Metodologi Pengembangan Perangkat Lunak

Karmilasari

Apa itu software?

- Program komputer dan seluruh dokumen yang terkait di dalamnya

- Produk perangkat lunak dapat dikembangkan untuk :
  - pelanggan tertentu (custom)
  - dikembangkan untuk pasar umum (generik)

Apa itu software engineering?

- Software engineering adalah suatu disiplin perekayasaan yang terkait dengan semua aspek produksi perangkat lunak

- Perekayasa perangkat lunak harus mengadopsi pendekatan yang sistematis dan terorganisir untuk pekerjaan mereka dengan menggunakan perkakas dan teknik tertentu tergantung pada masalah yang akan dipecahkan, kendala pengembangan dan sumber daya yang tersedia

Apa itu software process?

- Satu set kegiatan yang tujuannya adalah pengembangan atau evolusi dari perangkat lunak
- Aktivitas umum software processes :
  - Specification - sistem apa yang harus dikembangkan dan kendala pengembangannya
  - Development - produksi software system
  - Validation – pengecekan apakah software tersebut sudah sesuai dengan keinginan customer
  - Evolution – perubahan software dalam merespon perubahan permintaan

Apa itu software process model?

- Sebuah representasi sederhana dari proses perangkat lunak, yang disajikan dari perspektif tertentu
- Contoh process perspectives
  - Workflow perspective – urutan aktivitas
  - Data-flow perspective – alur informasi
  - Role/action perspective – siapa mengerjakan apa
- Generic process models
  - Waterfall
  - Evolutionary development
  - Formal transformation
  - Integration from reusable components

Apa itu metode software engineering?

- Pendekatan terstruktur untuk pengembangan perangkat lunak yang meliputi model sistem, notasi, aturan, saran desain dan petunjuk proses
- Model descriptions
  - Deskripsi dari model grafis yang harus diproduksi
- Rules
  - Batasan yang diterapkan pada model sistem
- Recommendations
  - Rekomendasi untuk mendapatkan desain yang bagus
- Process guidance
  - Arahan kegiatan

Atribut yang Dibutuhkan untuk Mengembangkan Software yang baik?

- Maintainability
  - Perangkat lunak harus berkembang untuk memenuhi perubahan kebutuhan
- Dependability
  - Software harus dapat dipercaya
- Efficiency
  - Software tidak memboroskan sumber daya sistem
- Usability
  - Perangkat lunak harus dapat digunakan oleh pengguna

Apa tantangan utama yang dihadapi software engineering?

- Mengatasi sistem pewarisan, mengatasi keragaman yang meningkat dan mengatasi tuntutan untuk mengurangi waktu pengiriman
- Legacy systems
  - Lama, sistem yang berharga harus dijaga dan diperbarui
- Heterogeneity
  - Sistem didistribusikan dan mencakup gabungan hardware dan software
- Delivery
  - Ada tekanan yang meningkat untuk pengiriman lebih cepat dari perangkat lunak

State of The Art
Metodologi Pengembangan Perangkat Lunak

- 1920an, alat bantu *flowchart* sudah mulai dikenal
- Metodologi pengembangan perangkat lunak mulai dikenal sejak tahun 1960an sejak diperkenalkannya SDLC (*System Development Life Cycle)*
- 1970an : Pemrograman Terstruktur
- 1980an : Metodologi Analisa dan Perancangan Sistem Terstruktur (*Structured System Analysis and Design Methodology / SSADM)*

State of The Art
Metodologi Pengembangan Perangkat Lunak

- 1990an :
  - Object Oriented Programming (OOP)
  - Rapid Application Development (RAD)
  - Scrum Development
  - Team Software Process (dibangun oleh Watts Humphey)
- 2000an :
  - Extreme Programming (1999)
  - Rational Unified Process /RUP (1998)
  - Agile Unified Process / AUP (2005)
  - Integrated Methodology (QAIassist –IM) (2007)

