**Nama : Muhammad Rosyid Maulana**

**NRP : 220441100019**

**Kelas : RPL-C**

**Topik : Rangkuman Chapter 2**

**SOFTWARE PROCESSES**

Proses perangkat lunak adalah serangkaian aktivitas terkait yang mengarah pada produksi produk perangkat lunak. Kegiatan ini melibatkan pengembangan perangkat lunak dari awal dalam bahasa pemrograman standar seperti Java atau C. Proses ini dapat dijelaskan dengan mengacu pada aktivitas, produk, peran, dan kondisi sebelum dan sesudahnya.

Aktivitas yang mendasari dalam rekayasa perangkat lunak melibatkan langkah-langkah berikut:

1. Spesifikasi Perangkat Lunak: Pada tahap ini, fungsionalitas perangkat lunak dan batasannya harus didefinisikan dengan jelas. Ini melibatkan identifikasi kebutuhan pelanggan dan menentukan apa yang perangkat lunak harus lakukan.
2. Perancangan dan Implementasi Perangkat Lunak: Setelah spesifikasi selesai, perangkat lunak harus dirancang dan diimplementasikan sesuai dengan spesifikasi tersebut. Ini mencakup menghasilkan kode sumber perangkat lunak dan menciptakan produk akhir yang memenuhi persyaratan.
3. Validasi Perangkat Lunak: Setelah perangkat lunak dibangun, tahap validasi harus memastikan bahwa perangkat lunak berfungsi dengan baik dan sesuai dengan harapan pelanggan. Ini melibatkan pengujian, pengujian kualitas, dan evaluasi produk perangkat lunak.
4. Evolusi Perangkat Lunak: Perangkat lunak cenderung perlu berubah seiring waktu untuk memenuhi perubahan kebutuhan pelanggan. Tahap evolusi melibatkan pemeliharaan perangkat lunak, peningkatan, dan perubahan sesuai permintaan pelanggan.

Selain aktivitas, deskripsi proses juga mencakup elemen-elemen berikut:

1. Produk: Produk merupakan hasil dari suatu proses atau aktivitas. Misalnya, dalam tahap perancangan, produknya bisa berupa model arsitektur perangkat lunak atau kode sumber yang telah dihasilkan.
2. Peran: Peran mencerminkan tanggung jawab orang-orang yang terlibat dalam proses. Contoh peran termasuk manajer proyek, manajer konfigurasi, pemrogram, dan lainnya. Setiap peran memiliki peranannya sendiri dalam menjalankan aktivitas yang diperlukan.
3. Kondisi Sebelum dan Sesudah: Pernyataan yang benar sebelum dan sesudah dilakukannya suatu aktivitas atau produksi produk. Sebelum memulai aktivitas, kondisi prasyarat harus terpenuhi. Setelah selesai, kondisi pasca mencerminkan apa yang telah dihasilkan. Misalnya, sebelum desain arsitektur dimulai, prasyaratnya mungkin adalah semua persyaratan telah disetujui oleh pelanggan; setelah aktivitas ini selesai, kondisi pasca mungkin berupa model UML yang menggambarkan arsitektur perangkat lunak telah ditinjau.

Ini adalah komponen-komponen penting dalam pemahaman proses perangkat lunak, yang membantu dalam merencanakan, mengelola, dan melaksanakan pengembangan perangkat lunak dengan sukses.

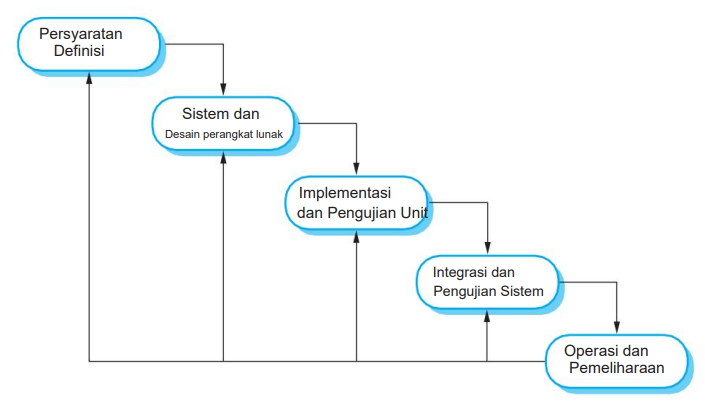
**SOFTWARE PROCESS MODELS**

Model proses perangkat lunak adalah representasi sederhana dari proses pengembangan perangkat lunak. Mereka mewakili sudut pandang tertentu dan memberikan pandangan terbatas tentang proses. Beberapa model yang dibahas meliputi:

1. Model Air Terjun (The Waterfall Model): Ini memecah proses menjadi fase-fase terpisah seperti spesifikasi, desain, implementasi, dan pengujian. Setiap fase dilakukan secara berurutan
2. Pengembangan Bertahap (Incremental Development): Pendekatan ini membangun sistem secara bertahap dengan menambahkan fungsionalitas pada setiap versi. Ini memungkinkan perkembangan yang berkelanjutan.
3. Rekayasa Perangkat Lunak Berorientasi Penggunaan Kembali (Reuse-Oriented Software Engineering): Berfokus pada penggunaan kembali komponen yang ada daripada membuat semuanya dari awal.

