UNIFIED SOFTWARE DEVELELOPMENT PROSES (USDP)

Oleh Kundang K Juman

Pengantar USDP :

Suatu proses pengembangan harus mampu melakukan spesifikasi terhadap apa yang dilakukan, pada saat kapan hal tersebut dikerjakan, bagaimana cara mengerjakannya dan siapa yang mengerjakan sehingga tujuannya dapat tercapai. Teknik manajemen proyek (Project Management technique) digunakan untuk mengatur, dan pengontrol proyek yang dikerjakan. Pada pengembangan perangkat lunak hal tersebut dilakukan dengan menggunakan salah satu tools Rational Unified Proses yang dikerluarkan oleh IBM berdasarkan pada Unified Software

Development Process (USDP) (Jacobson, et al., 1999). USDP dikembangkan oleh team yang membangun UML. USDP terdiri atas :

- Pengembangan secara Iteratif dan incremental (Iterative and Incremental Development)
- Pengembangan berdasarkan komponen (Component Based Development)
- Pengembangan berdasarkan kebutuhan (Requirment Driven Development)
- Kemampuan untuk melakukan konfigurasi (Configurability) • Arsitektur terpusat (Architecture centrism)
- Teknik pemodelan visual (Visual Modelling technique) USDP tidak mengikuti siklus hidup pengembangan perangkat lunak tradisional (waterfall model) melainkan mengadopsi pendekatan iterative dengan 4 fase utama. Setiap fase menggambarkan penekanan pada kegiatan yang penting pada pengembangan sistem. Fase-fase tersebut digambarkan dalam bentuk aliran kerja (workflows) dimana aliran tersebut merupakan serangkaian aktifitas.

Fase USDP Fase-fase pada USDP meliputi :

1. Inception
2. Elaboration
3. Construction
4. Transition

Fase, Workflow dan Iterasi Satu fase pengambangan terdiri atas beberapa aliran kerja (workflow). Usaha yang dilakukan seperti lama waktu pengerjaan untuk setiap aliran kerja bervariasi dari fase ke fase. Dalam satu fase bisa lebih dari satu iterasi.

Perbedaan USDP dan Siklus Hidup Waterfall Beberapa perbedaan antara USDP dan model Waterfall antara lain:

- Pada waterfall life cycle fase kegiatan dan aliran kerja terkait bersamaan sedangkan pada USDP antara fase dan aliran kerja terpisah

Pada fase requirement hanya melakukan aktivitas requirement saja sedangkan pada USDP ada serangkaian aktivitas yang dilakukan.

- Semua aktivitas requirement pada waterfall harus lengkap sebelum masuk ke fase analisis. Sedangkan USDP tidak harus. Requirements bisa dilakkukan pada setiap fase. Hanya bobot aktivitasnya yang berbeda.
- Pada sikllus hidup iterative, beberapa requirement dapat muncul pada saat analisa.

Prinsip Pokok USDP Pada prinsipnya USDP bukanlah suatu metodologi yang baru. USDP merupakan bentuk lain dari proses pengembangan sistem. Beberapa Karakteristik yang muncul dari USDP merupakan karakteristik umum dari metodologi yang sudah ada antara lain :

- Iterative
- Incremental
- Requirements Driven
- Component-based
- Architectural

Aktivitas Utama Proses pengembangan sistem USDP melibatkan aktitas utama, yaitu • Requirements capture and modeling

- Requirements Analysis
- System Design
- Class Design
- Data management design
- Construction
- Testing
- implementation masing-masing aktifitas saling berhubungan dan tergantung satu sama lain. Pada waterfall setiap kegiatan dijalankan secara berurutan (sequence). Berbeda dengan proses pengembangan iterative yang beberapa aktivitas bisa saling mendahului.

Requirements Capture and Modelling Requirement Capture and Modelling merupakan suatu proses untuk melakukan identifikasi kebutuhan perangkat lunak. Requirement di dokumentasikan dengan use cases. Model requirement dan fungsionalitasnya

Requirement Analysis Pada dasarnya, setiap use case menggambarkan satu buah user requirement utama. Setiap use case di analisa secara terpisah untuk mengidentifikasi objek yang diperlukan. Use case juga dianalisa untuk menentukan bagaimana objek tersebut berinteraksi dan bagaimana tanggung jawab setiap objek pada use case tersebut. Diagram komunikasi (Communication Diagram) digunakan untuk memodelkan interaksi dari setiap objek. Model pada setiap use case di integrasikan untuk mendapatkan analisa class diagram.

