2019

REKAYASA PERANGKAT LUNAK

HERI SANTOSO, M.Kom

PRODI ILMU KOMPUTER UINSU

KATA PENGANTAR

Puji syukur saya panjatkan ke hadirat Allah Subhana Wa Ta'ala karena atas rahmat dan karunia-Nya kami bisa menyelesaikan buku Rekayasa Perangkat Lunak untuk mahasiswa dan dosen di Prodi Ilmu Komputer UINSU Medan.

Buku ini dibuat untuk referensi materi perkuliahan Mahasiswa dan buku ajar bagi Dosen Pengampu Mata Kuliah Rekayasa Perangkat Lunak.

Kesuksesan belajar berawal dari kemauan dan ditunjang oleh berbagai sarana, salah satu diantaranya adalah buku. Harapan kami, buku ini dapat membantu mahasiswa memahami tentang Rekayasa Perangkat Lunak.

Akhir kata Penulis mengucapkan terima kasih kepada semua pihak yang telah membantu dalam pembuatan buku ini. Kritik dan saran sangat kami harapkanuntuk perbaikan buku ini di masa yang akan datang.

Medan, 2019

Penulis

DAFTAR ISI

BAB I
MACAM DARI PERANGKAT LUNAK DENGAN KOMPONEN-KOMPONENNYA.
APLIKASI YANG DIHASILKAN DENGAN PERANGKAT LUNAK.

Abstract
Membahas mengenai macam, komponen dan aplikasi perangkat lunak
Kompetensi
Mahasiswa dapat mengetahui komponen-komponen dari perangkat lunak dan aplikasi yang dihasilkannya.

MACAM DARI PERANGKAT LUNAK

Perangkat Lunak atau software computer dapat dikelompokkan dalam dua kelompok, yakni :

1. Perangkat lunak system (Softwaresystem)
2. Perangkat lunak Aplikasi (Softwareaplikasi)

Perangkat lunak system dibedakan menjadi 3, yaitu :

1. Systemoperasi
2. Bahasapemrograman
3. Utilitas

Sedangkan Perangkat lunak aplikasi dibedakan menjadi :

1. Aplikasiperkantoran
2. Aplikasimultimedia
3. Aplikasi internet danjaringan
4. Aplikasikhusus

Macam-macam Perangkat Lunak Sistem Operasi :

1. DOS (Disc Operating System); system operasi generasi awal yang dirancang untuk computer tunggal Personal Computer(PC).
2. Unix; Sistem operasi berbasis jaringan, merupakan system operasi tertua, dikeluarkan tahun 1960. Unix pertama kali digunakan oleh computer jenis IBM, HP dan SunSolaris.

Bagian-bagian Unix yaitu :

1. Unix
2. OpenBSD
3. FreeBSD

4. Windows 95/ windows 98/ windows Me/ windows XP

   Sistem operasi buatan Microsoft ini sangat populer karena tampilannya yang user friendly dengan menggunakan pendekatan GUI. Kelebihan windowsdibandingkan dengan system operasi lainnya yaitu:

   a) Multitasking; kemampuan yang memungkinkan penggunaan sejumlah program dalam waktubersamaan.
   b) Mendukung system kerja team atau workgroup dalam suatujaringan.

5. Macintosh;dikeluarkan pada Januari 1984
6. Linux; diperkenalkan pertama kali oleh Linus Torvalds tahun 1991. Linux menjadi pesaing utama system operasi windows karena mempunyai keunggulan yang sama yakni mendukung mendukung multitasking, user friendly, serta workgroup.

Perangkat Lunak Bahasa Pemrograman
Terdapat empat kelompok generasi bahasa pemrograman :

1. Generasi Pertama (Bahasa Mesin); pemrograman menggunakan bahasa mesin yang diwakili oleh bilangan biner 0 dan 1.
2. Generasi Kedua (Bahasa Assembly); pemrograman menggunakan perintah kata yang pendek.
3. Generasi Ketiga (Bahasa tingkat tinggi); pemrograman dengan bahasa yang procedural tertata dengan baik.
4. Generasi Keempat (Bahasa pemrograman yang berorientasi pada object).

APLIKASI YANG DIHASILKAN DENGAN PERANGKAT LUNAK

Perangkat Lunak Utilitas yaitu perangkat lunak yang ditujukan untuk menunjang fungsionalitas perangkat lunak system operasi. Contoh untuk melakukan kompresi data pada harddisk atau media penyimpanan lain, dapat dilakukan melalui perangkat lunak WinZip. Contoh lainnya apabila untuk menangkal virus diperlukan perangkat lunak antivirus[1].

Aplikasi Perkantoran

Aplikasi perkantoran yaitu perangkat lunak yang ditujukan untuk membatu tugas-tugas dalam dunia perkantoran.
Yang termasuk Aplikasi perkantoran , adalah :

[1] *Ian K Bray "An Introduction to Requirement Engineering". Vol 1 No .1, Addison- Wesley 2002, hal. 10*

1. Spreadsheet; Yang sering dipergunakan untuk menyelesaikan pekerjaan kantor khususnya dibidang hitung menghitung. Yang paling banyak digunakan adalah Microsoft excel.
2. Word processor; adalah aplikasi pengolah kata. Nama programnya adalah Microsoft word.
3. Program Presentasi; Sebuah aplikasi untuk membuat presentasi. Nama program yang paling populer yaitu Microsoft Power Point.
4. Data base manajemen system; adalah perangkat lunak untuk melaksanakan manajemen data. Yang paling banyak digunakan dalam office adalah Microsoft visual basic.

Aplikasi Multimedia

Aplikasi yang mendukung teknologi multimedia, seperti teks, suara, gambar, film. Macam-macam perangkat lunak multimedia :

1. Corel Draw dan adobe photoshop ; aplikasi untuk membuat desin gambar dan foto.
2. RealPlayer, Winamp, Windows media player; adalah program untuk memutar music dan film.
3. Adobe premiere; perangkat lunak untuk membuat dan mengedit film.
4. Macromedia flash MX; program untuk membuat berbagai animasi.

Perangkat Lunak aplikasi internet dan jaringan

Perangkat Lunak aplikasi internet dan jaringan yaitu perangkat lunak yang digunakan untuk mendukung pemanfaatn internet dan jaringan. Beberapa perngkat lunak yang terkait dengan internet dan jaringan antara lain :

1. Web browser; adalah program untuk mengakses informasi internet, contohnya Internet Explorer, Opera, Mozilla firefox.
2. E-mail software; perangkat lunak yang menyediakan fasilitas untuk berkomunikasi. Contohnya Microsoft outlook.
3. ICQ; merupakan singkatan "I Seek You" adalah sebuah program untuk berchatting.

Perangkat Lunak aplikasi Khusus

Adalah perangkat lunak yang ditujukan pada bidang-bidang spesifik, contohnya:

1. Program SPSS; untuk analisis data statistic.
2. Program Matematika dan MAPLE, perangkat lunak pada bidang kajian matematika.
3. Program AutoCad; adalah program untuk desain pada ilmu teknik arsitektur.
4. Program MYOB, DEA, GL; untuk keperluan akuntansi perusahaan.

BAB II
ARTI DAN DEFINISI DARI PERANGKAT LUNAK.
JENIS-JENIS PERANGKAT LUNAK.
PENTINGNYA REKAYASA PERANGKAT LUNAK.

Abstract
Membahas mengenai definisi, komponen dan aplikasi perangkat lunak.
Kompetensi
Mahasiswa dapat mengetahui komponen-komponen dari perangkat lunak dan aplikasi yang dihasilkannya.

SEJARAH REKAYASA PERANGKAT LUNAK

Rekayasa perangkat lunak telah berkembang sejak pertama kali diciptakan pada tahun 1940-an hingga kini. Fokus utama pengembangannya adalah untuk mengembangkan praktek dan teknologi untuk meningkatkan produktivitas para praktisi pengembang perangkat lunak dan kualitas aplikasi yang dapat digunakan oleh pemakai.

1945 - 1965: Awal

Istilah software engineering digunakan pertama kali pada akhir 1950-an dan awal 1960-an. Saat itu, masih terdapat debat tajam mengenai aspek engineering dari pengembangan perangkat lunak.

Pada tahun 1968 dan 1969, komite sains NATO mensponsori dua konferensi tentang rekayasa perangkat lunak, yang memberikan dampak kuat terhadap perkembangan rekayasa perangkat lunak. Banyak yang menganggap bahwa dua konferensi inilah yang menandai awal resmi profesi rekayasa perangkat lunak.

1965 - 1985: krisis perangkat lunak

Pada tahun 1960-an hingga 1980-an, banyak masalah yang ditemukan para praktisi pengembangan perangkat lunak. Banyak projek yang gagal, hingga masa ini disebut sebagai krisis perangkat lunak. Kasus kegagalan pengembangan perangkat lunak terjadi mulai dari projek yang melebihi anggaran, hingga kasus yang mengakibatkan kerusakan fisik dan kematian. Salah satu kasus yang terkenal antara lain meledaknya roket Ariane akibat kegagalan perangkat lunak.

1985 - kini: tidak ada senjata pamungkas

Selama bertahun-tahun, para peneliti memfokuskan usahanya untuk menemukan teknik jitu untuk memecahkan masalah krisis perangkat lunak.
Berbagai teknik, metode, alat, proses diciptakan dan diklaim sebagai senjata pamungkas untuk memecahkan kasus ini. Mulai dari pemrograman terstruktur, pemrograman berorientasi object, perangkat pembantu pengembangan perangkat lunak (CASE tools), berbagai standar, UML hingga metode formal diagung-agungkan sebagai senjata pamungkas untuk menghasilkan software yang benar, sesuai anggaran dan tepat waktu.

Pada tahun 1987, Fred Brooks menulis artikel No Silver Bullet, yang berproposisi bahwa tidak ada satu teknologi atau praktek yang sanggup mencapai 10 kali lipat perbaikan dalam produktivitas pengembangan perangkat lunak dalam tempo 10 tahun. Sebagian berpendapat, no silver bullet berarti profesi rekayasa perangkat lunak dianggap telah gagal. Namun sebagian yang lain justru beranggapan, hal ini menandakan bahwa bidang profesi rekayasa perangkat lunak telah cukup matang, karena dalam bidang profesi lainnya pun, tidak ada teknik pamungkas yang dapat digunakan dalam berbagai kondisi.

DEFINISI REKAYASA PERANGKAT LUNAK

Berikut adalah beberapa definisi dari istilah Rekayasa Perangkat Lunak :

- Menurut Stephen R.Schach
  sebuah disiplin dimana dalam menghasilkan perangkat lunak bebas dari kesalahan dan dalam pengiriman anggaran tepat waktu serta memuaskan keinginan pemakai.
- Menurut Fritz Bauer
  penetapan dan penggunaan prinsip rekayasa dalam rangka memperoleh perangkat lunak yang dapat dipercaya dan dapat bekerja secara efisien pada mesin nyata.
- Menurut IEEE 610.12
  sebuah studi pendekatan dan aplikasi secara sistematis, disiplin pengembangan operasi dan pemeliharaan PL yang kesemuanya itu merupakan aplikasi rekayasa yang berkaitan dengan PL.
- Menurut Wikipedia
  “Software engineering is the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software“. Secara ringkasnya adalah bahwa SE mencakup pembuatan, pengembangan, dan pemeliharaan suatu software.

Pembuatan meliputi bagaimana suatu software dibuat mulai dari user requirements, spesifikasi, desain, testing, dokumentasi (misal berupa manual pembuatan program), dan sebagainya. Sedangkan pengembangan adalah untuk menambah fitur-fitur baru yang belum ada pada versi sebelumnya. Pemeliharaan digunakan untuk memperbaiki bugs atau errors yang tidak ketahuan ketika dalam tahap pembuatan. Pemeliharaan ini biasanya dapat berupa Service Pack, dan sebagainya.

DEFINISI PERANGKAT LUNAK

Beberapa definisi dari perangkat lunak antara lain :

- Perangkat lunak adalah program komputer ditambah konfigurasi data dan file serta ditambahkan juga dokumentasi.
- Menurut Ian Sommerville :
  "Computer programs and associated documentation. Software products may be developed for a particular customer or may be developed for a general market." Program komputer dan dokumentasi yang terkait. Produk Software dapat dikembangkan untuk pelanggan tertentu atau mungkin dikembangkan untuk umum.
- Menurut Pressman dalam bukunya Software Engineering A Practioner's Approach, perangkat lunak didefinisikan lebih rinci lagi yaitu sebagai:
  - instruksi-instruksi yang jika dieksekusi akan memberikan layanan-layanan atau fungsi seperti yang diinginkan
  - struktur data yang memungkinkan program untuk memanipulasi informasi secara proporsional
  - dokumen-dokumen yang menggambarkan operasi dan kegunaan program
- Menurut IEEE (Standard Glossary of Software Engineering Terminology, 1990), perangkat lunak adalah program komputer, prosedur, dan dokumentasi serta data yang terkait dengan pengoperasian sistem komputer.