Kombinasi model air terjun dan pengembangan bertahap sering digunakan dalam pengembangan sistem besar. Bagian sistem yang sudah dipahami mungkin menggunakan model air terjun, sementara bagian yang lebih kompleks dikembangkan secara bertahap.

**(THE WATERFALL MODEL) MODEL AIR TERJUN**



Model air terjun adalah model proses pengembangan perangkat lunak yang pertama kali diterbitkan pada tahun 1970 oleh Royce. Model ini menggambarkan tahapan utama dalam pengembangan perangkat lunak sebagai berikut:

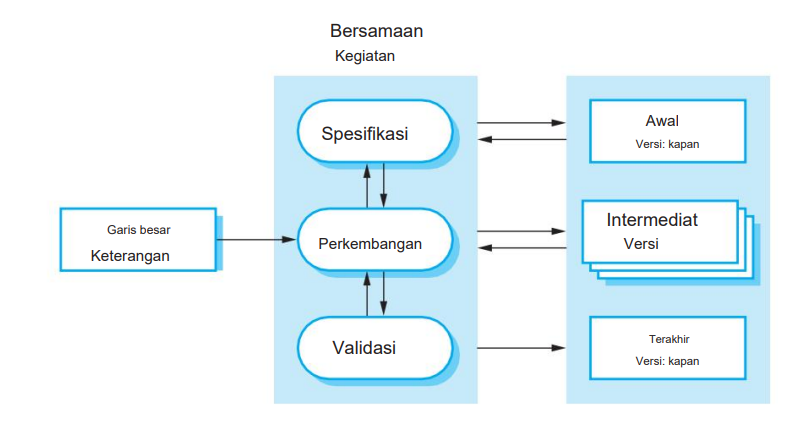
* 1. Analisis dan Definisi Persyaratan: Persyaratan sistem ditetapkan melalui konsultasi dengan pengguna, kemudian didefinisikan secara rinci.
  2. Desain Sistem dan Perangkat Lunak: Proses ini melibatkan alokasi kebutuhan perangkat keras dan perangkat lunak serta identifikasi abstraksi sistem perangkat lunak.
  3. Implementasi dan Pengujian Unit: Desain perangkat lunak direalisasikan sebagai program-program unit, yang kemudian diuji untuk memastikan memenuhi spesifikasinya.

1. Integrasi dan Pengujian Sistem: Unit-unit diintegrasikan menjadi sistem yang lengkap dan diuji untuk memenuhi persyaratan perangkat lunak.
2. Pengoperasian dan Pemeliharaan: Sistem diinstal dan digunakan, sementara pemeliharaan melibatkan perbaikan dan peningkatan.

Proses ini tidak bersifat linier, dan iterasi mungkin diperlukan. Dokumen persetujuan dibuat pada setiap tahapan, dan masalah yang belum terselesaikan dapat ditangani di tahap berikutnya. Model air terjun cocok digunakan ketika persyaratan sudah stabil.

Model air terjun memiliki keterbatasan, dan varian seperti pengembangan sistem formal cocok untuk sistem dengan persyaratan ketat dalam hal keselamatan, keandalan, atau keamanan. Namun, pendekatan ini umumnya hanya digunakan dalam proyek yang memerlukan spesialisasi khusus.

**INCREMENTAL DEVELOPMENT**



Pengembangan perangkat lunak berbasis pendekatan tangkas lebih sesuai untuk sebagian besar sistem bisnis, e-niaga, dan pribadi daripada model air terjun. Pendekatan ini memungkinkan pengembangan yang bertahap, refleksi cara kita menyelesaikan masalah dengan langkah-langkah berurutan dan fleksibilitas dalam melakukan perubahan saat ditemukan kesalahan. Dengan pengembangan bertahap, perubahan dapat dilakukan lebih ekonomis dan mudah selama proses pengembangan.

Keuntungan pengembangan bertahap dibandingkan dengan model air terjun mencakup:

1. Biaya Perubahan yang Lebih Rendah: Kurangnya analisis dan dokumentasi ulang mengurangi biaya untuk mengakomodasi perubahan kebutuhan pelanggan.
2. Masukan Pelanggan yang Lebih Baik: Pelanggan dapat memberikan masukan pada perangkat lunak yang sudah ada dengan melihat demonstrasi, lebih baik daripada hanya melihat dokumen desain.
3. Pengiriman Cepat: Pengiriman perangkat lunak yang bermanfaat kepada pelanggan dapat dilakukan lebih cepat, bahkan jika fungsionalitas belum sepenuhnya terimplementasi. Ini memungkinkan pelanggan mendapatkan nilai lebih awal.

