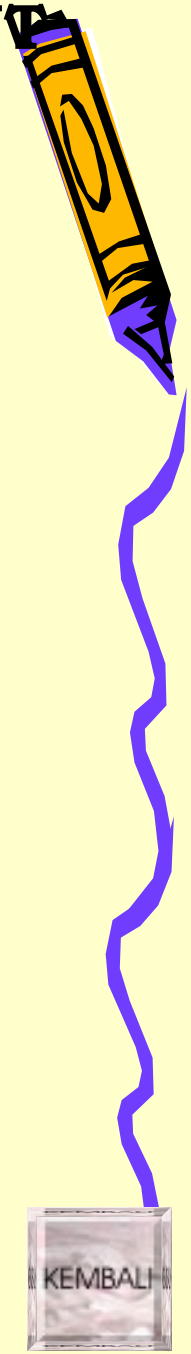
10. SOFTWARE CONFIGURATION MANAGEMENT



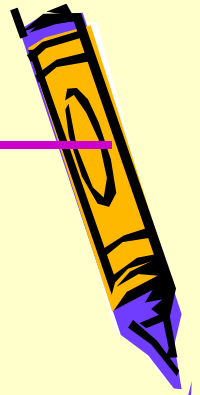
10.1. Pendahuluan

10.1.1 Perubahan

- Perubahan adalah hal yang tidak dapat dihindarkan ketika perangkat lunak komputer sedang dibuat.
- Perubahan^2 tersebut meningkatkan tingkat kebingungan di antara para software engineer yang berkerja pada proyek tersebut.
- Kebingungan muncul bila perubahan^2 tersebut tidak dianalisis sebelum perubahan tersebut dilaksanakan; dicatat sebelum diimplementasi, dilaporkan kepada yang ingin mengetahui, atau dikontrol dengan suatu cara yang akan meningkatkan kualitas & mengurangi error.
- Software configuration management (SCM) adalah kegiatan payung (umbrella activities) yang dilaksanakan selama proses perangkat lunak.

10.1.2 Tujuan SCM

- Karena perubahan dapat terjadi kapan saja, maka kegiatan SCM dibuat untuk;
  1) mengidentifikasi perubahan,
  2) mengontrol perubahan,
  3) mengimplementasikan perubahan dengan benar, dan
  4) melaporkan perubahan kepada pihak-pihak yang mempunyai kepentingan.



KEMBALI

10.1.3 Software Maintenance vs Software Configuration Management

- Penting untuk dibedakan dengan jelas antara software maintenance dan software configuration management.
- Software Maintenance adalah serangkaian aktivitas rekayasa perangkat lunak yang terjadi setelah perangkat lunak diserahkan ke pelanggan dan telah dioperasikan.
- Software configuration management adalah serangkaian kegiatan tracking & control yang dimulai ketika suatu proyek perangkat lunak dimulai dan berakhir ketika perangkat lunak sudah tidak beroperasi lagi.

10.1.4 Informasi dan Perubahan

- Keluaran dari proses perangkat lunak adalah informasi yang dapat dibagi ke dalam 3 kategori utama;
  1) program komputer (baik dalam bentuk source code maupun executable),
  2) dokumen[2] yang menjelaskan program komputer tersebut (yang ditargetkan baik untuk technical practitioners maupun users), dan
  3) data (yang diisikan dalam program atau dikeluarkan dari program).
- Item[2] yang terdiri dari semua informasi yang dihasilkan sebagai bagian dari proses perangkat lunak secara kolektif disebut Software Configuration Items (SCI).

- Selama proses - rekayasa SCI berkembang dengan pesat. System specification menghasilkan sebuah software project plan dan software requirements specification (juga dokumen[2] yang terkait dengan hardware), yang secara berurutan akan menghasilkan dokumen[2] untuk menciptakan suatu hirarki informasi.
- Perubahan masuk ke dalam proses rekayasa. Perubahan dapat terjadi kapan saja, untuk suatu alasan. Ini sesuai dengan Hukum 1 system engineering [BER80] yang menyatakan: "Tidak masalah dimana anda berada dalam siklus kehidupan sistem, sistem akan berubah, dan keinginan untuk mengubahnya akan selalu ada selama siklus hidup tersebut".

KEMBALI

10.2. Software Configuration Management

10.2.1 Sumber Dasar Perubahan

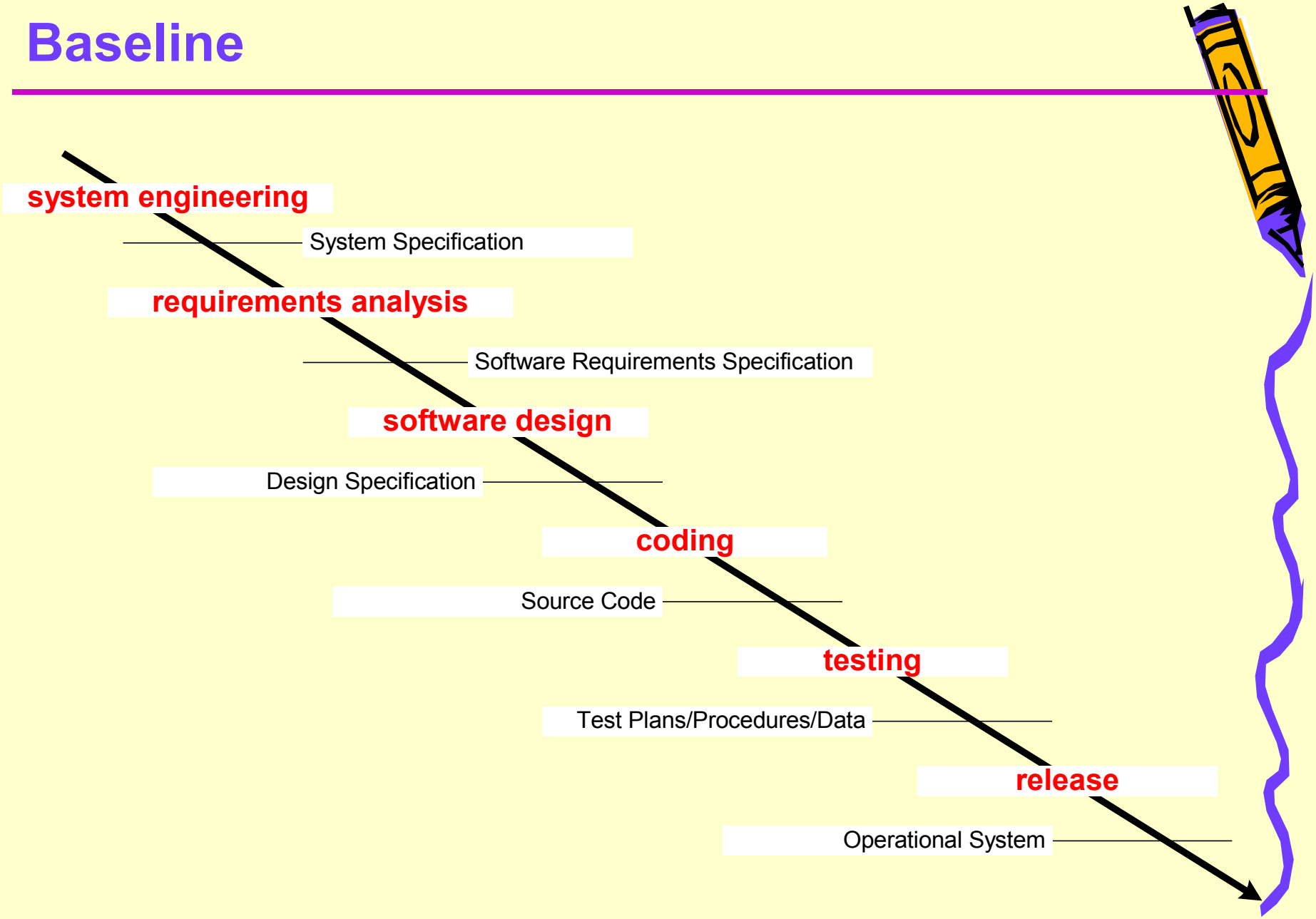
- Terdapat 4 sumber dasar perubahan.
  - Bisnis baru atau kondisi pasar yang mendiktekan perubahan[2] dalam produk atau aturan[2] bisnis.
  - Keinginan pelanggan baru yang meminta modifikasi data yang dihasilkan oleh sistem informasi; fungsionalitas yang diberikan oleh produk, atau layanan yang diberikan oleh suatu sistem berbasis komputer.
  - Reorganisasi dan/atau perampingan bisnis yang menyebabkan perubahan prioritas proyek atau struktur tim rekayasa perangkat lunak.
  - Kendala[2] anggaran atau jadwal yang menyebabkan redefinisi sistem atau produk.
- SCM adalah sekumpulan kegiatan yang telah dikembangkan untuk menangani perubahan[2] selama siklus hidup dari perangkat lunak komputer.
- SCM dapat dipandang sebagai kegiatan SQA yang dipakai selama proses perangkat lunak.