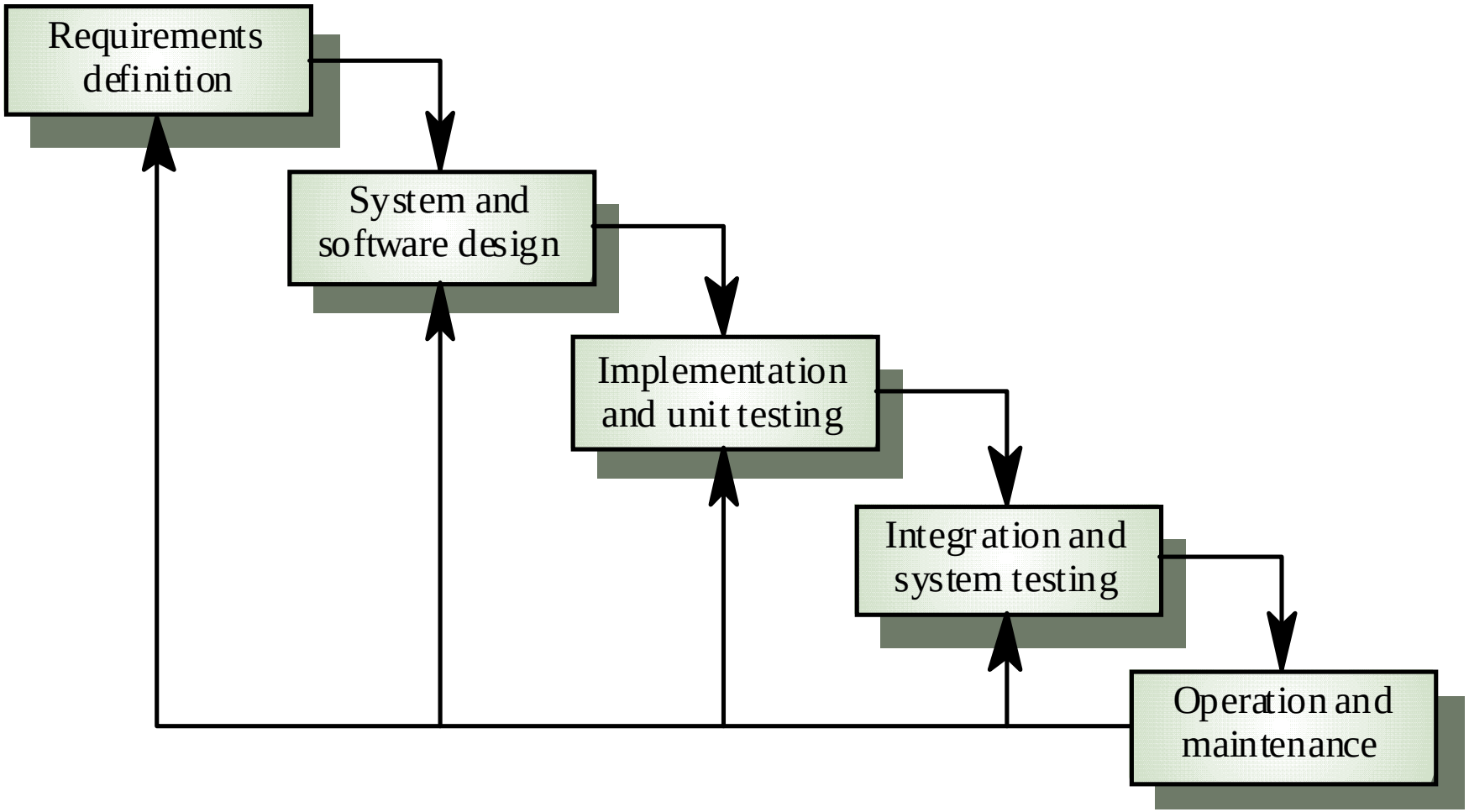
Software Process

- Satu set kegiatan terstruktur yang dibutuhkan untuk mengembangkan sistem perangkat lunak
  - Specification
  - Design
  - Validation
  - Evolution
- Sebuah model proses perangkat lunak adalah representasi abstrak dari suatu proses. Hal ini menyajikan gambaran tentang suatu proses dari beberapa perspektif tertentu

