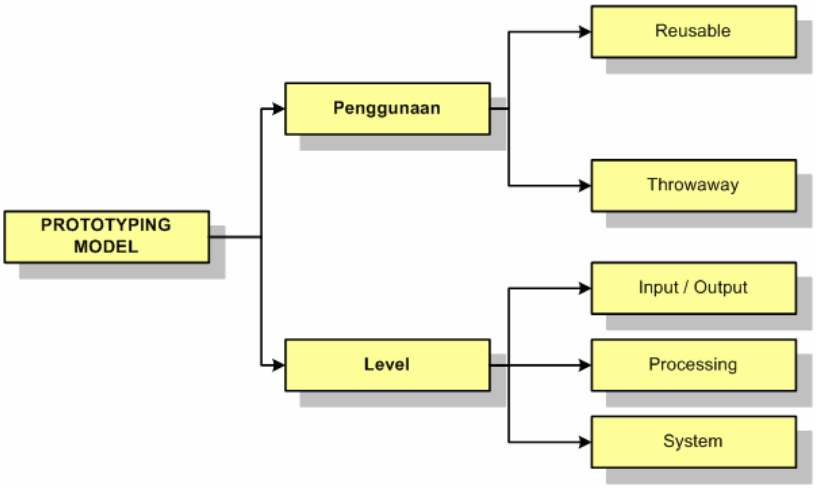


Gambar 2.3. The Waterfall Model

2.1.2. Prototyping model

Prototyping adalah s*alah satu pendekatan dalam rekayasa perangkat lunak yang secara langsung mendemonstrasikan bagaimana sebuah perangkat lunak atau komponen-komponen perangkat lunak akan bekerja dalam lingkungannya sebelum tahapan konstruksi aktual dilakukan* (Howard, 1997).

Prototyping model dapat diklasifikasikan menjadi beberapa tipe seperti terlihat pada gambar 2.4.



Gambar 2.4. Klasifikasi prototyping model (Harris, 2003)

- *Reusable prototype :*

  Prototype yang akan ditransformasikan menjadi produk final.

- *Throwaway prototype :*

  Prototype yang akan dibuang begitu selesai menjalankan maksudnya.

- *Input/output prototype :*

  Prototype yang terbatas pada antar muka pengguna (user interface).

- *Processing prototype :*

  Prototype yang meliputi perawatan file dasar dan proses-proses transaksi.

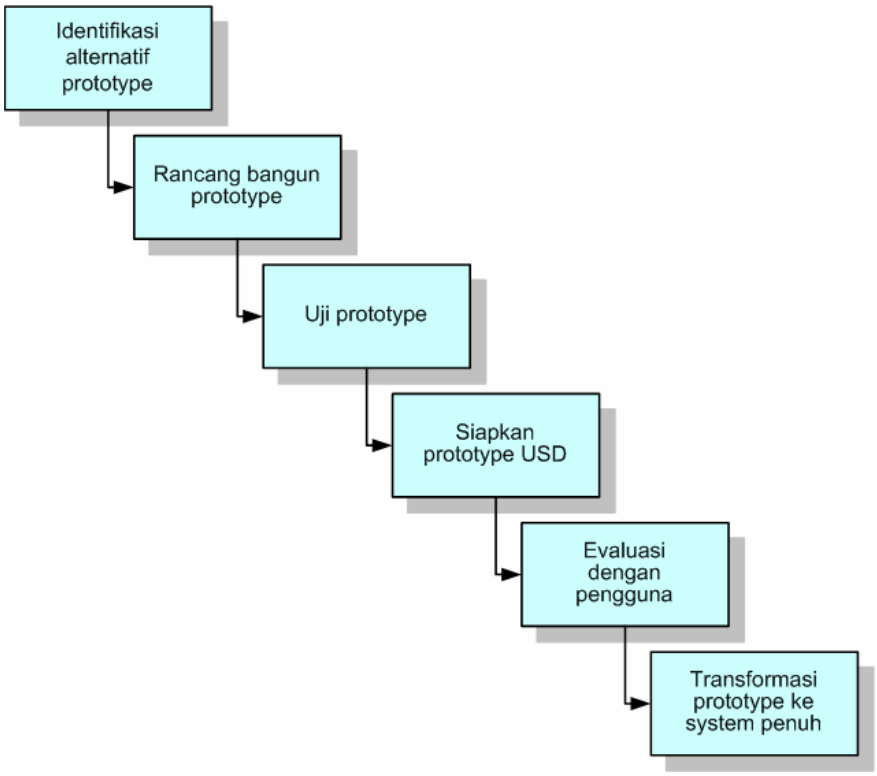
- *System prototype :*

  Prototype yang berupa model lengkap dari perangkat lunak.

