Pertemuan 14

IMPLEMENTASI dan PEMELIHARAAN

1. IMPLEMENTASI PL

IMPLEMENTASI

- Perancangan dan implementasi PL adalah tahap dalam proses RPL dimana dikembangkan sistem PL yang dapat dieksekusi.
- Implementasi adalah proses mewujudkan desain sebagai sebuah program.
- RPL mencakup semua kegiatan yang terlibat dalam pengembangan PL dari persyaratan awal sistem hingga pemeliharaan dan pengelolaan sistem yang digunakan.
- Implementasi dapat melibatkan pengembangan program atau menyesuaikan dan mengadaptasi sistem generik, *off-the-shelf* untuk memenuhi persyaratan khusus dari suatu organisasi.

IMPLEMENTASI (Lanjutan)

Aspek implementasi yang sangat penting untuk RPL:

1. ***Reuse***

   Sebagian besar PL modern dibangun dengan menggunakan kembali komponen atau sistem yang ada.

2. ***Configuration Management***

   Selama proses pengembangan, banyak versi yang berbeda dari setiap komponen PL.

3. ***Host-Target Development***

   Produksi PL biasanya tidak dijalankan pada komputer yang sama dengan lingkungan pengembangan PL. Pengembangan pada satu komputer (sistem *host*) dan dijalankan pada komputer yang terpisah (sistem target).

A. *Reuse*

Penggunaan ulang (*reuse*) PL dimungkinkan pada sejumlah level yang berbeda:

1. **Tingkat Abstraks**.

   Pada tingkat ini, tidak menggunakan *reuse software* secara langsung tetapi menggunakan pengetahuan abstraksi dalam desain PL menggunakan pola desain dan pola arsitektur.

2. **Tingkat Objek**.

   Pada tingkat ini, langsung menggunakan *reuse objects* dari *library* daripada menulis kode sendiri.

Reuse (Lanjutan)

3. **Tingkat Komponen**.

   Komponen adalah kumpulan objek dan kelas objek yang beroperasi bersama untuk menyediakan fungsi dan layanan terkait. Pada tingkat ini harus menyesuaikan dan memperluas komponen dengan menambahkan beberapa kode sendiri.

4. **Tingkat Sistem**.

   Pada tingkat ini, menggunakan kembali seluruh sistem aplikasi. Biasanya melibatkan beberapa jenis konfigurasi sistem. Dapat dilakukan dengan menambahkan dan memodifikasi kode atau dengan menggunakan antarmuka konfigurasi sistem sendiri.

Reuse (Lanjutan)

Keuntungan menggunakan *reuse software* yang ada:

a. Dapat mengembangkan sistem baru dengan lebih cepat
b. Dengan risiko pengembangan yang lebih sedikit dan juga biaya yang lebih rendah
c. Karena *reuse software* yang digunakan telah diuji dalam aplikasi lain, sehingga lebih dapat diandalkan daripada PL baru

Reuse (Lanjutan)

Biaya yang terkait dengan penggunaan kembali:

1. Biaya waktu yang dihabiskan dalam mencari PL untuk digunakan kembali dan menilai apakah sudah memenuhi kebutuhan atau tidak, dan menguji PL untuk memastikan bahwa dapat bekerja di lingkungan sistem.
2. Biaya membeli PL yang dapat digunakan kembali.
3. Biaya untuk adaptasi dan mengkonfigurasi komponen PL/sistem yang dapat digunakan kembali.
4. Biaya untuk mengintegrasikan elemen *reuse software* yang dapat digunakan satu sama lain (jika menggunakan PL dari sumber yang berbeda) dan dengan kode baru yang telah dikembangkan.

B. Manajemen Konfigurasi (*Configuration Management*)

- Manajemen konfigurasi merupakan proses rekayasa sistem untuk menetapkan dan mempertahankan konsistensi dari kinerja produk, fungsional, dan atribut fisik dengan persyaratan, desain, dan informasi operasional sepanjang hidupnya
- Tujuan dari manajemen konfigurasi adalah untuk mendukung proses integrasi sistem sehingga semua pengembang dapat mengakses kode dan dokumen dengan cara yang terkontrol, mencari tahu perubahan yang telah dibuat, dan mengkompilasi dan menghubungkan komponen untuk membuat sistem

Manajemen Konfigurasi (Lanjutan)

Tiga aktivitas dasar manajemen konfigurasi:

1. ***Version Management***, dukungan diberikan untuk melacak berbagai versi komponen PL, mencakup fasilitas untuk mengkoordinasikan pengembangan oleh beberapa programmer.
2. **Integrasi sistem**, dukungan disediakan untuk membantu pengembang menentukan versi komponen yang digunakan untuk membuat setiap versi sistem.
3. **Pelacakan masalah**, dukungan diberikan untuk memungkinkan *user* melaporkan *bug* dan masalah lain, dan memungkinkan semua pengembang untuk melihat siapa yang bekerja pada masalah ini dan memperbaikinya

