Pertemuan 2

SOFTWARE DEVELOPMENT LIFE CYCLE (SDLC)

POKOK BAHASAN

- Biaya PL
- Software Quality Attribute
- Standar kualitas
- Takaran Jaminan Kualitas
- CASE TOOLS
- Siklus Hidup Perangkat Lunak (SWDLC/Software Development Life Cycle)

BIAYA PERANGKAT LUNAK (SOFTWARE COST)

- Terkadang mendominasi biaya sistem secara keseluruhan
- Biaya terbesar untuk perangkat lunak terletak pada proses perawatan (maintenance) dibanding biaya pembuatannya (develop)
- Biaya pengadaan perangkat lunak yang di pasang pada PC sering lebih besar dibandingkan dengan harga perangkat kerasnya kec. Di negara-negara yang tidak menghargai HAKI
- Biaya perangkat lunak secara kasar sebesar 60% dari biaya untuk pembangunan dan 40% untuk pengujian
- Secara umum besarnya biaya bervariasi tergantung pada tipe sistem yang dibangun dan kebutuhan sistem seperti kinerja dan kehandalan sistem
- Biaya distribusi tergantung pada model pembangunan yang digunakan

SOFTWARE QUALITY ATTRIBUTE (1)

Ciri-ciri kualitas menurut lembaga penjamin mutu PL (ISO, ANSI, IEEE, dll):

- Correctness (kebenaran)
- Reliability (tahan uji)
- User Friendliness
- Maintenatibility (mudah dirawat)
- Portability (mudah di distribusikan)

UKURAN JAMINAN KUALITAS (1)

- **Ukuran membangun (constructive measures)**
  - Aplikasi yg konsisten pada metode di seluruh fase proses pembangunan
  - Penggunaan perlatan/ tools yang memadai
  - Pembangunan PL pd basis kualitas yg tinggi di akhir tahapan
  - Perawatan yang konsisten pada dokumenasi pengembangan
- **Ukuran analitik (analytical measures)**
  - Analisis program yang statis
  - Analisis program yang dinamis
  - Pemeilihan test case yang sistematis
  - Pencatatan yang konsisten pada analisis produk

UKURAN JAMINAN KUALITAS (2)

- **Ukuran Organisasi (Organization Measures)**
  - Pengalaman pengembang (developer) dalam mempelajari strategi dan teknik yang tepat dalam membangun PL

KRISIS PERANGKAT LUNAK

- Masalah yang muncul:
  - Estimasi jadwal dan biaya yang seringkali tidak tepat
  - Produktivitas orang-orang software yang tidak dapat mengimbangi permintaan software
  - Kualitas software yang kurang baik.

- Kurangnya pengetahuan tentang:
  - Bagaimana mengembangkan software
  - Bagaimana memelihara software yang ada, yang berkembang dalam jumlah besar
  - Bagaimana mengimbangi permintaan software yang makin besar.

KODE ETIK PROFESI

- Konfidensialitas (menghormati klien)
- Tidak boleh menerima pekerjaan di luar
- kompetensinya
- Hak kekayaan intelektual (HaKI)
- Penyalahgunaan komputer, hack, crack,

KODE ETIK INTERNASIONAL

- Digagas oleh masyarakat profesional di Amerika (1999) yang tergabung dalam ACM/IEEE
- Makna yang terkandung:
  - Prinsip-prinsip kesepakatan yang dihubungkan dengan tingkah laku dan keputusan yang dibuat oleh para ahli profesional
  - Masyarakat profesional: praktisi, pengajar, manajer, supervisor, pengambil kebijakan.