10.2.2 Baseline

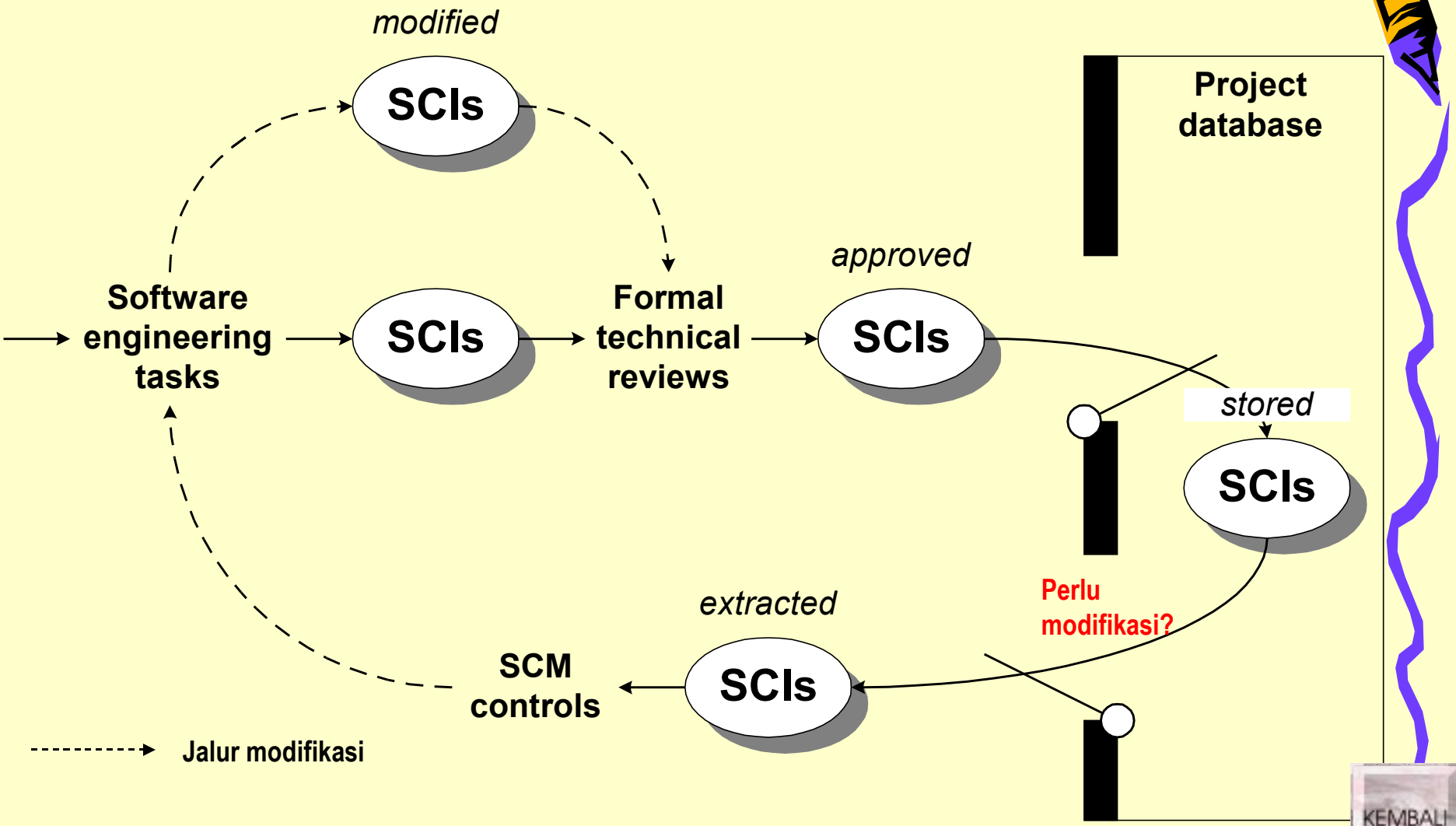
- Perubahan adalah kenyataan hidup dalam pengembangan perangkat lunak. Pelanggan ingin memodifikasi persyaratan[2]. Pengembang ingin memodifikasi metode[2] teknis. Manager ingin memodifikasi cara pendekatan proyek.
- Baseline adalah sebuah konsep SCM yang membantu dalam kontrol perubahan[2] tanpa secara serius menghalangi perubahan[2] yang dapat dijustifikasi.
- IEEE mendifinisikan baseline sebagai:

  Suatu spesifikasi atau produk yang telah direview secara formal dan disetujui bersama, yang selanjutnya berfungsi sebagai dasar bagi pengembangan lebih lanjut, serta dapat diubah hanya melalui prosedur[2] kontrol perubahan formal.
- Dalam konteks rekayasa perangkat lunak, baseline adalah milestone dalam rekayasa perangkat lunak yang ditandai dengan penyampaian (delivery) SCI dan persetujuan (approval) SCI tersebut yang diperoleh lewat suatu FTR.