PERANAN PERANGKAT LUNAK

Software saat ini memegang dua peran. Software sebagai sebuah produk dan, dalam waktu yang bersamaan, juga sebagai sarana untuk menghasilkan sebuah produk. Sebagai sebuah produk, software memberikan suatu kemampuan menghitung yang disatukan ke dalam hardware komputer, atau yang lebih luas lagi, pada jaringan komputer yang bisa diakses oleh hardware lokal. Software bisa berada melekat didalam sebuah telepone selular atau beroperasi didalam sebuah komputer mainframe, yang bekerja untuk merobah data menjadi informasi, memproduksi, mengatur, mendapatkan, mengubah, mempertunjukan, atau mengirimkan informasi yang sederhana, seperti sebuah bit tunggal atau, barangkali sekompleks pertunjukan dengan multimedia. Sebagai sarana yang digunakan untuk menghasilkan suatu produk, software bertindak sebagai basis untuk mengontrol komputer (sistem operasi), komunikasi informasi (jaringan), dan pembuat serta pengontrol dari program yang lainnya (software sebagai alat bantu (tools) dan lingkungan (environments)).

Software menghantar informasi, produk paling penting yang dihasilkan untuk kita, mengubah data pribadi (seperti misalnya transaksi keuangan pribadi) sehingga data itu bisa lebih berarti dalam konteks lokal; software juga mengelola informasi bisnis menjadikannya lebih bisa bersaing; juga menjadi penyedia pintu masuk ke jaringan informasi dunia(misal internet) dan yang bisa memenuhi permintaan informasi dalam berbagai bentuk.

Peran software komputer cukup banyak mengalami perubahan sepanjang kurun waktu tidak kurang dari 50 tahun. Dengan peningkatan kemampuan hardware yang

dramatis, perubahan yang terjadi lebih dalam lagi dari segi arsitektur komputer, kecepatan, peningkatan kapasitas memori dan penyimpanan, dan dengan beragamnya variasi pilihan input dan output menjadikan sistem berbasis komputer semakin lengkap dan kompleks. Kehandalan dan kompleksitas bisa menimbulkan kekaguman ketika suatu sistem yang berjalan dengan baik dan sukses, tetapi dia juga bisa menimbulkan masalah yang besar manakala mereka harus membangun sebuah sistem yang rumit.

PERKEMBANGAN PERANGKAT LUNAK

- Tahun-Tahun Awal (1950 - 1965)
  - Orientasi batch -> update data pada periode tertentu
  - Distribusi terbatas
  - PL dibuat menurut pesanan
- Era Kedua (1965 - 1975)
  - Multiuser -> ada pembagian hak akses, contoh : manager, karyawan
  - Real time -> update data langsung ketika ada perubahan
  - Database -> karena real time
  - Software produk
- Era Ketiga (1975 - 1989)
  - Sistem terdistribusi
  - Embedded Intelligence
  - Hardware biaya rendah -> kalau dulu mahal karena ukurannya sangat besar
- Era Keempat (1989 - sekarang)
  - Sistem desktop bertenaga kuat
  - Teknologi berorientasi objek (Object Oriented) -> kalau ada komponen rusak, tidak perlu membeli PL baru, cukup membeli komponen
  - Sistem pakar -> bertindak seperti pakar
  - Jaringan syaraf tiruan
  - Komputasi Paralel
  - Komputasi Jaringan
  -

KARAKTERISTIK PERANGKAT LUNAK

Untuk menambah pemahaman mengenai software (dan mengerti sepenuhnya Software Engineering), sangatlah penting untuk memeriksa ciri-ciri software yang membuat beda dengan barang-barang yang dibuat orang lainnya. Sewaktu software dibangun, proses kreasi manusia (analisa, perancangan, pembangunan, percobaan/tes) pada akhirnya diwujudkan kedalam bentuk fisik. Jika kita bangun sebuah komputer yang baru, mula-mula kita buat sketsa, gambar rancangan formal, dan dari bentuk dasar rancangan fisik tersebut, kemudian berkembang menjadi bentuk produk secara fisik (chips, circuit board, power supply, dll)[1].

Software adalah suatu kerangka berpikir atau logika bukan seperti elemen yang dapat dilihat secara fisik. Oleh karena itu software mempunyai ciri-ciri yang berbeda dibanding dengan perangkat keras (hardware) sebagai berikut :

[1]*Ian K Bray "An Introduction to Requirement Engineering". Vol 1 No .1, Addison- Wesley 2002, hal. 10*

1. Software is developed or engineered, it is not manufactured in the classical sense. (Perangkat Lunak dibangun dan dikembangkan, tidak dibuat dalam bentuk klasik.) Perangkat lunak adalah suatu produk yang lebih menekankan pada kegiatan rekayasa (engineering) dibandingkan kegiatan manufacturing (rancang bangun di pabrik). Dalam pembuatan perangkat lunak kualitas yang tinggi dicapai melalui perancangan yang baik, tetapi dalam fase perangkat keras, selalu saja ditemukan masalah kualitas yang tidak mudah untuk disesuaikan dengan perangkat lunak. Biaya untuk perangkat lunak dikonsentrasikan pada pengembangan. Hal ini berarti proyek perangkat lunak tidak dapat diatur seperti pengaturan pada proyek pemanufacturan.
2. Software doesn't "wear out". (Perangkat lunak tidak pernah usang)
   Perangkat lunak tidak rentan terhadap pengaruh lingkungan yang merusak yang mengakibatkan perangkat keras menjadi usang. Kesalahan-kesalahan yang tidak dapat ditemukan menyebabkan tingkat kegagalan menjadi sangant tinggi pada awal hidup program. Tetapi hal itu dapat diperbaiki (diharapkan tidak ditemukan lagi kesalahan lain) sehingga kurva menjadi mendatar.
3. Although the industry is moving toward component based construction most software continues to be custom built (Meskipun industri saat ini menuju pada pembangunan dengan component based namun sebagian besar software masih dibangun secara custom built.)

Kini paradigma baru mulai dikembangkan, yaitu konsep reuseability. Komponen software didisain dan diimplementasikan agar dapat digunakan kembali pada program yang berlainan.

KARAKTERISTIK PERANGKAT LUNAK BERKUALITAS

Perangkat lunak yang dikatakan bagus atau berkualitas memiliki karakteristik sebagai berikut :

- Maintainability
  adalah tingkat kemudahan perangkat lunak tersebut dalam mengakomodasi perubahan- perubahan
  Contoh : AVG merupakan software antivirus yang memiliki tingkat maintainability cukup tinggi. AVG dapat mengupdate dirinya sendiri selama komputer memiliki koneksi dengan internet atau dengan mendownload update terbarunya di situs AVG. update tersedia tiap hari dan merupakan salah satu kelebihan avg dibanding dengan beberapa antivirus lain dalam hal maintainability.
- Dependability
  adalah ketidakbergantungan perangkat lunak dengan elemen-elemen sistem lainnya atau sistem secara keseluruhan. Artinya kegagalan elemen lain tidak mempengaruhi performansi perangkat lunak

Contoh : AVG bergantung pada sistem operasi dan Selama Operating Sistem tidak ada masalah maka AVG tidak akan bermasalah

- Efficiency
  Menyangkut waktu eksekusi. Waktu eksekusi cukup singkat, dan saat melakukan scanning membutuhkan waktu yang lebih singkat bila dibandingkan dengan beberapa antivirus lain
- Usability
  adalah atribut yang menunjukkan tingkat kemudahan pengoperasian perangkat lunak.

Contoh : awalnya kita membutuhkan waktu agar terbiasa dengan Interface AVG. AVG Control-Center adalah komponen utama untuk mengontrol system AVG, dan berjalan tiap kali user melakukan login. Dengan menggunakan AVGCC settingan sistem AVG dapat diedit dan kita dapat monitoring status dari tiap komponen individual seperti status updatenya.

JENIS – JENIS PERANGKAT LUNAK

1. System software
   Melayani program-program yang lain, contoh :kompiler, editor, prosesor telekomunikasi, sistem operasi, driver. Areanya ditandai dengan eratnya interaksi dengan hardware komputer, penggunaan oleh banyak user, operasi konkuren yang membutuhkan penjadwalan, tukar-menukar sumber dan pengaturan proses yang canggih serta struktur data yang kompleks dan interface eksternal yang ganda.
2. Application Software
   Program stand alone yang dimanfaatkan untuk menyelesaikan kebutuhan spesifik dari bisnis. Aplikasi dalam area proses bisnis maupun data teknis yang ada di dalamnya digunakan untuk memfasilitasi operasional bisnis dan digunakan untuk memfasilitasi pengambilan keputusan ditingkat manajemen maupun teknis. Selain sebagai pengolah data konvensional, software aplikasi juga digunakan untuk mengontrol fungsi bisnis secara real time (misal : pemrosesan transaksi point of sale, kontrol pemrosesan manufaktur secara real time).
3. Engineering / scientific software
   Ditandai dengan algoritma numerik (number crunching). Memiliki jangkauan aplikasi mulai astronomi sampai vulkanologi, analisis otomatif sampai dinamika orbit pesawat ruang angkasa, dan biologi molekular sampai pabrik yang sudah diotomatisasi. Namun aplikasi baru dalam area teknik atau ilmu pengetahuan sedang bergerak menjauhi algoritma numerik yang konvensional.
4. Embedded software
   Ada dalam ROM, digunakan untuk mengontrol hasil serta sistem untuk keperluan konsumen dan pasar industri. Dapat melakukan fungsi terbatas serta fungsi esoterik

(contoh : key pad control microwave yang bisa mematikan otomatis sesuai waktu) atau memberikan kemampuan kontrol dan fungsi penting (contoh : fungsi digital dalam sebuah automobil seperti kontrol bahan bakar, autopilot, penampilan dashboard, sistem rem).

5. Product line software
   Dirancang agar dapat memiliki kemampuan khusus yang diperuntukkan bagi pelanggan yang berbeda. Product line software dapat difokuskan pada pasar terbatas (misal : pengontrolan persediaan produk), atau berfokus pada pasar konsumen massal/umum (misal : pengolah kata, spreadsheet, pengolahan database, komputer grafik, multimedia, entertainment serta aplikasi keuangan personal dan bisnis).
6. Aplikasi web
   Disebut “WebApps”. Dalam bentuk sederhana, WebApps merupakan kumpulan link files hypertext yang mempresentasikan informasi menggunakan text dan grafikal. Contoh : web 2.0.
7. AI (Artificial Intelligence) software
   Menggunakan algoritma non-numerik untuk menyelesaikan masalah kompleks yang tidak sesuai untuk perhitungan maupun analisis secara langsung. Contoh : sistem pakar, aplikasi dengan jaringan syaraf tiruan, image dan suara, pembuktian teorema, permainan game.

BAB III

MACAM DARI SIKLUS HIDUP PERANGKAT LUNAK (SWDLC/SOFTWARE DEVELOPMENT LIFE CYCLE)

Abstract

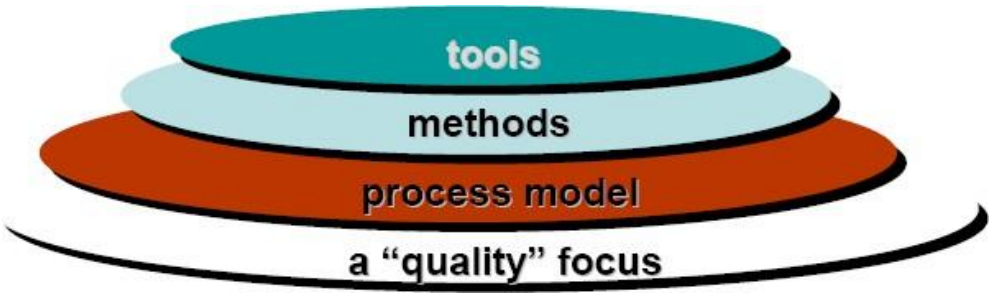
Membahas mengenai alur kehidupan yang membentuk perangkat lunak.

Kompetensi

Mahasiswa dapat memahami siklus hidup dari perangkat lunak dengan mengetahui dari segi software.

TEKNOLOGI BERLAPIS PENGEMBANGAN PERANGKAT LUNAK

Proses perangkat lunak adalah sebuah kerangka kerja untuk membangun perangkat lunak yang berkualitas tinggi. Gambar dibawah ini menunjukkan lapisan teknologi pada rekayasa Perangkat lunak[1].



Lapisan-lapisan Rekayasa Perangkat Lunak

- Dari Gambar tersebut dapat dilihat bahwa tujuan utama rekayasa perangkat lunak adalah pencapaian kualitas ("Quality Focus"). Kualitas ini diterjemahkan ke dalam ukuran-ukuran (metrics), meliputi maintainability, dependability, usability, dan efificiency yang sudah diterangkan di atas.
- Proses : mendefinisikan kerangka kerja (frame work) , sehingga pembangunan perangkat lunak dapat dilakukan secara sistematis.
- Metode : mendefinisikan bagaimana perangkat lunak dibangun, meliputi metode-metode yang digunakan dalam melakukan analisis kebutuhan, perancangan, implementasi dan pengujian. Sebagai contoh : metode terstruktur, metode berorientasi objek, dan lain-lain.
- Alat Bantu : perangkat yang bersifat otomatis maupun semi otomatis yang berfungsi mendukung tiap tahap pembangunan perangkat lunak. Contoh : CASE, CAD, dan lain-lain.

[1] *IEEE,"Software Requirements Engineering", Second Edition, IEEE Computer Society Press 2002,hal 55*.