System Design Pada bagian ini dilakukan indentifikasi dan dokumentasi standar pengembangan (rancangan interface standar, standar untuk conding).

Class Design Setiap model analisa dari use case akan diuraikan secara terpisah. Diagram iterasi digunakan untuk memperlihatkan komunikasi antara objek serta state diagram digunakan untuk menunjukkan perilaku suatu objek yang kompleks. Dari kedua model ini akan di dapat rancangan class diagram. Setiap class yang dihasilkan memiliki atribut dan operai secara spesifik

User Inteface Design User interface design diperoleh dari hasil realisasi functionalitas. User interface design menampilkan bentuk interaksi dengan user seperti menentukan posisi dan warna dari tombol buton di layar

Management Design Bagian ini fokus pada mekanisme dan implementasi dari sistem manajemen database yang digunakan. Teknik database seperti normalisasi, entity relational diagram sangat bermanfaat jika menggunakan model data relasional. Antara manajemen data dan rancangan class diagram masing-masing bediri secara terpisah.

Construction Bagian construction adalah bagian untuk membangun sebuah aplikasi dengan menggunakan teknologi tertentu. Setiap bagian dari sistem bisa saja menggunakan bahasa pemrograman yang berbeda. Misalkan interface menggunakan bahasa java dan databasenya menggunakan teknologi oracle.

Testing Sebelum sistem diberikan ke client maka harus di tes teelebih dahulu. Script testing diperoleh dari deskripsi use case yang sudah disepakati oleh client. Testing merupakan elemen yang penting dan harus dilakukan.

Implementation Akhir dari tahapan implementasi adalah dilakukan installasi dari berbagai komputer client yang akan digunakan. Termasuk didalamnya manajemen transisi ndari sistem lama ke sistem yang baru. Bagian ini melibatkan manajemen resiko serta pelatihan untuk staff

1. Pemodelan (Model) Definisi model sendiri mempunyai Pengertian yang beragam sesuai dengan dunianya mulai dari pengertian sehari-hari (everyday sense) sampai technical sense. Contoh dari everyday sense, adalah artis merupakan (foto) model yang mendapat peran untuk memamerkan model-model pakaian karya disainer terkenal. Model matematik hanya salah satu jenis dari model dalam lingkup technical sense. Dalam banyak aplikasi engineering, model didefinisikan sebagai representasi dari sistem yang disederhanakan. Representasi ini pun juga bermacam-macam mulai dari yang bersifat physical, pictorial, verbal, schematic dan symbolic dimana:

a. Physical yaitu dengan membuat scaleddown version dari sistem yang dipelajari (model pesawat, model kereta api), b. Pictorial, yaitu representasi dengan gambar untuk menggambarkan kontur permukaan bumi seperti peta topografi dan bola dunia. c. Verbal, yaitu representasi suatu sistem ke dalam kalimat verbal yang mengambarkan ukuran, bentuk dan karakteristik. d. Schematic, yaitu representasi dalam bentuk skema figurative misalnya model rangkaian listrik, model Atom Bohr dan lain-lain