CASE TOOLS

- CASE (Computer Aided Software Engineering)
  - Suatu peralatan baik HW maupun SW komputer yang digunakan untuk menyediakan pendukung otomatis dalam aktivitas pembangunan PL.
  - Tujuan
    meningkatkan produktivitas dalam proses pembangunan PL secara signifikan

- Dikelompokkan dalam 2 kategori:
  1. Upper-CASE
     Mendukung aktivitas proses pembangunan tahap awal (tahap analisis kebutuhan dan desain)
  2. Lower-CASE
     Mendukung aktivitas pembangunan di tahap akhir programming, debuging, dan testing)

CASE TOOLS (2)

Penggunaan CASE tools:

- Graphical Editors
- Data Dictionaries
- GUI Builders
- Debugger
- Automated Translators
- Compilator Integrated
- Instalator Kit

SOFTWARE DEVELOPMENT LIFE CYCLE (SDLC)

Proses Generik

- **Spesifikasi**

  Apa yang harus dilakukan oleh perangkat lunak dan batasan/kendala pengembangannya

- **Pengembangan**

  Proses memproduksi sistem perangkat lunak

- **Validasi**

  Pengujian perangkat lunak terhadap keinginan pengguna

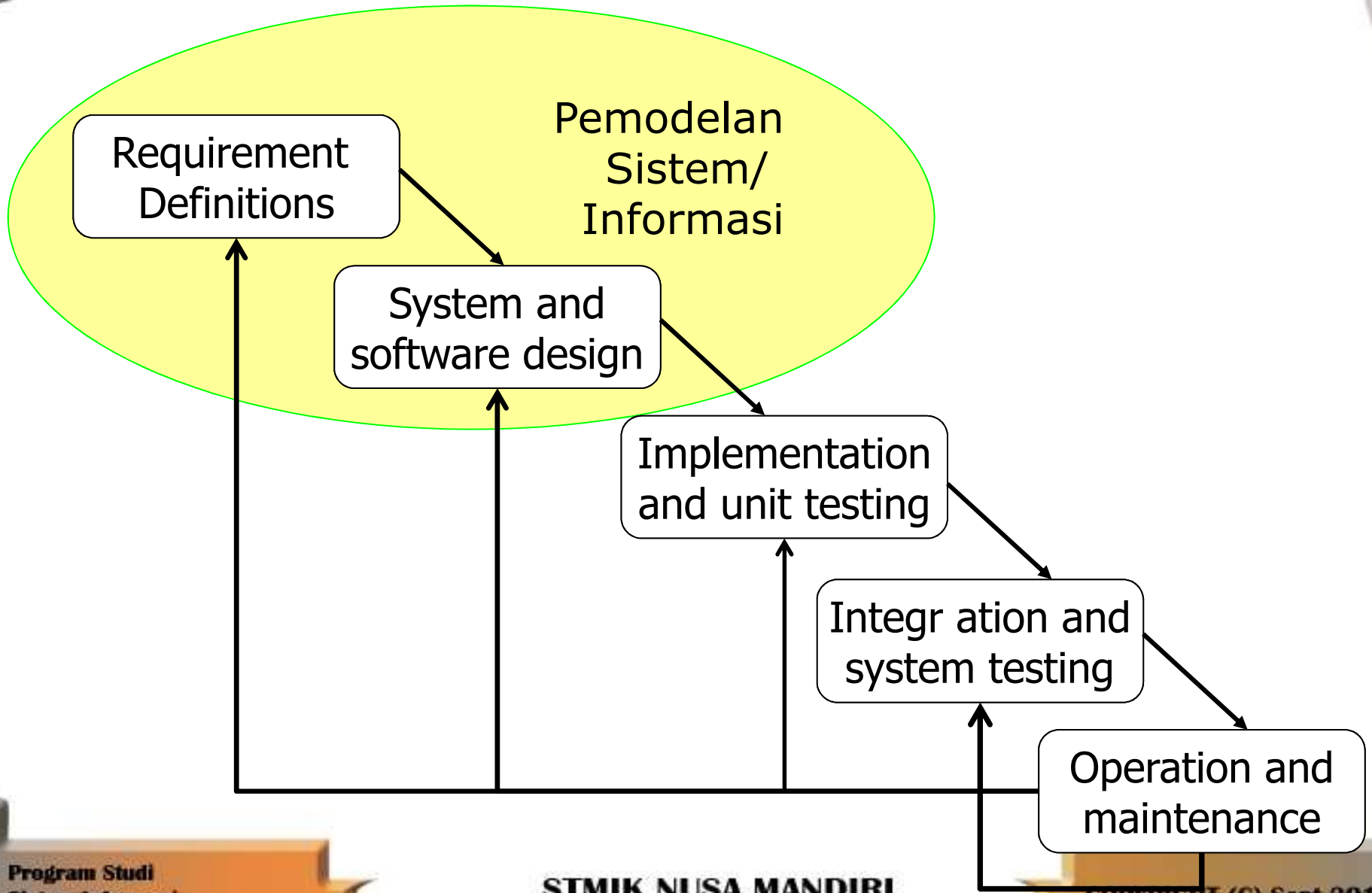
- **Evolusi**

  Perubahan perangkat lunak berdasarkan perubahan keinginan.

MODEL PROSES RPL

- Model Waterfall,
- Model Prototyping,
- Model Evolutionary
- Model Spiral
- Reuse Based Development

WATERFALL MODEL



WATERFALL MODEL (2)

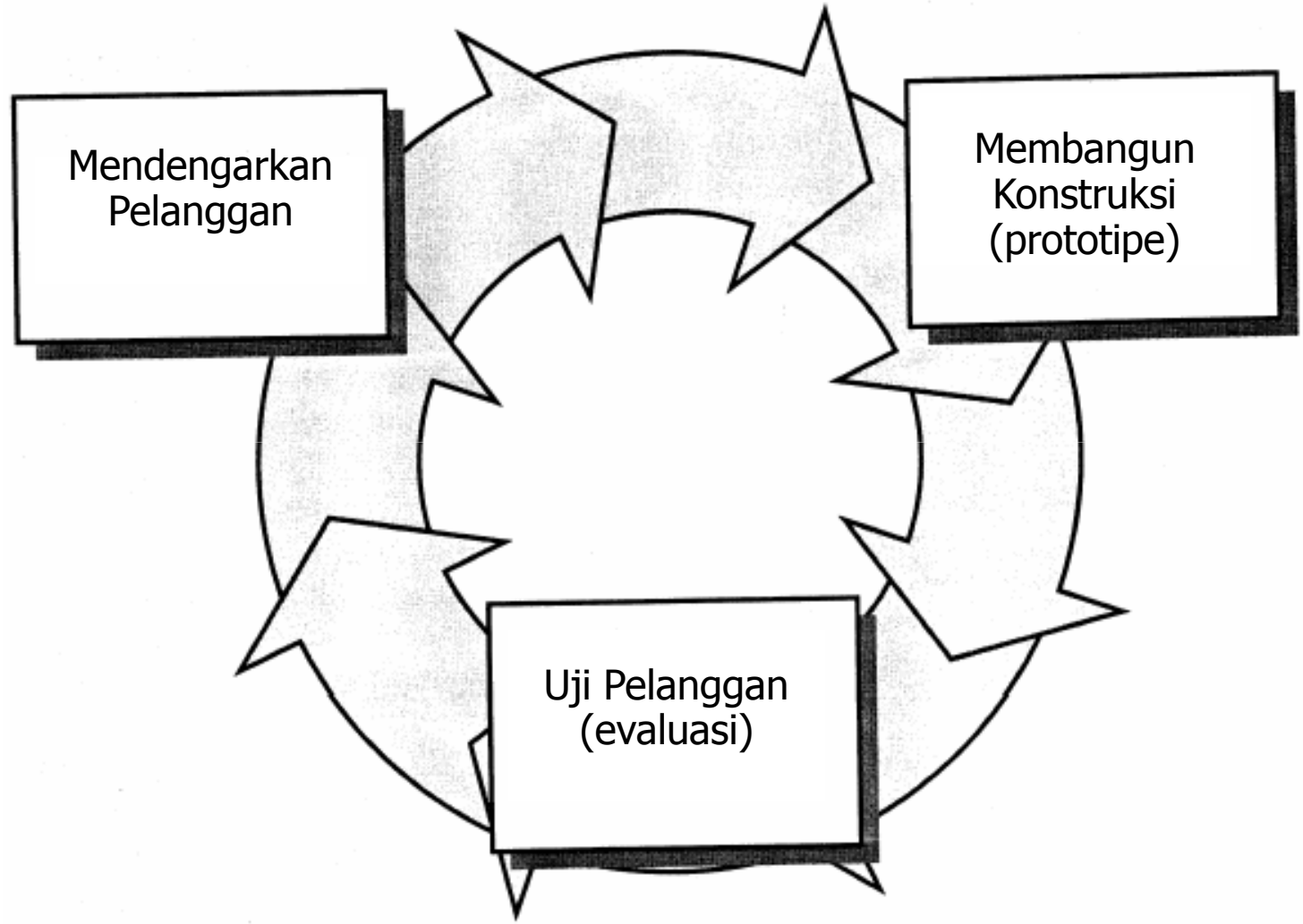
- Requirements Analysis And Definition
- System And Software Design
- Implementation And Unit Testing
- Integration And System Testing
- Operation And Maintenance