Baseline


system engineering
System Specification
requirements analysis
Software Requirements Specification
software design
Design Specification
coding
Source Code
testing
Test Plans/Procedures/Data
release
Operational System

10.2.3 SCI Baseline dan Database Proyek

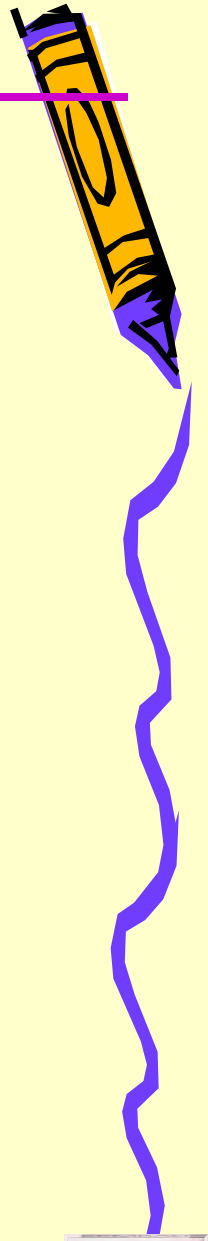


10.3 Software Configuration Item (SCI)

- SCI merupakan informasi yang diciptakan sebagai bagian dari proses rekayasa perangkat lunak.
- SCI berikut menjadi target bagi teknik[2] CM dan membentuk sekumpulan baseline.

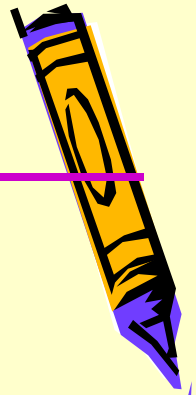
1. **System Specification**
2. **Software Project Plan**
3. **Software Requirement Specification**
   a. Graphical analysis model
   b. Process specifications
   c. Prototype(s)
   d. Mathematical specification
4. **Preliminary User Manual**
5. **Design Specification**
   a. Data design description
   b. Architectural design description
   c. Modul design descriptions
   d. Interface design descriptions
   e. Object description

Software Configuration Item (lanj)



6. **Source Code Listing**
7. **Test Specification**
   a. Test plan & procedure
   b. Test cases & recorded results
8. **Operation & Installation Manuals**
9. **Executable Program**
   a. Module executable code
   b. Linked modules
10. **Database Description**
    a. Schema & file structure
    b. Initial content
11. **As-built User Manual**
12. **Maintenance Documents**
    a. Software problem reports
    b. Maintenance requests
    c. Engineering change orders
13. **Standard & Procedure for Software Engineering**

10.4. SCM Process



10.4.1 Tanggung Jawab SCM

SCM adalah sebuah elemen penting dari SQA

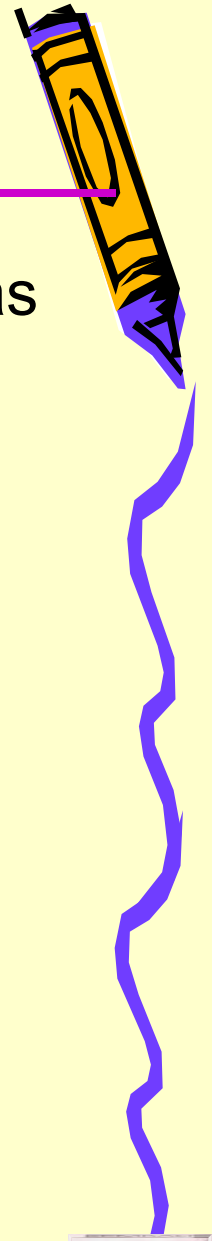
- Tanggung jawab utamanya adalah mengontrol perubahan.
- SCM juga bertanggung jawab untuk :
  - ✪ mengidentifikasi individual SCI & berbagai versi perangkat lunak,
  - ✪ meng-audit software configuration untuk memastikan bahwa dia telah dikembangkan dengan benar, dan
  - ✪ melaporkan semua perubahan yang telah dilakukan pada konfigurasi tersebut.

10.4.2 Pertanyaan Seputar SCM

- Diskusi mengenai SCM memperkenalkan sekumpulan pertanyaan kompleks sebagai berikut.
  - Bagaimana suatu organisasi mengidentifikasi dan menangani berbagai versi yang ada dari sebuah program (dan dokumentasinya) dalam suatu cara yang akan memungkinkan perubahan ditampung secara efisien?
  - Bagaimana suatu organisasi mengontrol perubahan sebelum dan setelah perangkat lunak dirilis ke pelanggan?
  - Siapa yang mempunyai tanggung jawab (otoritas) untuk approving (menyetujui) & prioritizing (memprioritaskan) perubahan?
  - Bagaimana kita yakin bahwa perubahan[2] tersebut telah dilakukan dengan benar?
  - Mekanisme apa yang dipakai untuk memberitahu yang lain bahwa perubahan telah dilakukan?

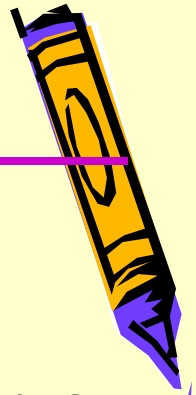
10.4.3 Tugas SCM

- Pertanyaan[2] tersebut membawa kepada definisi 5 tugas (task) SCM :
  - identifikasi,
  - kontrol versi,
  - kontrol perubahan,
  - auditing konfigurasi, dan
  - pelaporan.



KEMBALI

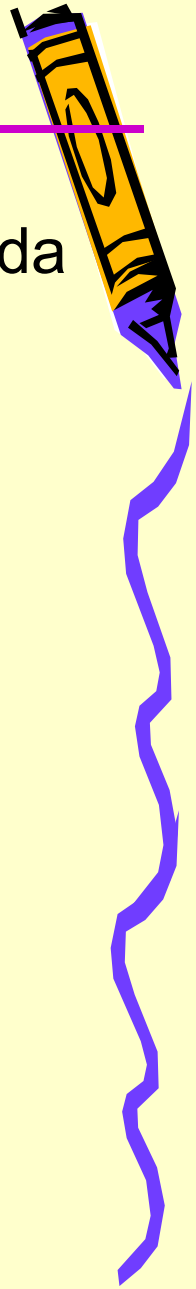
10.5 Identifikasi Objek dalam SC



10.5.1 Tipe Objek

- Untuk mengontrol & menangani SCI[2], masing[2] item harus diberi nama berbeda dan kemudian diorganisir dengan menggunakan metode berorientasi objek.
- Dua tipe objek dapat diidentifikasi: basic objects (obyek dasar) & aggregate objects (kumpulan obyek).
- Sebuah basic object adalah sebuah "unit of text" yang telah diciptakan oleh seorang software engineer pada saat (selama) analisis, design, coding, atau testing.
- Sebuah aggregate object adalah sekumpulan basic object dan aggregate objects lainnya.