Symbolic, yaitu representasi ke dalam simbol-simbol matematik dimana variable hasil karakterisasi proses atau sistem ke dalam variable formulasi menggunakan simbol-simbol matematik. Jadi Pemodelan merupakan suatu proses dalam representasi abstrak suatu model. Proses pemodelan menampilkan deskripsi suatu proses dari beberapa perspektif tertentu. Proses pemodelan perangkat lunak merupakan aktifitas yang saling terkait (koheren) untuk menspesifikasikan, merancang, implementasi dan pengujian sistem perangkat lunak. (www.ilmukomputer.com, Pemodelan Data, 2005). Pada tingkat teknik, rekayasa perangkat lunak dimulai dengan serangkaian tugas pemodelan. Model analisis sebenarnya merupakan serangkaian model yang merupakan representasi teknis yang pertama dari system. Di dalam suatu industri dikenal berbagai macam proses, demikian juga Halnya dengan industri perangkat lunak (Mukhamat Djafar, 2015). Perbedaan proses yang digunakan akan menguraikan aktivitas-aktivitas proses dalam cara-cara yang berlainan. Perusahaan yang berbeda menggunakan proses yang berbeda untuk menghasilkan produk yang sama. Tipe produk yang berbeda mungkin dihasilkan oleh sebuah perusahaan dengan menggunakan proses yang berbeda. Namun beberapa proses lebih cocok dari lainnya untuk beberapa tipe aplikasi. Jika proses yang salah digunakan akan mengurangi kualitas kegunaan produk yang dikembangkan. Karena banyaknya variasi dalam model proses yang digunakan maka tidak mungkin menghasilkan gambaran-gambaran yang reliabel untuk alokasi biaya dalam aktivitas-aktivitas ini. Modifikasi perangkat lunak biasanya lebih dari 60 % dari total biaya pembuatan perangkat lunak. Presentasi ini terus bertambah karena lebih banyak perangkat lunak dihasilkan dan dipelihara. Pembuatan perangkat lunak untuk suatu perubahan adalah penting. Proses perangkat lunak komplek dan melibatkan banyak aktivitas. Proses pemodelan analisis memiliki atribut dan karakteristik seperti: a. Understandability, yaitu sejauh mana proses secara eksplisit ditentukan dan bagaimana kemudahan definisi proses

itu dimengerti. b. Visibility, apakah aktivitas-aktivitas proses mencapai titik akhir dalam hasil yang jelas sehingga kemajuan dari proses Tersebut dapat terlihat nyata/jelas. 3 c. Supportability, yaitu sejauh mana aktivitas proses dapat didukung oleh CASE d. Acceptability, apakah proses yang telah ditentukan oleh insinyur dapat diterima dan digunakan dan mampu bertanggung jawab selama pembuatan produk perangkat lunak e. Reliability, apakah proses didesain sedikian rupa sehingga kesalahan proses dapat dihindari sebelum terjadi kesalahan pada produk. Robustness, dapatkah proses terus berjalan walaupun terjadi masalah yang tak diduga. f. Maintainability, Dapatkah proses berkembang untuk mengikuti kebutuhan atau perbaikan. g. Rapidity, bagaimana kecepatan proses pengiriman sistem dapat secara lengkap memenuhi spesifikasi. Tetapi Pada saat ini ada dua landskap pemodelan analisis. Yaitu yang pertama analisis terstrutur adalah metode pemodelan klasik. Dimana analisis terstruktur ini merupakan aktifitas pembangunan model, dan yang kedua adalah analisis berorientasi Objek. 2. Kebutuhan Sistem Berorientasi Objek Untuk mengetahui kebutuhan sistem berorientasi objek, maka perlu dilakukan analisis. Analisis yang dimaksudkan sebagai berikut: Analisis adalah Penguraian suatu pokok atas berbagai bagiannya dan penelaahan bagian itu sendiri serta hubungan antar bagi an untuk memperoleh pengertian yang tepat dan pemahaman arti keseluruhan. Studi dari suatu permasalahan dengan cara memilah-milah permasalahan tersebut sehingga dapat dipahami dan dievaluasi, sebelum diambil tindakan-tindakan tertentu. Agar kebutuhan sistem dapat diketahui, perlu digunakan suatu metode dalam analisisisnya. Metodologi adalah cara sistematis untuk mengerjakan pekerjaan analisis dan desain. Dengan metodologi, pihak yang membangun suatu sistem dapat merencanakan dan mengulangi pekerjaan di lain waktu. Metodologi menghilangkan kesalahpahaman dan menghilangkan perbedaan notasi untuk suatu hal yang sama. Metode yang digunakan harus sesuai dengan kebutuhan aplikasi yang akan dibangun. Selain itu metode juga harus mudah digunakan dan dimengerti oleh pengembang perangkat lunak. Metodologi orientasi objek yang digunakan dalam analisis berorientasi objek antara lain:

a. Metode Booch 4 Dikenal dengan nama Metode Desain Object Oriented. Metode ini menjadikan proses analisis dan desain ke dalam empat tahapan yang iteratif (dapat berulang), yaitu identifikasi kelas-kelas dan objek-objek, identifikasi semantik dan hubungan objek dan kelas tersebut, perincian interface dan implementasi. b.