Pendekatan ini sangat umum dalam pengembangan sistem aplikasi dan bisa berbentuk berdasarkan rencana atau pendekatan agile, atau campuran dari keduanya. Dalam pengembangan berbasis rencana, peningkatan sistem diidentifikasi sebelumnya, sedangkan dalam pendekatan agile, prioritas berdasarkan kemajuan dan kebutuhan pelanggan.

Namun, dari sudut pandang manajemen, pengembangan bertahap memiliki dua masalah:

1. Proses yang Tidak Terlihat: Manajer memerlukan hasil teratur untuk mengukur kemajuan, dan dalam pengembangan cepat, menghasilkan dokumen untuk setiap versi sistem bisa tidak efisien.
2. Penurunan Struktur Sistem: Sistem cenderung kehilangan strukturnya saat penambahan fungsionalitas baru. Untuk menghindari hal ini, perlu waktu dan biaya untuk pemfaktoran ulang perangkat lunak.

Masalah pengembangan bertahap menjadi lebih kompleks dalam sistem besar, kompleks, dan berumur panjang, di mana berbagai tim bekerja pada bagian-bagian sistem yang berbeda. Sistem besar memerlukan kerangka kerja yang stabil dan definisi jelas mengenai tanggung jawab tim yang berbeda dalam konteks arsitektur yang ada. Ini harus direncanakan terlebih dahulu, bukan dikembangkan secara bertahap.

**PENGEMBANGAN SOFTWARE REUSED – ORIENTED**

Reused – Oriented merupakan proses pengembangan software yang menggunakan kembali kode atau design yang sudah pernah dibuat. Reused -Oriented sudah banyak yang menggunakan pada abad ke-21. Pendekatan berorientasi penggunaan kembali bergantung pada basis besar komponen perangkat lunak yang dapat digunakan kembali dan kerangka kerja yang mengintegrasikan komposisi komponen-komponen ini.

1. Analisis komponen Mengingat spesifikasi kebutuhan, dilakukan pencarian komponen untuk mengimplementasikan spesifikasi tersebut. Biasanya, tidak ada yang sama persis dan komponen yang mungkin digunakan hanya menyediakan beberapa fungsi yang dibutuhkan.
2. Modifikasi persyaratan Pada tahap ini, persyaratan dianalisis menggunakan informasi tentang komponen yang telah ditemukan. Mereka kemudian dimodifikasi untuk mencerminkan komponen yang tersedia. Jika modifikasi tidak memungkinkan, aktivitas analisis komponen dapat dilakukan kembali untuk mencari solusi alternatif.
3. Desain sistem dengan penggunaan kembali Pada fase ini, kerangka sistem dirancang atau kerangka kerja yang sudah ada digunakan kembali. Para desainer memperhitungkan komponen-komponen yang digunakan kembali dan mengatur kerangka kerja untuk memenuhi hal ini. Beberapa perangkat
4. Pengembangan dan integrasi Perangkat lunak yang tidak dapat diperoleh dari luar dikembangkan, dan komponen serta sistem COTS diintegrasikan untuk menciptakan sistem baru. Integrasi sistem, dalam model ini, mungkin merupakan bagian dari proses pengembangan dan bukan aktivitas yang terpisah.

Ada tiga jenis komponen perangkat lunak yang dapat digunakan secara reused – oriented process:

1. Layanan web yang dikembangkan sesuai dengan standar layanan dan tersedia untuk pemanggilan jarak jauh.
2. Kumpulan objek yang dikembangkan sebagai suatu paket untuk diintegrasikan dengan a kerangka komponen seperti .NET atau J2EE.
3. Sistem perangkat lunak yang berdiri sendiri yang dikonfigurasi untuk digunakan pada lingkungan tertentu.

Rekayasa perangkat lunak yang berorientasi penggunaan kembali mempunyai keuntungan nyata dalam mengurangi jumlah perangkat lunak yang harus dikembangkan sehingga mengurangi biaya dan risiko.

**ACTIVITY PROCESS**

Activity Process adalah rangkaian aktivitas teknis, kolaboratif, dan manajerial yang disisipkan dengan tujuan keseluruhan untuk menentukan, merancang, mengimplementasikan, dan menguji sistem perangkat lunak. Empat kegiatan proses dasar spesifikasi, pengembangan, validasi, dan evolusi diatur secara berbeda dalam proses pengembangan yang berbeda.

