PERTEMUAN 18
(PENGGUNAAN KEMBALI (REUSE) PERANGKAT LUNAK)

A. TUJUAN PEMBELAJARAN

Setelah mempelajari materi ini mahasiswa diharapkan mampu untuk :

1. Dapat menjelaskan tentang penggunaan kembali perangkat lunak.

B. URAIAN MATERI

1. Penggunaan Kembali Perangkat Lunak *(Reuse Software)*

Perangkat lunak yang di gunakan kembali (*Software Reuse*) ialah suatu perjalanan yang membuat kembali dan untuk menghasilkan suatu pemrograman saat ini daripada membangun kerangka produk tanpa persiapan apa pun. Visi yang lugas namun luar biasa ini dipresentasikan pada tahun 1968. Secara keseluruhan, penggunaan kembali produk adalah penggabungan dan pemanfaatan sumber daya pemrograman dari kerangka kerja yang telah berkembang sebelumnya.

Pemrograman kembali, atau disebut penggunaan kembali kode, adalah pemanfaatan pemrograman yang ada, atau informasi pemrograman untuk memiliki pilihan untuk membuat pemrograman baru. Dalam banyak disiplin ilmu desain, kerangka kerja bekerja dengan menggabungkan bagian yang sudah ada dan telah digunakan pada kerangka kerja yang berbeda. Pemrograman komputer lebih bergantung pada pembuatan pemrograman secara lokal, namun akhir-akhir ini untuk meningkatkan kualitas, hasil pemrograman yang lebih murah, diperlukan siklus cepat yang bergantung pada gagasan penggunaan kembali pemrograman.

Penggunaan kembali telah dipoles sejak hari pertama pemrograman. Pengembang akan secara konsisten (menggunakan kembali) sebagian dari kode, format, kapasitas, dan teknik.

Hal-hal pemrograman yang dapat digunakan kembali atau informasi pemrograman benar-benar dapat disebut sumber daya pemrograman. Sumber daya mungkin berupa rencana, model pengujian, kebutuhan, atau rekayasa, dan sebagainya

a. Jenis Reuse Software

1) *Oportunistik Reuse* – Sebuah tim atau kelompok yang sudah bersiap memulai sebuah proyek, dan sadar akan adanya komponen yang dapat digunakan Kembali.

2) *Planning Reuse* - Sebuah kelompok merencanakan bagian-bagian penting sehingga akan digunakan kembali dalam usaha selanjutnya.

Bila menggunakan kembali oportunistik dapat dikategorikan lebih lanjut:

1) *Internal reuse* – bagian dari kumpulan di dalam bagian penggunaan kembali itu sendiri. sebuah pilihan yang mungkin terjadi dalam sebuah bisnis, karena sebuah pertemuan mungkin hanya menganalisis bagian dasar dari sebuah tugas

2) *External reuse* – izin yang dipilih oleh sekelompok orang luar di bagian yang dipilih. Biaya suku cadang yang diizinkan oleh pihak luar biasanya berkisar antara 1% - 20% dari jumlah yang dikeluarkan untuk tumbuh di dalam. Kelompok juga harus fokus pada waktu yang diperlukan untuk menemukan, mempelajari, dan menggabungkan bagian-bagian.

b. Manfaat Reuse Software

1) Biaya
2) Waktu
3) Reliabilitas
4) sebagian memiliki wawasan dan pengujian informasi asli tidak memberikan
5) note: catatan: Waktu tambahan yang diharapkan untuk rencana penggunaan kembali (pembuat)
6) menyisihkan sedikit upaya untuk merencanakan dengan penggunaan kembali (pembeli)

Alasan penggunaan kembali adalah untuk mengurangi biaya, waktu, tenaga, dan bahaya, dan untuk membangun efisiensi, kualitas, eksekusi, dan interoperabilitas.

