PERTEMUAN KE-16
PEMELIHARAAN PERANGKAT LUNAK

16.1 TUJUAN PEMBELAJARAN

Adapun tujuan pembelajaran yang akan dicapai sebagai berikut:

16.1. Pemeliharaan Perangkat Lunak

16.2. Karakteristik pemeliharaan Perangkat Lunak

16.3. Perekayasaan kembali

16.4. Penggunaan kembali Perangkat Lunak

16.2 URAIAN MATERI

Tujuan Pembelajaran 16.1:
Pemeliharaan Perangkat Lunak

Pemeliharaan Perangkat Lunak adalah proses umum pengubahan/pengembangan perangkat lunak setelah diserahkan ke konsumen. Perubahan mungkin berupa perubahan sederhana untuk membetulkan error koding atau perubahan yg lebih ekstensif untuk membetulkan error perancangan/perbaikan signifikan untuk membetulkan error spesifikasi/akomodasi persyaratan baru.

Ada 4 katagotri pemeliharaan software yaitu :

1. Corrective Maintenance, perubahan yang dilakukan guna memperbaiki kesalahan.
2. Adaptive Maintenance, perawatan berdasarkan perubahan lingkungan.
3. Perfective Maintenance, perubahan untuk meningkatkan kualitas sistem tanpa merubah fungsinya.
4. Preventive Maintenance, Meningkatkan reliability, future maintainability, future enhancement (reverse engineering dan re-engineering)

Kenapa biaya pemeliharaan lebih tinggi dari pada biaya pengembangan, berikut adalah beberapa faktor yang menyebabkannya:

1. Stabilitas Tim, biasanya tim pengembang dan tim pemelihara adalah orang yang berbeda karena tim pengembang biasanya sudah lari ke proyek baru sehingga tim pemeliharanya tidak begitu paham atas sistem yang dikembangkan.
2. Tanggung Jawab Kontrak, kontrak bagi pengembang dan pemelihara kebanyakan terpisah atau diberikan kepada perusahaan yang berbeda dan bahkan bukan pengembang sistem aslinya, akibatnya tidak ada insentif bagi pengembang untuk membuat sistem yang mudah untuk diubah.
3. Keahlian Staff, staff pemelihara kebanyakan tidak berpengalaman dalam hal pemeliharaan software dan staff pemelihara sering diaangap tidak memerlukan keahlian yang mendalam di bidang software.
4. Umur dan Struktur Program, program yang sudah tua biasanya strukturnya sudah terdegradasi oleh perkembangan jaman sehingga sangat sulih dipahami oleh pemelihara.

Beberapa permasalahan yang sering muncul dalam pemeliharaan software:

a. Kesulitan melakukan pelacakan evolusi software pd versi sebelumnya,
b. Kesulitan pelacakan pada proses pengembangan software,
c. Sulit untuk mengerti program orang lain,
d. Tidak adanya dokumentasi yang baik,
e. Tidak adanya nara sumber,
f. Kebanyakan software dirancang tanpa adanya pemikiran untuk diubah.

Tujuan Pembelajaran 16.2:

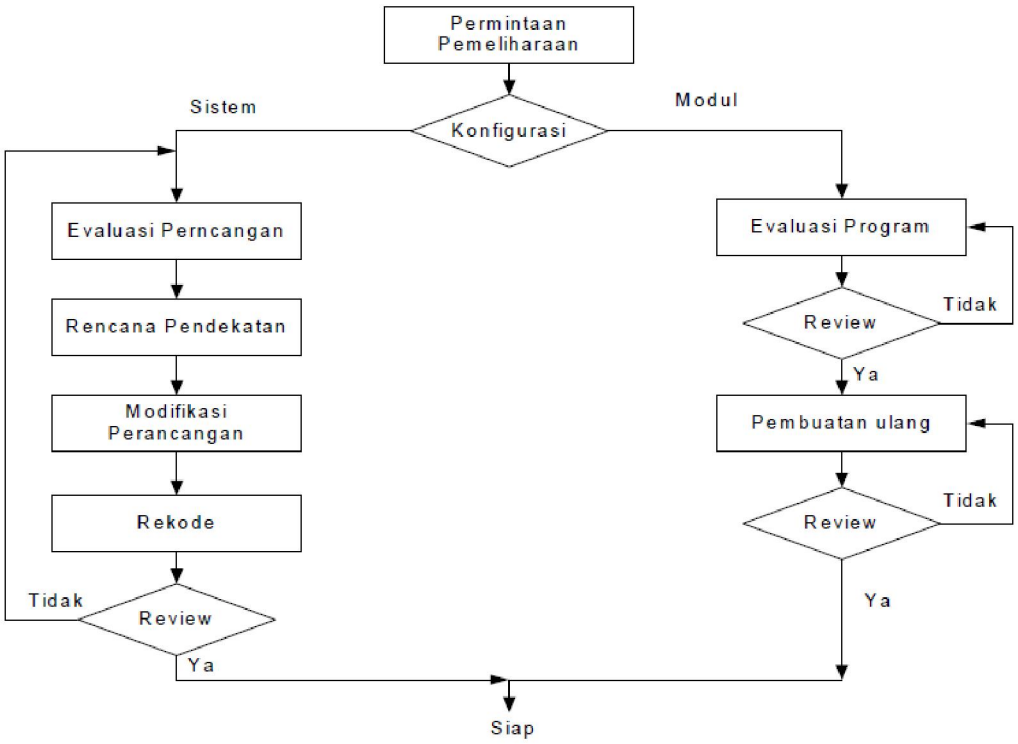
Karakteristik pemeliharaan Perangkat Lunak

Karakteristik pemeliharaan dapat dibedakan menjadi 2 yaitu:

1. Pemeliharaan terstruktur dan tidak terstruktur
2. Biaya pemeliharaan.

A. Pemeliharaan terstruktur dan tidak terstruktur

a. Pemeliharaan terstruktur



Pada gambar di atas terlihat pemeliharaan terstruktur dimulai dari permintaan akan pemeliharaan dan menentukan konfigurasi dari perangkat lunak yang akan diadakan pemeliharaan. Jika merupakan seluruh perangkat lunak maka tindakan yang diambil adalah evaluasi perancangan dan menentukan rencana pendekatan yang akan digunakan untuk melakukan pemeliharaan. Kemudian dilanjutkan dengan melaku-kan modifikasi+perancangan dan penulisan ulang program (rekode). Langkah terakhir adalah me-review program yang telah ditulis. Jika diterima maka berarti tugas pemeliharaan telah selesai. Sedangkan jika konfigurasi merupakan program per modul maka kegiatan yang dilakukan adalah evaluasi program. Jika diperlukan modifikasi yang cukup besar maka tindakan yang diambil adalah pembuatan ulang yang dilanjutkan dengan review hasil. Jika hasil akhir memenuhi kriteria maka berarti perangkat lunak siap.

b. Pemeliharaan tidak terstruktur

- Tidak mempunyai dokumentasi yang baik
- Tidak menggunakan metodologi perancangan
- Tidak mengikuti langkah-langkah di atas

B. Biaya pemeliharaan

Biaya pemeliharaan perangkat lunak yang dikeluarkan dalam pase pemeliharaan meningkat dengan cepat. Selain biaya yang umum dalam pase pengembangan Bering timbul biaya-biaya tak berwujud (intangible cost). Biaya-biaya ini ditimbulkan karena:

- Ketidakpuasan pemakai (user) akibat tidak selesainya perangkat lunak sesuai dengan waktu yang telah ditentukan pada pase pemeliharaan
- Pengurangan kualitas perangkat lunak
- Penambahan tenaga kerja baru

Tujuan Pembelajaran 16.3:

Perekayasaan kembali

Rekayasa Ulang Sistem atau Reengineering adalah suatu proses yang kerap disamakan dengan me-maintenance sistem (dalam hal ini yang kita bicarakan adalah sistem berbasis komputer). Menurut Pressman Reengineering adalah proses membangun kembali sistem dimana produk yang dihasilkan diharapkan bertambah fungsionalitasnya, semakin baik performa & keandalannya, serta meningkatkan kemampuan maintainability-nya.

Menurut Sommerville menyatakan bahwa kenapa suatu sistem harus di-reengineering, alasanya diantaranya :

1. Requirement baru terbentuk selaman sistem tersebut digunakan
2. lingkungan bisnis sistem tersebut berubah
3. terdapat eror yang harus segera diperbaiki
4. ada komputer atau hardware baru yang ditambahkan ke sistem
5. Performa dan keandalan sistem harus segera ditingkatkan.