Metode Rumbaugh (Object Modelling Technique - OMT) Metode ini berdasarkan pada analisis terstruktur dan pemodelan entityrelationship. Tahapan utama dalam metodologi ini adalah analisis, desain sistem dan desain objek, dan implementasi. Keunggulan metode ini adalah dalam penotasian yang mendukung semua konsep object oriented. c. Metode Jacobson (Object Oriented Software Engineering - OOSE) Metode yang mengandung elemen-elemen dari Object Oriented lainnya. Metode ini memberi penekanan lebih pada use-case. OOSE memiliki tiga tahapan yaitu membuat model requirement dan analisis, desain dan implementasi, dan model pengujian (tes model). Keunggulan metode ini adalah mudah untuk dipelajari karena memiliki notasi yang sederhana, mencakup seluruh tahapan dalam rekayasa software. d. Metode Coad dan Yourdon Metode ini didasarkan pada pemodelan Object Oriented dan entityrelationship. Metode ini mempunyai perancangan yang berfokus pada empat komponen yaitu Problem domain componet, Human interaction componet, Data management component dan Task management component. e. Metode Wirfs-Brock Responsibility Driven Design/-Class Responsibility Collaboration (RDD/CFC) Metode ini diarahkan pada desain, tetapi sangat berguna untuk memunculkan ide dalam tahap analisis. Keunggulannya adalah mudah digunakan, metode ini juga mengidentifikasikan hirarki kelas dan subsistem-subsistem. f. Metode Shlair-Mellor Object Oriented Analysis/Design (OOA/D) Metode yang menggunakan teknik pemodelan informasi tradisional yang menjelaskan entitas dalam sistem, menggunakan state diagram untuk memodelkan keadaan (state) entitas, menggunakan data flow diagram untuk memodelkan alur data dalam sistem. Metode ini menghasilkan tiga jenis model yaitu: information model, state model dan process model. Keunggulan metode ini adalah dalam

memandang masalah dari sudut pandang yang berbeda, mudah dibuat (dikonversi) dari metode struktural. 3. Alur Kerja Sistem Berorientasi Objek Siklus pemodelan atau hubungan dengan langkah-langkah pemodelan dalam mengembangkan suatu sistem sebagai berikut: a. Rekayasa pemodelan sistem Yaitu menyangkut pengumpulan kebutuhan (requirement gathering) pada level sistem dengan sejumlah analisis serta top desain. b. Analisis Yaitu kebutuhan Perangkat Lunak, proses requirement gathering difokuskan, khususnya pada Perangkat lunak. Untuk memahami sifat program yang dibangun, analis harus memahami domain informasi, tingkah laku, unjuk kerja, dan interface yang diperlukan. Kebutuhan sistem maupun

Perangkat Lunak didokumentasikan dan direview bersama user c. Desain Memiliki fokus terhadap 4 hal, yaitu: a. Desain database. b. Arsitektur Perangkat Lunak. c. Arsitektur interface d. Algoritma prosedural. Proses desain menerjemahkan kebutuhan kedalam representasi Perangkat Lunak sebelum dimulai coding. 4. Unified Modeling Language (UML) Unified Modeling Language (UML) adalah notasi yang lengkap untuk membuat visualisasi model suatu sistem. Sistem informasi dan fungsi, tetapi secara normal digunakan untuk memodelkan sistem komputer (Kasman Arif, 2014). Didalam pemodelan objek guna menyajikan sistem yang berorientasi pada objek dan pada orang lain, akan sangat sulit dilakukan jika pemodelan tersebut dilakukan dalam bentuk kode bahasa pemrograman. Kesulitan yang muncul adalah timbulnya ketidakjelasan dan salah interprestasi didalam pembacaan kode pemrograman untuk pemodelan objek tersebut (Verdi Yasin, 2012) Unified Modeling Language (UML) adalah bahasa pemodelan untuk sistem atau perangkat lunak yang berparadigma berorientasi objek" (Adi Nogroho, 2010: 6). Unified Modeling Language (UML) disebut bahasa pemodelan bukan metode. 6 Bahasa pemodelan (sebagaian besar grafik) merupakan notasi dari metode yang digunakan untuk mendesain secara cepat. Bahasa pemodelan merupakan bagian terpenting dari metode. Ini merupakan bagian kunci untuk komunikasi. Pemodelan ini merupakan bahasa standar untuk digunakan dalam visualisasi, spesifikasi, pembentukan dan pendokumentasian alat – alat dari sistem perangkat lunak. a. UML sebagai bahasa pemodelan Unified Modeling Language (UML) merupakan bahasa pemodelan yang memiliki pembendaharan kata dan cara untuk mempresentasikan secara fokus pada konseptual dan fisik dari suatu sistem. Contoh untuk sistem software yang intensif membutuhkan bahasa yang menunjukkan pandangan yang berbeda dari arsitektur sistem, ini sama seperti menyusun atau mengembangkan sistem development life cycle. Dengan Unified Modeling Language (UML) akan memberitahukan kita bagaimana untuk membuat dan membaca bentuk model yang baik, tetapi Unified Modeling Language (UML) tidak dapat memberitahukan model apa yang akan dibangun dan kapan akan membangun model tersebut. b. UML sebagai bahasa untuk mengambarkan sistem UML tidak hanya merupakan rangkain simbol grafikal, cukup dengan setiap simbol pada notasi UML merupakan penerapan simantik yang baik. UML menggambarkan modael yang dapat dimengerti dan dipresentasikan kedalam model tekstual bahasa pemrograman. Contohnya kita dapat menduga suatu model dari sistem yang