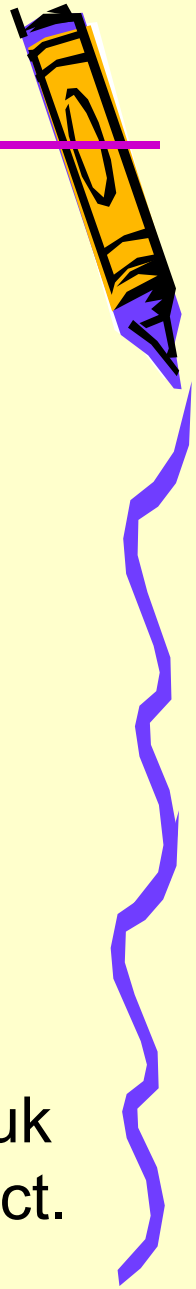
10.5.2 Keunikan Objek



Setiap object mempunyai sekumpulan fitur yang berbeda yang mengidentifikasikannya secara unik;

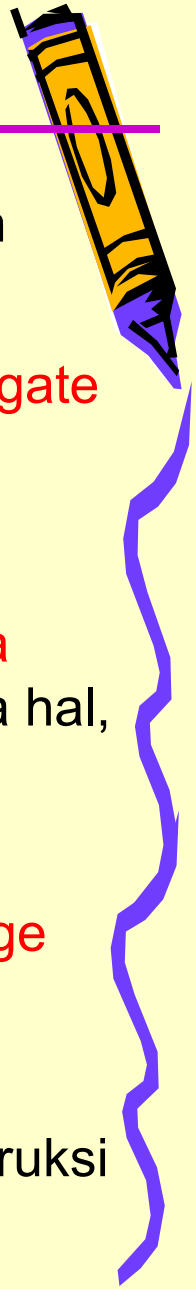
- sebuah nama (object name),
- suatu deskripsi (object description),
- suatu daftar sumber daya (resources list) dan
- suatu realisasi (realization)

10.5.2 Keunikan Objek (lanj)



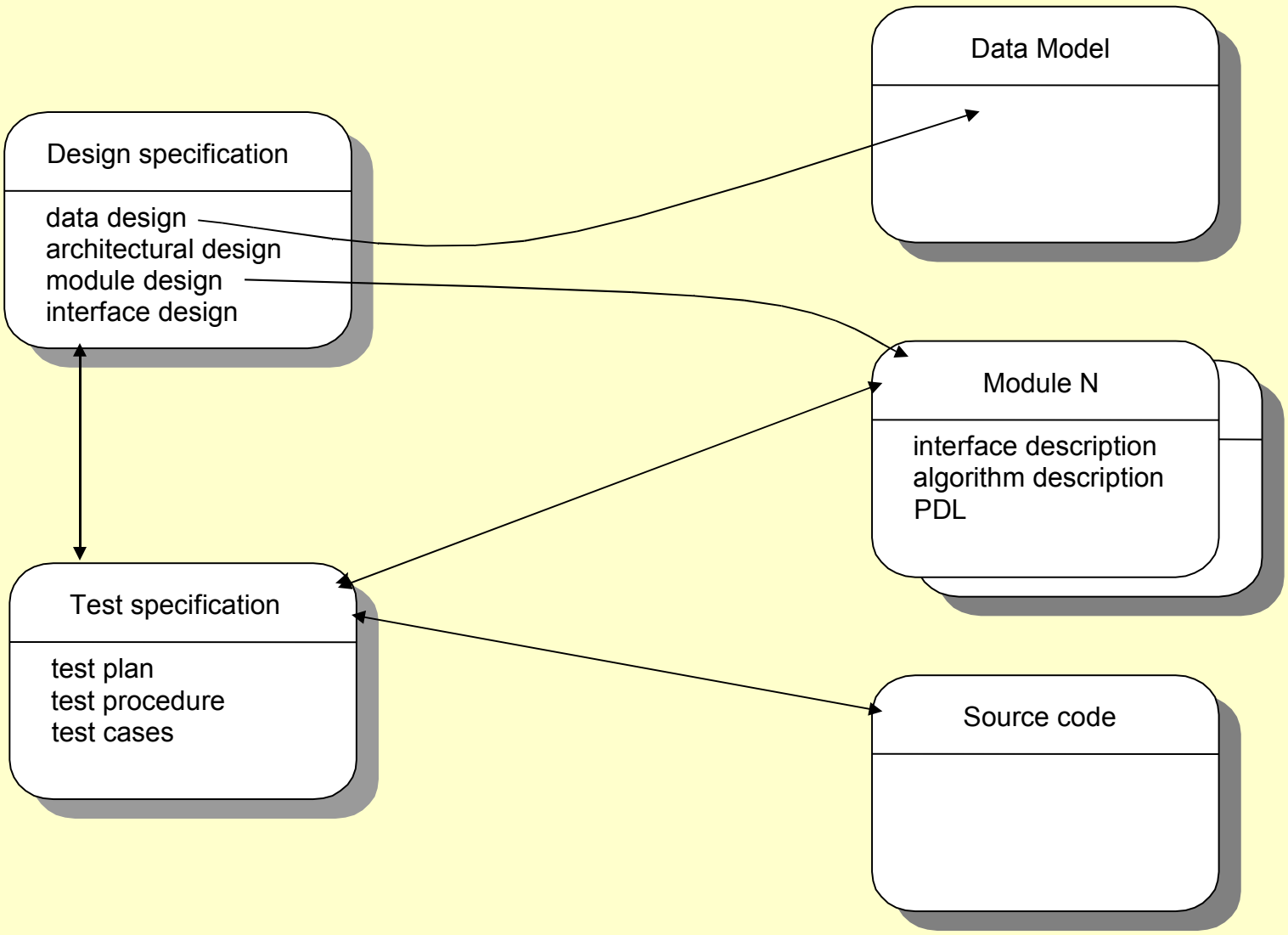
- Object name adalah sebuah string karakter yang mengidentifikasi object secara tidak samar.
- Object description adalah sebuah list item2 data yang mengidentifikasi:
  - tipe SCI (misal dokumen, program, data) yang direpresentasikan oleh object;
  - suatu project identifier; dan informasi perubahan dan/atau versi.
- Resources adalah entitas yang disediakan, diproses, diacu atau sebaliknya diperlukan oleh object.
- Realisasi adalah sebuah pointer pada “unit of text” untuk suatu basic object dan null untuk suatu aggregate object.

10.5.3 Hubungan Antar-Objek



- Configuration object identification juga harus mempertimbangkan hubungan yang ada antara “named objects”
- Sebuah object dapat diidentifikasi sebagai <part-of> suatu aggregate object.
- Hubungan <part-of> menentukan sebuah hirarki object[2].
- Hubungan di antara object[2] dalam suatu hirarki objek tidak hanya sepanjang direct path dari hirarchical tree, tetapi dalam beberapa hal, object[2] diinterrelasikan melewati cabang[2] dari object hirarchy.
- Interrelationships antara configuration objects dapat direpresentasikan dengan sebuah module interconection language (MIL).
- MIL menggambarkan interdependencies di antara configuration objects dan memungkinkan suatu versi dari suatu sistem dikonstruksi secara otomatis.