**SPESIFIKASI PERANGKAT LUNAK**

Spesifikasi perangkat lunak atau rekayasa persyaratan adalah proses memahami dan mendefinisikan layanan apa yang diperlukan dari sistem dan mengidentifikasi kendala pada pengoperasian dan pengembangan sistem. Proses rekayasa persyaratan bertujuan untuk menghasilkan dokumen persyaratan yang disepakati yang menentukan sistem yang memenuhi persyaratan pemangku kepentingan.

Ada empat aktivitas utama dalam proses rekayasa persyaratan:

1. Studi kelayakan Sebuah perkiraan dibuat mengenai apakah kebutuhan pengguna yang teridentifikasi dapat dipenuhi dengan menggunakan teknologi perangkat lunak dan perangkat keras saat ini. Studi ini mempertimbangkan apakah sistem yang diusulkan akan hemat biaya dari sudut pandang bisnis dan apakah sistem tersebut dapat dikembangkan sesuai dengan keterbatasan anggaran yang ada. Studi kelayakan harus relatif murah dan cepat. Hasilnya harus menjadi dasar pengambilan keputusan apakah akan melanjutkan analisis yang lebih rinci atau tidak.
2. Pemerolehan dan analisis persyaratan Ini adalah proses memperoleh persyaratan sistem melalui pengamatan terhadap sistem yang ada, diskusi dengan calon pengguna dan penyedia, analisis tugas, dan sebagainya. Hal ini mungkin melibatkan pengembangan satu atau lebih model dan prototipe sistem. Ini membantu Anda memahami sistem yang akan ditentukan.
3. Spesifikasi persyaratan Spesifikasi persyaratan adalah kegiatan menerjemahkan informasi yang dikumpulkan selama kegiatan analisis ke dalam suatu dokumen yang mendefinisikan serangkaian persyaratan. Dua jenis persyaratan dapat dimasukkan dalam dokumen ini. Persyaratan pengguna adalah pernyataan abstrak dari persyaratan sistem untuk pelanggan dan pengguna akhir sistem; persyaratan sistem adalah penjelasan lebih rinci tentang fungsionalitas yang akan disediakan.
4. Validasi persyaratan Kegiatan ini memeriksa persyaratan realisme, konsistensi, dan kelengkapan. Selama proses ini, kesalahan dalam dokumen persyaratan pasti ditemukan. Kemudian harus dimodifikasi untuk memperbaiki masalah ini.

Analisis persyaratan berlanjut selama definisi dan spesifikasi dan persyaratan baru terungkap selama proses berlangsung. Oleh karena itu, kegiatan analisis, definisi, dan spesifikasi disisipkan. Dalam metode agile, seperti pemrograman ekstrim, persyaratan dikembangkan secara bertahap sesuai dengan prioritas pengguna dan perolehan persyaratan berasal dari pengguna yang merupakan bagian dari tim pengembangan.

**DESIGN DAN IMPLEMENTASI PERANGKAT LUNAK**

Tahap implementasi pengembangan perangkat lunak adalah proses mengubah spesifikasi sistem menjadi sistem yang dapat dieksekusi. Desain perangkat lunak melibatkan deskripsi struktur perangkat lunak, model data, antarmuka antar komponen, dan mungkin algoritma. Perangkat lunak sering berinteraksi dengan sistem lain, membentuk platform perangkat lunak di mana perangkat lunak dijalankan.

Empat aktivitas yang mungkin menjadi bagian dari proses desain sistem informasi adalah:

1. Desain Arsitektur: Identifikasi struktur keseluruhan sistem, komponen utama, hubungan, dan distribusinya.
2. Desain Antarmuka: Menentukan antarmuka antar komponen sistem dengan spesifikasi yang jelas. Antarmuka presisi memungkinkan pengembangan bersamaan.
3. Desain Komponen: Merancang cara pengoperasian setiap komponen sistem, bisa berupa pernyataan fungsionalitas atau model desain terperinci.
4. Desain Basis Data: Merancang struktur data sistem dan representasinya dalam basis data, tergantung pada penggunaan ulang database yang ada atau pembuatan database baru.

Metode terstruktur untuk desain dikembangkan pada tahun 1970an dan 1980an dan menjadi dasar bagi UML dan desain berorientasi objek. Pengembangan berbasis model (MDD) adalah evolusi dari metode terstruktur, di mana model perangkat lunak dibuat pada tingkat abstraksi yang berbeda.

Pengembangan program mengikuti proses desain sistem, dengan beberapa program biasanya dirancang secara rinci sebelum implementasi dimulai, terutama pada sistem kritis terhadap keselamatan. Pemrograman adalah aktivitas pribadi dan tidak ada proses umum yang diikuti. Debugging melibatkan membuat hipotesis tentang perilaku program, menguji hipotesis ini untuk menemukan kesalahan. Selama debug, mungkin diperlukan penelusuran kode program secara manual.