berbasis web tetapi tidak secara langsung dipegang dengan mempelajari code dari sistem. c. UML sebagai bahasa untuk menspesifikasikan sistem UML membangun model yang sesuai dan lengkap. Pada faktanya UML menunjukan semua spesifikasi keputusan analisis, desain dan implementasi yang penting yang harus dibuat pada saat pengembangan dan penyebaran dari sistem software intensif. d. UML sebagai bahasa untuk pendokumentasian sistem UML menunjukan dokumentasi dari arsitektur sistem dan detail dari semuanya. Tujuan Unified Modeling Language (UML) diantaranya adalah. a. Memberikan model yang siap pakai, bahasa pemodelan visual yang ekspresif untuk mengembangkan sistem dan yang dapat saling menukar model dengan mudah dan dimengerti secara umum. 7 b. Memberikan bahasa pemodelan yang bebas dari berbagai bahasa pemrograman dan proses rekayasa. c. Menyatukan praktek – praktek terbaik yang terdapat dalam pemodelan. 5. UML dengan aplikasi StarUML Pemodelan merupakan suatu hal yang tidak bisa dilepaskan dari pembangunan aplikasi. Sebagai cikal-bakal dari suatu aplikasi, proses memodelkan tentu bukan hal yang mudah. Namun seiring berkembangnya teknologi, pemodelan yang notabene memakan banyak waktu bisa diselesaikan lebih cepat dan terorganisasi. Hal tersebut bisa terjadi dengan bantuan aplikasi pemodelan. StarUML adalah software permodelan yang mendukung UML (Unified Modeling Language). Berdasarkan pada UML version 1.4 dan dilengkapi 11 macam diagram yang berbeda, mendukung notasi UML 2.0 dan juga mendukung pendekatan MDA (Model Driven Architecture) dengan dukungan konsep UML. StarUML dapat memaksimalkan produktivitas dan kualitas dari suatu software project. Jika dikomputer Anda telah terpasang StarUML, maka silakan melanjutkan ke bahasan berikut. a. Use Case Diagram Use Case diagram merupakan suatu diagram yang menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Sebuah use case dapat memrepresentasikan interaksi antara aktor dengan sistem. Use Case Diagram adalah abstraksi dari interaksi antara system dan actor. Use case bekerja dengan cara mendeskripsikan tipe interaksi antara user sebuah system dengan systemnya sendiri melalui sebuah cerita bagaimana sebuah system dipakai. Use case merupakan kontruksi untk mendeskripsikan bagaimana system akan terlihat di mata user. Sedangkn use case diagram memfalisitasi komunikasi di antara analis dan pengguna serta analis dan client. Penjelasan bagian bagian use case diagram, ada 6 tool yang terpenting pada use case diagram : 1) System