C. *Host-Target Development*

- PL dikembangkan pada satu komputer (*host*), tetapi berjalan pada mesin yang terpisah (*target*).
- *Platform* lebih dari sekedar perangkat keras, termasuk sistem operasi yang terinstal ditambah perangkat lunak pendukung lainnya seperti DBMS, *platform* pengembangan, dan lingkungan pengembangan interaktif.
- *Platform* pengembangan dan eksekusi adalah sama, sehingga memungkinkan untuk mengembangkan PL dan mengujinya di mesin yang sama. Tetapi terkadang sering berbeda sehingga perlu memindahkan PL yang dikembangkan ke *platform* eksekusi untuk menguji atau menjalankan simulator pada mesin untuk pengembangan

Host-Target Development (Lanjutan)

Platform pengembangan PL harus menyediakan berbagai alat untuk mendukung proses RPL, termasuk:

1. Sebuah kompilator terintegrasi dan sistem pengeditan yang dirancang secara sintaks yang memungkinkan untuk membuat, mengedit, dan mengkompilasi kode.
2. Sistem *debug* bahasa.
3. Alat pengeditan grafis, seperti alat untuk mengedit UML.
4. Alat pengujian, seperti JUnit yang dapat menjalankan serangkaian tes secara otomatis pada versi baru program.
5. Alat dukungan proyek yang membantu mengatur kode untuk berbagai proyek pengembangan.

Host-Target Development (Lanjutan)

- Diperlukan keputusan tentang bagaimana PL yang dikembangkan akan digunakan pada *platform* target.
- Pertimbangan dalam membuat keputusan adalah:
  1. Persyaratan perangkat keras dan perangkat lunak dari suatu komponen
  2. Ketersediaan persyaratan sistem
  3. Komunikasi komponen, jika ada tingkat lalu lintas komunikasi yang tinggi antar komponen

2. PEMELIHARAAN (*MAINTENANCE*)

PEMELIHARAAN (*MAINTENANCE*)

Pemeliharaan PL adalah suatu aktivitas yang sangat luas yang sering digambarkan mencakup semua pekerjaan yang dibuat di suatu sistem setelah PL beroperasi.

Aktivitas meliputi:

a. Penambahan atau perbaikan program, seperti penambahan fungsi baru, dan perbaikan tampilan.
b. Perbaikan terhadap kesalahan yang timbul
c. Penghapusan kemampuan kualitas
d. Peningkatan pencapaian & memperluas daya guna untuk memenuhi kebutuhan *user* yang semakin bertambah
d. Menyesuaikan PL untuk memenuhi lingkungan yang berubah

A. Kategori Pemeliharaan PL

- **Korektif** adalah perbaikan program akibat adanya kesalahan
- **Adaptif** adalah penyesuaian dengan lingkungan yang baru, seperti penerapan pada *platform* di lingkungan yang baru, format tampilan printer, dll
- **Perfective** terjadi pada saat pengguna sistem atau *stakeholder* merubah *requirement* dari sistem yang dibangun
- **Preventif** berhubungan dengan prediksi yang akan datang, seperti penggunaan anti virus untuk keamanan data, *back-up* data dan program

Contoh Tugas-Tugas Pemeliharaan

Corrective Maintenance
- Diagnose and fix logic errors
- Replace defective network cabling
- Restore proper configuration settings
- Debug program code
- Update drivers
- Install software patch

Adaptive Maintenance
- Add online capability
- Create new reports
- Add new data entry field to input screen
- Install links to Web site
- Create employee portal

Perfective Maintenance
- Install additional memory
- Write macros to handle repetitive tasks
- Compress system files
- Optimize user desktop settings
- Develop library for code reuse
- Install more powerful network server

Preventive Maintenance
- Install new antivirus software
- Develop standard backup schedule
- Implement regular defragmentation process
- Analyze problem report for patterns
- Tighten all cable connections

a. Pemeliharaan Korektif (*Corrective Maintenance*)

- Pekerjaan pemeliharaan sistem harus dilakukan terlebih dahulu di lingkungan pengujian, dan kemudian dimigrasikan ke operasional sistem.
- Situasi terburuk adalah kegagalan sistem. Jika keadaan darurat terjadi, tim pemeliharaan mencoba memperbaiki masalah dengan segera, sementara permintaan sistem tertulis disiapkan dan ditambahkan ke *log* pemeliharaan.
- Ketika sistem beroperasi kembali, tim pemeliharaan menentukan penyebabnya, menganalisa masalah, dan mendesain solusi permanen. Kemudian memperbarui file data, menguji sistem secara menyeluruh, dan menyiapkan dokumentasi lengkap.