**SOFTWARE VALIDATION**

Software Validation, atau secara lebih umum, verifikasi dan validasi (V&V), bertujuan untuk menunjukkan bahwa suatu sistem sesuai dengan spesifikasinya dan memenuhi harapan pelanggan sistem.

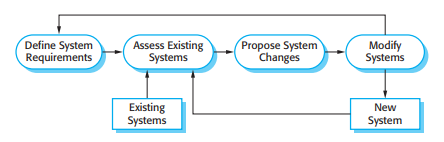
Ada tiga tahapan pengujian yang perlu dipahami dalam pengembangan perangkat lunak:

1. Pengujian Pengembangan: Pada tahap ini, komponen-komponen yang membentuk sistem diuji oleh orang yang mengembangkan sistem. Setiap komponen diuji secara independen, tanpa melibatkan komponen sistem lain. Fokus utamanya adalah memastikan bahwa setiap komponen berfungsi dengan baik dan sesuai dengan spesifikasinya.
2. Pengujian Sistem: Tahap berikutnya adalah pengujian sistem, di mana komponen sistem diintegrasikan untuk membentuk sistem lengkap. Tujuannya adalah menemukan kesalahan yang muncul akibat interaksi antar komponen dan masalah antarmuka. Pengujian sistem juga bertujuan untuk memeriksa apakah sistem memenuhi persyaratan fungsional dan non-fungsionalnya, serta menguji properti sistem yang muncul. Ini mencakup uji integrasi, uji regresi, dan uji stres, di antara lain.
3. Pengujian Penerimaan: Tahap terakhir dalam proses pengujian sebelum sistem diterima untuk digunakan secara operasional adalah pengujian penerimaan. Pada tahap ini, sistem diuji dengan data yang diberikan oleh pelanggan sistem daripada dengan data uji yang disimulasikan. Pengujian penerimaan memungkinkan pengguna akhir untuk menguji sistem dengan data nyata, yang dapat mengungkapkan kesalahan dan kelalaian dalam definisi persyaratan sistem. Hal ini juga membantu mengidentifikasi masalah persyaratan di mana fasilitas sistem tidak memenuhi kebutuhan pengguna atau kinerja sistem tidak dapat diterima.

Secara keseluruhan, pengujian pengembangan fokus pada komponen, pengujian sistem fokus pada integrasi dan persyaratan, sementara pengujian penerimaan adalah tahap akhir yang melibatkan pengguna akhir dan data nyata untuk memastikan kesiapan sistem sebelum digunakan secara operasional.

**SOFTWARE EVOLUTION**

Evolusi perangkat lunak adalah proses berkelanjutan di mana perangkat lunak terus mengalami perubahan sebagai tanggapan terhadap perubahan kebutuhan dan kebutuhan pelanggan. Fleksibilitas perangkat lunak memungkinkan perubahan yang lebih mudah dan lebih ekonomis dibandingkan dengan perubahan pada perangkat keras sistem. Oleh karena itu, penting untuk memandang perangkat lunak sebagai entitas yang terus berkembang sepanjang masa hidupnya, daripada memisahkannya menjadi dua proses terpisah.



**MENGHADAPI PERUBAHAN**

Persyaratan sistem berubah ketika bisnis yang memperoleh sistem merespons tekanan eksternal dan prioritas manajemen berubah. Saat teknologi baru menjadi tersedia, kemungkinan desain dan implementasi yang baru muncul. Oleh karena itu, tidak peduli model proses perangkat lunak apa yang digunakan, sangat penting bahwa model tersebut dapat mengakomodasi perubahan dalam perangkat lunak yang sedang dikembangkan.

Perubahan dalam proyek perangkat lunak dapat menambah biaya karena seringkali memerlukan pekerjaan ulang. Dua pendekatan utama untuk mengurangi biaya rework adalah penghindaran perubahan dan toleransi perubahan.

1. Penghindaran Perubahan: Ini melibatkan aktivitas yang dapat menduga perubahan sebelum rework signifikan diperlukan. Misalnya, pengembangan prototipe sistem memungkinkan pengguna untuk eksperimen dengan sistem sebelum pengiriman, sehingga persyaratan dapat disempurnakan sebelum biaya produksi perangkat lunak yang tinggi dikeluarkan.
2. Toleransi Perubahan: Dalam proses ini, perubahan dapat diakomodasi dengan biaya yang relatif rendah. Ini sering melibatkan pengembangan bertahap, di mana perubahan dapat diimplementasikan dalam tahap yang belum dikembangkan atau hanya memengaruhi tahap kecil dari sistem.