Menyatakan batasan sistem dalam relasi dengan actor-actor yang menggunakannya (di luar sistem) dan fitur-fitur yang harus disediakan (dalam sistem). Digambarkan dengan segi empat yang membatasi semua use case dalam sistem terhadap pihak mana sistem akan berinteraksi. Sistem disertai label yang menyebutkan nama dari sistem, tapi umumnya tidak digambarkan karena tidak terlalu memberi arti tambahan pada diagram. 2) Actor Aktor adalah segala hal diluar sistem yang akan menggunakan sistem tersebut untuk melakukan sesuatu. Bisa merupakan manusia, sistem, atau device yang memiliki peranan dalam keberhasilan operasi dari sistem. Cara mudah untuk menemukan aktor adalah dengan bertanya hal-hal berikut: SIAPA yang akan menggunakan sistem? APAKAH sistem tersebut akan memberikan NILAI bagi aktor? 3) Use case Mengidentifikasi fitur kunci dari sistem. Tanpa fitur ini, sistem tidak akan memenuhi permintaan user/actor. Setiap use case mengekspresikan goal dari sistem yang harus dicapai. Diberi nama sesuai dengan goal-nya dan digambarkan dengan elips dengan nama di dalamnya. Fokus tetap pada goal bukan bagaimana mengimplementasikannya walaupun use case berimplikasi pada prosesnya nanti. Setiap use case biasanya memiliki trigger/pemicu yang menyebabkan use case memulai (misalnya,Pasien mendaftar dan membuat janji baru atau meminta untuk membatalkan atau mengubah janji yang sudah ada).ada 2 triger pertama triger eksternal, seperti pelanggan memesan atau alarm kebakaran berbunyi, kedua triger temporal, seperti tanggal pengembalian buku terlewati di perpustakaan atau keterlambatan bayar sewa. 4) Assosiation Mengidentifikasikan interaksi antara setiap actor tertentu dengan setiap use case tertentu. Digambarkan sebagai garis antara actor terhadap use case yang bersangkutan. Asosiasi bisa berarah (garis dengan anak panah) jika komunikasi satu arah, namun umumnya terjadi kedua arah (tanpa anak panah) karena selalu diperlukan demikian. 5) Dependency Dependensi <>

) Mengidentifikasi hubungan antar dua use case di mana yang satu memanggil yang lain. b) Jika pada beberapa use case terdapat bagian yang memiliki aktivitas yang sama maka bagian aktivitas tersebut biasanya dijadikan use case tersendiri dengan relasi dependensi setiap use case semula ke use case yang baru ini sehingga memudahkan pemeliharaan. c) Digambarkan dengan garis putus-putus bermata panah dengan notasi <> pada garis. d) Arah mata panah sesuai dengan arah pemanggilan. Dependensi <> a) Jika pemanggilan memerlukan adanya kondisi

tertentu maka berlaku dependensi <>. b) Note: konsep “extend” ini berbeda dengan “extend” dalam Java! c) Digambarkan serupa dengan dependensi <> kecuali arah panah berlawanan 6) Generalization Mendefinisikan relasi antara dua actor atau dua use case yang mana salah satunya meng-inherit dan menambahkan atau override sifat dari yang lainnya. Penggambaran menggunakan garis bermata panah kosong dari yang meng-inherit mengarah ke yang di-inherit Berikut ialah contoh sederhana cara membuat use case diagram dengan starUML: 1) Buka aplikasi starUML 2) Pada tampilan awal, pilih model yang terletak pada toolbar, lalu Add Diagram dan pilih Use Case Diagram

Prinsip Desain Perangkat lunak

Desain perangkat lunak berupa model dan proses. Proses desain merupakan serangkaian langkah / tahap / step iteratif yang memungkinkan desainer menggambarkan semua aspek perangkat luank yang dibangun . Sedangkan Model Desain merupakan ekivalen rencana arsitek untuk membangun suatu sistem interface. model desain memeluai dengan menyajikan totalitas dari hal-hal yang akan dibangun.

