Evolusi Perangkat Lunak

Team Teaching Mata Kuliah Rekayasa Perangkat Lunak
Jurusan Teknologi Informasi

Politeknik Negeri Malang

Perubahan Perangkat Lunak

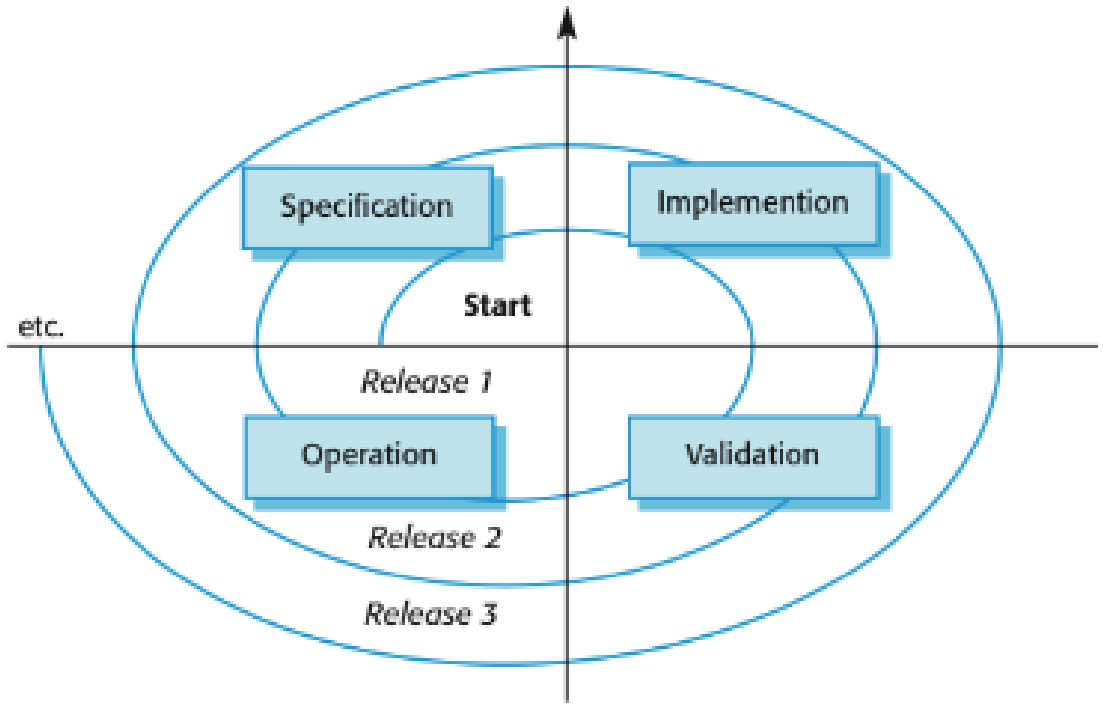
- Alasan mengapa perubahan perangkat lunak tidak dapat dihindari:
  - Adanya kebutuhan baru ketika perangkat lunak digunakan;
  - Lingkungan bisnis berubah;
  - Error/Bug harus diperbaiki;
  - Komputer dan peralatan baru ditambahkan ke sistem;
  - Kinerja atau keandalan sistem mungkin perlu ditingkatkan.
- Masalah utama bagi semua organisasi adalah mengimplementasikan dan mengelola perubahan pada sistem perangkat lunak yang ada.

Pentingnya Perubahan Perangkat Lunak

- Organisasi memiliki **investasi yang besar** pada sistem perangkat lunak mereka .
- Perangkat Lunak adalah aset bisnis yang kritis. Sehingga untuk menjaga PL harus diubah dan diperbarui.
- Sebagian besar **anggaran** perangkat lunak di perusahaan besar dikhususkan untuk mengubah dan mengembangkan perangkat lunak yang sudah ada **daripada mengembangkan perangkat lunak baru**.