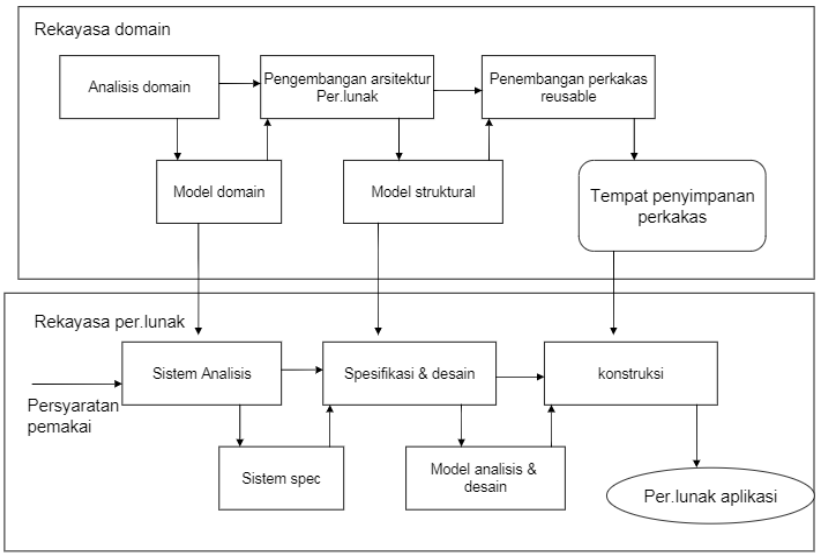
c. Perancangan Reuse Software

1) Mencirikan antarmuka / spesifikasi
2) Mencirikan pelaksanaan

3) pemrograman yang diatur/diperoleh objek
4) tata letak generik/format
5) modul peluruhan melewati batas/melewati batas
6) fitur

d. Siklus Penggunaan Kembali Pemrograman

Peralatan yang dapat digunakan kembali seperti Rencana pelaksanaan, Penawaran, Teknik, Model dan kebutuhan tertentu, Rencana, Kode sumber, Dokumentasi khusus dan klien, Antarmuka manusia, Informasi, Eksperimen Model Proses



Gambar 18. 1 *Rekaysa Domain*

e. Tujuan

Untuk membedakan, mengembangkan, membuat daftar, dan menumbuhkan sekelompok peralatan dalam suatu produk dengan penggunaan pemrograman masa depan dalam bidang penggunaan tertentu, dengan adanya beberapa latihan khususnya yaitu

1) menganalisis,
2) perangkat yang dibangun dan
3) Perluasan.

f. Perjalanan Analisis Domain

Didefinisikan prosesnya dengan langkah sebagai berikut:

1) Di tentukannya sebuah domain untuk diteliti.

2) mengkategorikan sebuah item yang akan pilah pada sebuah domain.
3) di kumpukannya sebuah hasil yang sepresentatif mungkin pada sebuah aplikasi di dalam domain.
4) menganalisis pada setiap aplikasi didalam sebuah hasil.
5) Mengembangkan suatu analisis pada sebuah proyek.

g. Fungsi-Fungsi Karakterisasi

Susunan kualitas ruang untuk peralatan yang dapat digunakan kembali dapat dialamatkan sebagai {Dp} di mana Dpi , membahas atribut area tertentu ;

Nilai yang dialokasikan untuk Dpi menunjukkan skala ordinal yang berarti pentingnya merek untuk perangkat p.

Jenis skala:

1) tidak penting
2) Penting tetapi dalam suatu kondisi diluar biasanya.
3) Penting, dan dapat di ubah-ubah walaupun bebeda,
4) sangat penting apabila suatu Software tidak memiliki karakter ini maka penggunaan Kembali tidak Efisien
5) sangat penting apabila perangkat lunak tidak memiliki karakteristik ini reuse tidak efektif, reuse tidak disetujui.

Ketika pemograman baru, *w,* akan mencangkup area Aplikasi, serangkaian karakteristik domain ditarik untuknya. Dpi dan Dwi kemudian dibandingkan untuk menentukan apakah perkakas yang ada (*p*) dapat secara efektif digunakan lagi didalam aplikasi *w*

Produk	Proses	Personil
Stabilitas persyaratan	Model proses	Motivasi
Per.lunak saat ini	Konformansi proses	Pendidikan
Batasan memori	Lingkungan proyek	Pengalaman - domain aplikasi - proses - platform - bahasa
Ukuran aplikasi	Batasan jadwal	
Kompleksitas interface	Batas anggaran	
Bhs. Pemrograman	produktivitas	
Keamanan/reliabilitas		
Lifetime requirement		Produktivitas tim
Kualitas produk		
Reliabilitas produk		

Gambar 18. 2 *Tabel Karateristik Domain*

h. Pemodelan Struktural dan Point Struktur

Tampilan utama adalah contoh yang bergantung pada metodologi perancangan area aplikasi yang bekerja dengan anggapan bahwa setiap

ruang aplikasi memiliki desain pengulangan (pekerjaan, informasi, perilaku) yang berpotensi untuk digunakan kembali.