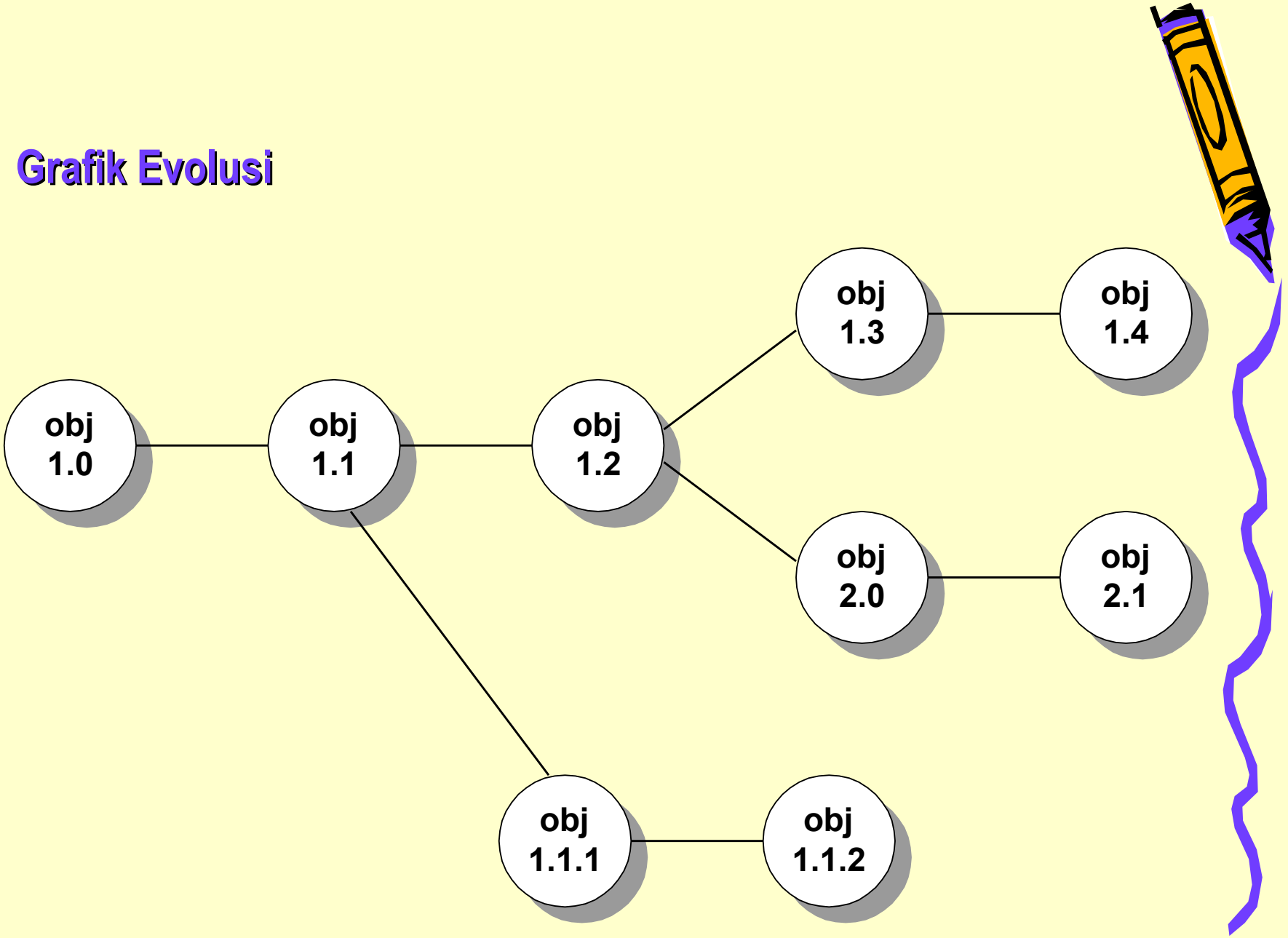
Gambar : Configuration Objects



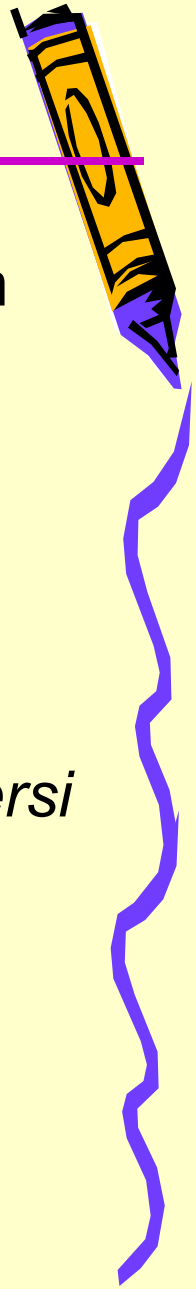
10.5.4 Evolusi Objek

- Skema identifikasi untuk objek[2] perangkat lunak harus mengenali bahwa objek[2] berevolusi selama proses perangkat lunak.
- Sebelum suatu objek dijadikan baseline, dia boleh berubah berkali-kali, dan bahkan setelah suatu baseline ditetapkan, perubahan[2] mungkin cukup sering.
- Dimungkinkan untuk menciptakan sebuah evolution graph untuk suatu object. Evolution graph menggambarkan sejarah perubahan dari object (lihat gbr).
- Dimungkinkan juga perubahan[2] dapat dilakukan pada sembarang versi, tetapi tidak perlu pada semua versi.

Grafik Evolusi



10.6 Kontrol Versi (Version Control)



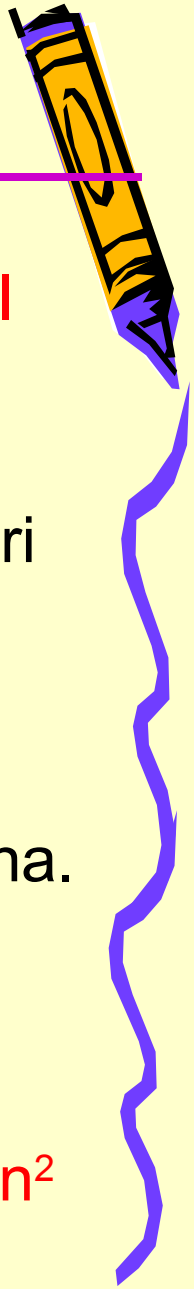
- Clemm[CLE89] menggambarkan version control dalam konteks SCM sbb:

  Configuration management mengijinkan seorang user untuk menentukan konfigurasi alternatif dari sistem software melalui pemilihan versi2 yang sesuai. Hal ini didukung oleh atribut2 yang terkait dengan masing2 versi software, dan kemudian juga mengijinkan suatu konfigurasi ditentukan (dan dikonstruksi) dengan menggambarkan serangkaian atribut yang diinginkan.