Tahap-tahap dalam *prototyping* boleh dikata merupakan tahap-tahap yang dipercepat. Strategi utama dalam *prototyping* adalah kerjakan yang mudah terlebih dahulu dan sampaikan hasil kepada pengguna sesegera mungkin. Harris (2003) membagi prototyping dalam enam tahapan seperti terlihat pada gambar 2.5.

Tahapan-tahapan secara ringkas dapat dijelaskan sebagai berikut:

- *Identifikasi kandidat prototyping*. Kandidat dalam kasus ini meliputi *user interface* (menu, dialog, input dan output), file-file transaksi utama, dan fungsi-fungsi pemrosesan sederhana.
- *Rancang bangun prototype dengan bantuan software* seperti *word processor, spreadsheet, database*, pengolah grafik, dan software CASE (*Computer-Aided System Engineering*).
- *Uji prototype* untuk memastikan prototype dapat dengan mudah dijalankan untuk tujuan demonstrasi.
- *Siapkan prototype USD (User's System Diagram)* untuk mengidentifikasi bagian-bagian dari perangkat lunak yang di-*prototype*-kan.
- *Evaluasi dengan pengguna* untuk mengevaluasi *prototype* dan melakukan perubahan jika diperlukan.
- *Transformasikan prototype menjadi perangkat lunak yang beroperasi penuh* dengan melakukan penghilangan kode-kode yang tidak dibutuhkan, penambahan program-program yang memang dibutuhkan dan perbaikan dan pengujian perangkat lunak secara berulang.

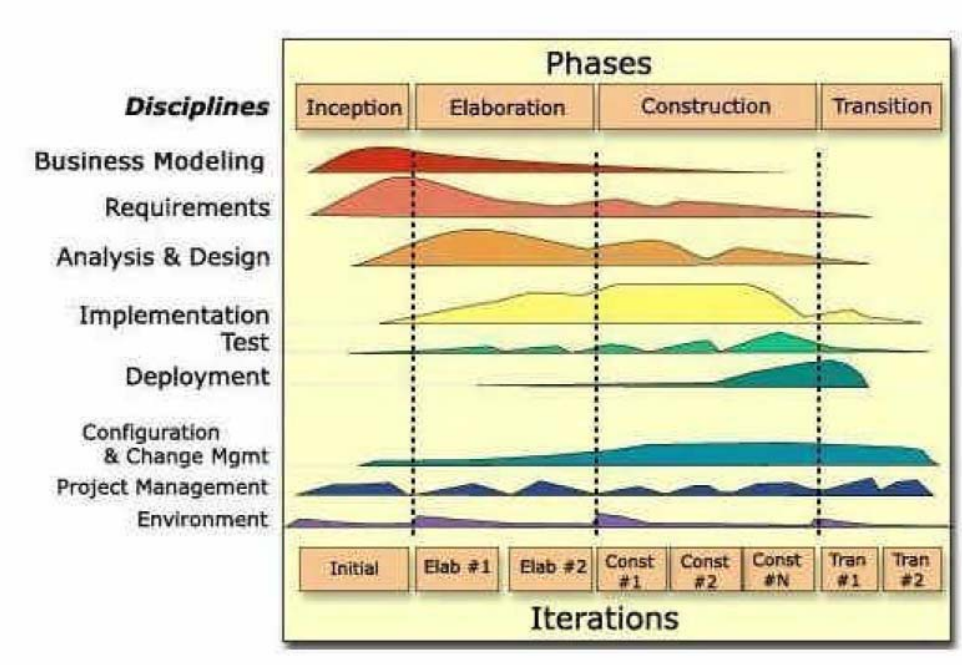


Gambar 2.5. Tahapan-tahapan prototyping model (Harris, 2003)

2.1.3. Unified Process dan Unified Modeling Language

Unified Process (UP) atau kadang disebut sebagai *Unified Software Development Process* (USDP) adalah kerangka proses pengembangan yang bersifat *use-case-driven*, berpusat pada arsitektur perangkat lunak, interatif dan tumbuh-kembang (Alhir, 2005). Kerangka pengembangan ini termasuk baru dalam metodologi pengembangan perangkat lunak. UP dapat diaplikasikan pada berbagai skala proyek, mulai dari skala kecil sampai dengan skala besar.