Struktur titik memiliki 3 kualitas:

1) Struktur titik adalah refleksi yang seharusnya memiliki jumlah model yang terbatas.
2) Antarmuka ke desain titik harus dasar.
3) Struktur titik harus mengeksekusi penyembunyian data yang disimpan di dalam struktur titik itu sendiri.

2. Membangun Komponen Reuse Software

a. Analisis dan Desain untuk Reuse

Idealnya, model analisis di analisis untuk menentukan elemenelemen model yang menunjuk ke perkakas reusable yang ada. Kunci untuk desain reuse: (Design For Reuse/DFR)

1) Data standar : Domain aplikasi, dan struktur data global standar (struktur file atau database) lengkap harus diidentifikasi.
2) Protokol interface standar : Tiga tingkat protokol harus dibangun; sifat interface intramodular, desain interface teknis eksternal dan interface manusia mesin.
3) Template program : Model struktur dapat berfungsi sebagai template untuk desain arsitektur dari program baru.

1) Metode Konstruksi

Konstruksi dapat dilakukan dengan bahasa pemrograman generasi ketiga yang konvensional, bahasa generasi keempat dan generator kode, teknik pemrograman visual dan peranti yang lebih canggih. Serangkaian komponen generik dan dapat diadaptasikan yang disebut kerangka (frame) atau teknologi kerangka.

2) Pengembangan Berbasis Komponen

Pendekatan konstruksi kadang diacukan ke pengembangan berbasis komponen atau perangkat lunak komponen. Implementasi pengembangan berbasis komponen:

a) *Model pertukaran data* : Mekanisme yang memungkinkan pemakai dan aplikasi berinteraksi dan mentransfer data untuk semua komponen reusable.
b) *Otomasi* : Berbagai peranti macro dan script harus diimplementasi untuk memfasilitasi interaksi antar komponen reusable.
c) *Penyimpanan terstruktur* : Data heterogen (grafis, suara, teks dan numeris) dikumpulkan dan diakses sebagai struktur data tunggal.
d) *Model obyek yang mendasari* : Interface Definition Language (IDL)
e) *OpenDoc* : Standar dokumen gabungan dan perangkat lunak komponen.
f) *OMG/CORBA* : Object Request Broker/ORB memberi pelayanan komponen reusable berkomunikasi dengan komponen lain tanpa melihat lokasi didalam sistem.
g) *OLE 2.0 (Object Linking and Embedding)* : Merupakan bagian dari
h) Component Object Model (COM).

b. Keuntungan Reuse Software
1) Pengembangan sistematis komponen dapat digunakan kembali.
2) Sistematis penggunaan kembali komponen-komponen ini sebagai blok bangunan untuk menciptakan sistem baru.

Sebuah komponen dapat digunakan kembali kode, tetapi manfaat yang lebih besar menggunakan kembali datang dari yang lebih luas dan tampilan tingkat tinggi dari apa yang dapat digunakan kembali.

Keuntungan utama untuk menggunakan kembali perangkat lunak adalah untuk:

1) Perangkat lunak meningkatkan produktivitas.
2) Mempersingkat waktu pengembangan perangkat lunak.
3) Meningkatkan interoperabilitas sistem software.
4) Mengembangkan software dengan lebih sedikit orang.
5) Pindahkan personel lebih mudah dari proyek ke proyek.
6) Mengurangi pengembangan perangkat lunak dan biaya pemeliharaan.
7) Memproduksi lebih banyak perangkat lunak standar.
8) Menghasilkan perangkat lunak kualitas yang lebih baik dan memberikan keunggulan kompetitif yang kuat.