WATERFALL MODEL (3)

Problems Model Waterfall

1. Jarang sekali proyek yang prosesnya bisa dilakukan secara sequencial.
2. Sukar bagi customer untuk secara explisit mengemukakan semua kebutuhannya.
3. Customer harus sabar.
4. Developer sering menunda pekerjaan. Anggota tim harus menunggu anggota lainnya
5. menyelesaikan tugasnya.

PROTOTYPE MODEL



PROTOTYPE MODEL (2)

- Prototype Paradigm dimulai dengan mengumpulkan kebutuhan-kebutuhan customer.
- Developer dan customer bertemu dan mendefinisikan obyektif software secara menyeluruh, mengidentifikasi kebutuhan-kebutuhan yang diketahui dari area pekerjaan.
- Setelah itu dibuat Quick Design.
- Quick Design difokuskan pada representasi aspek software yang bisa dilihat customer/user (misal: format input dan output).
- Quick Design cenderung ke pembuatan prototipe.
- Prototipe dievaluasi customer/user dan digunakan untuk menyempurnakan kebutuhan software yang akan dikembangkan.

PROTOTYPE MODEL (2)

- Sering terjadi customer menjabarkan objektif umum mengenai software yang diminta, tetapi tidak bisa mendefinisakan input, proses, output yang diminta secara detail.
- Disisi lain, developer menjadi tidak yakin terhadap efisiensi algoritma, kemampuan adaptasi terhadap sistem operasi, atau bentuk interaksi mesin dengan orang.
- Untuk mengatasi situasi tersebut, bisa digunakan pendekatan Prototype Paradigm.