Daur hidup UP secara umum akan tampak seperti pada bagan di Gambar 2.6. Bagan ini biasa disebut sebagai "*hump chart*". Pada bagan ini terlihat ada empat tahap pengembangan yaitu *inception, elaboration, construction* dan *transition*. Selain itu tampak pula sejumlah aktivitas (*disciplines*) yang harus dilakukan sepanjang pengembangan perangkat lunak, yaitu, *business modeling, requirements, analysis and design, implementation, test*. Tahap dan aktivitas tersebut akan dilakukan secara iteratif (Ambler, 2005).



Gambar 2.6. RUP Life Cycle (Ambler, 2005).

Penjelasan singkat untuk empat tahapan dalam UP adalah sebagai berikut:

- *Inception*. Tahapan ini merupakan tahapan paling awal dimana aktivitas penilaian terhadap sebuah proyek perangkat lunak dilakukan. Tujuannya adalah untuk mendapatkan kesepakatan dari stakeholder sehubungan dengan tujuan dan dana proyek.
- *Elaboration*. Tujuan dari tahap ini adalah untuk mendapatkan gambaran umum kebutuhan, persyaratan dan fungsi-fungsi utama perangkat lunak. Hal ini penting untuk mengetahui secara lebih baik resiko-resiko proyek, baik meliputi resiko arsitektur perangkat lunak, perencanaan, maupun implementasi. Pada tahap ini telah dimulai rancang bangun perangkat lunak secara iterative melalui aktivitas-aktivitas seperti *business modeling, requirements, analysis* dan *design* meskipun baru pada tahap awal.
- *Construction*. Tujuan dari tahapan ini adalah membangun perangkat lunak sampai dengan saat perangkat lunak tersebut siap digunakan. Titik berat tahapan ini adalah pada penentuan tingkat prioritas kebutuhan / persyaratan, melengkapi spesifikasinya, analisis lebih dalam, disain solusi yang memenuhi kebutuhan dan persyaratan, pengkodean dan pengujian perangkat lunak. Jika dimungkinkan versi awal dari perangkat lunak diuji cobakan untuk mendapatkan masukan dari pengguna.

- *Transition.* Tahap ini difokuskan pada bagaimana menyampaikan perangkat lunak yang sudah jadi pada pengguna. Perangkat lunak akan secara resmi diuji oleh baik oleh penguji (tester) yang kompeten maupun oleh pengguna. Beberapa aktivitas seperti pemindahan pusat data dan pelatihan pengguna dan staf pendukung harus dilakukan pada tahap ini.

Dalam pengembangan perangkat lunak dengan menggunakan UP, maka tidak lepas dari penggunaan notasi-notasi yang biasa disebut sebagai **UML (*Unified Modeling Language*)**. Meskipun UP mensyaratkan penggunaan UML, namun UML sendiri dapat digunakan pada berbagai metodologi yang lain bahkan dapat digunakan pada bidang selain sistem informasi. UML adalah bahasa pemodelan standar atau kumpulan teknik-teknik pemodelan untuk men-spesifikasi, mem-visualisasi, meng-konstruksi dan mendokumentasi hasil kerja dalam pengembangan perangkat lunak (Fowler, 2004). UML lahir dari penggabungan banyak bahasa pemodelan grafis berorientasi obyek yang berkembang pesat pada akhir tahun 1980an dan awal 1990an.