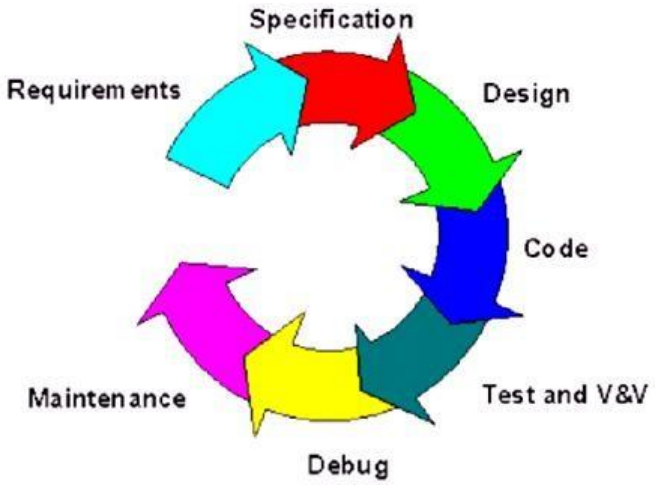
DEFINISI *SOFTWARE PROCESS*

Software process dapat didefinisikan sebagai berikut :

- Merupakan suatu deskripsi proses yang dijadikan panduan kerja bagi para software engineer dengan memetakan peran dan tanggung jawab mereka masing-masing.
- Merupakan sekumpulan aktifitas yang ditujukan untuk melakukan pengembangan maupun evolusi software.
- Merupakan suatu urutan langkah yang diperlukan dalam pengembangan atau pemeliharaan software.
- Merupakan kerangka teknis dan manajemen untuk menerapkan metode, alat bantu(tool) serta komponen SDM dalam pengerjaan software.
- (Menurut Ian Sommerville) : merupakan sekumpulan proses dan hasil yang terkait dengan proses tersebut dalam rangka pengembangan produk software.

AKTIFITAS UMUM DALAM *SOFTWARE PROCESS*

Untuk mengembangkan perangkat lunak secara memadai, proses pengembangan perangkat lunak harus didefinisikan terlebih dahulu. Usaha yang berhubungan dengan rekayasa perangkat lunak dapat dikategorikan ke dalam beberapa aktifitas dengan tanpa mempedulikan area aplikasi, ukuran proyek atau kompleksitasnya. Berikut adalah aktifitas umum yang terdapat dalam proses pengembangan perangkat lunak :



1. Requirement

   Merupakan aktifitas dimana didefinisikan mengenai “apa” (what) yang akan dibangun terkait dengan produk perangkat lunak yang akan dihasilkan. Kebutuhan dari persepsi pelanggan (requirement) didefinisikan dan disepakati. Dari aktifitas ini akan diperoleh pernyataan global mengenai kegunaan sistem serta ketersediaan sumber daya yang akan

mendukung pembangunan sistem seperti : kebutuhan sumber daya waktu, biaya dan tenaga (manusia).

2. Specification
   Merupakan aktifitas dimana kebutuhan pelanggan (requirement) yang telah ditetapkan ditransformasikan ke dalam kebutuhan sistem. Dari aktifitas ini akan diperoleh spesifikasi detil mengenai produk perangkat lunak yang akan dibangun antara lain seputar fungsionalitasnya (mengidentifikasikan informasi apa yang akan diproses, fungsi dan unjuk kerja apa yang dibutuhkan, tingkah laku sistem seperti apa yang diharapkan), kebutuhan perangkat keras dan perangkat lunak pendukung dalam pembangunannya, dan lain-lain.
3. Design
   Merupakan aktifitas dimana hasil analisis kebutuhan dan spesifikasi sistem dibentuk dalam suatu model. Pengembang harus mendefinisikan bagaimana data dikonstruksikan, bagaimana fungsi-fungsi diimplementasikan sebagai sebuah arsitektur perangkat lunak, bagaimana detail prosedur akan diimplementasikan, bagaimana interface ditandai (dikarakterisasi). Dalam membuat pemodelan, pengembang dapat menggambarkan pemodelan berdasarkan prilaku sistem ataupun secara struktural. Dari aktifitas ini akan diperoleh penggambaran sistem dalam bentuk model semacam use case diagram, data flow diagram, sequence diagram, entity relationship diagram, dan lain-lain.
4. Code
   Merupakan aktifitas dimana hasil rancangan (model) dari tahapan sebelumnya diterjemahkan dalam bentuk coding program pada sebuah bahasa pemrograman.
5. Test (verification & validation)
   Setelah rancangan diterjemahkan ke dalam bentuk coding program selanjutnya akan dilakukan proses pengujian untuk memastikan apakah aplikasi yang dibangun sudah sesuai dengan spesifikasi dan kebutuhan pelanggan yang ditahapan awal ditetapkan.
6. Debug
   Merupakan tahapan dimana akan dilakukan proses perbaikan yang diperlukan apabila pada fase pengujian masih ditemukan adanya kesalahan.
7. Maintenance
   Aktifitas ini berfokus pada perubahan (change), yang dihubungkan dengan koreksi kesalahan, penyesuaian yang dibutuhkan ketika lingkungan perangkat lunak berkembang, serta perubahan sehubungan dengan perkembangan yang disebabkan oleh perubahan kebutuhan pelanggan. Fase pemeliharaan mengaplikasikan lagi langkah-langkah pada fase definisi dan fase pengembangan, tetapi semuanya tetap bergantung pada konteks perangkat lunak yang ada. Ada empat tipe perubahan yang terjadi selama masa fase pengembangan yaitu :
   a. Koreksi (corrective)

Meskipun dengan jaminan kualitas yang terbaik, sepertinya pelanggan akan tetap menemukan cacat pada perangkat lunak. Pemeliharaan korektif mengubah perangkat lunak, membetulkan cacat atau rusak.

b. Adaptasi (adaptive)
Dari waktu ke waktu, lingkungan original (contohnya CPU, sistem operasi, aturan-aturan bisnis, karakterisasi produk eksternal) dimana perangkat lunak dikembangkan akan terus berubah. Pemeliharaan adaptif menghasilkan modifikasi kepada perangkat lunak untuk mengakomodasi perubahan pada kebutuhan fungsional original.

c. Pengembangan (perfective)
Ketika perangkat lunak dipakai, pelanggan akan mengenali fungsi-sungsi tambahan yang memberi mereka keuntungan. Pemeliharaan perfektif memperluas perangkat lunak sehingga melampaui kebutuhan fungsi originalnya.

d. Pencegahan (preventive)
Keadaan perangkat lunak semakin memburuk sehubungan dengan waktu, dan karena itu preventive maintenance yang sering juga disebut software engineering (rekayasa perangkat lunak), harus dilakukan untuk memungkinkan perangkat lunak melayani kebutuhan para pemakainya. Pada dasarnya preventive maintenance melakukan perubahan pada program komputer sehingga bisa menjadi lebih mudah untuk dikoreksi, disesuaikan dan dikembangkan.

Aktifitas dasar dalam software process menurut Ian Sommerville antara lain :

1. Specification
   Aktifitas dimana pelanggan dan pengembang membuat definisi mengenai software yang akan dibangun dan juga batasan pada pengembangannya.
2. Development
   Aktifitas dimana software diproduksi (didisain dan diprogram)
3. Validation
   Aktifitas dimana dilakukan proses pengecekan apakah software yang dibangun sudah sesuai dengan keinginan pelanggan.
4. Evolution
   Aktifitas dimana dilakukan perubahan/modifikasi terhadap software untuk diadaptasikan terhadap perubahan kebutuhan dari pelanggan.

Aktifitas atau langkah-langkah yang dijabarkan dalam pengembangan perangkat lunak tersebut harus diimbangi dengan sejumlah aktifitas pelindung (umbrella activities). Kegiatan-kegiatan khusus di dalam kategori ini menyangkut :

- Manajemen proyek PL : melindungi agar PL yang ada hasilnya bagus
- Formal technical review, contoh : menemui user dan mengecek kebutuhannya untuk analisis. Jika tidak dilakukan, nanti kita buat PL yang sesuai pikiran kita,

bukan sesuai dengan kebutuhan user.

- Software quality assurance (jaminan kualitas PL) : langkah supaya PL berkualitas
- Manajemen konfigurasi PL : bagaimana PL bisa dikonfigurasikan atau dibuat
- Pembuatan dan penyiapan dokumen : sebagai senjata jika user tiba-tiba meminta tambahan fungsi PL
- Reusability management (manajemen reusabilitas): komponen PL bisa dipakai ulang
- Measurement (pengukuran) : harus ada dalam setiap tahap
- Risk Management (manajemen resiko) : risiko harus diantisipasi supaya tidak gagal Aktifitas pelindung diaplikasikan ke seluruh proses perangkat lunak.

KARAKTERISTIK *SOFTWARE PROCESS* YANG BAIK

- Understandability
  Proses secara eksplisit didefinisikan sehingga mudah dipahami bagi siapapun yang terlibat di dalam proses pengembangan.
- Visibility
  Aktifitas proses memberi hasil yang jelas sehingga kemajuan proses dapat terlihat dari luar pihak pengembang.
- Supportability
  Proses dapat didukung oleh teknologi semacam CASE tools.
- Acceptability
  Penerimaan atas proses yang terdefinisi dan yang digunakan oleh software engineer selama pembangunan produk perangkat lunak.
- Reliability
  Proses didisain dengan suatu metode untuk menghindar dari kesalahan, dan apabila ada kesalahan dapat terdeteksi sedini mungkin sebelum mengakibatkan cacat pada produk akhir.
- Robustness
  Proses dapat terus dilanjutkan meskipun terdapat masalah yang tidak diharapkan muncul.
- Maintainability
  Proses dapat mengadaptasi terhadap permintaan perubahan ataupun perbaikan
- Rapidity
  Proses dapat diselesaikan dalam waktu yang relatif cepat

MODEL PROSES RPL

Model proses disebut juga dengan aliran kerja (workflow), yakni tata cara bagaimana elemen-elemen proses berhubungan satu dengan lainnya. Aliran kerja ini dapat juga disebut dengan siklus hidup (life-cycle) sistem yang dimulai dari sejak sistem diajukan untuk dibangun hingga saat ia ditarik dari peredaran.

Fungsi utama model proses pengembangan perangkat lunak adalah :

- menentukan tahap-tahap yang diperlukan untuk pengembangan perangkat lunak.
- Menentukan urutan pelaksanaan dari tahap-tahap tersebut dalam rangka pengembangan perangkat lunak.
- menentukan kriteria transisi/perpindahan dari satu tahap ke tahap berikutnya.

Untuk menentukan mana model yang terbaik, kita harus tahu apa kelebihan dan kekurangan model-model proses tersebut. Akan tetapi, model proses biasanya bukan ditentukan mana yang terbaik atau tidak, tetapi ditentukan oleh karakteristik dari berbagai macam faktor, misalnya tim SE-nya, atau software-nya sendiri, waktu untuk melakukan SE, kebijakan-kebijakan dari perusahaan, dan sebagainya. Dengan menggunakan model proses yang terbaru pun, ketika diaplikasikan ke dalam perusahaan misalnya, tetapi kalau tim SE-nya tidak siap dengan kondisi yang mengharuskan menggunakan model proses tersebut, tentunya tidak mungkin bisa dilakukan. Kalaupun dipaksakan tentu saja hasilnya tidak akan maksimal. Akan tetapi di perusahaan lain, bisa jadi menerapkan model proses yang sama, tetapi hasilnya bagus, karena tim SE-nya siap atau scope softwarenya berbeda[1].

[1] *IEEE,"Software Requirements Engineering", Second Edition, IEEE Computer Society Press 2002,hal 55*.

BAB IV
TUJUAN PERENCANAAN PROYEK.
RUANG LINGKUP PROYEK PERANGKAT LUNAK.
SUMBER DAYA YANG DIBUTUHKAN.
ESTIMASI PROYEK PERANGKAT LUNAK.

Abstract
Membahas mengenai tujuan, ruang lingkup dan sumberdaya yang dibutuhkan dalam proyek perangkat lunak.
Kompetensi
Mahasiswa dapat membuat perencanaan proyek perangkat lunak.

TUJUAN PERENCANAAN PROYEK

Proses manajemen proyek perangkat lunak dimulai dengan beberapa aktivitas yang secara kolektif disebut dengan project planning (perencanaan proyek). Aktivitas ini dimulai dengan estimasi, yang merupakan gambaran dimana kita melihat masa depan serta menerima tingkat ketidakpastian sebagai bahan pembicaraan. Perencanaan proyek memberikan sebuah peta jalan bagi suksesnya rekayasa perangkat lunak.

Observasi pada Estimasi

Estimasi yang diperlukan dalam perancangan proyek perangkat lunak di antaranya adalah sumber daya, biaya, dan jadwal sebagai usaha dalam pengembangan perangkat lunak, mengakses informasi historis yang baik, dan keberanian untuk melakukan pengukuran kuantitatif bila hanya data kualitatif saja yang ada. Berikut adalah yang menimbulkan ketidakpastian dalam estimasi :

- Project complexity (kompleksitas proyek) berpengaruh kuat terhadap ketidakpastian yang inheren dalam perencanaan. Komplekitas ini merupakan pengukuran relatif yang dipengaruhi oleh kebiasaan dengan usaha yang dilakukan sebelumnya.
- Project size (Ukuran proyek) Merupakan faktor penting yang dapat mempengaruhi akurasi estimasi. Bila ukuran bertambah maka ketergantungan di antara berbagai elemen perangkat lunak akan meningkat dengan cepat.
- Structural uncertainty (Ketidakpastian struktural) Tingkat ketidakpastian strutural

juga berpengaruh dalam risiko estimasi. Dengan melihat kembali, kita dapat mengingat lagi hal-hal yang terjadi dan dapat menghindari tempat-tempat dimana masalah muncul.