Sommerville lebih menitikberatkan pada Software Evolution dimana software-software kuno yang vital harus di-rekayasa ulang agar dapat menjawab tantangan di masa kini dan masa depan.

Sedangkan Pressman membagi proses Reenginering kedalam 2 tahap yaitu Business Process reengineering dan Software Reenginering. Kedua hal tersebut memang berbeda dan harus dibedakan, jika business process Reenginering dilakukan oleh business analyst maka software Reenginering dilakukan oleh programer dan tim-timnya. Business process Reenginering terdiri dari Business definition, Process Identification, Process Evaluation, Process specification & design, Prototyping, Refinement & Instalation.

Sedang software Reenginering terdiri atas Inventory Analysis, Document Restructuring, Reverse Engineering, Code Restructuring, Data Restructuring, Forward Engineering.

Jika kedua pakar diatas lebih banyak menerangkan secara teoritis mengenai Reenginering,sedangkan Joseph Fong dalam bukunya Information System Reenginering and Integration, lebih banyak menerangkan secara praktis bagaimana proses-proses Reenginering itu dilalui. Joseph Fong banyak menerangkan bagaimana mengubah basis data misal dari skema Entity Relationship Diagram ke skema Object Oriented dan ke skema XML. Sungguh suatu teknik rumit tapi posible untuk dicoba dan berharga untuk dimiliki.

Tujuan Pembelajaran 16.4:

Penggunaan kembali Perangkat Lunak

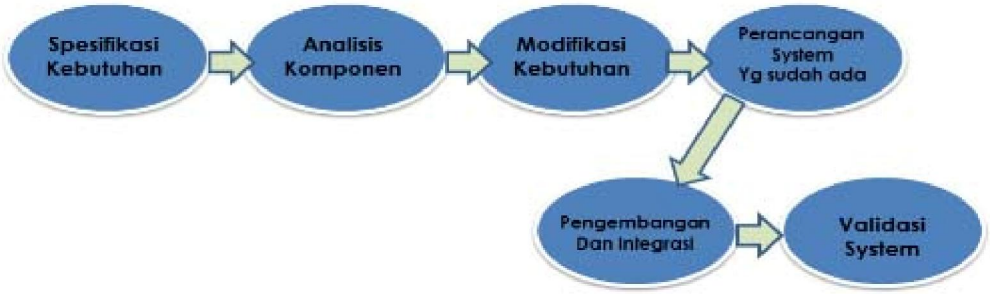
Dalam pengembangan perangkat lunak ada beberapa metode pendekatan yang digunakan, seperti : Waterfall, reuse, prototyping, spiral. Masing-masing dari metode tersebut memiliki kelebihan dan kekurangan masing-masing, hal tersebut tentu disesuaikan dengan waktu, budget dan pengalaman dari developer dalam mengembangkan sebuah software.

Penggunaan kembali perangkat lunak (software reuse) adalah proses menggunakan system perangkat lunak dari perangkat lunak yang ada dari pada

membangun system perangkat lunak dari awal. Ini sederhana namun memiliki visi yang kuat dan diperkenalkan pada tahun 1968.

Penggunaan kembali perangkat lunak adalah proses dimana suatu organisasi mendefinisikan prosedur sistematis untukmenspesifikasikan, membuat, mengklasifikasikan, menggunakan, dan mengadaptasiartifak perangkat lunak untuk menggunakannya pada aktivitas pengembangan perangkat lunak.

Dengan kata lain software reuse adalah integrasi dan penggunaan asset perangkat lunak dari sebuah system yang dikembangkan sebelumnya. Tahapan yang dilakukan untuk pengembangan software reuse, adalah :



Tahapan tersebut terdiri dari :

1. Spesifikasi Kebutuhan. Dilakukan untuk memenuhi kebutuhan yang diinginkan dan mendefinisikan kebutuhan yang diperlukan untuk mengatasi masalah yang terjadi pada system sebelumnya.
2. Analisis komponen. Hal ini dilakukan untuk meyakini bahwa spesifikasi kebutuhan telah didefinisikan dan user telah menyelaraskan terhadap kriteria spesifikasi kebutuhan yang baik.
3. Modifikasi kebutuhan. Kebutuhan dianalisis menggunakan informasi tentang komponen yang didapat, kemudian dimodifikasi untuk merefleksikan komponen yang ada. Jika modifikasi tidak mungkin dilakukan, maka kegiatan analisis komponen bisa diulang untuk mencari solusi alternatif.
4. Perancangan system. Proses ini dapat didefinisikan sebagai penggambaran, pembuatan sketsa atau pengaturan dari beberapa komponen yang sudah ada sebelumnya menjadi satu kesatuan yang

utuh dan berfungsi, perancangan system dapat menentukan bagaimana suatu system akan menyelesaikan apa yang mesti diselesaikan.