Model Spiral

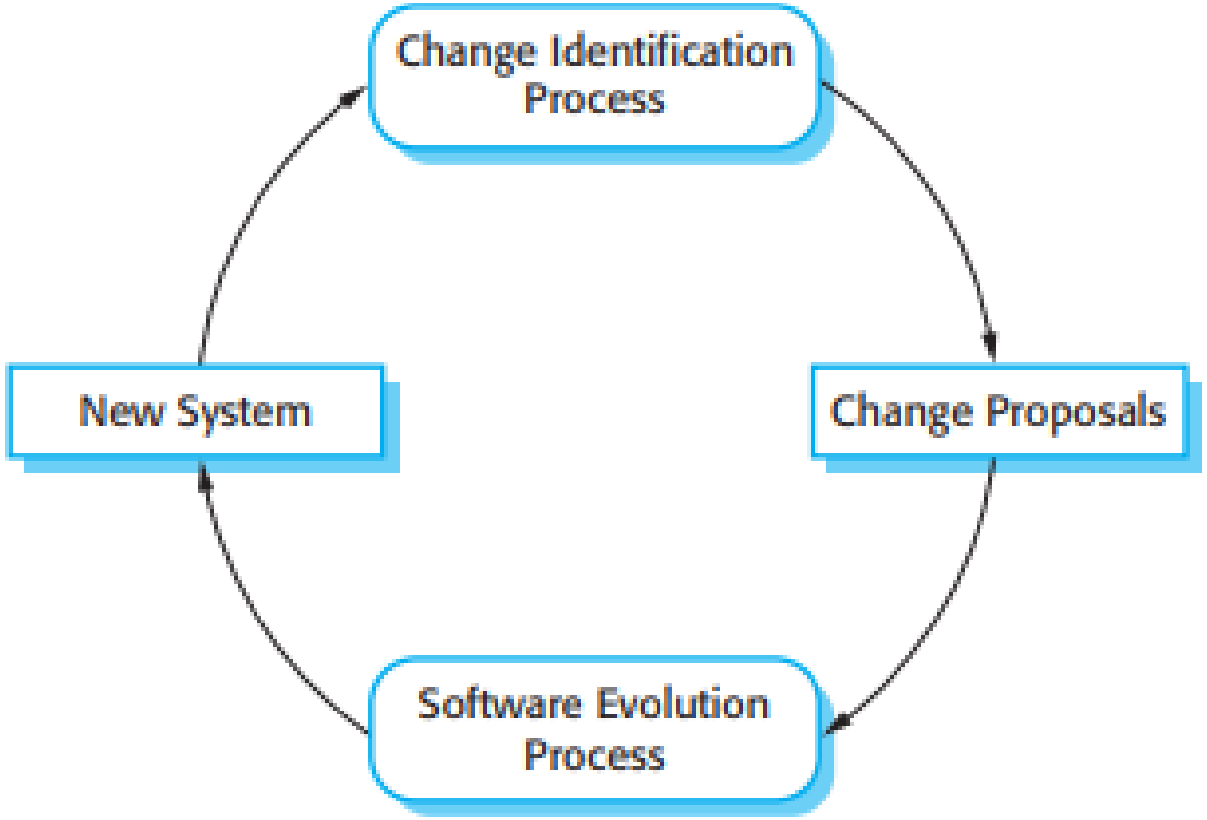
- Model spiral menggambarkan bagaimana sistem perangkat lunak berkembang melalui serangkaian *release* yang berurutan.