Risiko diukur melalui tingkat ketidakpastian pada estimasi kuantitatif yang dibuat untuk sumber daya, biaya, dan jadwal.Bila ruang lingkup proyek atau syarat proyek tidak dipahami dengan baik, maka risiko dan ketidakpastian menjadi sangat tinggi. Perencana perangkat lunak harus melengkapi fungsi, kinerja, dan definisi interface(yang diisikan ke dalam spesifikasi sistem). Pendekatan-pendekatan rekayasa perangkat lunak modern (seperti model proses evolusioner) memakai pandangan pengembangan yang interaktif. Dengan pandangan semacam ini dimungkinkan untuk melihat estimasi dan merevisinya bila customer mengubah kebutuhannya.

Tujuan Perencanaan Proyek

Tujuan perencanaan proyek perangkat lunak adalah untuk menyediakan sebuah kerangka kerja yang memungkinkan manajer membuat estimasi yang dapat dipertanggungjawabkan mengenai sumber daya, biaya dan jadwal. Tujuan perencanaan dicapai melalui suatu proses penemuan informasi yang menunjuk ke estimasi yang dapat dipertanggungjawabkan.

Ruang Lingkup Perangkat Lunak

Penentuan ruang lingkup perangkat lunak merupakan aktivitas pertama dalam perencanaan proyek perangkat lunak. Ruang lingkup perangkat lunak menggabarkan fungsi, kinerja, batasan, interface dan reliabilitas. Fungsi yang digambarkan dalam statmen ruang lingkup dievaluasi dan disaring untuk memberikan awalan yang lebih detail pada saat estimasi dimulai. Pertimbangan kinerja melingkupi pemrosesan dan kebutuhan waktu respon. Batasan ini mengidentifikasi dari batas yang ditempatkan pada perangkat lunak oleh perangkat keras eksternal, memori, atau sistem informasi yang ada.

Mencari Informasi Yang Dibutuhkan Untuk Ruang Lingkup

Teknik yang banyak dipakai secara umum untuk menjembatani jurang komunikasi antara pelanggan dan pengembang serta untuk memulai proses komunikasi adalah dengan melakukan BAB atau wawancara pendahuluan. Gause & weinberg mengusulkan bahwa analis harus memulai dengan mengajukan pertanyaan-pertanyaan bebas konteks, yaitu serangkaian pertanyaan yang akan membawa pada pemahaman mendasar terhadap masalah, orang yang menginginkan suatu solusi, sifat solusi yang diharapkan, dan efektivitas BAB itu. Beberapa pertanyaan bebas konteks pada pelanggan yang meliputi tujuan keseluruhan, serta keuntungan :

- Siapa di belakang permintaan kerja ini?
- Siapa yang akan memakai solusi ini?
- Apakah yang akan menjadi keutungan ekonomi dari sebuah solusi yang sukses?

- Adakah sumberdaya lain bagi solusi ini?

Beberapa contoh pertanyaan yang memungkinkan analis untuk memahami masalah lebih baik :

- Bagaimanakah anda menandai output yang baik yang akan dimunculkan oleh sebuah solui yang baik?
- Masalah apa yang akan dituju oleh solusi ini?
- Dapatkah anda memperlihatkan atau menggambarkan lingkungan di mana solusi akan dipakai?
- Adakah batasan atau isu kinerja khusus yang akan mempengaruhi cara pendekatan terhadap solusi?

Beberapa pertanyaan yang berfokus pada efektivitas BAB :

- Apakah anda orang yang tepat untuk menjawab pertanyaan ini? Apakah anda resmi?
- Apakah pertanyaan saya relevan dengan problem yang anda punyai?
- Apakah saya terlalu banyak pertanyaan?
- Apakah ada orang lain yang dapat menyedikan informasi tambahan?
- Adakah sesuatu yang lain yang dapat saya tanyakan kepada anda?

Bagian Question dan Answer hanya akan digunakan untuk BAB pertama yang kemudian diganti dengan format BAB yang mengkombinasikan elemen- elemen penyelesaian masalah, negoisasi, dan spesifikasi. Sejumlah peneliti lepas mengembangkan pedekatan yang berorientasi pada tim terhadap pengumpulan kebutuhan yang dapat deiterapkan untuk membangun ruang lingkup sebuah proyek, yang disebut teknik spesifikasi aplikasi yang teraplikasi (FAST).

Sumber Daya

Mengestimasi sumber daya yang dibutuhkan untuk menyelesaikan usaha pengembangan perangkat lunak yang meliputi manusia, komponen perangkat lunak, dan peranti perangkat keras/perangkat lunak.

Memperlihatkan sumber daya pengembangan sebagai sebuah piramid. Peranti perangkat keras dan perangkat lunak berada pada fondasi dari piramida dan menyediakan infrastruktur untuk mendukung usaha pengembangan(lingkungan pengembang).

Dalam tingkat yang lebih tinggi terdapat komponen perangkat lunak reuseable – blok bangungan perangkat lunak yang dapat mengurangi biaya pengembangan secara dramatis dan mempercepat penyampaian. Dan di puncak terdapat sumber daya utama yaitu manusia. Masing-masing sumber daya ditentukan dengan empat karakteristik :

- Deskripsi sumber daya.
- Statemen ketersediaan.
- Waktu kronologis sumber daya diperlukan.
- Durasi waktu sumber daya diaplikasikan.

Sumber daya manusia

Perencanaan sumber daya manusia memulai dengan mengevaluasi ruang lingkup serta memilih kecakapan yang dibutuhkan untuk mnyelesaikan pengembangan. Baik posisi organisasi maupun specialty. Jumlah orang yang diperlukan untuk sebuah proyek perangkat lunak dapat ditentukan setelah estimasi usaha pengembangan dibuat.

Sumber daya perangkat lunak reusable

Kreasi dan penggunaan kembali blok bangunan perangkat lunak yang seharusnya dikatalog menjadi referensi yang mudah, distandarisasi untuk aplikasi yang mudah, dan divalidasi untuk integrasi yang mudah. Ada empat kategori sumber daya perangkat lunak yang harus dipertimbangkan pada saat perencanaan berlangsung, yaitu :

- Komponen off-the-self Perangkat lunak yang ada dapat diperoleh dari bagian ketiga atau telah dikembangkan secara internal untuk proyek sebelumnya.
- Komponen full-experience Spesifikasi, kode, desain atau pengujian data yang sudah ada yang dikembangkan pada proyek yang lalu yang serupa dengan perangkat lunak yang akan dibangun pada proyek saat ini.
- Komponen partial-experience Aplikasi, kode, desain, atau data pengujiaan yang ada pada proyek yang lalu yang dihubungkan dengan perangkat lunak yang dibangun untuk proyek saat ini, tetapi akan membutuhkan modifikasi substansial.
- Komponen baru Komponen perangkat lunak yang harus dibangun oleh tim perangkat lunak khususnya adalah untuk kebutuhan proyek sekarang .

Lebih baik mengkhususkan syarat sumber daya perangkat lunak dari awal. Dengan cara ini evaluasi teknis dari semua alternatif dapat dilakukan dan akuisisi secara berkala dapat terjadi.

Sumber daya lingkungan

Lingkungan yang mendukung poyek perangkat lunak, yang disebut juga software engineering environment (SEE), menggabungkan perangkat lunak dan perangkat keras. Karena sebagian besar organisasi perangkat lunak memiliki konstituen ganda yang memerlukan akses ke SEE, maka perencana proyek harus menentukan jendela waktu yang dibutuhkan bagi perangkat keras dan perangkat lunak serta membuktikan bahwa sember-sumber daya tersebut dapat diperoleh.

Pada saat sebuah sistem berbasis komputer akan direkayasa, tim perangkat lunak mungkin membutuhkan akses ke elemen perangkat keras yang sedang dikembangkan oleh tim rekayasa yang lain.

Estimasi Proyek Perangkat Lunak

Biaya perangkat lunak terdiri dari presentase kecil pada biaya sistem berbasis komputer secara keseluruhan. Kesalahan estimasi biaya yang besar dapat memberikan perbedaan antara keuntungan dan kerugian. Estimasi proyek perangkat lunak dapat ditranformasi dari suatu seni yang misterius ke dalam langkah-langkah yang sistematis yang memberikan estimasi dengan risiko yang dapat diterima. Sejumlah pilihan untuk mencapai estimasi biaya dan usaha yang dapat dipertanggung jawabkan[1] :

1. Menunda etimasi sampai akhir proyek
2. Mendasarkan etimasi pada proyek-proyek yang mirip yang sudah pernah dilakukan sebelumnya
3. Menggunakan "teknik dekomposisi" yang relatif sederhana untuk melakukan estimasi biaya dan usaha proyek
4. Menggunakan satu atau lebih model empiris bagi estimasi usaha dan biaya perangkat lunak.

[1] IEEE,"Software Requirements Engineering", Second Edition, IEEE Computer Society Press 2002,hal 55.

BAB V
ANALISIS KEBUTUHAN PERANGKAT LUNAK. TEKNIK KOMUNIKASI DAN PRINSIP ANALISIS. PEMBUATAN MODEL PROTOTYPE PERANGKAT LUNAK.

Abstract
Membahas mengenai Analisa kebutuhan yang dilakukan agar perangkat lunak yang dibuat dapat memenuhi.

Kompetensi
Mahasiswa dapat menganalisa, menguasai teknik perangkat lunak dan membuatnya.

ANALISIS KEBUTUHAN PERANGKAT LUNAK

Definisi

Proses manajemen proyek perangkat lunak dimulai dengan kegiatan project planning (perencanaan proyek). Yang pertama dari aktifitas ini adalah estimation (perkiraan). Estimasi membawa resiko yang inheren (dari diri sendiri) dan resiko inilah yang membawa ketidakpastian. Yang mempengaruhi estimasi :

- Project complexity (kompleksitas proyek)
- Project size (ukuran proyek)
- Struktural uncertainty (ketidakpastian struktural)

Tujuan Perencanaan Proyek Perangkat Lunak :
menyediakan sebuah kerangka kerja yang memungkinkan manajer membuat estimasi yang dapat dipertanggungjawabkan terhadap sumber daya, biaya dan jadwal pada awal proyek yang dibatasi oleh waktu.
Aktifitas Perencanaan Proyek PL

1. Menentukan ruang lingkup PL
2. Mengestimasi sumber daya yang dibutuhkan

Ruang Lingkup PL

Ruang lingkup PL menggambarkan : fungsi, kinerja, batasan, interface dan reliabilitas. Fungsi yang digambarkan dlm statemen ruang lingkup dievaluasi untuk memberikan awalan yang lebih detail pada saat dimulai estimasi.

Kinerja melingkupi pemrosesan dan kebutuhan waktu respon. Batasan mengidentifikasi batas yang ditempatkan pada PL oleh perangkat keras eksternal, memori atau sistem lain.

Informasi yang dibutuhkan (awal BAB antara pelanggan dan pengembang). Pertanyaan berfokus pada pelanggan, tujuan keseluruhan serta keuntungan.

- Siapa di belakang permintaan kerja ini?
- Siapa yang akan memakai solusi ini?
- Apakah keuntungan ekonomi dari solusi yang sukses?
- Adakah sumber daya lain bagi solusi ini? * Pertanyaan yang memungkinkan analis memahami masalah lebih baik dan pelanggan menyuarakan persepsi tentang sebuah solusi.
- Bagaimana Anda (pelanggan) menandai output yg baik yg akan dihasilkan oleh sebuah solusi yg baik?
- Masalah apa yang dituju solusi ini?
- Dapatkah anda menggambarkan lingkungan dimana solusi akan dipakai?
- Adakah batasan atau isu kinerja khusus yg akan mempengaruhi PL berinteraksi dengan elemen sistem berbasis komputer.

Konsep sebuah interface diinterpretasi untuk menentukan:

1. Hardware yg mengeksekusi PL dan device yg dikontrol secara tidak langsung oleh PL
2. Software yg sudah ada dan harus dihubungkan dengan PL yg baru
3. Manusia yg menggunakan PL melalui keyboard atau perangkat I/O lain
4. Prosedur

Sumber Daya

1. Manusia
2. Perangkat Lunak Kategori yg diusulkan BEUNATAN
   - Komponen Off-the-self
   - Komponen Full-Experience
   - Komponen Partial-Experience
   - Komponen Baru
3. Lingkungan (Software Engineering Environment - SEE), menggabungkan PL dan Perangkat Keras.

Estimasi biaya dan usaha dapat dilakukan dengan cara :

1. Menunda estimasi sampai akhir proyek.
2. Berdasarkan estimasi pada proyek yg mirip sebelumnya.
3. Menggunakan 'teknik dekomposisi' yg relatif sederhana u/ estimasi biaya dan usaha proyek.

4. Menggunakan satu atau lebih model empiris bagi estimasi usaha dan biaya PL.

Akurasi estimasi proyek PL didasarkan pada :

1. Tingkat dimana perencana telah dengan tepat mengestimasi ukuran produk yg akan dibuat.
2. Kemampuan mengestimasi ukuran ke dalam kerja manusia, waktu kalender, dan dolar.
3. Tingkat dimana rencana proyek mencerminkan kemampuan tim PL.
4. Stabilitas syarat produk serta lingkungan yg mendukung usaha pengembangan PL.