3. Pengklasifikasian Dan Pengambilan Kembali Komponen

a. Penggambaran Komponen Reuse

Konsep harus menggambarkan isi komponen, yaitu: isi dan konteks. Skema klasifikasi komponen perangkat lunak dibagi menjadi tiga kategori:

1) *Enumerated Classification* : Komponen digambarkan dengan menentukan struktur hirarki dimana kelas dan sub kelas yang bervariasi dari komponen diterapkan.
2) *Faceted Classification* : Sebuah area domain dianalisis dan serangkaian ciri deskriptif dasar diidentifikasi. Ciri-ciri tersebut disebut faceted.
3) *Klasifikasi atribut-nilai* : Sama dengan klasifikasi faced.

Lingkungan Reuse

1) Database komponen
2) Sistem manajemen pustaka
3) Sistem pemanggilan kembali perangkat lunak (OBR)
4) Peranti CASE

b. Dampak Terhadap Ekonomi

1) Pengaruh Terhadap Kualitas Produktivitas dan Biaya
   a) Kualitas : Dengan reuse cacat ditemukan dan dieliminasi sebagai hasilnya kualitas komponen meningkat
   b) Produktivitas : Lebih sedikit waktu yang digunakan untuk membuat rencana, model, dokumen, kode dan data yang diperlukan untuk membuat sistem.
   c) Biaya : Penghematan waktu neto untuk reuse diperkirakan dengan memproyeksikan bila proyek dikembangkan sejak awal.
2) Analisis Biaya Dengan Menggunakan Poin Struktur
   Idealnya adaptasi, integrasi dan biaya pemeliharaan yang berhubungan pada masing-masing komponen pada suatu pustaka reuse disimpan dari masing-masing contoh panggunaan, lalu dianalisa untuk mengembangkan proyeksi biaya bagi contoh reuse selanjutnya.

Usaha keseluruhan = Enew + Eadapt +Eint

Dimana

Enew adalah usaha untuk komponen perangkat lunak baru.
Eadapt adalah usaha untuk menyelesaikan reuse dari tiga point struktur yaitu SP1, SP2, SP3
Eint adalah usaha untuk mengintgrasikan SP1, SP2, SP3
SP1, SP2 dan SP3 ditentukan dengan mengambil rata-rata data historis.

c. Metrik Reuse

Untuk mengukur manfaat reuse pada sistem berbasis komputer. Reuse pada sistem S diekspresikan sebagai:

$$Rb(S) = [Cnoreuse – Creuse]/Cnoreuse$$

Dimana Cnoreuse adalah biaya pengembangan tanpa reuse
Creuse adalah biaya pengembangan dengan reuse
Rb(S) dieksresikan sebagai nilai nondimensional dalam range:
$0 \leq Rb(S) \leq 1$
Semakin tinggi nilai Rb(S), reuse akan semakin menarik.
Pengukuran reuse dalam sistem OO disebut *reuse leverage* ditentukan sebagai:
Rlev = OBJreused / OBJbuilt
dimana OBJreused adalah jumlah objek yang digunakan kembali
OBJbuilt adalah jumlah objek yang dibangun.
Bila Rlev bertambah Rb juga bertambah

d. Pentingnya Integrasi Reuse Software Ke Dalam Budaya Perusahaan

Software Reuse adalah strategi penting untuk semua kelompok pengembangan perangkat lunak. Dengan menggunakan kembali kode sementara pindah ke platform generasi berikutnya, perusahaan dapat memanfaatkan investasi perangkat lunak yang ada mereka dan mengurangi waktu ke pasar. Namun, banyak perusahaan sedang berjuang untuk sepenuhnya menggunakan kembali kode menerapkan seluruh organisasi mereka. Untuk mencapai efisien dan metodis penggunaan kembali kode, organisasi harus mengintegrasikan tujuan ini ke dalam budaya mereka. Kode

Reusing memberikan manfaat terbesar untuk sebuah organisasi jika hal itu dilakukan secara sistematis, daripada secara sporadis dan opportunistically. Namun, ada banyak hal yang dapat mencegah penggunaan kembali kode sistematis, baik teknis dan non-teknis.