10.6.1 Versi PL

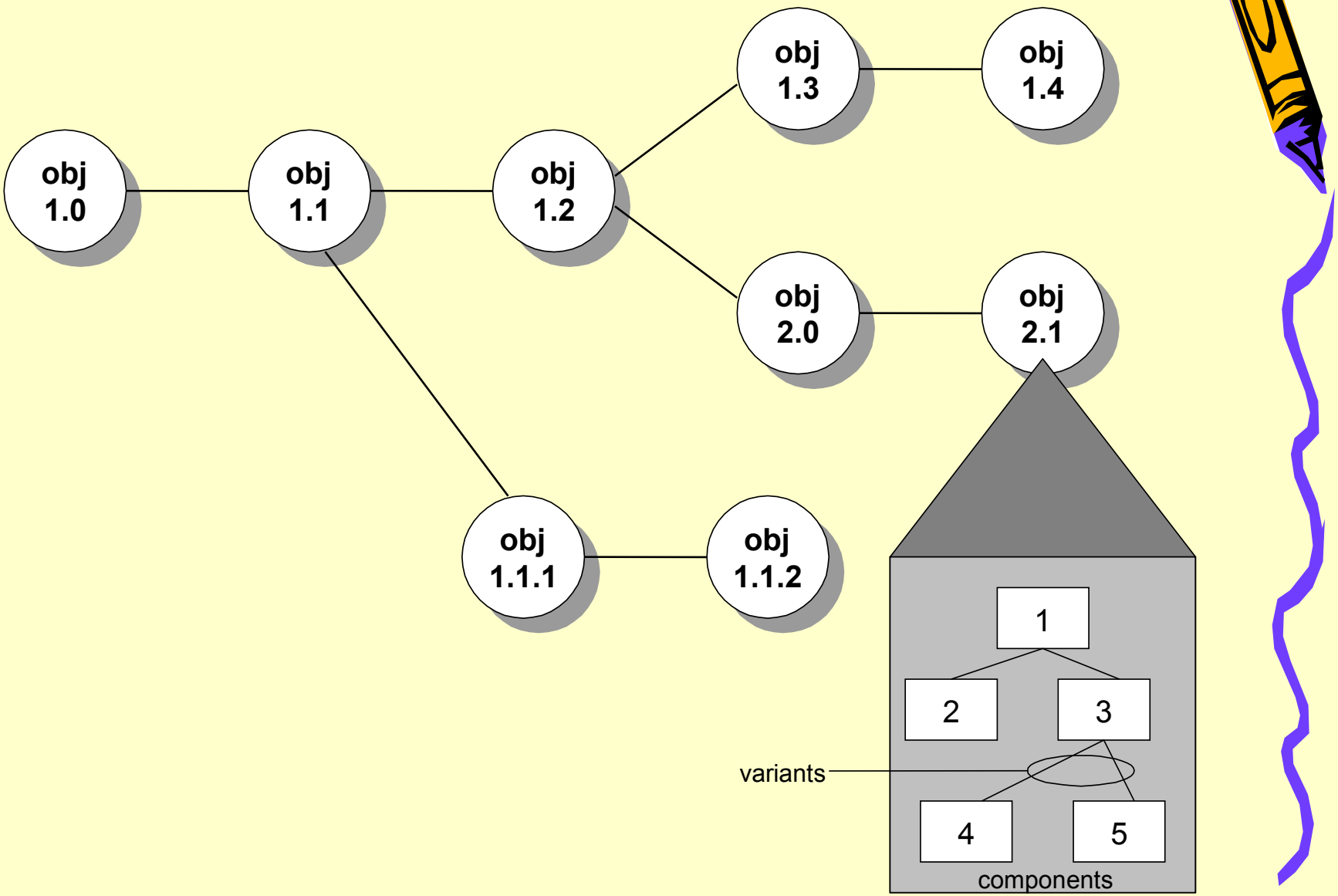
- Atribut[2] yang dikemukakan di atas dapat sesederhana seperti hanya sebuah nomor versi tertentu yang terkait pada masing[2] objek atau sekompleks seperti suatu string dari variable boolean (switches) yang menunjukkan tipe tertentu dari perubahan[2] fungsional yang telah dilakukan pada sistem.
- Salah satu representasi dari versi[2] yang berlainan dari suatu sistem adalah evolution graph (lihat gbr).
- Masing[2] node pada graph tersebut adalah sebuah aggregate object, yaitu satu versi lengkap (utuh) dari perangkat lunak.

10.6.2 Komponen

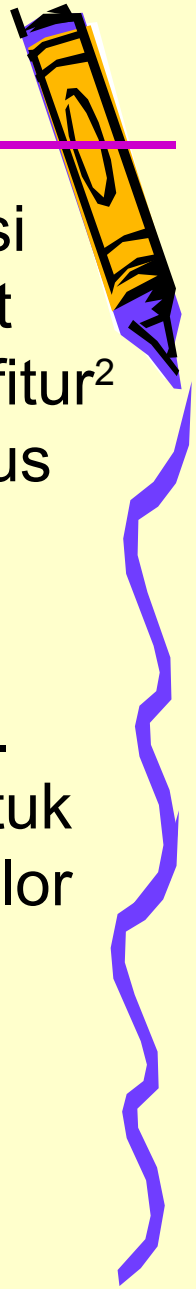


- Masing[2] versi perangkat lunak adalah sekumpulan SCI (source code, dokumen[2], data) dan setiap versi dapat terdiri dari variant[2] yang berbeda.
- Untuk melukiskan konsep ini, pikirkan sebuah versi dari sebuah program sederhana yang tersusun dari komponen[2] 1, 2, 3, 4, dan 5 (lihat gbr).
- Komponen 4 hanya dipakai bila software diimplementasikan dengan menggunakan display warna. Komponen 5 diimplementasikan bila tampilan yang dipakai monochrome.
- Oleh karena itu, dua variant dari versi tersebut dapat ditentukan: (1) komponen[2] 1, 2, 3, dan 4; (2) komponen[2] 1, 2, 3, dan 5.