Putnam dan Myers mengusulkan 4 masalah penentuan ukuran :

- Fuzzy-logic sizing (logika kabur)
  Perencana harus mengidentifikasi tipe aplikasi, membuat besarannya dalam skala kuantitatif kemudian dibandingkan dengan rentang orisinil.
- Function point sizing
  Perencana mengembangkan estimasi berdasarkan karakteristik domain informasi. - Standard component sizing PL dibangun dari sejumlah 'komponen standar' yg umum (subsistem, modul, laporan, program interaktif).
- Change sizing
  Digunakan jika PL yang ada harus dimodifikasi dengan banyak cara sebagai bagian dari proyek.

Data baris kode (LOC) dan titik fungsi (FP) pada estimasi proyek digunakan sbg :

1. variabel estimasi yg dipakai untuk mengukur masing-masing elemen PL.
2. metrik baseline yang dikumpulkan dari proyek yg lalu dan dipakai dengan variabel estimasi untuk mengembangakan proyeksi kerja dan biaya.

Expected Value untuk variabel estimasi :

EV = (Sopt + 4Sm + Spess) / 6
EV = Expected value
Sopt = Estimasi optimistik
Sm = Estimasi paling sering S
pess = Estimasi pesimistik

Apakah estimasi ini benar ? ' Kita tidak yakin!' Bagaimanapun canggih teknik estimasi harus di-cross-check dengan pendekatan lain.

Keputusan MAKE-BUY

Pada aplikasi PL, dari segi biaya sering lebih efektif membeli dari pada mengembangkan sendiri. Manajer RPL dihadapkan pada keputusan make- buy dengan pilihan :

1. PL dapat dibeli (atau lisensi) off-the-self.
2. Komponen PL full-experience dan partial-experience, dapat diperoleh dan kemudian dimodifikasi dan integrasi untuk memenuhi kebutuhan sendiri.
3. PL dapat dibuat custom-built oleh kontraktor luar untuk memenuhi spesifikasi pembeli.

Untuk produk PL yang mahal, langkah-langkah di bawah ini dapat dipetimbangkan:

1. Kembangkan spesifikasi untuk fungsi dan kinerja PL yg diperlukan.
2. Perkirakan biaya internal untuk pengembangan dan tanggal penyampaian.
3. a. Pilih tiga atau empat calon aplikasi yang paling cocok dengan aplikasi anda.
   b. Pilih komponen yang reusable yg dapat membantu konstruksi aplikasi yg diperlukan.
4. Kembangkan sebuah matriks perbandingan untuk membandingkan calon PL.
5. Evaluasi masing-masing paket PL berdasarkan kualitas produk sebelumnya, dukungan penjual, arah proyek, reputasi dsb.
6. Hubungi pemakai PL lain dan mintalah pendapat mereka.

Pada analisis akhir, keputusan make-buy berdasarkan kondisi sbb:

1. Tanggal penyampaian
2. Biaya yang diperlukan
3. Dukungan

Membuat Pohon Keputusan

Rekayasa atau organisasi PL dapat menggunakan teknik statistik analisis pohon keputusan dengan pilihan [1]:

1. membangun sistem X dari permulaan.
2. menggunakan lagi komponen partial experience yang ada untuk membangun sistem
3. membeli sebuah produk perangkat lunak yang dapat diperoleh dan dimodifikasi untuk memenuhi kebutuhan lokal.
4. mengkontrakkan pengembangan PL ke vendor luar.

Bila sistem dibangun dari permulaan, hanya 70% probabilitasnya sehingga pekerjaan menjadi sulit. Perencana proyek dapat memproyeksikan usaha pengembangan yang sulit berbiaya $450.000, usaha yang sederhana diperkirakan berbiaya $380.000.

Expected value untuk biaya dihitung sepanjang cabang pohon keputusan, adalah :

Expected Cost = ∑ (jalur probabilitas)i * (biaya jalur terestimasi)i

dimana i adalah garis edar pohon keputusan.

Contoh : expected costbuild = 0.30 ($380 K) + 0.70 ($450 K) = $ 429 K

expected costreuse = 0.40 ($275 K) + 0.60 (0.20 ($310 K) + 0.80 ($490 K))

= $ 382 K

expected costbuy = 0.70 ($210 K) + 0.30 ($400 K) = $ 267 K e

xpected costcontract = 0.60 ($350 K) + 0.40 ($500 K) = $ 410 K

Berdasar biaya probabilitas dan proyeksi, expected cost yang paling rendah adalah pilihan buy

Catatan : Banyak kriteria yang harus dipertimbangakan, bukan hanya biaya, seperti pengalaman pengembang/ vendor/ kontraktor, penyesuaian kebutuhan,kecenderungan perubahan dapat mempengaruhi keputusan akhir!.

[1] IEEE,"Software Requirements Engineering", Second Edition, IEEE Computer Society Press 2002,hal 55.

BABVI

SPESIFIKASI PERANGKAT LUNAK

KAJIAN SPESIFIKASI PERANGKAT LUNAK

Abstract

Bahasan mengenai spesifikasi perangkat lunak yang dipergunakan untuk memenuhi kebutuhan pemakai.

Kompetensi

Mahasiswa dapat menjabarkan spesifikasi Perangkat Lunak dan melakukan pengkajian spesifikasi Perangkat Lunak

SPESIFIKASI PERANGKAT LUNAK

Definisi

Kebutuhan Perangkat Lunak adalah kondisi, kriteria, syarat atau kemampuan yang harus dimiliki oleh perangkat lunak untuk memenuhi apa yang disyaratkan atau diinginkan pemakai[1].

Jenis perangkat lunak :

1. Kebutuhan fungsional : kebutuhan yang berkaitan dengan fungsi atau proses transformasi yang harus mampu dikerjakan oleh perangkat lunak.
   contoh : perangkat lunak harus dapat menyimpan semua rincian data pesanan pelanggan
2. Kebutuhan Antarmuka : kebutuhan yang menghubungkan perangkat lunak dengan elemen perangkat keras, perangkat lunak, atau basis data.
   contoh : perangkat untuk input data dapat berupa keyboard, mouse, dan scanner.
3. Kebutuhan unjuk kerja : kebutuhan yang menetapkan karakteristik unjuk kerja yang harus dimiliki oleh perangkat lunak
   contoh : perangkat lunak harus bisa mengolah data sampai 1juta record untuk tiap transaksi

1. PENDAHULUAN

Dalam Pembuatan Spesifikasi Kebutuhan Perangkat Lunak (SKPL) yang dapat menggambarkan kebutuhan pengguna dan memberikan arah agar perancangannya sesuai dengan rancangan.

Tujuan

Membahas mengenai perangkat lunak yang kebutuhan softwarenya ada pada dokumen ini. Gambarkan lingkup dari produk yang dilingkupi oleh SKPL ini, khususnya jika gambaran SKPL hanya bagian dari sebuah sistem atau sub system.

---

[1] IEEE,"Software Requirements Engineering", Second Edition, IEEE Computer Society Press 2002,hal 58.

Ruang Lingkup Perangkat Lunak

Memberikan gambaran singkat mengenai perangkat lunak yang akan dikembangkan, termasuk keuntungan, tujuan dan sasaran. Terangkan juga hubungan perangkat lunak dengan sasaran perusahaan atau strategi binis.

Target Audience

Untuk target audience menjelaskan siapa yang harus memahami dan menggunakan dokumen. Dapat menunjukkan bagaimana perbedaan dari masing-masing audience tersebut dalam memperlakukan dokumen .

2. DESKRIPSI UMUM

Tentang Perangkat Lunak

Menggambarkan secara apa adanya keadaan perangkat lunak yang ditetapkan dalam SKPL. Sebagai contoh, perangkat lunak yang mengganti sebagian sistem yang ada. Jika SKPL mendefinisikan komponen dari sistem yang besar, kebutuhan dari sistem tersebut di bagi menjadi dua, yaitu fungsionalitas perangkat lunak dan identifikasi antarmuka. Diagram sederhana dapat membantu untuk menunjukkan komponen utama dari sistem keseluruhan, interkoneksi subsistem dan antarmuka eksternal[1].

Fungsi-fungsi Perangkat Lunak

Berisi ringkasan dari fungsi-fungsi utama perangkat lunak. Dapat berupa ringkasan yang bersifat high level . Aturlah fungsi-fungsi tersebut agar mudah dipahami saat membaca SKPL . Gambarkan dengan menggunakan seperti data flow diagram level 0 atau object class diagram.

Karakteristik dan Klasifikasi Pengguna

Memperkenalkan variasi klasifikasi pengguna yang akan mempergunakan perangkat lunak ini. Klasifikasi pengguna bisa dibedakan berdasarkan banyak pengguna, kumpulan pengguna fungsi perangkat lunak, keahlian teknis, keamanan atau pembagian hak akses.

Terangkan karakteristik dan kebutuhan yang berhubungan dengan setiap klasifikasi pengguna. Membedakan klasifikasi pengguna yang sangat penting untuk perangkat lunak ini, dengan klasifikasi pengguna yang dianggap kurang penting untuk dipuaskan.

Lingkungan Operasi

Merupakan gambaran lingkungan dimana perangkat lunak ini akan beroperasi, termasuk platform perangkat keras, versi dan sistem operasi, dan berbagai software atau aplikasi lain yang diperlukan untuk mendampinginya.

[1] IEEE,"Software Requirements Engineering", Second Edition, IEEE Computer Society Press 2002,hal 59.

Batasan Desain dan Implementasi

Menggambarkan beberapa item atau isu yang dapat membatasi pengembangan perangkat lunak. Hal ini termasuk: kebijakan regulasi perusahaan, keterbatasan perangkat keras (timing requirements, memory requirements), antarmuka pada aplikasi lain, teknologi tertentu, tools, dan database yang digunakan, operasi paralel, kebutuhan bahasa, protokol komunikasi, pertimbangan keamanan, konvensi desain atau standart pemrograman (contohnya jika organisasi customer/pengguna akan bertanggung jawab dalam pemelihara perangkat lunak yang telah diberikan)

Dokumentasi Bagi Pengguna

Berisi daftar komponen-komponen dokumentasi yang diperuntukkan kepada pengguna (seperti: user manual, bantuan on-line, dan tutorials) dan yang akan diberikan bersama-sama perangkat lunaknya.

Asumsi dan Ketergantungan

Berisi daftar beberapa asumsi yang akan mempengaruhi beberapa kebutuhan dalam SKPL. Termasuk di dalamnya third-party atau komponen komersil dalam perencanaan yang digunakan, isu-isu atau batasan tentang pengembangan atau lingkungan operasi. Pekerjaan pembuatan SKPL ini akan terpengaruh jika asumsi- asumsinya tidak benar, tidak shared, atau berubah. Tunjukkan juga ketergantungan terhadap faktor eksternal, seperti komponen-komponen perangkat lunak yang dimaksudkan untuk digunakan pada proyek lain, kecuali kalau telah disiapkan pada dokumen lain (misalnya dalam dokumen perencanaan proyek).

3. KEBUTUHAN ANTARMUKA EKSTERNAL

Antarmuka Pengguna

Menguraikan karakteristik logik dari setiap antarmuka antara produk perangkat lunak dan penggunanya. Bisa berupa contoh gambar screen, beberapa standar GUI atau arahan bentuk yang harus diikuti, batasan screen layout, standart buttons dan function (misal help) yang akan kelihatan pada setiap screen, keyboard shortcuts, standart tampilan error message, dan yang lainnya. Tentukan komponen perangkat lunak yang diperlukan

untuk antarmuka pengguna. Detail dari desain antarmuka pengguna ada pada dokumen terpisah yaitu spesifikasi antarmuka pengguna.

Antarmuka Perangkat Keras

Gambarkan karakteristik logik dan fisik dari setiap antarmuka antara produk perangkat lunak dan komponen perangkat keras dari sistem. Boleh berupa tipe peralatan pendukung, data alamiah dan kontrol interaksi antara perangkat lunak dan perangkat keras, dan protokol komunikasi yang digunakan.

[1]Ian K Bray "An Introduction to Requirement Engineering". Vol 1 No .1, Addison- Wesley 2002, hal. 12

Antarmuka perangkat lunak

Menjelaskan koneksi antara perangkat lunak ini dengan komponen perangkat lunak tertentu lainnya (nama dan versi), termasuk database, sistem operasi, tools, libraries, dan komponen komersial yang terintegrasi. Tunjukkan item-item data atau pesan yang datang kepada sistem dan hasilnya dan gambaran dari penggunaan setiap hasil tersebut. Gambaran kebutuhan servis dan komunikasi. Menunjuk pada dokumen yang menguraikan detail pemrograman aplikasi interface protocol. Identifikasi data yang akan dibagi antar komponen perangkat lunak. Jika mekanisme pembagian data harus terimplementasi dengan cara yang khusus (contoh, penggunaan lingkungan data global sistem operasi multitasking), terutama batasan implementasinya.

Antarmuka Komunikasi

Menguraikan asosiasi kebutuhan dengan beberapa fungsi komunikasi yang dibutuhkan oleh perangkat lunak ini, termasuk e-mail, web browser, protokol komunikasi network server, forms elektronik, dan lain sebagainya. Identifikasi beberapa hal yang berhubungan dengan format message. Identifikasi bebrapa standart komunikasi yang akan digunakan, seperti FTP atau HTTP. Menetapkan keamanan komunikasi atau isu tentang encrypsi, kecepatan transfer data, dan mekanisme sinkronisasi.