1) Software reuse - Masalah Teknis

Di sisi teknis ada banyak perbedaan antara sistem operasi, seperti tingkat prioritas tugas yang ditawarkan oleh masing-masing OS, yang membuat memodifikasi kode untuk platform yang berbeda membosankan dan rumit. Ini telah membawa perlunya port Cots alat yang secara otomatis akan menjelaskan perbedaan-perbedaan dalam sistem operasi untuk membuat port bekerja lebih cepat dan lebih mudah.

Untuk menghindari masalah port sama sekali, organisasi melihat kebutuhan untuk solusi abstraksi untuk melindungi kode mereka terhadap perubahan platform masa depan. Namun, mengembangkan sebuah antarmuka abstraksi menggunakan OS asli API tidak akan memberikan portabilitas dan kinerja yang diperlukan dalam aplikasi embedded. Sebaliknya, tingkat rendah pendekatan perlu diambil untuk menjamin bahwa sumber daya sistem operasi dasar seperti benang, Semaphore, dan mutex akan berperilaku sama di seluruh platform dan kinerja yang tidak terkena dampak. Jika, membangun dan memelihara rumah yang di-abstraksi untuk beberapa sistem operasi memerlukan banyak waktu, uang, dan sumber daya.

Para pengembang harus memiliki pengetahuan yang terperinci dari masing-masing sistem operasi dan melakukan banyak pengujian untuk memverifikasi portabilitas platform yang berbeda, yang mengakibatkan biaya tinggi. Inilah sebabnya mengapa banyak perusahaan yang berubah ke arah abstraksi Cots lapisan yang tetap dipertahankan, diuji, dan diverifikasi oleh pihak ketiga, daripada mengambil fokus jauh dari kompetensi inti organisasi. Common Menggunakan API (yang disediakan oleh OS Cots abstraksi) di seluruh platform juga mengurangi potensi kurva pembelajaran saat mengembangkan sistem operasi baru, sehingga membuat lebih mudah untuk menggunakan kembali kode mengadopsi.

Sama seperti menggunakan kembali kode pada sistem operasi yang berbeda memiliki tantangan sendiri, menggunakan kembali kode

ketika pindah ke bahasa yang berbeda juga menyajikan kesulitan. Sebagai contoh, banyak perusahaan yang sekarang bergerak menjauh dari Ada yang lebih modern bahasa C, karena kurangnya dukungan untuk programmer dan Ada. Organisasi-organisasi ini memanfaatkan konversi bahasa Cots alat untuk konversi otomatis untuk menghindari penulisan ulang.

2) Software Reuse – Masalah Industri

Di sisi non-teknis, sementara tingkat atas eksekutif dan lembaga pemerintah mungkin akan melihat manfaat penggunaan kembali kode, ada tujuan kurangnya kesesuaian dengan kelompok rekayasa dan subkontraktor. Berkali-kali kelompok-kelompok ini mempunyai hambatan psikologis untuk menggunakan kembali kode. Mereka mungkin keliru berpikir bahwa menggunakan kembali kode akan menyebabkan bakat-bakat mereka tidak lagi diperlukan. Namun, dengan menggunakan kembali kode warisan mereka dengan cepat dan efisien dengan menggunakan kembali kode Cots solusi, mereka dapat menyumbangkan bakat untuk proyek-proyek baru dan pengembangan produk, bukannya macet oleh port melelahkan bekerja.

Organisasi mungkin juga perlu mengubah kebijakan dan standar produktivitas untuk secara efektif menggunakan kembali kode mengintegrasikan ke dalam budaya mereka. Alih-alih berfokus pada berapa banyak baris kode baru pengembang berkontribusi, mereka mungkin perlu untuk memberikan penghargaan lebih pendek kali untuk penyelesaian proyek. Ini akan memotivasi para pengembang untuk menggunakan alat-alat port Cots sehingga mereka dapat menggunakan kembali sebanyak mungkin lebih cepat untuk memenuhi tenggat waktu sebelumnya. Hal ini akan menyebabkan penyelesaian proyek lebih banyak, lebih banyak produk baru, dan akhirnya lebih banyak kesempatan untuk mendapatkan pangsa pasar yang lebih besar dalam organisasi industri.