Secara sederhana UML digunakan untuk menggambar sketsa sistem. Pengembang menggunakan UML untuk menyampaikan beberapa aspek dari sebuah perangkat lunak melalui notasi grafis. UML mendefinisikan notasi dan semantik. Notasi merupakan sekumpulan bentuk khusus yang memiliki makna tertentu untuk menggambarkan berbagai diagram piranti lunak dan semantik mendefinisikan bagaimana bentuk-bentuk tersebut dapat dikombinasikan. Ada beberapa jenis diagram yang disediakan dalam UML, antara lain adalah:

- *Use-case diagram*. Diagram ini berguna untuk menggambarkan interaksi antara pengguna dengan sebuah perangkat lunak
- *Activity diagram*. Diagram ini berguna untuk menggambarkan prosedur-prosedur perilaku perangkat lunak.
- *Class diagram*. Diagram ini berguna untuk menggambarkan class, fitur, dan hubungan-hubungan yang terjadi. Pada diagram ini pendekatan berorientasi obyek memegang peranan yang sangat penting.
- *Sequence diagram*. Diagram ini berguna untuk menggambarkan interaksi antar obyek dengan penekanan pada urutan proses atau kejadian.
- *State machine diagram*. Diagram ini digunakan untuk menggambarkan bagaimana suatu kejadian mengubah obyek selama masa hidup obyek tersebut.
- *Component diagram*. Diagram ini berguna untuk menggambarkan struktur dan koneksi komponen.

2.2. TAHAPAN REKAYASA PERANGKAT LUNAK

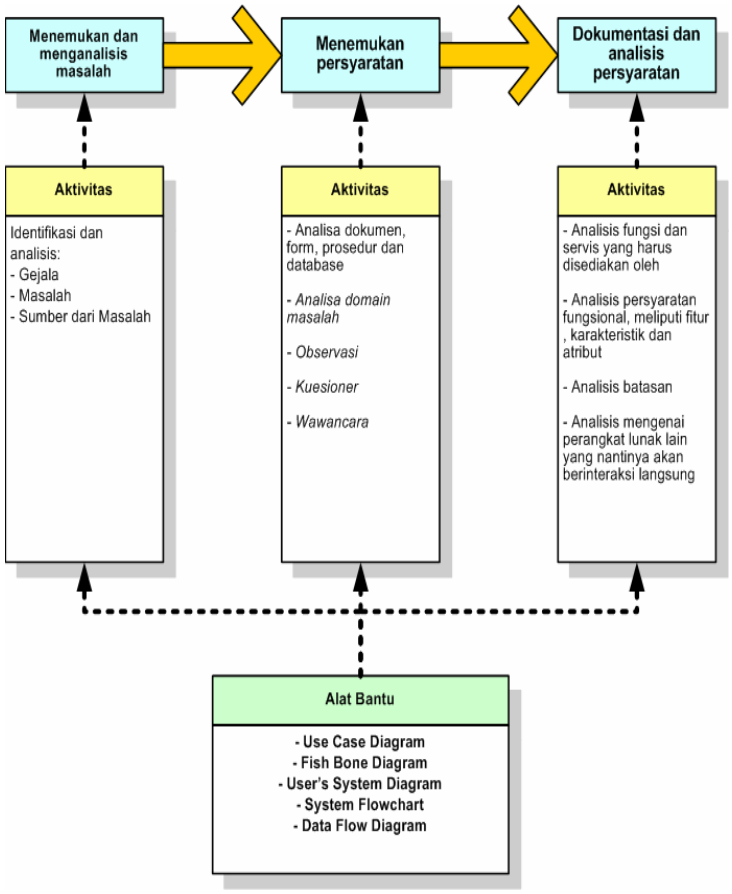
Seperti telah disebutkan, meskipun dalam pendekatan berbeda-beda, namun model-model di atas memiliki kesamaan, yaitu menggunakan pola tahapan *analysis – design – coding(construction) – testing – maintenance.*

2.2.1. Analisis

Analisis sistem adalah sebuah teknik pemecahan masalah yang menguraikan sebuah sistem menjadi komponen-komponennya dengan tujuan mempelajari seberapa bagus komponen-komponen tersebut bekerja dan berinteraksi untuk meraih tujuan mereka.

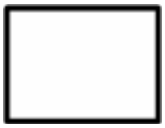
Analisis mungkin adalah bagian terpenting dari proses rekayasa perangkat lunak. Karena semua proses lanjutan akan sangat bergantung pada baik tidaknya hasil analisis. Tahapan-tahapan dalam analisis rekayasa perangkat lunak secara ringkas dapat dilihat pada Gambar 2.7.

Ada satu bagian penting yang biasanya dilakukan dalam tahapan analisis yaitu pemodelan proses bisnis. **Model proses** adalah model yang memfokuskan pada seluruh proses di dalam sistem yang mentransformasikan data menjadi informasi (Harris, 2003). Model proses juga menunjukkan aliran data yang masuk dan keluar pada suatu proses. Biasanya model ini digambarkan dalam bentu Diagram Arus Data (Data Flow Diagram / DFD). DFD meyajikan gambaran apa yang manusia, proses dan prosedur lakukan untuk mentransformasi data menjadi informasi.