4. FEATURE SISTEM

Adalah bagian untuk mengilustrasikan kebutuhan fungsional perangkat lunak dengan mengelompokkan secara feature sistem, yaitu servis utama yang disediakan oleh perangkat lunak. Pengelompokan featur sistem pada bab ini sebaiknya dengan use case, jenis operasi, user class, object class, hirarki fungsionalitas atau kombinasinya, apapun yang membuat dapat lebih mengetahui tentang perangkat lunak tersebut.

5. KEBUTUHAN NONFUNGSIONAL LAINNYA

Kebutuhan Kinerja

Jika ada kebutuhan kinerja perangkat lunak yang kondisinya bervariasi, nyatakan dan terangkan dasar pemikirannya, agar dapat membantu pengembang dalam memahami tujuan dan pemilihan desain yang cocok. Terutama yang berhubungan dengan waktu untuk sistem real time. Buatlah kebutuhan yang sedemikian jelas dan mungkin. Pernyataan kebutuhan kinerja untuk satu kebutuhan fungsional atau feature.

Kebutuhan Keamanan

Spesifikasikan kebutuhan yang mementingkan kemungkinan hilang, rusak atau kesalahan akan hasil dari penggunaan perangkat lunak. Tentukan beberapa usaha perlindungan atau aksi yang harus dilakukan untuk mencegahnya. Tunjuklah beberapa kebijakan eksternal atau regulasi isu tentang keamanan yang mempengaruhi penggunaan dan desain perangkat lunak. Temukan beberapa setifikasi keamanan yang dapat memberikan kepuasan.

Kebutuhan Perlindungan Keamanan

Spesifikasikan kebutuhan yang concern pada keamanan atau isu privasi di sekitar penggunaan perangkat lunak atau proteksi oleh perangkat lunak pada penggunaan atau pembuatan data. Tentukan kebutuhan autentifikasi identitas pengguna. Tunjuklah beberapa kebijakan eksternal atau regulasi yang berisi isu-isu keamanan yang mempengaruhi penggunaan perangkat lunak. Temukan beberapa setifikasi keamanan atau privasi yang harus memuaskan.

Atribut Kualitas Perangkat Lunak

Spesifikasikan beberapa tambahan karakteristik kualitas dari perangkat lunak yang penting bagi pengguna atau pengembang. Pertimbangkan tentang adaptability, availability, correctness, flexibility, interoperability, maintainability, portability, reliability, reusability, robustness, testability dan usability. Tuliskan pertimbangan-pertimbangan tersebut agar menjadi spesifik, kuantitatif dan memungkinkan untuk diverifikasi. Setidaknya, klarifikasikan preferensi relatif dari variasi antribut, seperti lebih mudah menggunakannya dari pada mempelajarinya.

Aturan Penggunaan

Daftar beberapa prinsip pengoperasian perangkat lunak, seperti fungsi-fungsi yang dapat dilakukan seseorang pada situasi tertentu. Ingat, bukan untuk kebutuhan funsional, tetapi yang menyatakan beberapa kebutuhan fungsional tertentu sebagai sebuah aturan.

6. KEBUTUHAN LAIN

Tentukan beberapa kebutuhan lain yang tidak tercover pada SKPL ini. Mungkin bisa termasuk kebutuhan database, kebutuhan menginternasionalisasikan, kebutuhan

legal/hukum, penggunaan kembali pada sebuah proyek, dan sebagainya. Ditambah beberapa bagian yang relevan untuk SKPL tersebut.

MODEL ANALISIS

Dapat digambarkan dengan menggunakan model analisis seperti data flow diagram, class diagram, state-transition diagram, atau entity relationalship diagram.

DAFTAR KEBUTUHAN

Berisi daftar nomer-nomer kebutuhan yang dapat ditunjukkan pada SKPL, sehingga dapat ditelurusi asalnya.

BABVII
DESAIN PERANGKAT LUNAK DAN REKAYASA PERANGKAT LUNAK
PRINSIP DESAIN DAN KONSEP DESAIN

Abstract
Materi ini membahas mengenai pembuatan desain perangkat lunak dan merekayasakannya, Mengenal prinsip desain besarta konsepnya.

Kompetensi
Mahasiswa dapat mendisain perangkat lunak dengan cara merekayasakan disesuaikan dengan konsep

DESAIN PL

Pada bab ini perancangan desain yang akan dibahas merupakan perancangan terstruktur lanjutan tahapan analisa terstruktur. Perancangan perangkat lunak merupakan inti teknik dari proses rekayasa perangkat lunak dan merupakan aktivitas pertama dari tiga aktivitas teknik – perancangan, pembuatan kode, dan pengujian – yang diperlukan untuk membangun dan menguji perangkat lunak.

Pengertian

Perancangan perangkat lunak dapat didefinisikan sebagai

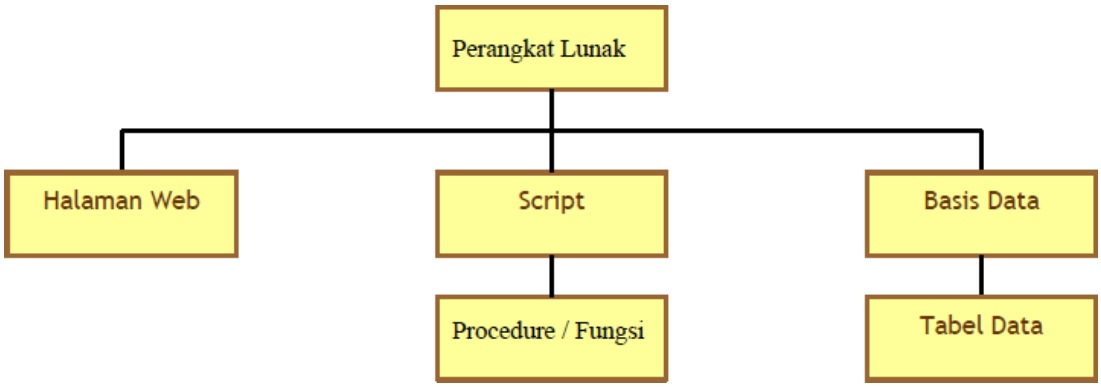
- Proses untuk mendefinisikan suatu model atau rancangan perangkat lunak dengan menggunakan teknik dan prinsip tertentu sedemikian hingga model atau rancangan tersebut dapat diwujudkan menjadi perangkat lunak.
- Proses mendefinisikan arsitektur perangkat lunak, komponen, modul, antarmuka, pendekatan pengujian, serta data untuk memenuhi kebutuhan yang sudah ditentukan sebelumnya. [IEE98]
- Proses bertahap dimana semua kebutuhan yang ada diterjemahkan menjadi suatu cetak biru yang akan digunakan untuk mengkonstruksi perangkat lunak. [PRE01]

Tujuan dilakukannya perancangan oleh seorang designer system (software engineer) adalah

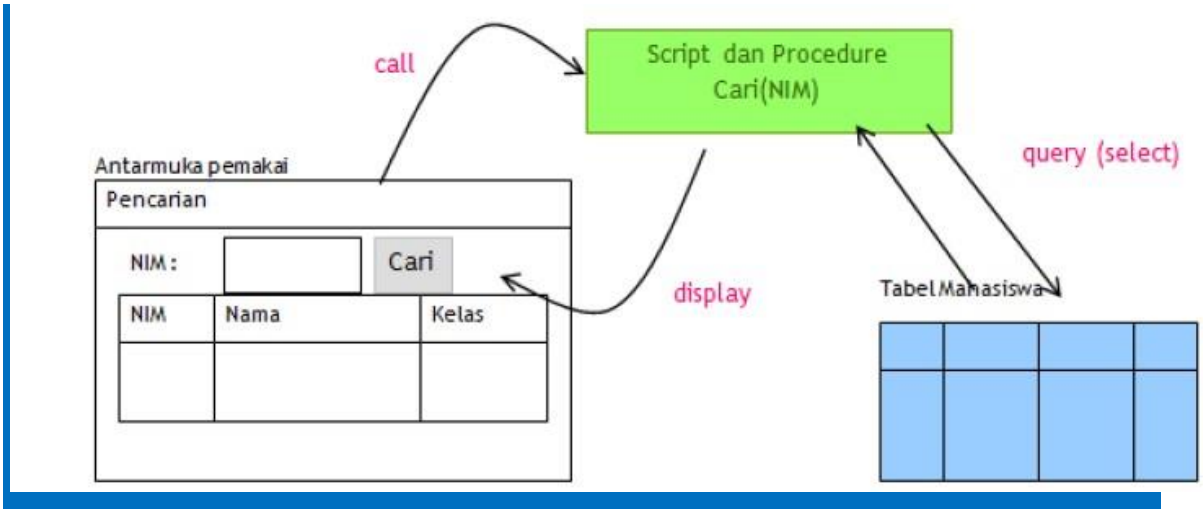
- Mendekomposisi sistem (perangkat lunak) menjadi komponen-komponennya (data, antarmuka, prosedur, arsitektur). Sebagai gambaran, pada gambar 5.1 menunjukkan dekomposisi perangkat lunak menjadi halaman web (antarmuka), script (prosedur) dan basisdata/tabel data (desain data).
- Menentukan relasi antar komponen.
- Menentukan mekanisme komunikasi antar komponen. Sebagai gambaran, pada gambar berikut yang menunjukkan mekanisme dan relasi antar komponenperangkat

lunak yaitu relasi antarmuka pemakai ke prosedur/script untuk meminta sebuah data yang diinginkan pengguna serta bagaimana sebuahprosedur mengakses tabel data agar dapat ditampilkan sesuai dengan permintaan pemakai pada antarmuka pemakai.

- Menentukan antarmuka komponen.
- Menjelaskan fungsionalitas masing-masing komponen.



Gambar 1 Dekomposisi perangkat lunak menjadi komponen-komponennya.



Gambar 2 Mekanisme dan relasi antar komponen perangkat lunak.

Prinsip Perancangan

Perancangan perangkat lunak merupakan model dan proses. Proses perancangan merupakan serangkaian langkah yang memungkinkan seorang desainer menggambarkan semua aspek perangkat lunak yang dibangun, sedangkan model perancangan hampir sama dengan rencana arsitek untuk sebuah rumah yaitu memulai dengan menyajikan totalitas hal yang akan dibangun (misal pandangan 3 dimensi dari rumah yang akan dibangun, setelah

itu akan disaring hal-hal yang memberikan panduan bagi pembangunan setiap detail dari rumah, seperti layout ruangan, layout pipa dan lainnya). Sama halnya dengan model perancangan yang dibuat untuk perangkat lunak memberikan berbagai pandangan yang berbeda terhadap program komputer.

Ada beberapa prinsip yang dikemukakan oleh Davis [DAV95] yang perlu diketahui oleh desainer untuk dapat mengendalikan proses perancangan, yaitu

- Perancangan harus dapat ditelusuri sampai ke model analisis.
- Perancangan tidak boleh berulang, maksudnya dapat mengunakan kembali rancangan yang sudah ada sebelumnya (reusable component).
- Perancangan dapat diperbaiki atau diubah tanpa merusak keseluruhan sistem.
- Perancangan harus dinilai kualitasnya pada saat perancangan, bukan setelah sistem jadi dengan kata lain siap diimplementasikan.
- Perancangan harus mempunyai beberapa pendekatan alternatif rancangan.
- Perancangan harus mengungkap keseragaman dan integrasi
- Perancangan harus meminimalkan kesenjangan intektual antara perangkat lunak dan masalah yang ada didunia nyata. Maksudnya perancangan perangkat lunak harus mencerminkan struktur domain permasalahan.
- Perancangan bukanlah pengkodean dan pengkodean bukanlah perancangan.
- Perancangan harus dikaji untuk meminimalkan kesalahan-kesalahan konseptual. Desainer harus menekankan pada hal-hal yang penting seperti elemen-elemen konseptual (ambiguitas, inkonsisten).

Jika prinsip perancangan diatas diaplikasikan dengan baik, maka desainer telah mampu menciptakan sebuah perancangan yang mengungkapkan faktor-faktor kualitas eksternal dan internal[MEY88]. Faktor-faktor eksternal adalah sifat-sifat perangkat lunak yang dapat diamati oleh pemakai (misal kecepatan, reliabilitas, ketepatan, usabilitas). Sedangkan faktor internal lebih membawa pada perancangan berkualitas tinggi dan perspektif teknis dari perangkat lunak yang sangat penting bagi para perekayasa perangkat lunak. Untuk mencapai kulitas faktor internal, seorang desainer harus memahami konsep-konsep dari perancangan perangkat lunak.

Konsep Perancangan

Pada dasarnya konsep perancangan memberikan kerangka kerja atau pedoman untuk mendapatkan perangkat lunak yang bisa berjalan dengan baik. Ada beberapa konsep perancangan yang dikemukakan oleh Pressman [PRE01] dan perlu dipahami oleh seorang desainer agar mendapatkan perancangan yang berkualitas tinggi yaitu

1. Abstraksi

Abstraksi merupakan cara untuk mengatur kompleksitas sistem dengan menekankan karakteristik yang penting dan menyembunyikan detail dari implementasi. Tiga mekanisme dasar dari abstraksi yaitu :

a. Abstraksi Prosedural, urutan instruksi yang mempunyai sebuah nama yang menggambrakan fungsi tertentu

b. Abstraksi Data, kumpulan data yang mempunya nama yang menggambarkan objek data.
c. Abstraksi Control, mengimplikasikan sebuah mekanisme kontrol dari program.