3) Fokus pada asset software khusus untuk domain

Aset bisnis adalah apa yang membuat aplikasi atau lini produk Anda unik, organisasi Anda istimewa, dan pada akhirnya membedakan Anda

dari pesaing. Semakin cepat Anda dapat mengembangkan, merilis, dan secara berulang meningkatkan aset perangkat lunak yang relevan dengan domain Anda, semakin cepat Anda akan memenuhi perubahan kebutuhan bisnis dan menyenangkan pelanggan Anda. Jika Anda hanya berfokus pada membangun aset bisnis yang dapat digunakan kembali, kemungkinan besar aset tersebut relevan dengan hasil bisnis Anda dan dapat digunakan kembali untuk proyek masa depan.

Terlalu sering, pengembang dalam semangat mereka untuk membuat karya teknis, fokus pada membangun komponen dan layanan yang dapat digunakan kembali untuk masalah yang memiliki solusi di dalam perusahaan Anda atau dari komunitas sumber terbuka. Sekarang, jika harus, Anda harus - tetapi cobalah untuk menghindari membuat kode baru yang solusinya sudah ada. Bukankah itu tentang penggunaan kembali perangkat lunak?

Ketika Anda menyadari perlunya aset perangkat lunak yang dapat digunakan kembali, itu adalah kunci untuk memetakan strategi realisasi. Jika Anda mendekati realisasi aset big bang, Anda akhirnya bisa membuat aset perangkat lunak yang tidak relevan dengan kebutuhan mendesak proyek Anda dan menambah risiko jadwal yang signifikan karena peningkatan waktu desain, pengembangan, dan pengujian. Either way, Anda akan menghabiskan banyak sumber daya berharga.

Alih-alih mengurangi risiko ini dengan mengembangkan aset yang dapat digunakan kembali selama beberapa iterasi. Ambil contoh aset yang dapat digunakan kembali yang mengirimkan pemberitahuan kepada pengguna Anda. Mari kita beri nama aset *Pemberitahu Bisnis* . Alih-alih mencoba membangun 100 lonceng dan peluit yang bisa dimiliki aset ini, kami membuat rencana sederhana untuk mengembangkannya selama beberapa iterasi.

Iterasi #1 - pemberitahuan melalui email untuk acara bisnis yang telah ditentukan sebelumnya
Iterasi #2 - izinkan pengguna untuk memilih pemberitahuan di luar acara bisnis
Iterasi #3 - biarkan pengguna menentukan aturan bisnis khusus untuk menghasilkan acara bisnis

Iterasi #4 - mengirim pemberitahuan di konsol web atau aplikasi pesan instan

Iterasi #5 - memungkinkan pengguna untuk menyertakan teman mereka saat menerima notifikasi.

Iterasi #6 - integrasikan notifikasi dengan alur kerja untuk ditinjau oleh orang pendukung

Jelas ini hanyalah sebuah contoh dan fitur yang Anda masukkan ke dalam iterasi tertentu didasarkan pada kebutuhan bisnis, prioritas, kemudahan implementasi, dan pertimbangan lainnya. Misalnya, Anda dapat mengurangi investasi pada suatu aset jika tidak lagi menjadi prioritas atau sebaliknya. Takeaway adalah apa pun yang ingin Anda bangun sebagai aset yang dapat digunakan kembali, rencanakan evolusinya melalui beberapa iterasi. Anda akan mengurangi risiko, tetap fleksibel dengan persyaratan dan kebutuhan bisnis Anda, dan hanya membangun aset yang ingin Anda investasikan.

C. SOAL LATIHAN/TUGAS

1. Sebutkan jenis – jenis reuse software ?
2. Jelaskan masalah Teknik pada reuse software ?
3. Bagaimana tahapan analisis dan desain pada reuse software ?
4. Apa klasifikasi untuk reuse software ?
5. Metode apa yang sering digunakan untuk reuse software ?

D. REFERENSI

1. https://www.scribd.com/presentation/541849567/PERTEMUAN-9-SOFTWARE-REUSE-20-NOV-2021
2. https://ejurnal.itats.ac.id/iptek/article/view/252