Gambar 2.7. Tahapan dan aktifitas dalam analisis.

Umumnya ada empat notasi yang sering digunakan dalam DFD seperti tampak Gambar 2.8.

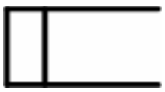


External Entitiy

External Entity melambangkan sumber data (dari mana data berasal) atau penerima informasi (tujuan akhir dari data). Contoh external entity antara lain konsumen yang memesan suatu produk, manajer yang mengevaluasi laporan penjualan mingguan, dan lain-lain.

Process

Proses adalah serangkaian langkah yang dilakukan untuk memanipulasi data, misalnya pengumpulan, pengurutan, pemilihan, pelaporan, peringkasan, analisis dan lain-lain.



Data Store

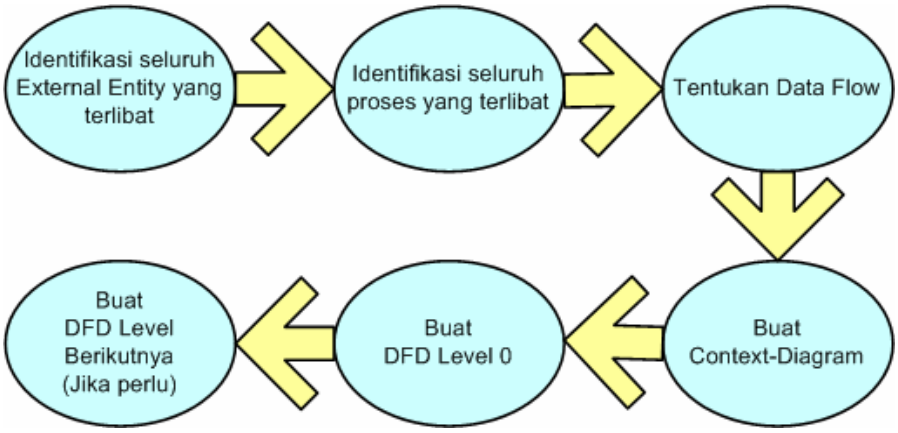
Data store adalah tempat untuk menyimpan data untuk digunakan kemudian. Nama yang pada data store ini merupakan abstraksi dari data yang disimpan. Namun detil / item data apa saja yang ada, bagaimana cara akses, atau bagaimana mengorganisasinya tidak dijelaskan dalam notasi ini.

Data Flow

Data flow menunjukkan aliran data dari satu tempat ke tempat lain. Perpindahan data ini dapat dari external entity ke proses, antar proses satu dengan yang lain, dari proses ke data store. Dalam penggambarannya setiap data flow harus diberi label yang menunjukkan data apa yang mengalir.

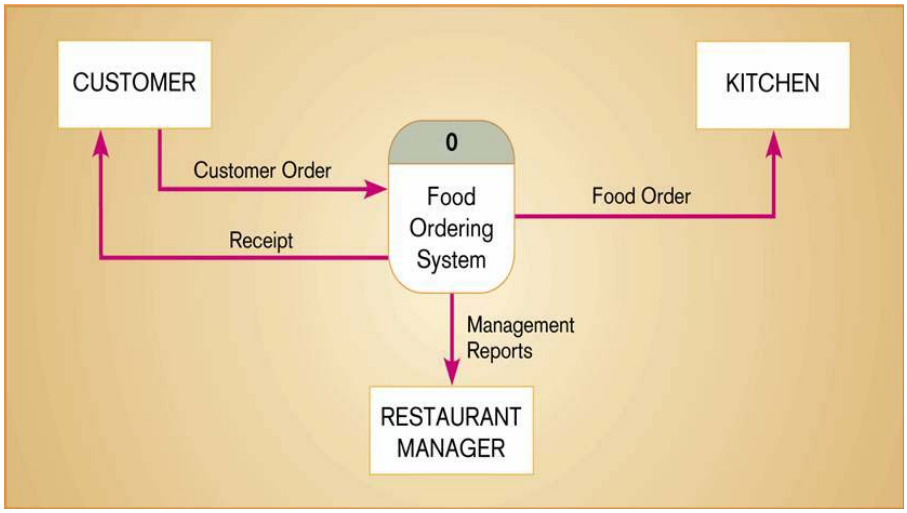
Gambar 2.8. Notasi pada DFD.