2. Dekomposisi

Dekomposisi merupakan mekanisme untuk merepresentasikan detail-detail dari fungsionalitas. Dengan adanya dekomposisi membantu para desainer mengungkapkan detail tingkat rendah ketika perancangan sedang berjalan. Jadi dekomposisi membagi perancangan secara top-down/menyaring tingkat detail dari prosedural.

3. Modularitas

Mekanisme membagi perangkat lunak ke dalam elemen-elemen kecil dan dapat dipanggil secara terpisah, biasanya elemen ini sering disebut dengan modul.

Modularitas merupakan karakteristik penting dalam perancangan yang baik karena

a. Menyediakan pemisahaan fungsionalitas yang ada pada perangkat lunak.
b. Memungkinkan pengembang mengurangi kompleksitas dari sistem.
c. Meningkatkan skalabilitas, sehingga perangkat lunak dapat dikembangkan oleh banyak personal.

Modularitas perangkat lunak ditentukan oleh coupling dan cohesion:

a. Coupling: derajat ketergantungan antar modul yang berinteraksi.
b. Cohesion: derajat kekuatan fungsional dalam suatu modul

Modul yang baik harus mempunyai chesion yang tinggi dan coupling yang rendah.

Faktor-faktor yang mempengaruhi coupling:

a. Banyaknya data yang dilewatkan antar modul (passing parameter)
b. Banyaknya kontrol data yang dilewatkan antar modul.
c. Banyaknya data global yang digunakan bersama oleh beberapa modul.

4. Arsitektur Perangkat Lunak

Arsitektur perangkat lunak merupakan struktur hirarki dari komponen program (modul), cara bagaimana komponen tersebut berinteraksi dan struktur data yang digunakan oleh komponen.

5. Hirarki Kontrol

Hirarki kontrol disebut juga dengan struktur program, yang merepresentasikan oraganisasi (hirarki) komponen program (modul) serta mengimplikasikan suatu hirarki kontrol. Hirarki kontrol tidak mengimplikasikan aspek prosedural dari perangkat lunak, seperti urutan proses, kejadian/urutan keputusan, atau pengulangan operasi.

Hirarki kontrol juga merepresentasikan dua karakteristik yang berbeda dari arsitektur peragkat lunak yaitu visibilitas dan konektivitas. Visibilitas menunjukkan serangkaian komponen program yang dapat diminta dan dipakai sebagai data oleh komponen yang diberikan dan dilakukan secara tidak langsung. Sedangkan konektivitas mengindikasikan serangkaian komponen program yang diminta secara tidak langsung atau digunakan data oleh sebuah modul yang ditetapkan.

6. **Partisi Struktural**

Struktur program harus dipartisi secara horisontal maupun struktural. Partisi ini membagi cabang-cabang yang terpisah dari hirarki modul untuk menjadi sebuah fungsi program. Ada beberapa keuntungan yang didapat mempartisi arsitektur secara horisontal, yaitu

a. menghasilkan perangkat luank yang mudaj diuji.
b. menghasilkan penyebaran efek samping yang sedikit.
c. menghasilkan perangkat lunak yang lebih mudah diperluas.
d. menghasilkan perangkat lunak yang lebih mudah dipelihara.

Selain struktur program bisa dipartisi secara horisontal bisa juga dipartisi secara vertikal, dimana kontrol dan kerja dari arsitektur program didistribusikan secara top-down.

7. Struktur Data

Struktur data merepresentasikan hubungan logis antara elemen-elemen data. Selain itu struktur data juga menentukan organisasi, metode akses, tingkat assosiativitas dan alternatif pemrosesan untuk informasi.

8. **Prosedur Perangkat Lunak**

Prosedur perangkat lunak lebih berfokus pada detail-detail pemrosesan dari masing-masing modul. Prosedur harus memberikan spesifikasi yang teliti terhadap pemrosesan, mencakup event, keputusan, operasi, dan struktur data.

9. **Penyembunyian Informasi**

Sebuah mekanisme perancangan modul sehingga informasi yang terkandung dalam modul tidak dapat diakses oleh modul lain yang tidak berkepentingan dengan informasi tersebut[1]. Informasi yang disembunyikan terdiri dari :

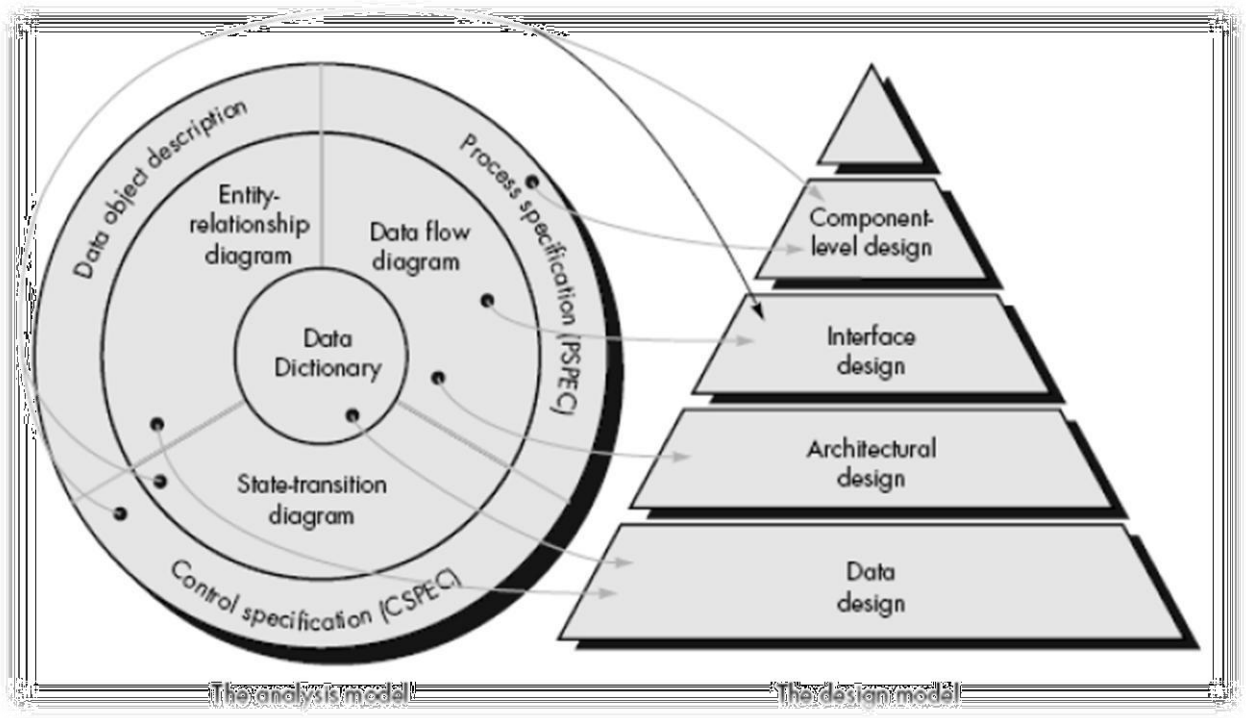
a. Representasi data
b. Algoritma seperti teknik pengurutan dan pencarian
c. Format masukan dan keluaran
d. Perbedaan mekanisme/kebijakan
e. Antarmuka modul tiangkat rendah Ada beberapa alasan kenapa konsep perancangan ini perlu dipahami oleh desainer yaitu
    1. Mengatur sistem perangkat lunak yang kompleks.

2. Meningkatkan kualitas faktor dari perangkat lunak.
3. Memudahkan penggunaan kembali simantik sistem atau perangkat lunak.
4. Memecahkan permasalahan-permasalahan perancangan yang ada pada umumnya.

[1]Ian K Bray "An Introduction to Requirement Engineering". Vol 1 No .1, Addison- Wesley 2002, hal. 13

Transformasi Model Analisa ke Perancangan

Masing-masing elemen pada model analisis akan memberikan informasi yang diperlukan untuk menciptakan model perancangan. Pada gambar berikut menunjukkan bagaimana menterjemahkan model analisis kenjadi empat model perancangan.



Gambar 3 Transformasi model analisis ke model perancangan perangkat lunak

Pada tahap perancangan ini akan dihasilkan empat model/objek perancangan, yaitu

- Perancangan data, yang berupa tabel-tabel basis data / file data konvensional Dan struktur data internal (jika diperlukan).
- Perancangan arsitektur yang berupa Structure chart dan struktur menu program (sebagai pelengkap).
- Perancangan antarmuka (interface).
- Perancangan level komponen/prosedural yang berupa spesifikasi program (algoritma).

BABVIII
DESAIN MODULAR EFEKTIF, MODEL DESAIN DAN DOKUMEN DESAIN

Abstract
Materi ini membahas mengenai desain modular efektif , model desain dan dokumentasi desain.

Kompetensi
Mahasiswa dapat membuat berbagai desain untuk keperluan membuat Rekayasa Perangkat Lunak

DESAIN SOFTWARE DAN SOFTWARE ENGINEERING

Desain ->Langkah pertama dalam fase pengembangan untuk prodengineer manapun. Bertindak sebagai pondasi untuk semua software engineering dan pemeliharaan software dengan langkah-langkah yang mengikutinya.

Tujuan seorang perancang adalah menghasilkan suatu model(atau penyajian) dari sebuah entitas yang akan dibangun

Input desain software : Model analisa kebutuhan dan dokumentasi spesifikasi Output desain software : Model desain dan dokumentasi spesifikasi desain
Desain - menerjemahkan kebutuhan ke model desain lengkap untuk sebuah produk software.

- menyediakan representasi software yang dapat diduga untuk kualitas.

Desain Software
Sejumlah metode desain dapat digunakan untuk menghasilkan desain software:

- Desain data: mentransformasi model domain informasi ke struktur data
- Desain arsitektur: menggambarkan relasi antar elemen struktural utama program
- Desain interface : mendeskripsikan bagaimana software berkomunikasi with user
- Desain prosedur: mentransformasikan elemen structural arsitektur program ke sebuah deskripsi prosedur dari komponen software.

Evolusi desain software:

- Konstruksi program modulardan metode perbaikan top-down
- Pemrograman terstruktur
- Perpindahan data flow/data structure ke sebuah defenisi desain
- Pendekatan berorientasi object

Fitur umum metode desain software:

- Sebuah mekanisme untuk perpindahan sebuah model analisa ke representasi desain
- Sebuah notasi untuk merepresentasi komponen fungsional dan interface-nya.
- Heuristic untuk perbaikan dan partisi
- Panduan untuk assessment kualitas

Proses Desain

Desain software --> sebuah proses iteratif dimana kebutuhan diterjemahkan ke sebuah "blueprint" untuk mengkonstruksi software.

Desain direpresentasikan pada level atas abstraction. Saat iterasi desain terjadi, perbaikan subsequent membawa ke representasi desain pada level abstraksi yang lebih bawah.

Kualitas desain sangat penting.

Dua metode digunakan untuk mengecek kualitas:

a) Tinjauan ulang teknis formal, dan
b) desain walkthrough

Tiga fitur umum sebuah desain yang baik dari McGlaughlin's [McG91]:

- Desain harus mengimplementasikan semua kebutuhan(eksplisit/implisit)
- Desain harus dapat dibaca dan dimengerti
- Desain harus menyediakan suatu gambar lengkap software pada aspek aspek data, fungsi dan perilaku.

Kualitas Desain

Untuk mengevaluasi sebuah desain software, sebauh criteris kualitas desain dapat digunakan.

Berikut panduan untuk sebuah desain yang baik:

- Sebuah desain harus A design should exhibit a hierarchical organization about software
- Sebuah desain harus modular berdasarkan pada partisi logical.
- Sebuah desain terdiri dari data dan abstraksi prosedural.
- Sebuah desain harus membawa ke modul dengan fitur fungsional yang independen. Sebuah desain harus membawa interface yang sederhana antar modul.
- Sebuah desain harus diturunkan menggunakan metode yang dapat berulang.

Proses desain software memacu desain yang baik dalam aplikasi dengan prinsip desain fundamental, metodologi yang sistematik, dan melalui review(pengulangan).

Prinsip Desain

David memaparkan sekumpulan prinsip untuk desain software:

Proses desain seharusnya tidak bertahan dari "tunnel vision".

- Desain harus dapat di trace ke model analisa.
- Desain seharusnya tidak menemukan/invent wheel.

- Desain harus "memperkecil jarak intellectual" antara software dan masalah-masalah pada dunia nyata.
- Desain harus mengeluarkan uniformity dan integrasi.
- Desain harus terstruktur untuk mengakomodasi perubahan.
- Desain harus terstruktur untuk mendegradasi secara halus(degrade gently).
- Desain bukan coding.
- Desain harus di-assessed untuk kualitas.
- Desain harus diulang untuk meminimalis kesalahan konsep.

Faktor kualitas eksternal: diobservasi oleh user.
Faktor kualitas internal: penting untuk engineer.

Konsep Desain

Abstraksi:
Tiap langkah pada proses software engineering adalah sebuah perbaikan pada tingkatan abstraksi dari solusi software.

- Abstraksi data: Sekumpulan data
- Abstraksi prosedural: Satu urutan instruksi pada sebuah fungsi spesifik
- Abstraksi kontrol: Sebuah program mengkontrol mekanisme tanpa menspesifikasikan detail internal.