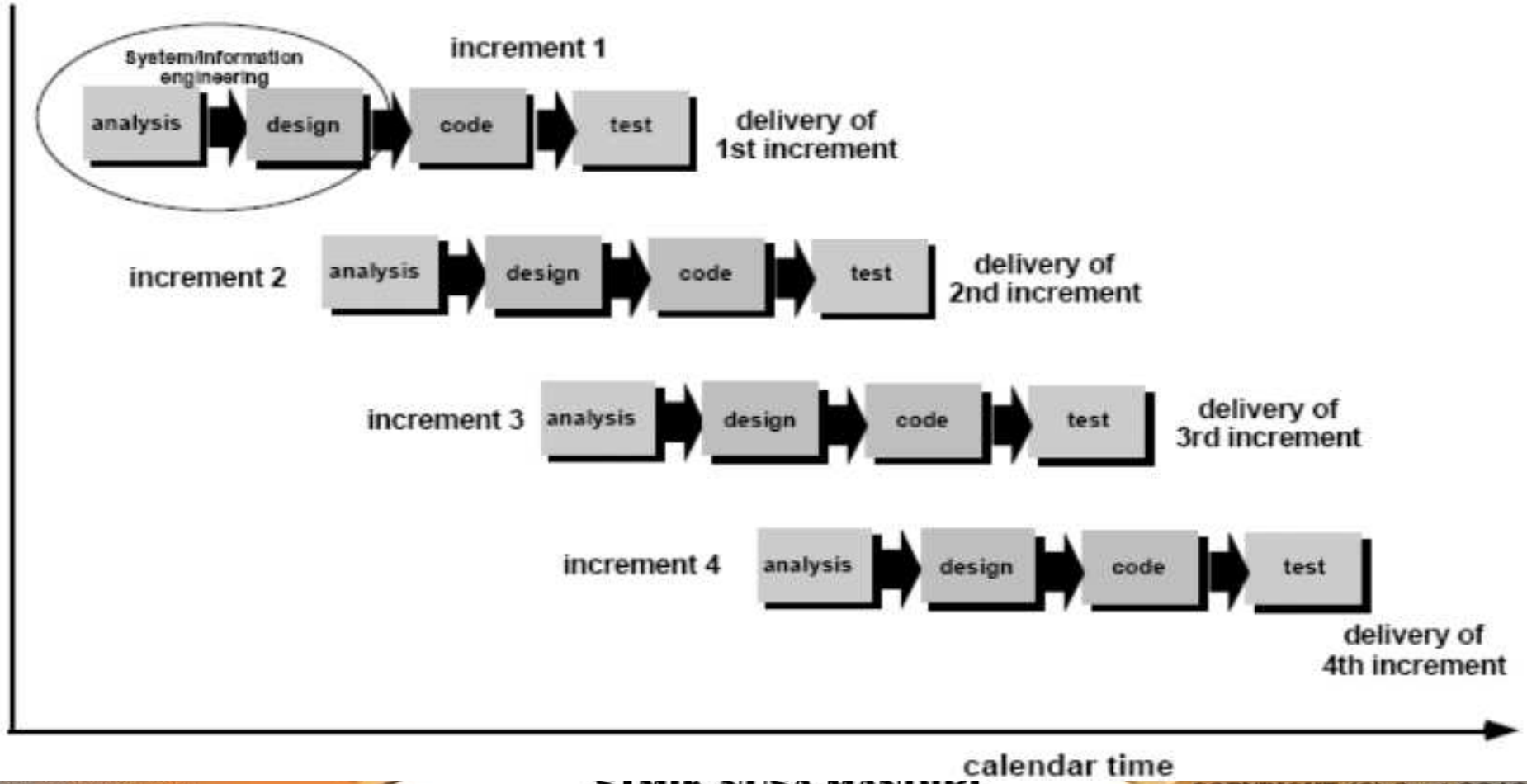
PROTOTYPE MODEL (3)

Problems Prototyping Model

- Customer melihat prototipe tersebut sebagai versi dari software.
  - Pada saat produk tersebut harus dibangun ulang supaya level kualitas bisa terjamin,
  - Customer akan mengeluh dan meminta sedikit perubahan saja supaya prototipe tersebut bisa berjalan.
- Development membuat implemetasi yang kompromitas dengan tujuan untuk memperoleh prototipe pekerjaan secara cepat.
  - Dampaknya adalah sistem operasi atau bahasa pemrograman yang dipergunakan tidak tepat, algoritma tidak efisien.

EVOLUTIONARY MODEL

Evolutionary Model (Incremental)



EVOLUTIONARY MODEL INCREMENTAL (2)

Penjelasan :

1. Kombinasikan elemet-element dari waterfall dengan sifat iterasi/perulangan.
2. Element-element dalam waterfall dikerjakan dengan hasil berupa produk dengan
3. Spesifikasi tertentu, kemudian proses dimulai dari fase pertama hingga akhir dan menghasilkan produk dengan spesifikasi yang lebih lengkap dari yang sebelumnya.
4. Demikian seterusnya hingga semua spesifikasi memenuhi kebutuhan yang ditetapkan oleh pengguna.