Selain itu, dua metode yang mendukung menghadapi perubahan adalah:

1. Prototipe Sistem: Dalam metode ini, versi awal dari sistem atau sebagian sistem dikembangkan dengan cepat untuk menguji persyaratan pelanggan dan desain. Ini memungkinkan pelanggan untuk bereksperimen dan menyempurnakan persyaratan mereka sebelum pengiriman, mengurangi jumlah perubahan setelahnya.
2. Pengiriman Bertahap: Dalam pendekatan ini, sistem disampaikan kepada pelanggan dalam tahap-tahap untuk mendapatkan komentar dan eksperimen. Ini mendukung baik penghindaran perubahan maupun toleransi perubahan, menghindari komitmen dini terhadap persyaratan seluruh sistem.
3. Penting juga untuk mempertimbangkan refactoring, yang melibatkan peningkatan struktur dan organisasi program, sebagai mekanisme penting untuk mendukung toleransi perubahan. Semua ini memungkinkan proyek perangkat lunak untuk lebih fleksibel dalam menghadapi perubahan yang tak terhindarkan.

**PROTOTYPING**

Prototipe perangkat lunak adalah versi awal dari sistem perangkat lunak yang digunakan untuk mendemonstrasikan konsep, mencoba opsi desain, dan memahami lebih banyak tentang masalah dan solusi yang mungkin. Pengembangan prototipe dengan cepat dan secara iteratif penting agar biaya terkendali, dan pemangku kepentingan sistem dapat bereksperimen dengan prototipe pada awal proses perangkat lunak.

Selain itu, ada pendekatan yang mengintegrasikan pengembangan prototipe dalam proses perangkat lunak:

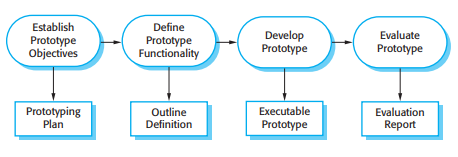
1. Penghindaran perubahan: Prototipe membantu dalam proses rekayasa persyaratan dengan menggali dan memvalidasi persyaratan sistem.
2. Desain sistem: Prototipe digunakan untuk menjelajahi desain perangkat lunak tertentu dan mendukung desain antarmuka pengguna.

Penting untuk menentukan tujuan prototipe dari awal proses. Prototipe tidak dapat memenuhi semua tujuan, sehingga perlu menjelaskan tujuan yang spesifik. Selain itu, tahap akhir adalah evaluasi prototipe, di mana pengguna harus diberikan pelatihan dan tujuan prototipe digunakan untuk membuat rencana evaluasi.

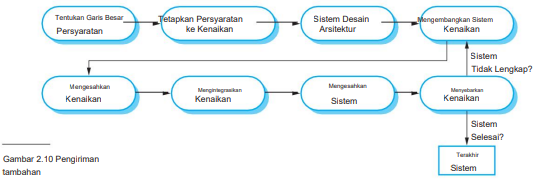
Ada masalah umum dalam penggunaan prototipe, seperti ketidakcocokan pengguna prototipe dengan penggunaan sistem akhir, kurangnya dokumentasi, degradasi struktur sistem, dan relaksasi standar kualitas organisasi.

Prototipe tidak selalu harus berupa perangkat lunak yang dapat dieksekusi. Mock-up berbasis kertas dari antarmuka pengguna sistem dapat efektif dalam membantu pengguna memperbaiki desain antarmuka dan menjalankan skenario penggunaan. Pendekatan lain adalah "Wizard of Oz prototype," di mana hanya antarmuka pengguna dikembangkan, dan respons pengguna disimulasikan oleh orang yang menginterpretasikan input pengguna.

Berikut adalah proses pengembangan prototipe :



**PENGIRIMAN TAMBAHAN**



Pengiriman tambahan (Gambar 2.10) adalah pendekatan pengembangan perangkat lunak di mana beberapa peningkatan yang dikembangkan dikirimkan ke pelanggan dan diterapkan untuk digunakan dalam lingkungan operasional.