b. Pemeliharaan Adaptif (*Adaptive Maintenance*)

- Pemeliharaan adaptif menambahkan peningkatan pada operasional sistem dan membuat sistem lebih mudah digunakan berupa peningkatan fitur baru/perubahan.
- Misal: layanan baru, teknologi manufaktur baru, atau dukungan untuk operasi berbasis web baru.
- Pemeliharaan adaptif membutuhkan lebih banyak sumber daya departemen IT daripada pemeliharaan korektif.
- Pemeliharaan adaptif bisa lebih sulit daripada pengembangan sistem baru karena penyempurnaan harus bekerja dalam batasan sistem yang ada/baru.

c. Pemeliharaan Perfektif (*Perfective Maintenance*)

- Melibatkan perubahan operasional sistem agar lebih
- efisien, dapat diandalkan, dan dapat dipelihara.
- Permintaan untuk pemeliharaan korektif dan adaptif biasanya berasal dari pengguna, sedangkan departemen IT biasanya memulai pemeliharaan perfektif.
- Pemeliharaan perfektif dapat meningkatkan keandalan sistem. Misalnya, masalah *input* dapat menyebabkan program berhenti secara tidak normal, sehingga diperlukan program yang dapat menangani masalah tsb.
- Semakin banyak program berubah, semakin besar ketidakefisienan dan sulit dipertahankan.

d. Pemeliharaan Preventif (*Preventive Maintenance*)

- Untuk menghindari masalah, pemeliharaan preventif membutuhkan area analisis dimana masalah mungkin terjadi.
- Pemeliharaan preventif menghasilkan peningkatan kepuasan pengguna, *downtime* yang menurun, dan pengurangan biaya.
- Pemeliharaan harus dilayani oleh teknisi yang ahli sehingga kualitas pemeliharaan akan langsung mempengaruhi keberhasilan organisasi.

3. PEMELIHARAAN MANAJEMEN

A. TIM PEMELIHARAAN

1. **Systems Administrator**

   Bertanggung jawab untuk pemeliharaan rutin dan berwenang mengambil tindakan pencegahan untuk menghindari keadaan darurat. Seperti kerusakan server, pemadaman jaringan, insiden keamanan, dan kegagalan perangkat keras.

2. **Systems Analyst**

   Bertugas menyelidiki dan menemukan sumber masalah dengan menggunakan keterampilan analisis dan sintesis. Analisis: memeriksa keseluruhan unsur-unsur individu. Sintesis: mempelajari bagian-bagian untuk memahami keseluruhan sistem.

Pemeliharaan Manajemen (Lanjutan)

3. Programmer

- Programmer aplikasi bekerja pada pengembangan dan pemeliharaan sistem baru.
- Programmer sistem berkonsentrasi pada perangkat lunak dan utilitas sistem
- Programmer basis data fokus pada pembuatan dan dukungan sistem basis data skala besar.

B. PERMINTAAN PEMELIHARAAN

Pengguna mengirimkan sebagian besar permintaan untuk pemeliharaan korektif dan adaptif ketika sistem tidak berfungsi dengan baik, atau jika mereka menginginkan fitur baru.

1. **Determinasi Awal**

   Ketika pengguna mengajukan permintaan pemeliharaan, administrator membuat penentuan awal, jika permintaan memerlukan perhatian segera, administrator akan mengambil tindakan sekaligus.

Pemeliharaan Manajemen (Lanjutan)

2. **Komite Peninjau Sistem**

   Ketika suatu permintaan melebihi tingkat biaya yang telah ditentukan atau melibatkan perubahan konfigurasi utama, komite peninjau sistem akan menetapkan prioritas, atau menolaknya.

3. **Penyelesaian Tugas**

   Administrator sistem bertanggung jawab untuk mempertimbangkan pengalihan tugas di antara staf IT atau membatasi tugas pemeliharaan kepada individu atau tim tertentu agar tugas dapat diselesaikan dengan baik.

Pemeliharaan Manajemen (Lanjutan)

4. User Notification

Pengguna yang memulai permintaan pemeliharaan mengharapkan tanggapan yang cepat, terutama jika situasi tersebut secara langsung mempengaruhi pekerjaan mereka. Bahkan ketika tindakan korektif tidak dapat terjadi dengan segera, pengguna akan menghargai umpan balik dari administrator sistem dan harus terus diberitahu tentang keputusan atau tindakan yang akan mempengaruhi pengguna.