EVOLUTIONARY MODEL INCREMENTAL (3)

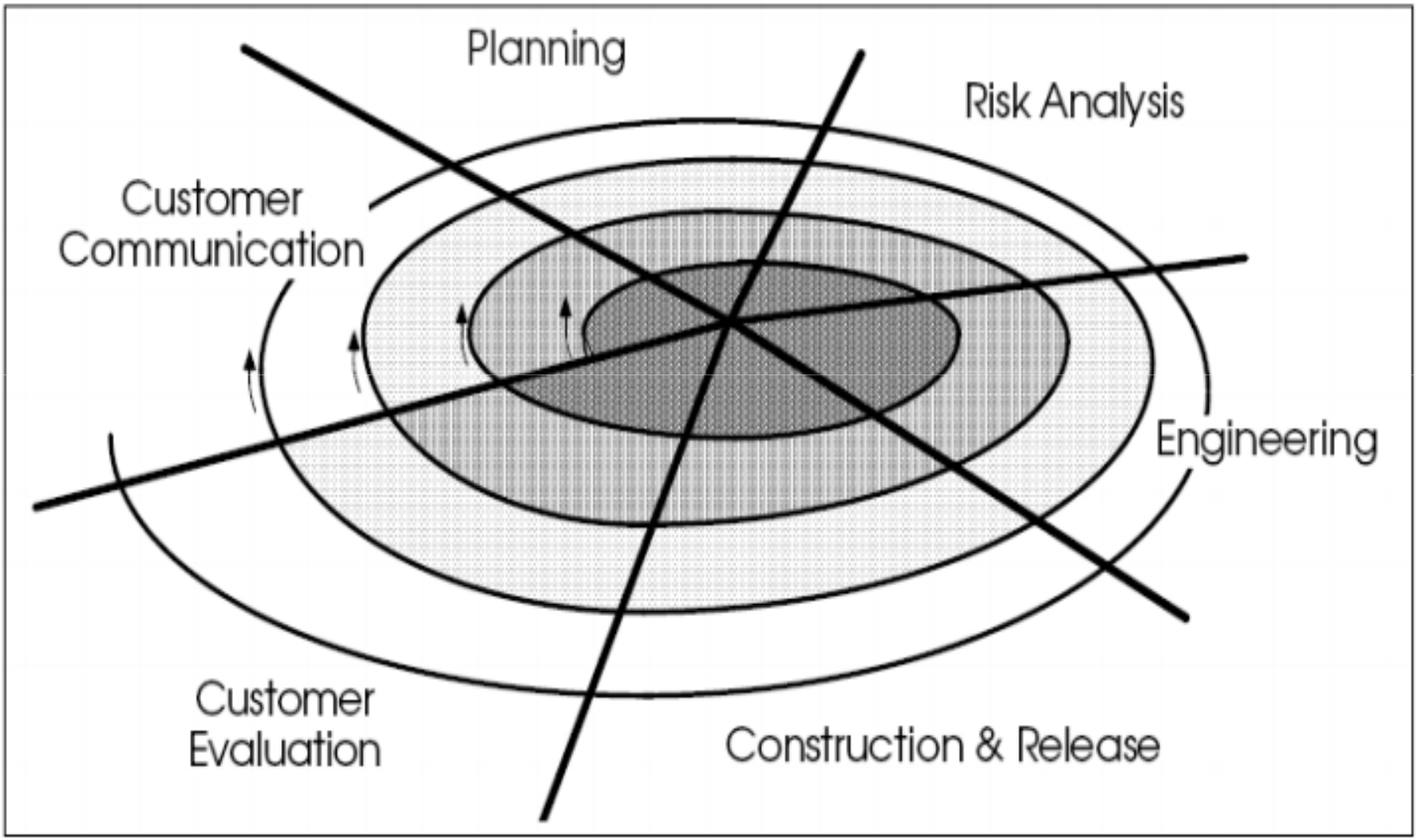
5. Produk hasil increment pertama biasanya produk inti (core product). Produk tersebut digunakan oleh pengguna atau menjalani review/ pengecekan detil. Hasil review tsb menjadi bekal untuk pembangunan pada increment berikutnya, sampai produk yang komplit dihasilkan.
6. Model ini cocok jika jumlah anggota tim pengembang/pembangun PL tidak cukup.
7. Mampu mengakomodasi perubahan secara fleksibel.
8. Produk yang dihasilkan pada increment pertama bukanlah prototype, tapi produk yang sudah bisa berfungsi dengan spesifikasi dasar.