Prinsip-prinsip desain dasar untuk mengendalikan proses desain . Davis [DAV95] mengusulkan serangkaian prinsip bagi desain perangkat lunak , antara lain sebagai berikut :

- Proses desain tidak boleh menderita karena "tunnel vision"
- Desain harus dapat ditelusuri sampai pada model analisis.
- Desain tidak boleh "berulang"
- Desain harus " Meminimalkan kesenjangan intelektual"
- Dsain harus mengungkapkan keseragaman dan integritasi.
- Desain harus terstruktur untuk mengakomodasi perubahan.
- Desain harus terstruktu untuk berdegradasi dengan baik , bahkan pada saat data dan event" menyimpang , / sedang menghadapi kondisi operasional.
- Desain bukanlah pengkodean
- Desain harus dinilai kualitasnya pada saat desain dibuat , bukan setelah jadi

· Desain harus dikaji untuk meminimalkan kesalahan-kesalahan konseptual (semantik).

jika prinsip diatas sudah diterapkan dengan benar , maka RPL mampu menciptakan suatu
desain yang mengungkapkan baik itu faktor kualitas internal dan juga eksternal
Setiap model yang dikembangkan mempunyai karakteristik sendiri. Namun secara umum ada persamaannya, 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 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 dirupiah-kan. Namun efek dari penggunaan perangkat lunak yang telah dikembangkan haruslah memberi nilai tambah bagi organisasi. Hal ini dapat Setiap model yang dikembangkan mempunyai karakteristik sendiri-sendiri.

Model proses perangkat lunak masih menjadi object penelitian, tapi sekarang ada banyak model umum atau paradigma yang berbeda dari pengembangan perangkat lunak, antara lain:

A. Waterfall Model

Sebuah pendekatan pengembangan perangkat lunak sistematik dan sekuensial. Disebut juga “Classic Life Cycle”. Disebut waterfall (berarti air terjun) karena memang diagram tahapan prosesnya mirip dengan air terjun yang bertingkat, berikut diagram tahapannya :

Aktivitas Waterfall Model

- Requirements analysis and definition : mengumpulkan kebutuhan secara lengkap kemudian dianalisis dan didefinisikan kebutuhan yang hasrus dipenuhi oleh program yang akan dibangun.
- System and software design : desain dikerjakan setelah kebutuhan selesai dikumpulkan secara lengkap.
- Implementation and unit testing : desain program diterjemahkan ke dalam kode-kode dengan menggunakan bahasa pemrograman yang sudah ditentukan. Program yang dibangun langsung diuji.
- Integration and system testing : penyatuan unit-unit program kemudian diuji secara keseluruhan (system testing)
- Operation and maintenance : mengoperasikan program dilingkungannya dan melakukan pemeliharaan, seperti penyesuaian atau perubahan karena adaptasi dengan situasi sebenarnya.

Keunggulan dari waterfall:

- Software yang dikembangkan dengan metode ini biasanya menghasilkan kualitas yang baik.

- Dokumen pengembangan sistem sangat terorganisir, karena setiap fase harus terselesaikan dengan lengkap sebelum melangkah ke fase berikutnya.

Kekurangan dari waterfall:

- Perubahan sulit dilakukan karena sifatnya yang kaku.
- Karena sifat kakunya, model ini cocok ketika kebutuhan dikumpulkan secara lengkap sehingga perubahan bisa ditekan sekecil mungkin. Tapi pada kenyataannya jarang sekali konsumen/pengguna yang bisa memberikan kebutuhan secara lengkap, perubahan kebutuhan adalah sesuatu yang wajar terjadi.
- Waterfall pada umumnya digunakan untuk rekayasa sistem yang besar dimana proyek dikerjakan di beberapa tempat berbeda, dan dibagi menjadi beberapa bagian sub-proyek.

Evolutionary Software Process Models

Bersifat iteratif/ mengandung perulangan. Hasil proses berupa produk yang makin lama makin lengkap sampai versi terlengkap dihasilkan sebagai produk akhir dari proses. Dua model dalam **evolutionary software process model** adalah:

1. Incremental Model

Incremental Model merupakan gabungan antara model linear sekuensial dan prototyping. Setiap linear sekuen menghasilkan produk yang deliveriables. Increment pertama merupakan produk inti yang mengandung persyaratan/kebutuhan dasar.

2. Spiral Model

Proses digambarkan sebagai spiral. Setiap loop mewakili satu fase dari software process. Loop paling dalam berfokus pada kelayakan dari sistem, loop selanjutnya tentang definisi dari kebutuhan, loop berikutnya berkaitan dengan desain sistem dan seterusnya.