5. Pengembangan dan integrasi. Teknisi mengidentifikasikan bagaimana data dikonstruksikan, bagaimana fungsi-fungsi diimplementasikan sebagai sebuah arsitektur perangkat lunak, bagaimana rancangan diterjemahkan dalam bahasa pemograman.
6. Validasi system. Tahap ini system diuji kelayakannya untuk menjamin bahwa system berjalan sesuai dengan apa yang diinginkan user.

Kelebihan Software Reuse antara lain:

1. Meningkatkan kepercayaan. Software yang akan digunakan kembali telah di tes dan dicoba pada sistemnya, sehingga lebih bisa dipercaya dari software baru. awal pembuatan dari software mendeteksi berbagai kesalahan desain dan implementasi. Ini kemudian diperbaiki, yang megurangi tingkat kegagalan saat software di gunakan kembali.
2. Mengurangi resiko. Jika sotware telah ada, ada pengurangan biaya dalam pembuatan software. Ini adalah factor yang penting untuk manajemen proyek untuk mengurang estimasi biaya proyek disisi kesalahan software. Hal ini lebih terlihat saat sejumlah besar komponen software digunakan kembali.
3. Lebih efektif untuk para spesialis. Para spesialis tidak perlu melakukan pekerjaan yang sama pada proyek berbeda. Para spesialis bisa menggunakan software sebelumnya dengan mengkapsulasi program mereka.
4. Standar pelaksanaan. Beberapa standar, seperti standar user interface, bisa diimplementasikan sebagai standard reusable component. Sebagai contoh interface menu bisa diimplementasikan memggunakan reusable component, semua aplikasi menyajikan format menu yang sama. Standar interface ini meningkatkan

keyakinan user untuk mengurangi kesalahan ketika medapati interface yang familiar.

5. Percepatan pengembangan. Membawa software ke pasaran secepat mungkin adalah lebih penting dari semua biaya pengembangan. Reuse softrware dapat mempercepat produksi karena waktu pengembangan dan pengesahan software bisa dikurangi.

Kekurangan Software Reuse antara lain:

1. Meningkatkan biaya perawatan. Biaya perawatan mungkin akan bertambah saat reuse elemen dari suatu sistem memjadi semakin tidak sesuai dengan perubahan system.
2. Kekurangan tool pendukung. Toolset mungkin tidak support pembangunan software dengan model reuse. Ini mungkin sulit atau tidak mungkin untuk mengintegrasi tool – tool ini dengan sistem component library.
3. Sindrome Not-Invented-here. Beberapa software engineer kadang – kadang lebih suka menulis ulang reuse component sebagian dengan alasan dapat meningkatkan kegunaan reusable component, sebagian melakukan dengan kepercayaan bahwa fakta menulis original software adalah lebih menantang dari menggunakan software orang lain.
4. Membuat dan merawat komponen library. Menyusun sejumlah reusable componenet library dan menjamin pengembang bisa mengunakan library ini bisa menjadi mahal. Teknik umum kita untuk mengklasifikasi, mengkatalog, dan megambil komponen software adalah belum tepat.
5. Menemukan, mengerti dan mengadaptasikan komponen reusable. Komponen software harus ditemukan di library, dimengerti dan, kadang, diadaptasikan untuk bekerja di lingkungan baru. Engineer harus yakin untuk menemukan komponen di library sebelum mereka akan menyertakan komponen sebagai bagian dari proses pembangunan software mereka.

16.3 LATIHAN SOAL/TUGAS

1. Berikan contoh penggunaan ulang perangkat lunak dalam mengembangkan aplikasi sistem informasi akademik

16.4 DAFTAR PUSTAKA

1. Pressman, R. S. (2015). Software Engineering. A Practitioner's Approach (8th ed.). New York: McGraw-Hill Education.
2. Sommerville, I. (2011). Software Engineering (9th ed.). Boston: Addison-Wesley.