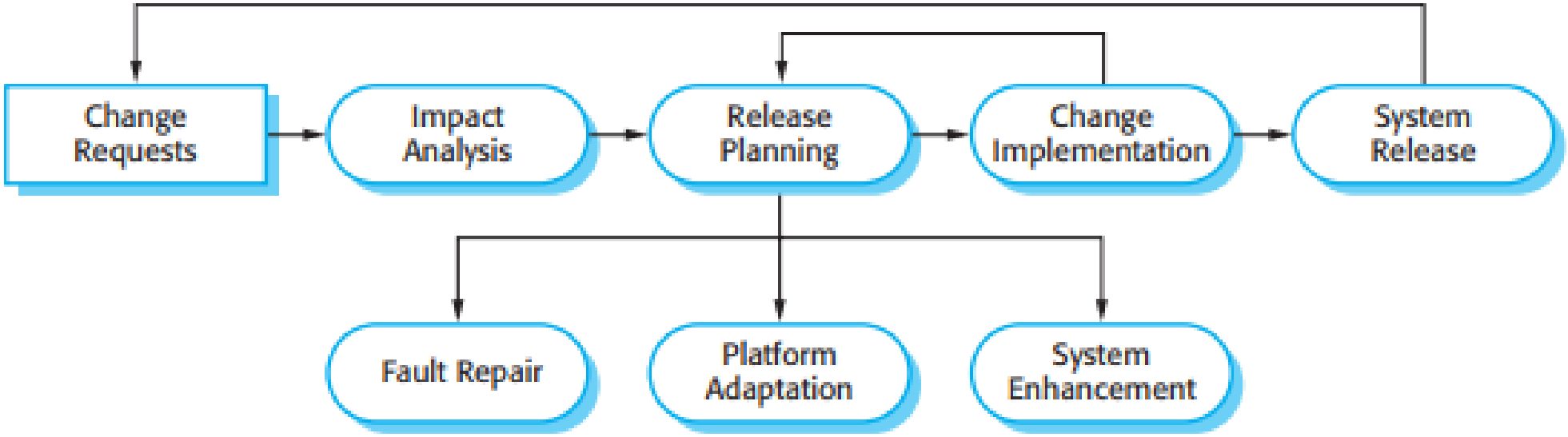
Proses Evolusi

- Proses evolusi perangkat lunak bergantung pada:
  - Jenis perangkat lunak yang sedang dipelihara;
  - Proses pengembangan yang digunakan;
  - Keterampilan dan pengalaman orang-orang yang terlibat.
- Pengajuan perubahan menjadi penggerak evolusi sistem.
  - Berkaitan dengan komponen-komponen yang dipengaruhi oleh perubahan tersebut, sehingga memungkinkan estimasi biaya dan dampak perubahan.
- Identifikasi perubahan dan evolusi berlanjut sepanjang masa sistem.

Proses identifikasi perubahan dan evolusi



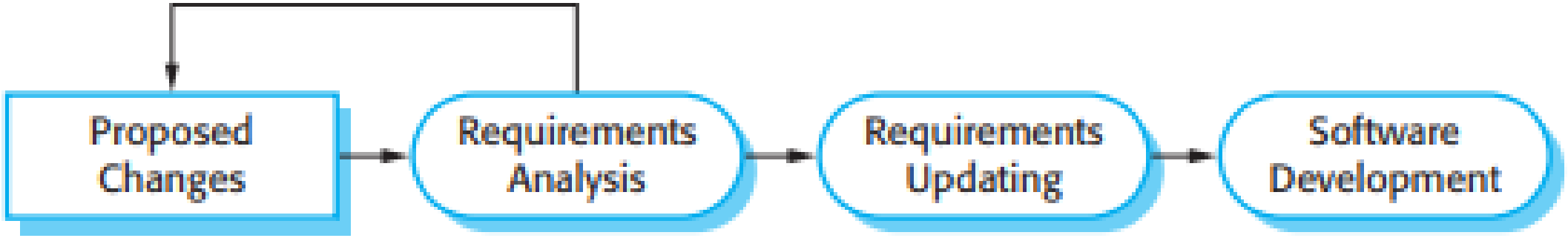
Proses Evolusi Perangkat Lunak



Change implementation

- Implementasi perubahan dapat dilihat sebagai iterasi dari proses pengembangan di mana revisi-revisi pada sistem dirancang, diimplementasikan, dan diuji.
- Tahap pertama dari implementasi perubahan mungkin melibatkan pemahaman program, terutama jika pengembang sistem asli tidak bertanggung jawab atas implementasi perubahan.
- Selama fase pemahaman program, harus dipahami bagaimana program tersebut terstruktur, bagaimana program memberikan fungsionalitas, dan bagaimana perubahan yang diusulkan dapat mempengaruhi program tersebut.

Change implementation



Dinamika evolusi program

- Dinamika evolusi program adalah studi tentang proses perubahan sistem.
- Persyaratan sistem kemungkinan akan berubah saat sistem sedang dikembangkan karena environment berubah, sehingga sistem yang disampaikan tidak akan memenuhi kebutuhan.
- Sistem terhubung erat dengan environment.
- Lehman dan Belady mengusulkan bahwa ada beberapa 'hukum' yang berlaku bagi semua sistem saat berkembang.
- ‘Hukum’ tersebut berlaku untuk sistem besar yang dikembangkan oleh organisasi besar.