Refinement(perbaikan): Perbaikan sebenarnya adalah sebuah proses dari elaborasi. Stepwise dari perbaikan adalah sebuah strategi desain top-down yang dikeluarkan oleh Niklaus [WIR71].
Arsitektur sebuah program dikembangkan dengan sukses memperbaiki level detail procedural.
Proses perbaikan program analog terhadap proses perbaikan dan partisi yang digunakan selama analisa kebutuhan. Perbedaan utama berada pada level detail implementasi, disamping pendekatannya.
Abstraksi dan perbaikan secara komplemen dikonsep.
Abstraksi memungkinkan seorang desainer untuk menspesifikasi prosedur dan data w/o detail. Perbaikan membantu desainer untuk me-reveal detail low-level.

Konsep Desain – Modularity
Konsep modularity telah dipakai selama hampir empat dekade. Software dibagi komponen dengan nama dan alamat yang berbeda, disebut modul.
Meyer mendefenisikan lima kriteria yang memungkinkan kita mengevaluasi sebuah metode desain dengan bergantung kepada kemampuannya mendefenisikan five sebuah sistem modular efektif:

Modular decomposability: sebuah metode desain menyediakan sebuah mekanisme sistematik untuk men- decompose atau membagi masalah ke sub-masalahà mengurangi kompleksitas dan mendapatkan modularity

Modular composability: sebuah metode desain memungkinkan komponen desain yang telah ada dirakit ke sebuah sistem baru.
Modular understandability: sebuah modul dapat dimengerti sebagai sebuah unit yang berdiri sendiri dan akan lebih mudah membangun dan mengubahnya.

Modular continuity: perubahan kecil terhadap kebutuhan sistem menghasilkan perubahan pada tiap modul, dibanding perubahan system-wide.
Modular protection: sebuah kondisi aberrant terjadi dalam sebuah modul dan efeknya di-constrain dalam modul.

Arsitektur Software

Arsitektur software adalah struktur hirarki dari komponen program components dan interaksinya.

Shaw dan Garlan menggambarkan sekumpulan properti desain arsitektur: Properti structural : Desain arsitektur mendefenisikan komponen system dan interaksinya.
Properti extra-functional : Desain arsitektur harus meng-address bagaimana arsitektur desain menerima kebutuhan untuk performance, kapasitas, ketersediaan(reliability), adaptability, keamanan.
Kumpulan sistem yang berhubungan:
desain arsitektur harus menggambar pola yang dapat diulang pada desain dengan sistem yang sama.

Metode desain arsitektural yang berbeda:
Model structural: merepresentasikan arsitektur sebagai sebuah koleksi terorganisir dari komponen.
Model framework: meningkatkan level desain abstraksi dengan mengidentifikasi secara berulang
framework desain arsitektur(pola-pola)
Model dinamis: meng-address aspek-aspek perilaku arsitektur program
Model prosess: fokus pada desain bisnis atau proses teknis
Model functional: dapat digunakan untuk merepresentasikan hierarki fungsional sebuah sistem

Partisi Structural

Struktur program harus dipartisi secara horizontal dan vertikal.

(1) Partisi horizontal menggambarkan cabang-cabang yang terbagi dari hierarki modular untuk tiap fungsi program utama.

Cara termudah adalah mempartisi sebuah sistem menjadi: input, transformasi data(pemrosesan), dan output

Keuntungan dari partisi horizontal:
- mudah untuk diuji, di-maintain, dan extend
- efek yang lebih kecil pada perubahan propagasi atau error propagasi.

Kerugian: lebih banyak data dilewatkan melalui interface modul
--> menyulitkan kontrol keseluruhan dari aliran program.

(2) Partisi vertical memaparkan kontrol dan work harus terdistribusi top-down dalam struktur program.

Keuntungan: baik pada kesesuain untuk perubahan:
- mudah untuk me-maintain perubahan
- mengurangi pengaruh perubahan dan propagasi.

Desain Modular Efektif

Informasi yang disimpan: Modul-modul seharusnya dispesifikasi dan didesain sehingga detail internal dari modul darus dapat terlihat atau dapat diakses terhadap modul lainnya.
Keuntungan utama: mengurangi pengaruh perubahan pada testing(pengujian) dan maintenance(perawatan).
Independen fungsional: Mendesain modul-modul berdasarkan fitur fungsional independen.

Keuntungan utama: modularity efektif

Cohesion: sebuah ekstensi alami dari konsep informasi yang disimpan sebuah modul dapat menampilkan sejumlah tusk

Sebuah modul cohesive menampilkan sebuah task tunggal dalam sebuah prosedur dengan interaksi kecil dengan lainnya.

Goal: untuk mencapai cohesion yang tinggi untuk modul-modul pada sebuah sistem.

Tipe-tipe yang berbeda dari cohesion:

- coincidentally cohesive: sekumpulan task yang berhubungan satu sama lain Secara bebas.
- logically cohesive: koneksi logical antara elemen-elemen pemrosesan
- communication cohesion: data di-share antar elemen-elemen pemrosesan.
- procedural cohesion: pemesanan antara elemen-elemen pemrosesan

Desain Modular Efektif

Suatu ukuran interkoneksi antar modul-modul dalam sebuah struktur program. Coupling tergantung pada kompleksitas interface antar modul.

Goal: mencoba untuk coupling terendah yang mungkin antar modul.

Coupling yang baik à mengurangi atau mencegah pengaruh perubahan dan efek ripple. mengurangi biaya pada perubahan program, testing, maintenance

Tipe-tipe coupling:

- data coupling: parameter passing atau interaksi data
- control coupling: share logical kontrol yang berhubungan (untuk sebuah data kontrol)
- common coupling: sharing data umum
- content coupling: modul, penggunaan data atau pengontrolan informasi di-maintain modul lain.

BABIX
DESAIN PERANGKAT LUNAK DAN REKAYASA PERANGKAT LUNAK
PRINSIP DESAIN DAN KONSEP DESAIN

Abstract
Materi ini membahas mengenai pembuatan desain data dan arsitektur sampai dengan pembuatan program

Kompetensi
Mahasiswa dapat membuat desain data, proses data dan pasca pemrosesan data untuk optimasi desain arsitektur dan coding

DESAIN PERANGKAT LUNAK

1. Desain Data (Data Design)
2. Desain Arsitektur (Architectural Design)
3. Desain Antar Muka (Interface Design)
4. Desain Prosedural (Procedural Design)

1. Desain Data

Desain data adalah aktivitas pertama dan terpentig dari empat aktivitas desain yang dilakukan selama rekayasa perangkat lunak. Proses pemilihan struktur dalam menentukan desain yang paling efisien sesuai kebutuhan. Tujuannya untuk mendapatkan struktur data yang baik sehingga diperoleh program yang lebih modular dan mengurangi kompleksitas pengembangan software.

Prinsip Mendesain Data:

- Prinsip analisis sistematika yang diaplikasikan pada fungsi dan perilaku harusnya juga diaplikasikan pada data.
- Semua struktur data dan operasi yang akan dilakukan pada masing-masing struktur data harus didentifikasi.
- Kamus data harus dibangun dan digunakan untuk menentukan baik data maupun desain program.
- Keputusan desain data tingkat rendah harus ditunda sampai akhir proses desain.
- Representasi struktur data hanya boleh diketahui oleh modul-modul yang menggunakan secara langsung data yang diisikan didalam struktur tersebut.
- Pustaka struktur data dan operasi yang berguna yang dapat diaplikasikan pada

struktur data tersebut harus dikembangkan.
- Desain perangkat lunak dan bahasa pemrograman harus mendukung spesifikasi dan realisasi dari tipe-tipe data abstrak.

2. Desain Arsitektur

Desain arsitektur adalah untuk mengembangkan struktur program modular dan merepresentasikan hubungan kontrol antar modul. Metode desain yang disajikan pada bagian ini mendorong prekayasa perangkat lunak untuk berkosentrasi pada desain arsitektur sebelum mencemaskan masalah perpipaan. Faktor seleksi yang penting untuk suatu metode desain adalah luasnya apliksi dimana aplikasi dapat diaplikasikan. Desain berorientasi pada aliran data dapat menyetujui rentang area aplikasi yang luas.

Proses Desain Arsitektur

Desain yang berorientasi pada aliran data merupakan suatu metode desain arsitektur yang mengijinkan transisi yang baik dari model analisis ke deskripsi desain dari struktur program. Transisi dari aliran informasi (yang ditujukan sebagai diagram aliran data) kestruktur dilakukan bagian dari proses 5 langkah:

1. Tipe aliran informasi dibangun.
2. Batas aliran diindikasikan.
3. DFD dipetakan didalam struktur program.
4. Hirarki kontrol ditentukan dengan pemfaktoran.
5. Struktur resultan disaring atau diperhalus dengan menggunakan pengukuran desain dan heuristik.

Pasca Pemrosesan Desain

Aplikasi dari pemetaan transaksi dan transformasi yang berhasil kemudian ditambahkan pada dokumentasi tambahan yang dibutuhkan sebagai bagian dari desain arsitektur. Setelah struktur dikembangkan dan disaring, tugas – tugas berikut harus dilakukan:

1. Mengembangkan narasi pemerosesan untuk masing – masing modul.
2. Menyediakan deskripsi interface untuk masing – masing modul.
3. Menentukan struktur data local dan global.
4. Mencatat semua batasan desain.
5. Mengkaji desain.
6. Mempertimbangkan “optimasi” (bila perlu dan dibenarkan).

Optimasi Desain Arsitektur

Desainer perangkat lunak harus memperhatikan perkembangan representasi perangkat lunak yang akan memenuhi semua fungsi dan persyaratan kinerja dan penerimaan jasa berdasarkan pengukuran desain kualitas.

Usul pendekatan berikut ini untuk perangkat lunak kinerja – kritis dalam optimasi desain arsitektur:

1. Kembangkan dan saringlah struktur program tanpa memperhatikan optimasi kinerja – kritis.
2. Gunakan peranti CASE yang mensimulasi kinerja run – time untuk menisolasi area inesifiensi.
3. selama iterasi desain selanjutnya, pilihlah modul yang dicurigai dan dengan hati – hati kembangkanlah prosedur (algoritma – algoritma) untuk efisiensi waktu.
4. Kodekan sebuah bahasa pemerograman yang sesuai.
5. Instrumentasikan perangkat lunak untuk mengisolasi modul yang menjelaskan utilisasi proses yang berat.
6. Bila perlu, Desain ulang atau kodekan kembali bahasa yang tergantung pada mesin untuk meningkatkan efisiensi.

3. Desain Interface

Memberikan suatu gambaran mengenai struktur program kepada perekayasa perangkat lunak. Fokus Desain Interface :

1. Desain interface antar modul
2. Desain interface antara perangkat lunak dan entitas eksternal (produser & konsumen)
3. Desain interface manusia dengan komputer

Desain Interface Manusia-Mesin

Ada empat model yang berbeda pada saat manusia-komputer/ human-komputer interface (HCL) akan didesain. Perekayasa perangkat lunak menciptakan sebuah model desain, perekayasa perangkat lunak membangun model pemakai, pemakai akhir mengembangkan citra mental yang sering disebut user's model atau perception, dan implementer sistem menciptakan system image.

Model desain dari keseluruhan sistem menggabungkan data, arsitektur, interface, dan representasi prosedural dari perangkat lunak.

Model pemakai menggambarkan profil para pemakai akhir dari sistem. Untuk membangun interface pemakai yang efektif, semua desain harus dimulai dengan suatu pemahaman terhadap pemakai yang dimaksudkan, meliputi profil, usia, jenis kelamin.

Para pemakai juga dapat dikategorikan sebagai:

- Orang baru
- Pemakai intermiten yang banyak pengetahuan

- Pemakai yang banyak pengetahuan dan sering

Persepsi sistem (model pemakai) merupakan citra sistem yang ada dikepala seorang pemakai akhir. Sebgai contoh, bila pemakai pengelola kata tersebut, persepsi sistem akan menuntun respon tersebut.
Citra sistem merangkai manifestasi bagian luar dari sistem berbasis computer (tampilan luar dan "rasa" interface), dengan semua informasi yang mendukung (buku-buku, manual, pita video) yang menggambarkan sintaksis dan semantik sistem[1].

4. Desain Prosedural

Tujuan: untuk menetapkan detail algoritma yang akan dinyatakan dalam suatu bahasa tertentu. Desain prosedural dilakukan setelah diselesaikannya perancangan desain data, arsitektur, dan antar muka software.

Coding

Program Design Language (PDL)
Bahasa keseluruhan yang sintaksnya dari bahasa tertentu (pemrograman terstruktur).

[1]Ian K Bray "An Introduction to Requirement Engineering". Vol 1 No .1, Addison- Wesley 2002, hal. 14

DAFTAR PUSTAKA

Bray, Ian K. An Introduction to Requirement Engineering, 1st published, Addison- Wesley, 2002

IEEE Std. 1233, 1998 Edition IEEE Guide for Developing System Requirements Specifications

IEEE, Software Requirements Engineering, Second Edition, IEEE Computer Society Press, 2002.

Kotonya, Gerald and Sommerville, Ian. Requirement Engineering: Processes and Techniques, John Wiley & Sons Ltd, 1998

Software Engineering Ian Sommerville

Software Engineering Roger S.Pressman