Dalam pembuatan DFD ada beberapa tahapan yang dilakukan secara berurutan. Gambar 2.9. menunjukkan urutan tahapan tersebut.



Gambar 2.9. Tahapan pembuatan DFD.

Context diagram adalah DFD ruang lingkup dari sistem yang menunjukkan batas-batas sistem, *external entitiy* yang berinteraksi dengan sistem dan aliran data utama antara *external entity* dengan sistem. *Context diagram* menggambarkan keseluruhan sistem dalam suatu proses tunggal. Gambar 2.10 menunjukkan sebuah contoh *context diagram.*

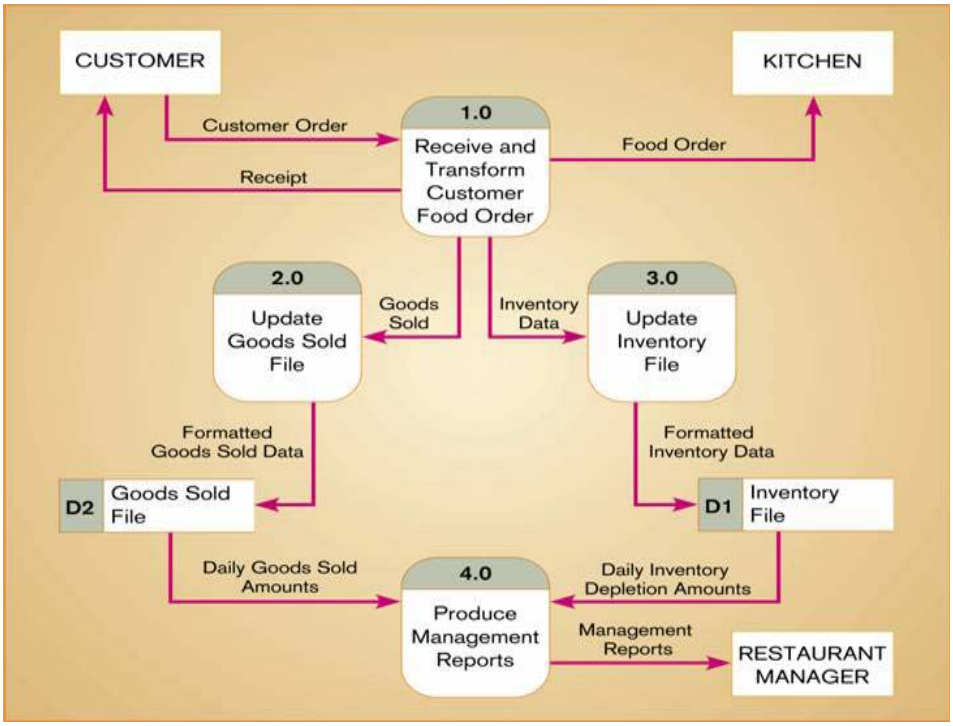


Gambar 2.10. *Context diagram* sistem pemesanan makanan (Hoffer et al., 2002).

Context diagram pada Gambar 2.10 tampak hanya ada satu proses tunggal yang merepresentasikan sistem yang dimodelkan. Pada proses ini diberi notasi angka 0 untuk menunjukkan ini adalah level paling abstrak dari sistem. Selain itu ada tiga *external entity* yaitu *customer, kitchen* dan *restaurant*

manager. Ketiganya dapat berperan sebagai sumber data (dalam contoh di atas adalah *customer*) atau sebagai penerima informasi (dalam contoh di atas *customer, kitchen*, dan *restaurant manager*). *Data flow* yang tampak pada gambar menunjukkan ada satu *data flow* yang masuk ke sistem dan ada tiga *data flow* yang keluar dari sistem. Masing-masing *data flow* diberi label yang menunjukkan data apa yang sedang mengalir.