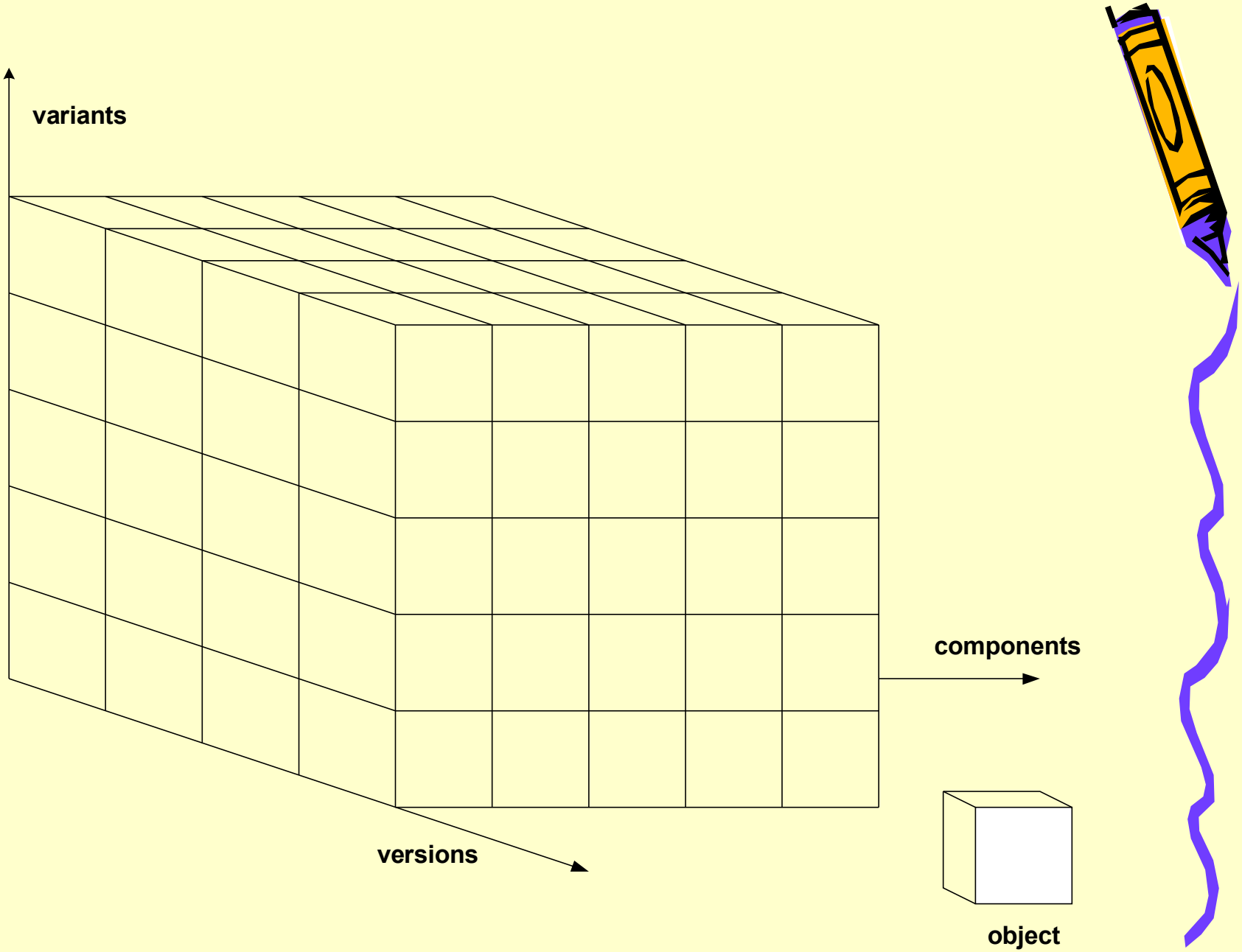
Grafik Evolusi Revisi Perangkat Lunak



10.6.3 Varian



- Untuk membangun varian yang sesuai dari suatu versi tertentu dari suatu program, masing[2] komponen dapat diberikan suatu “attribute-tuple”, yaitu sebuah lis dari fitur[2] yang akan menentukan suatu komponen tertentu harus dipakai bila suatu varian tertentu dari suatu versi perangkat lunak dibuat.
- Satu atau lebih atribut diberikan untuk masing[2] varian. Sebagai contoh, suatu atribut ‘color’ dapat dipakai untuk menentukan komponen yang harus disertakan bila color display yang harus didukung.
- Cara lain untuk mengkonseptualisir hubungan antara komponen, varian[2], dan versi[2] (revisi[2]) adalah merepresentasikannya sebagai “object pool”


variants
components
versions
object

10.6.4 Hub Objek Konfigurasi, Komponen, Varian, dan Versi

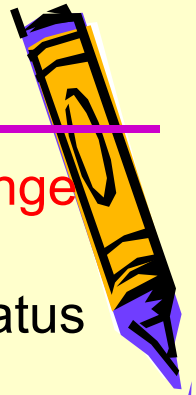
- Hubungan antara configuration object dan komponen^2, varian^2, dan versi^2 dapat direpresentasikan sebagai sebuah ruang tiga dimensi.
- Sebuah versi dapat dipilih dari beberapa varian yang ada (sb vertikal) yang terdiri dari beberapa komponen.
- Sebuah komponen tersusun dari sekumpulan objek^2 pada tingkat level revisi yang sama.
- Sebuah varian adalah sekumpulan objek^2 yang berbeda pada tingkat revisi yang sama, dan oleh karena itu berada berdampingan secara paralel dengan variant^2 lain.
- Sebuah versi baru ditentukan bila perubahan^2 besar (utama) dilakukan pada satu atau lebih object.

KEMBALI

10.7 Kontrol Perubahan

- Untuk proyek pengembangan perangkat lunak yang besar, perubahan yang tidak terkontrol membawa pada kekacauan (chaos).
- Change control menggabungkan prosedur manusia dan tool otomatis guna memberikan sebuah mekanisme untuk mengontrol perubahan.
- Suatu change request di submit dan dievaluasi untuk menilai manfaat teknisnya, efek samping yang potensial, dampak keseluruhan pada configuration objects dan fungsi[2] sistem lainnya.

10.7.1 Proses Kontrol Perubahan



- Hasil dari evaluasi tersebut dipresentasikan sebagai sebuah Change Report yang dipakai oleh Change Control Authority (CCA), yaitu seseorang atau grup yang membuat keputusan akhir terhadap status dan prioritas dari perubahan tersebut.
- Sebuah Engineering Change Order (ECO) dibuat untuk setiap perubahan yang telah disetujui.
- ECO menjelaskan perubahan^2 yang harus dibuat, kendala^2 yang harus diperhatikan, dan kriteria^2 untuk review & audit.
- Objek yang akan diubah di'checked out' dari basis data proyek, dilakukan perubahan, dan kegiatan^2 SQA yang bersesuaian dilaksanakan.
- Objek tersebut kemudian di'checked-in' ke basis data dan mekanisme version control yang sesuai dipakai untuk menciptakan versi berikutnya dari perangkat lunak tersebut.
- Proses check-in dan check-out mengimplementasikan dua elemen penting dari kontrol perubahan, yaitu access control & synchronization control.