C. Transformasi Formal

Metode ini berbasiskan pada transformasi spesifikasi secara matematik melalui representasi yang berbeda untuk suatu program yang dapat dieksekusi. Trasformasi

menyatakan spesifikasi program Menggunakan pendekatan ‘Cleanroom’ untuk pengembangan Metode ini mempunyai keterbatasan dalam pemakaiannya. Keunggulannya adalah mengurangi jumlah kesalahan pada sistem sehingga penggunaan utamanya adalah pada sistem yang kritis. Hal ini menjadi efektif dari segi biaya.emakaian model pengembangan formal memerlukan tingkat kerahasian sebelum digunakan.Permasalahan dalam model pengembangan metode formal:

- Memerlukan keahlian khusus dan pelatihan untuk mengaplikasikannya
- Sulit menentukan beberapa aspek dari suatu sistem seperti user interface

D. Model Rapid Aplication Development (RAD)

Rapid Aplication Development (RAD) adalah sebuah model proses perkembangan software sekuensial linier yang menekankan siklus perkembangan yang sangat pendek. Model RAD ini merupakan sebuah adaptasi “kecepatan tinggi” dari model sekuensial linier di mana perkembangan cepat dicapai dengan menggunakan pendekatan kontruksi berbasis komponen. Jika kebutuhan dipahami dengan baik, proses RAD memungkinkan tim pengembangan menciptakan “sistem fungsional yang utuh” dalam periode waktu yang sangat pendek (kira-kira 60

Bussiness modeling

Aliran informasi di antara fungsi – fungsi bisnis dimodelkan dengan suatu cara untuk menjawab pertanyaan – pertanyaan berikut : informasi apa yang mengendalikan proses bisnis? Informasi apa yang di munculkan? Siapa yang memunculkanya? Ke mana informasi itu pergi? Siapa yang memprosesnya?

Data modeling

Aliran informasi yang didefinisikan sebagai bagian dari fase bussiness modelling disaring ke dalam serangkaian objek data yang dibutuhkan untuk menopang bisnis tersebut. Karakteristik (disebut atribut) masing–masing objek diidentifikasi dan hubungan antara objek – objek tersebut didefinisikan.

Prosess modeling

Aliran informasi yang didefinisikan di dalam fase data modeling ditransformasikan untuk mencapai aliran informasi yang perlu bagi implementasi sebuah fungsi bisnis. Gambaran pemrosesan diciptakan untuk menambah, memodifikasi, menghapus, atau mendapatkan kembali sebuah objek data.

Aplication generation

RAD mengasumsikan pemakaian teknik generasi ke empat. Selain menciptakan perangkat lunak dengan menggunakan bahasa pemrograman generasi ketiga yang konvensional, RAD lebih banyak memproses kerja untuk memkai lagi komponen program yang ada (pada saat memungkinkan) atau menciptakan komponen yang bisa dipakai lagi (bila perlu). Pada semua kasus, alat – alat bantu otomatis dipakai untuk memfasilitasi konstruksi perangkat lunak.

Testing and turnover

Karena proses RAD menekankan pada pemakaian kembali, banyak komponen program telah diuji. Hal ini mengurangi keseluruhan waktu pengujian. Tetapi komponen baru harus di uji dan semua interface harus dilatih secara penuh.

Prototyping Model

Kadang-kadang klien hanya memberikan beberapa kebutuhan umum software tanpa detil input, proses atau detil output. Di lain waktu mungkin dimana tim pembangun (developer) tidak yakin terhadap efisiensi dari algoritma yang digunakan, tingkat adaptasi terhadap sistem operasi atau rancangan form user interface. Ketika situasi seperti ini terjadi model prototyping sangat membantu proses pembangunan software.

Pengumpulan kebutuhan: developer dan klien bertemu dan menentukan tujuan umum, kebutuhan yang diketahui dan gambaran bagian-bagian yang akan dibutuhkan berikutnya. Detil kebutuhan mungkin tidak dibicarakan disini, pada awal pengumpulan kebutuhan.

- Perancangan : perancangan dilakukan cepat dan rancangan mewakili semua aspek software yang diketahui, dan rancangan ini menjadi dasar pembuatan prototype.
- Evaluasi prototype: klien mengevaluasi prototype yang dibuat dan digunakan untuk memperjelas kebutuhan software.

Refrensi :

Rekayasa Perangkat Lunak (RPL) Roger Presssman