* **Pengiriman tambahan memiliki sejumlah keunggulan:**

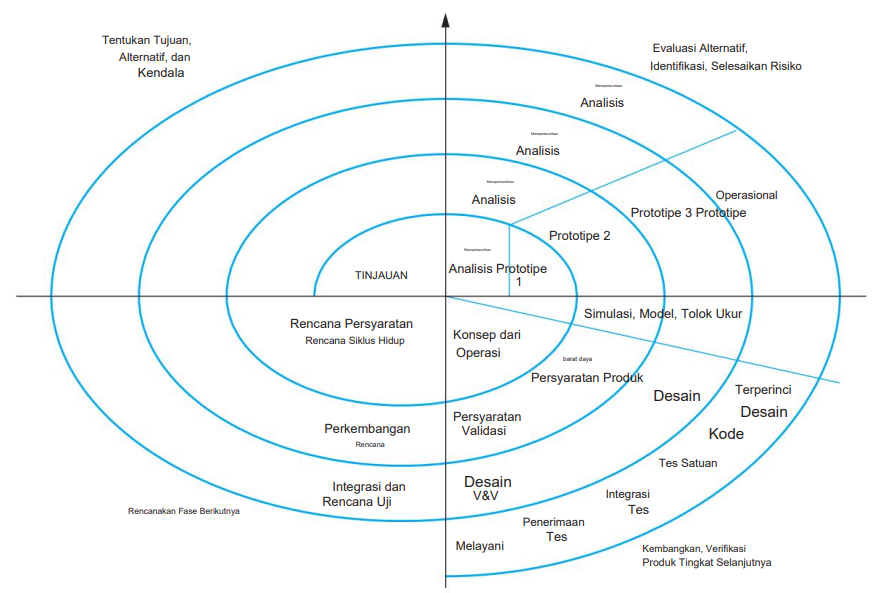
1. Pelanggan dapat menggunakan peningkatan awal sebagai prototipe dan mendapatkan pengalaman yang menginformasikan kebutuhan mereka untuk peningkatan sistem selanjutnya. Tidak seperti prototipe, ini adalah bagian dari sistem nyata sehingga tidak ada pembelajaran ulang ketika sistem lengkap sudah tersedia.
2. Pelanggan tidak perlu menunggu sampai keseluruhan sistem selesai sebelum mereka dapat memperoleh manfaat darinya. Peningkatan pertama memenuhi persyaratan paling penting sehingga mereka dapat segera menggunakan perangkat lunak tersebut.
3. Proses tersebut mempertahankan manfaat pembangunan bertahap sebagaimana mestinya relatif mudah untuk memasukkan perubahan ke dalam sistem.

* **Namun, ada masalah dengan pengiriman tambahan:**

1. Sebagian besar sistem memerlukan seperangkat fasilitas dasar yang digunakan oleh berbagai bagian sistem. Karena persyaratan tidak didefinisikan secara rinci sampai suatu penambahan akan dilaksanakan, akan sulit untuk mengidentifikasi fasilitas umum yang dibutuhkan oleh semua penambahan tersebut.
2. Inti dari proses berulang adalah spesifikasi dikembangkan bersamaan dengan perangkat lunak. Namun, hal ini bertentangan dengan model pengadaan di banyak organisasi, di mana spesifikasi sistem yang lengkap merupakan bagian dari kontrak pengembangan sistem. Dalam pendekatan inkremental, tidak ada spesifikasi sistem yang lengkap sampai kenaikan akhir ditentukan. Hal ini memerlukan bentuk kontrak baru, yang mungkin sulit diakomodasi oleh pelanggan besar seperti lembaga pemerintah.

**MODEL SPIRAL BOEHM**

Kerangka proses perangkat lunak berbasis risiko (model spiral) diusulkan oleh Boehm (1988). Di sini, proses perangkat lunak direpresentasikan sebagai suatu spiral, bukan suatu rangkaian aktivitas dengan beberapa kemunduran dari satu aktivitas ke aktivitas lainnya. Setiap loop dalam spiral mewakili fase proses perangkat lunak. Dengan demikian, perulangan terdalam mungkin berkaitan dengan kelayakan sistem, perulangan berikutnya dengan definisi persyaratan, perulangan berikutnya dengan desain sistem, dan seterusnya.



Setiap putaran dalam spiral dibagi menjadi empat sektor:

1. Penetapan tujuan Tujuan khusus untuk fase proyek tersebut telah ditentukan.
2. Penilaian dan pengurangan risiko Untuk setiap risiko proyek yang teridentifikasi, analisis terperinci dilakukan. Langkah-langkah diambil untuk mengurangi risiko.
3. Pengembangan dan validasi Setelah evaluasi risiko, model pengembangan sistem dipilih. Jika risiko keselamatan menjadi pertimbangan utama, pembangunan berdasarkan transformasi formal mungkin merupakan proses yang paling tepat, dan seterusnya. Jika risiko utama yang teridentifikasi adalah integrasi sub-sistem, model air terjun mungkin merupakan model pengembangan terbaik untuk digunakan.
4. Perencanaan Proyek ditinjau dan diambil keputusan apakah akan melanjutkan putaran spiral selanjutnya.

Setelah risiko dinilai, beberapa pengembangan dilakukan, diikuti dengan aktivitas perencanaan untuk tahap proses berikutnya. Secara informal, risiko berarti sesuatu yang bisa saja salah. Misalnya, jika tujuannya adalah menggunakan bahasa pemrograman baru, risikonya adalah kompiler yang tersedia tidak dapat diandalkan atau tidak menghasilkan kode objek yang cukup efisien. Risiko menyebabkan usulan perubahan perangkat lunak dan masalah proyek seperti pembengkakan jadwal dan biaya, sehingga minimalisasi risiko adalah aktivitas manajemen proyek yang sangat penting.