Model Process Generic Software

- Waterfall model
  - Memisahkan dan membedakan fase spesifikasi dan pengembangan
- Evolutionary development
  - Spesifikasi dan pengembangan interleave
- Formal systems development
  - Model sistem matematik ditransformasikan ke implementasi
- Reuse-based development
  - Sistem dirakit dari komponen yang ada

Waterfall model



Fase Waterfall model

- Mendefinisikan dan Menganalisis kebutuhan
- Perancangan System dan Software
- Pengujian unit dan Implementasi
- Pengujian Sistem dan Integrasi
- Pengoperasian dan Pemeliharaan
- Kelemahan dari model air terjun adalah sulitnya mengakomodasi perubahan setelah proses sedang berlangsung

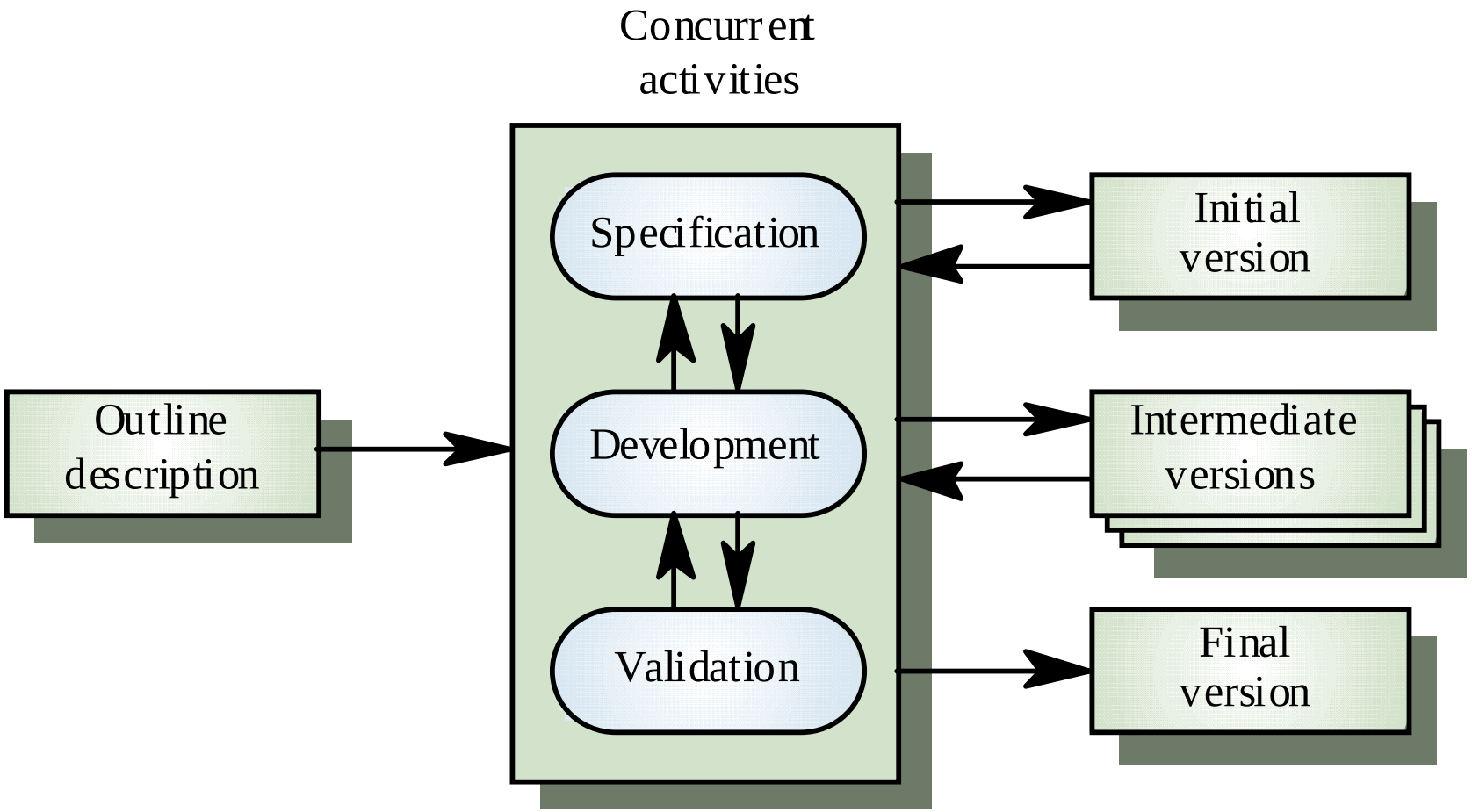
Masalah Waterfall model

- Partisi yang tidak fleksibel dari proyek pada tahap yang berbeda
- Hal ini membuat sulit untuk merespon kebutuhan pelanggan yang berubah
- Oleh karena itu, model ini hanya sesuai ketika persyaratan dipahami dengan baik

Evolutionary development

- Exploratory development
  - Penetapan tujuan dikerjakan bersama dengan pelanggan, termasuk pengembangan sistemnya,dari awal hingga akhir. Dimulai dengan pemahaman kebutuhan yang baik.