Setelah context diagram terbentuk dengan benar maka langkah selanjutnya adalah merinci context diagram tersebut dalam DFD Level 0. DFD Level 0 adalah DFD yang merepresentasikan proses-proses, data flow dan data storage utama di dalam sistem. DFD Level 0 ini akan digunakan sebagai dasar untuk membangun DFD yang level dibawahnya (Level 1, 2, 3, .. dst) atau biasa disebut sebagai dekomposisi DFD. Gambar 3.10 merupakan DFD level 0 dari context-diagram pada gambar 2.11.



Gambar 2.11. DFD Level 0.

Pada gambar 2.11 tampak adanya pemecahan pada proses dari yang semula hanya satu menjadi empat. Masing-masing proses diberi nomor kode 1.0, 2.0, 3.0 dan 4.0. Jumlah *external entity* harus tetap yaitu 3 demikian pula *data flow* yang keluar dan masuk (input dan output) ke dalam sistem harus sama dengan pada context diagram. Sedangkan *data flow* yang berada di dalam sistem (yang mengalir antar proses dan atau data storage) tergantung pada proses dan *data storage* yang terlibat. Ada dua *data storage* yaitu *Goods Sold*

File dan *Inventory File*. Kedua *data storage* ini digunakan untuk menyimpan data dari suatu proses. Data ini juga akan dibaca / diakses oleh proses yang lain. Sebagai contoh *data storage Inventory File* berisi data hasil proses 3.0 (*Update Inventory File*). Data ini akan digunakan proses 4.0 (*Produce Management Reports*) untuk membuat laporan yang akan disampaikan pada *Restaurant Manager*.

DFD level berikutnya yaitu level 1, 2 dan seterusnya diperlukan apabila level sebelumnya dirasa kurang detil. Sebagai contoh apabila DFD level 0 (Gambar 14.12) dirasa belum cukup detil menunjukkan arus data yang mengalir, maka dapat dibuat detilnya pada DFD level 1. Bagian yang harus didetilkan biasanya adalah proses. Detil pada level berikutnya, mungkin pada semua proses atau hanya pada proses-proses tertentu saja. DFD pada level 0 maupun level di bawahnya memiliki kesamaan aturan yang tersaji berikut pada tabel berikut ini.

Tabel 2.1. Aturan-aturan dalam DFD

Kelompok	Aturan
Umum	• input-input ke suatu process akan selalu berbeda dengan output-outputnya • obyek obyek (External Entity, Process, Data Storage, dan Data Flow) yang ada pada suatu DFD selalu memiliki nama yang unik
External Entity	• nama yang dipakai pada External Entity selalu menggunakan kata benda • data tidak boleh mengalir secara langsung dari External Entity yang satu ke External Entity yang lain
Process	• nama yang dipakai pada Process selalu menggunakan kata kerja • tidak ada Process yang hanya menghasilkan output • tidak ada Process yang hanya menerima input
Data Storage	• nama yang dipakai pada Data Storage selalu menggunakan kata benda • data tidak boleh mengalir secara langsung dari Data Storage yang satu ke Data Storage yang lain • data tidak boleh mengalir secara langsung dari External Entity ke Data Storage demikian juga sebaliknya.
Data Flow	• nama yang dipakai pada Data Flow selalu menggunakan kata benda • Data Flow di antara dua notasi hanya memiliki satu arah aliran • Percabangan (fork) menunjukkan adanya data yang persis sama yang mengalir dari suatu tempat ke dua atau lebih tempat yang lain • Penggabungan (join) menunjukkan adanya data yang persis sama yang mengalir dua atau lebih tempat menuju satu tempat yang lain • Data Flow menuju Data Storage berarti terjadi update data • Data Flow dari Data Storage berarti terjadi pembacaan / pengambilan data

2.2.2. Disain

Disain perangkat lunak adalah tugas, tahapan atau aktivitas yang difokuskan pada spesifikasi detil dari solusi berbasis computer (Whitten et al, 2004).

Disain perangkat lunak sering juga disebut sebagai *physical design*. Jika tahapan analisis sistem menekankan pada masalah bisnis (business rule), maka sebaliknya disain perangkat lunak fokus pada sisi teknis dan implementasi sebuah perangkat lunak (Whitten et al, 2004).