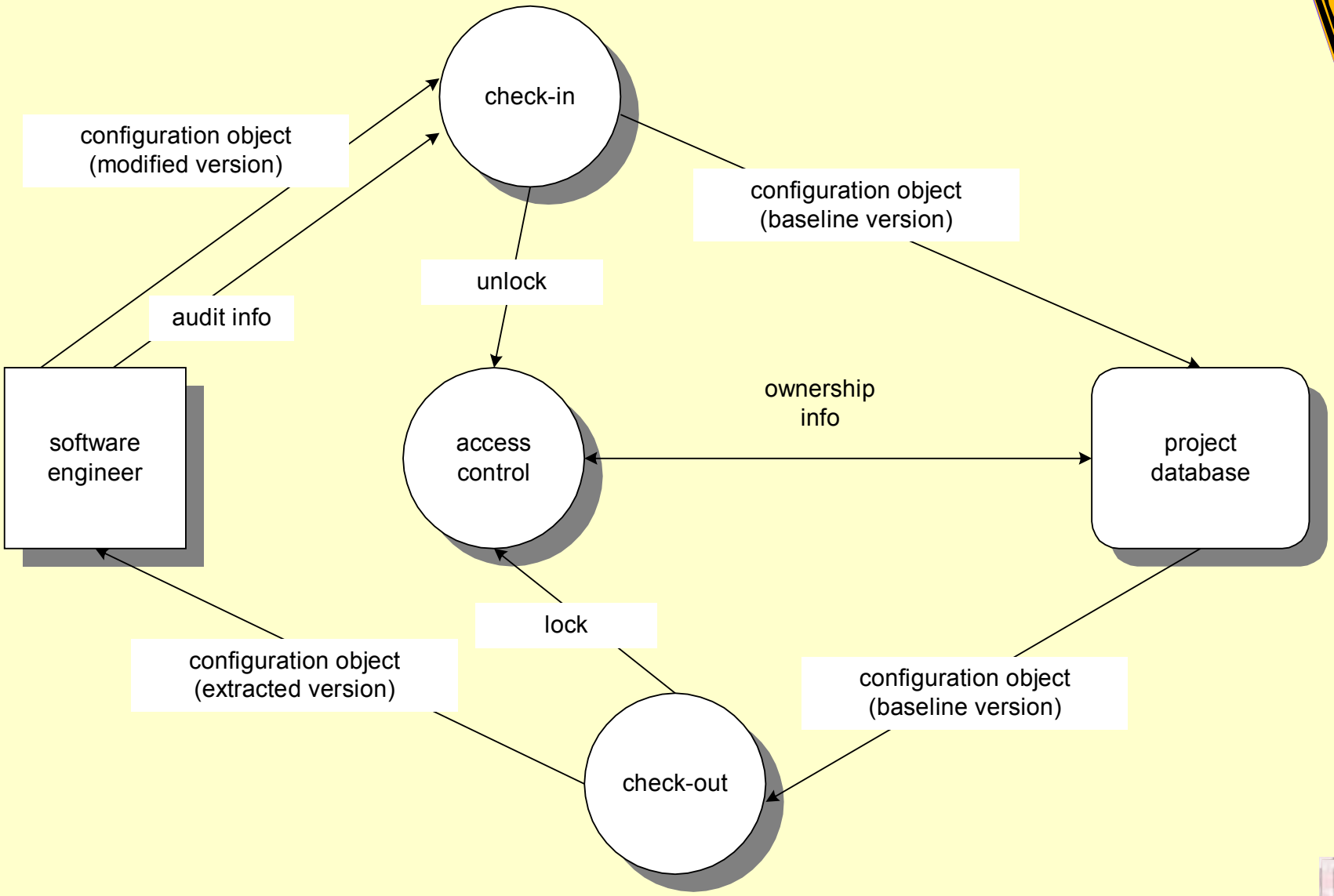
10.7.2 Kontrol Akses & Sinkronisasi

- Kontrol akses mengatur perekayasa perangkat lunak yang mempunyai otoritas untuk mengakses dan memodifikasi suatu configuration object tertentu (khusus).

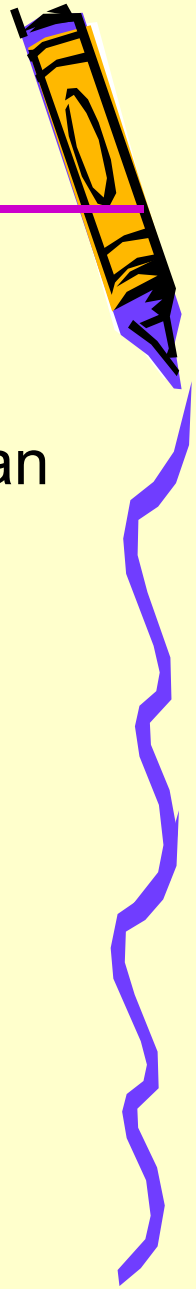
- Kontrol sinkronisasi membantu untuk memastikan bahwa paralel changes yang dilakukan oleh dua orang yang berbeda tidak saling overwrite satu terhadap lainnya.

- Aliran kontrol akses & sinkronisasi dilukiskan secara skematik pada gbr berikut.

Change Control

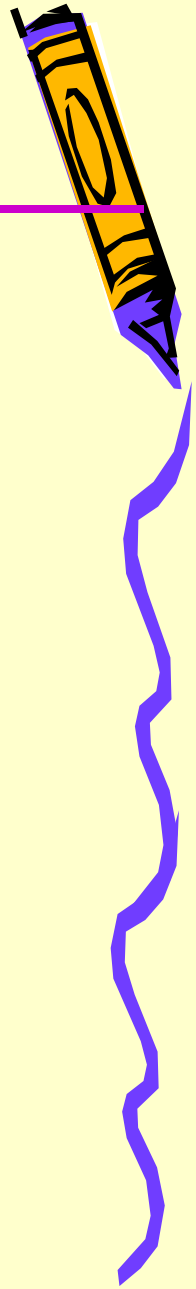


10.8 Audit Konfigurasi



- Identifikasi, kontrol versi, dan kontrol perubahan membantu pengembang perangkat lunak untuk mempertahankan aturan, bila tidak akan mendatangkan situasi yang chaotic & fluid.
- Walaupun demikian, bahkan mekanisme kontrol yang berhasil mentrack perubahan hanya sampai ECO dibangkitkan.
- Bagaimana kita dapat menjamin bahwa perubahan tersebut telah diimplementasikan dengan benar? Jawabnya adalah dua hal: (1) FTR, dan (2) software configuration audit.

10.8.1 Proses Audit Konfigurasi



- FTR memusatkan pada ketepatan (correctness) teknis dari configuration object yang telah dimodifikasi.
- Para reviewer menilai SCI tersebut untuk menentukan konsistensi terhadap SCI[2] lain, penghilangan, dan potensial side effects. FTR harus dilaksanakan untuk semua perubahan termasuk yang paling sepele.
- Software configuration audit melengkapi (menyempurnakan) FTR dengan penilaian suatu configuration object untuk karakteristik[2] yang pada umumnya tidak dipertimbangkan selama review.

10.8.2 Pertanyaan dalam Proses Audit Konfigurasi

- Audit tersebut menanyakan dan menjawab pertanyaan[2] berikut:
  1. Apakah perubahan yang ditentukan dalam ECO telah dilakukan? Apakah suatu modifikasi tambahan telah disertakan?
  2. Apakah FTR telah dilakukan untuk menilai ketepatan teknis?
  3. Apakah standard[2] software engineering telah diikuti dengan benar?
  4. Apakah perubahan tersebut telah ditandai dalam SCI? Apakah tanggal perubahan dan orang yang membuat perubahan telah dicatat? Apakah atribut[2] configuration object mencerminkan perubahan tersebut?
  5. Apakah prosedur[2] SCM untuk pencatatan perubahan, perekaman, dan pelaporan telah diikuti?
  6. Apakah semua SCI yang terkait telah diupdate dengan benar?

KEMBALI

10.8.3 Pelaporan Status Konfigurasi (Status Accounting)

- Setiap kali suatu SCI diberi identifikasi baru atau diupdate, sebuah Configuration Status Reporting CSR entry dibuat.
- Setiap kali suatu perubahan disetujui oleh CCA (yaitu, suatu ECO dikeluarkan), sebuah entry CSR dibuat.
- Setiap kali suatu configuration audit dilakukan, hasil²nya dilaporkan sebagai bagian dari task CSR.
- Output dari CSR dapat diletakkan dalam basis data on-line shg pengembang perangkat lunak atau pengelola (maintainers) dapat mengakses informasi perubahan dengan keyword category.

KEMBALI

10.9 SCM Standards

- Beberapa standard SCM awal, seperti MIL-STD-483, DOD-STD-480A, dan MIL-STD-1521A, memusatkan pada software yang dikembangkan untuk aplikasi militer.
- Standar ANSI/IEEE yang lebih baru, seperti ANSI/IEEE Std. No. 828 - 1983, Std. No. 1042 - 1987, dan Std. No. 1028 - 1988, dapat dipakai untuk software komersial dan direkomendasikan baik untuk organisasi[2] rekayasa perangkat lunak besar & kecil.

***