- Throw-away prototyping
  - Tujuan adalah untuk memahami kebutuhan sistem. Dimulai dengan pemahaman kebutuhan yang kurang

Evolutionary development



Evolutionary development

- Masalah
  - Kurangnya visibilitas proses
  - Sistem ini sering kurang terstruktur
  - Keterampilan khusus (misalnya dalam bahasa untuk rapid prototyping) mungkin diperlukan
- Applicability
  - Untuk sistem interaktif yang kecil atau menengah
  - Untuk bagian dari sistem yang besar (misalnya user interface)
  - Untuk sistem yang berumur pendek

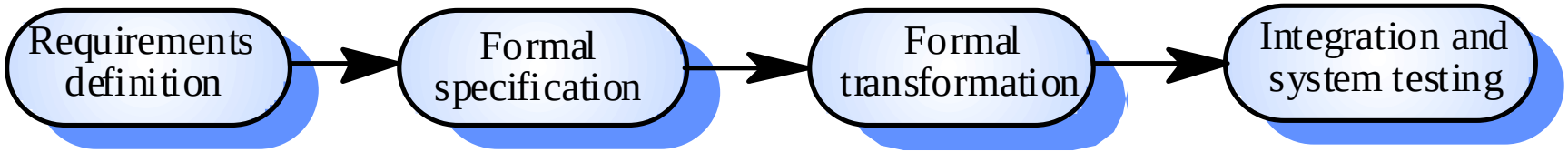
Formal systems development

- Berdasarkan pada transformasi spesifikasi matematika melalui representasi yang berbeda untuk program dieksekusi
- Transformasi adalah ‘corectness-preserving' sehingga sangat mudah untuk menunjukkan bahwa program tersebut sesuai dengan spesifikasinya
- Embodied in the ‘Cleanroom’ approach to software development

Penggunaan formal methods

- Metode formal telah membatasi penerapan praktis
- Manfaat utamanya adalah mengurangi jumlah kesalahan dalam sistem sehingga daerah utama mereka adalah penerapan sistem kritis
- Penggunaan metode formal memiliki biaya-efektif