EVOLUTIONARY MODEL INCREMENTAL (4)

Kekurangan Incremental Model:

- Hanya cocok untuk proyek berukuran kecil (tidak lebih dari 200.000 baris coding)
- Mungkin terjadi kesulitan untuk memetakan kebutuhan pengguna ke dalam rencana spesifikasi masing-masing hasil increment

EVOLUTIONARY MODEL SPIRAL



EVOLUTIONARY MODEL SPIRAL (2)

Penjelasan :

- **Customer Comunication**

  Membangun komunikasi yang baik dengan pelanggan

- **Planning**

  Mendefinisikan sumber, batas waktu, informasi-informasi lain seputar proyek

- **Risk Analyst**

  Identifikasi resiko management dan teknis

- **Engineering**

  Pembangunan contoh-contoh aplikasi misalnya prototype

- **Construction and release**

  Pembangunan, test, install dan report

- **Customer Evaluation**

  Mendapatkan feedback dari pengguna berdasarkan evaluasi pada fase engineering dan fase instalasi

EVOLUTIONARY MODEL SPIRAL (3)

- Pada model spiral, resiko sangat dipertimbangkan. Resiko adalah sesuatu yang mungkin mengakibatkan kesalahan.
- Model spiral merupakan pendekatan yang realistik untuk Perangkat Lunak berskala besar.
- Pengguna dan pembangun bisa memahami dengan baik software yang dibangun karena setiap kemajuan yang dicapai selama proses dapat diamati dengan baik. Namun demikian, waktu yang cukup panjang mungkin bukan pilihan bagi pengguna, karena waktu yang lama sama dengan biaya yang lebih besar.

REUSE BASED

A. Software Re-engineering

- **Apakah itu?**

  Restrukturisasi atau menulis ulang sebagian atau keseluruhan dari sistem yang telah ada tanpa merubah fungsionalitasnya.

- **Kapan?**

  Ketika sebagian tetapi tidak semua sub sistem yg besar membutuhkan perawatan yg sering

  Ketika HW dan SW sudah lama hampir tak berfungsi

- **Bagaimana?**

  Sistem bisa di restrukturisasi dan didokumentasi ulang untuk membuat menjadi mudah dalam perawatan