**RATIONAL UNIFIED PROCESS (RUP)**

Rational Unified Process (RUP) adalah teknik pengembangan aplikasi perangkat lunak dengan banyak alat untuk membantu dalam pengkodean produk akhir dan tugas yang terkait dengan tujuan ini.

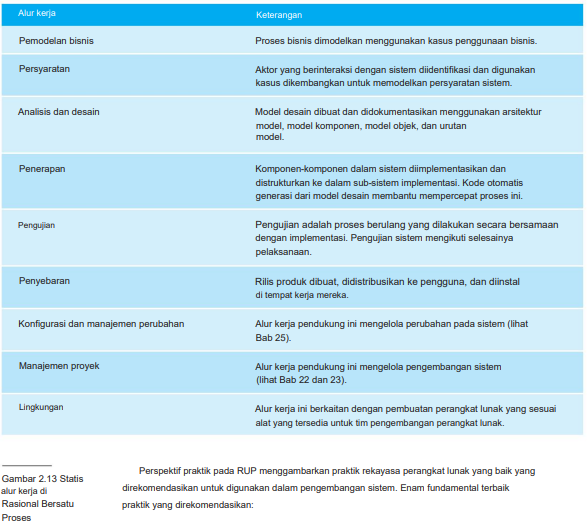
RUP adalah pendekatan berorientasi objek yang digunakan untuk memastikan manajemen proyek yang efektif dan produksi perangkat lunak berkualitas tinggi. RUP mengakui bahwa model proses konvensional menyajikan pandangan tunggal proses. Sebaliknya, RUP biasanya digambarkan dari tiga perspektif:

1. Perspektif dinamis, yang menunjukkan fase model dari waktu ke waktu.
2. Perspektif statis, yang menunjukkan proses kegiatan yang dilakukan.
3. Perspektif praktik, yang menyarankan praktik-praktik baik untuk digunakan selama proses berlangsung.

RUP adalah model bertahap yang mengidentifikasi empat fase terpisah dalam proses perangkat lunak. Namun, berbeda dengan model air terjun yang fase-fasenya disamakan dengan aktivitas proses, fase-fase dalam RUP lebih berkaitan erat dengan bisnis dibandingkan masalah teknis. Gambar 2.11 menunjukkan tahapan-tahapan dalam RUP. Ini adalah:

1. Inception Tujuan dari fase inception adalah untuk membangun kasus bisnis untuk sistem
2. Elaborasi Tujuan dari fase elaborasi adalah untuk mengembangkan pemahaman tentang domain masalah, menetapkan kerangka arsitektur untuk sistem, mengembangkan rencana proyek, dan mengidentifikasi risiko utama proyek.
3. Konstruksi Tahap konstruksi melibatkan perancangan sistem, pemrograman, dan pengujian. Bagian dari sistem dikembangkan secara paralel dan terintegrasi selama fase ini.
4. Transisi Fase terakhir dari RUP berkaitan dengan perpindahan sistem dari komunitas pengembangan ke komunitas pengguna dan membuatnya berfungsi dalam lingkungan nyata

Keuntungan dalam menyajikan tampilan dinamis dan statis adalah bahwa tahapan proses pengembangan tidak terkait dengan alur kerja tertentu. Setidaknya pada prinsipnya, semua alur kerja RUP mungkin aktif di semua tahapan proses. Pada fase awal proses, sebagian besar upaya mungkin akan dihabiskan pada alur kerja seperti pemodelan bisnis dan persyaratan, dan pada fase selanjutnya, pada pengujian dan penerapan.



Perspektif praktik pada RUP menggambarkan praktik rekayasa perangkat lunak yang baik yang direkomendasikan untuk digunakan dalam pengembangan sistem. Enam fundamental terbaik praktik yang direkomendasikan:

1. Mengembangkan perangkat lunak secara berulang Merencanakan peningkatan sistem berdasarkan pelanggan prioritas dan mengembangkan fitur sistem dengan prioritas tertinggi di awal proses pengembangan.
2. Kelola persyaratan Dokumentasikan secara eksplisit persyaratan pelanggan dan melacak perubahan pada persyaratan ini. Menganalisis dampak perubahan pada sistem sebelum menerimanya.
3. Gunakan arsitektur berbasis komponen Susun arsitektur sistem menjadi koponents, seperti yang dibahas sebelumnya dalam bab ini.
4. Perangkat lunak model visual Gunakan model UML grafis untuk menyajikan statis dan pandangan dinamis dari perangkat lunak.
5. Verifikasi kualitas perangkat lunak Pastikan perangkat lunak memenuhi kualitas organisasi standar.
6. Mengontrol perubahan pada perangkat lunak Mengelola perubahan pada perangkat lunak menggunakan sistem manajemen perubahan dan prosedur serta alat manajemen konfigurasi.