Output utama dari tahapan disain perangkat lunak adalah spesifikasi disain. Spesifikasi ini meliputi spesifikasi disain umum yang akan disampaikan kepada stakeholder sistem dan spesifikasi disain rinci yang akan digunakan pada tahap implementasi. Spesifikasi disain umum hanya berisi gambaran umum agar stakeholder sistem mengerti akan seperti apa perangkat lunak yang akan dibangun. Biasanya diagram USD tentang perangkat lunak yang baru merupakan point penting dibagian ini. Spesifikasi disain rinci atau kadang disebut disain arsitektur rinci perangkat lunak diperlukan untuk merancang sistem sehingga memiliki konstruksi yang baik, proses pengolahan data yang tepat dan akurat, bernilai, memiliki aspek *user friendly* dan memiliki dasar-dasar untuk pengembangan selanjutnya.

Desain arsitektur ini terdiri dari desain database, desain proses, desain *user interface* yang mencakup desain *input, output form dan report,* desain hardware, software dan jaringan. Desain proses merupakan kelanjutan dari pemodelan proses yang dilakukan pada tahapan analisis.

2.2.3. Konstruksi

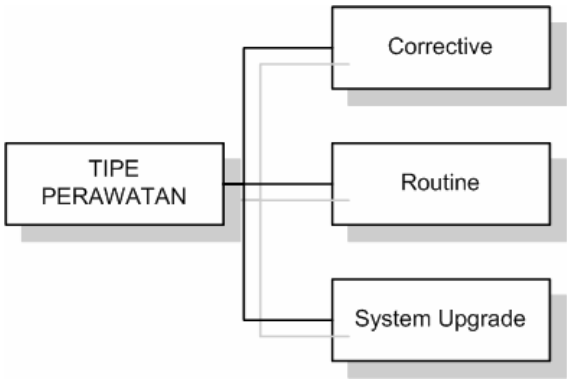
Konstruksi adalah tahapan menerjemahkan hasil disain logis dan fisik ke dalam kode-kode program computer. Buku ini sebagian besar berisi tentang bagian ini.

2.2.4. Pengujian

Pengujian sistem melibatkan semua kelompok pengguna yang telah direncanakan pada tahap sebelumnya. Pengujian tingkat penerimaan terhadap perangkat lunak akan berakhir ketika dirasa semua kelompok pengguna menyatakan bisa menerima perangkat lunak tersebut berdasarkan criteria-kriteria yang telah ditetapkan.

2.2.5. Perawatan dan Konfigurasi

Ketika sebuah perangkat lunak telah dianggap layak untuk dijalankan, maka tahapan baru menjadi muncul yaitu perawatan perangkat lunak. Ada beberapa tipe perawatan yang biasa dikenal dalam dunia perangkat lunak seperti terlihat pada diagram di Gambar 2.12.



Gambar 3.12. Tipe-tipe perawatan.

- Tipe perawatan *corrective* dilakukan jika terjadi kesalahan atau biasa dikenal sebagai bugs. Perawatan bisa dilakukan dengan memperbaiki kode program, menambah bagian yang dirasa perlu atau malah menghilangkan bagian-bagian tertentu.
- Tipe perawatan *routine* biasa juga disebut preventive maintenance dilakukan secara rutin untuk melihat kinerja perangkat lunak ada atau tidak ada kesalahan.
- Tipe perawatan *sistem upgrade* dilakukan jika ada perubahan dari komponen-komponen yang terlibat dalam perangkat lunak tersebut. Sebagai contoh perubahan platform sistem operasi dari versi lama ke versi baru menyebabkan perangkat lunak harus diupgrade.

2.3. RINGKASAN

- Model-model rekayasa perangkat lunak pada umumnya mengacu pada model proses pengembangan sistem yang disebut ***System Development Life Cycle (SDLC).***
- Model pengembangan perangkat lunak yang sekarang umum digunakan adalah *The Waterfall Model, Prototyping,* dan *Unified Process (UP).*
- Tahapan-tahapan utama dalam rekayasa perangkat lunak meliputi : analisis, disain, konstruksi, pengujian dan perawatan.

2.4. SOAL-SOAL LATIHAN

1. Sebutkan tahapan-tahapan dalam System Developmen Life Cycle (SDLC)
2. Sebutkan persamaan karakteristik yang dimiliki oleh model-model pengembangan perangkat lunak.
3. Sebutkan lima model pengembangan perangkat lunak yang anda ketahui.
4. Apakah yang dimaksud tahapan konstruksi pada rekayasa perangkat lunak?
5. Gambarkan notasi-notasi dalam Data Flow Diagram yang anda ketahui.