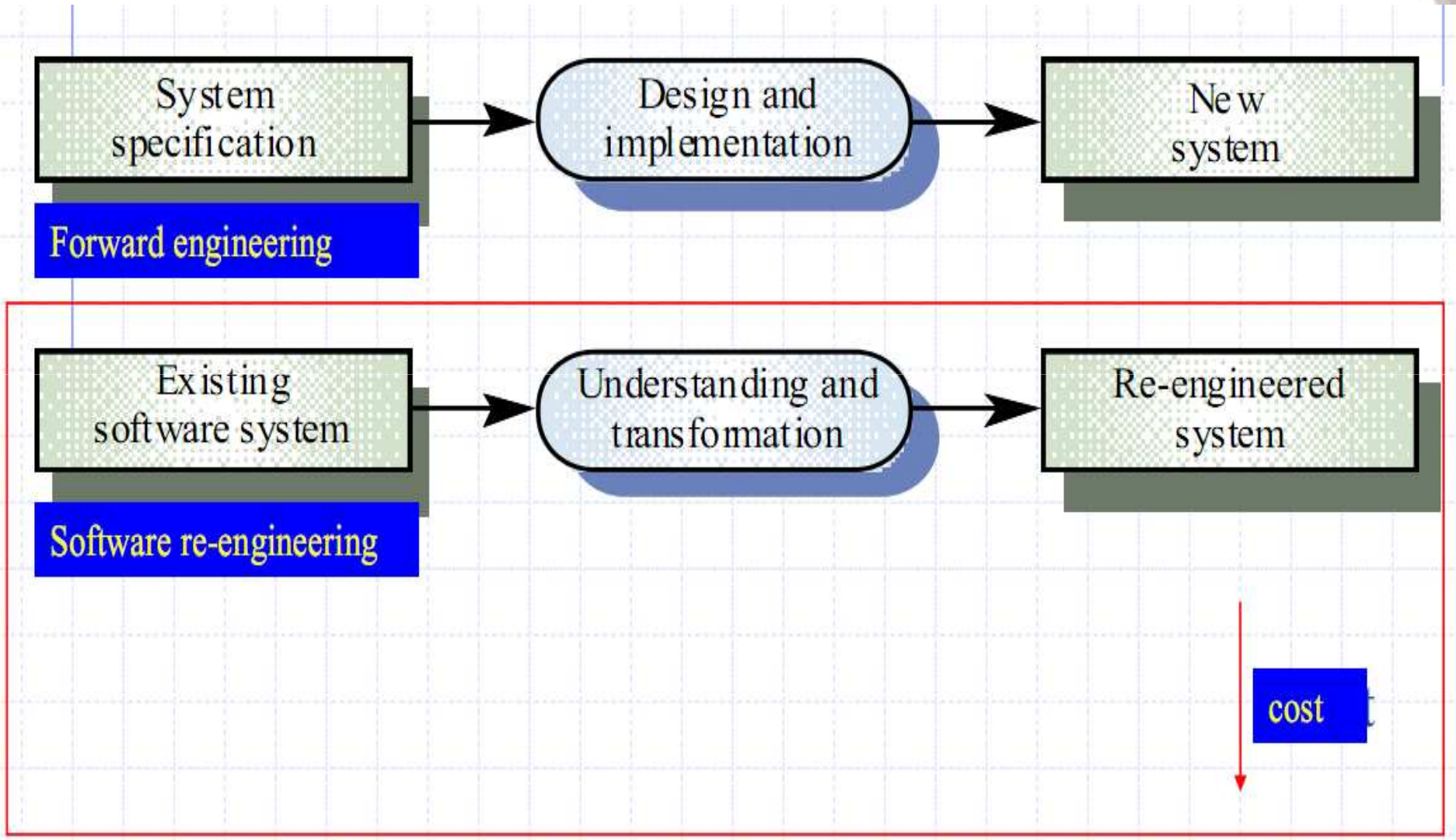
REUSE BASED (2)

- **Software Re-engineering (lanjutan)**
  - Mengapa?
    - Mengurangi resiko

      SW yang baru dibangun membawa resiko yg tinggi
    - Mengurangi biaya

      Biaya untuk re-engineering sering lebih kecil dibanding membangun SW baru.

REUSE BASED (3)



REUSE BASED (3)

B. Reverse Engineering

- Analisis SW kembali dalam tahap pemahaman dlm desain dan spesifikasinya
- Bisa sebagian proses re-engineering atau sebagian spesifikasi sistem untuk diimplementasi ulang
- Membangun database dan bangkitkan program informasi dari proses ini
- Mengapa?
  - ✓ Kode aslinya telah dalam keterbatasan
  - ✓ Perawatan terbentur pada struktur dan program yang rusak sehingga membutuhkan kerja yg sangat keras
  - ✓ Program secara otomatis distrukturisasi ulang untuk menghilangkan beberapa bagian yang tidak beres dalam kondisi yang sangat kompleks.

Tugas: II (10%)

Diskusi Teori Konsep Dasar Perangkat Lunak hasil dan hasil diskusi dibacakan perkelompok dan dikumpulkan