Law	Description
Continuing change	A program that is used in a real-world environment must necessarily change, or else become progressively less useful in that environment.
Increasing complexity	As an evolving program changes, its structure tends to become more complex. Extra resources must be devoted to preserving and simplifying the structure.
Large program evolution	Program evolution is a self-regulating process. System attributes such as size, time between releases, and the number of reported errors is approximately invariant for each system release.
Organizational stability	Over a program's lifetime, its rate of development is approximately constant and independent of the resources devoted to system development.
Conservation of familiarity	Over the lifetime of a system, the incremental change in each release is approximately constant.
Continuing growth	The functionality offered by systems has to continually increase to maintain user satisfaction.
Declining quality	The quality of systems will decline unless they are modified to reflect changes in their operational environment.
Feedback system	Evolution processes incorporate multi agent, multi loop feedback systems and you have to treat them as feedback systems to achieve significant product improvement.

Software maintenance

- Pemeliharaan perangkat lunak berfokus pada melakukan modifikasi sebuah perangkat lunak saat sedang digunakan.
- Perangkat lunak khusus / custom : mengubah PL saat sedang digunakan
- Perangkat lunak general / umum : mengalami evolusi untuk menciptakan versi-versi baru.
- Maintenance umumnya tidak melibatkan perubahan besar pada arsitektur sistem.
- Perubahan diimplementasikan dengan memodifikasi komponen yang sudah ada dan menambahkan komponen baru ke dalam sistem.

Jenis-jenis Pemeliharaan PL

- *Fault repair* → Pemeliharaan untuk **memperbaiki kesalahan perangkat lunak**: mengubah sistem untuk memperbaiki kekurangan (bugs, errors, performance issues) untuk memenuhi kebutuhan.
- *Environmental adaptation* → Pemeliharaan untuk **menyesuaikan perangkat lunak dengan lingkungan operasional** yang berbeda: mengubah sistem agar beroperasi dalam lingkungan yang berbeda (hardware, sistem operasi, dll.) dengan implementasi awal.
- *Functionality addition* → Pemeliharaan untuk **menambah atau memodifikasi fungsionalitas sistem**: memodifikasi sistem untuk memenuhi kebutuhan baru.

Maintenance costs

- Biaya pemeliharaan biasanya lebih besar daripada biaya pengembangan (2x hingga 100x tergantung pada aplikasinya).
- Biaya dipengaruhi oleh faktor teknis dan non-teknis;
- Biaya cenderung meningkat seiring perangkat lunak dipelihara.
- Pemeliharaan dapat mengganggu struktur perangkat lunak sehingga pemeliharaan lebih lanjut menjadi lebih sulit.
- Perangkat lunak yang menua dapat memiliki biaya dukungan yang tinggi (misalnya, bahasa pemrograman lama, perubahan teknologi, dll.).

Technical cost factors

- **Kompleksitas kode**: Struktur kontrol dan logika yang kompleks sulit dipahami dan oleh karena itu sulit untuk diubah.
- **Perubahan dalam bahasa pemrograman**: Jika kode yang akan diubah ditulis dalam bahasa yang tidak lagi umum digunakan, sulit untuk diubah. Lebih sulit lagi jika kode baru harus ditulis dalam bahasa yang berbeda.
- **Perubahan dalam infrastruktur perangkat lunak**: Jika middleware, atau library yang digunakan telah berubah, maka para programmer harus memahami bagaimana perangkat lunak yang akan diubah berinteraksi dengan komponen tersebut.
- **Kualitas pemformatan dan gaya pengkodean**: contoh saat penamaan yang buruk digunakan.
- **Usia dan struktur program**: Seiring berjalannya waktu, struktur program tersebut mengalami penurunan kualitas dan menjadi lebih sulit dipahami dan diubah..

Human cost factors

- **Stabilitas tim**: biaya pemeliharaan akan berkurang jika staf yang sama terlibat dalam proses maintenance, Biaya pemeliharaan akan lebih tinggi jika pengembang asli tidak tersedia.
- **Tanggung jawab kontraktual**: Jika pengembang asli tidak diharapkan bertanggung jawab untuk perubahan di masa depan sehingga tidak ada rancangan untuk perubahan di masa depan.
- **Keterampilan staf**: staf pemeliharaan mungkin kurang berpengalaman dan memiliki pengetahuan domain yang terbatas.

Pertanyaan?