Formal systems development



Penggunaan formal specification

- Spesifikasi formal melibatkan investasi lebih banyak usaha dalam fase awal dari pengembangan perangkat lunak
- Hal ini mengurangi kesalahan persyaratan dalam hal analisis rinci persyaratan
- Ketidaklengkapan dan inkonsistensi dapat ditemukan dan diselesaikan
- Ketidaklengkapan dan inkonsistensi dapat ditemukan dan diselesaikan

List specification

LIST (Elem)

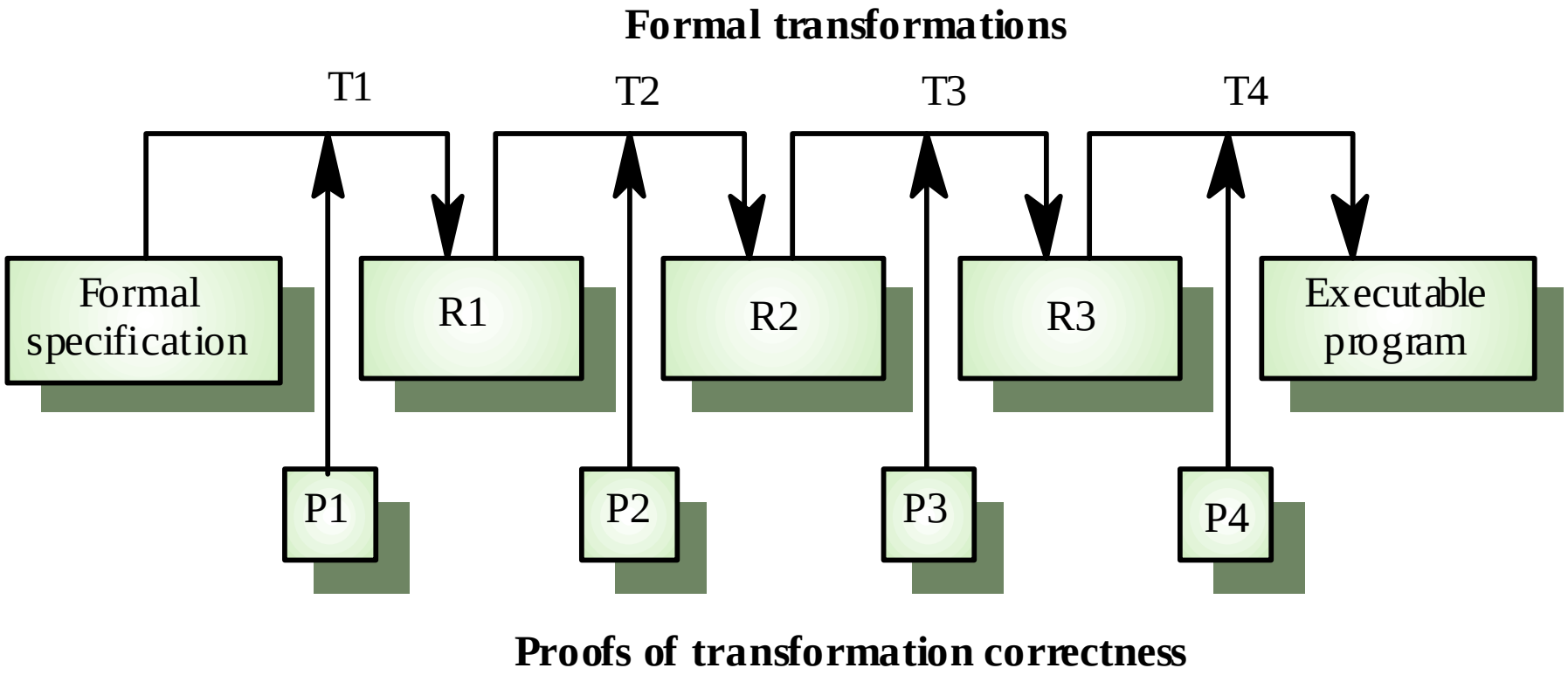
sort List
imports INTEGER

Defines a list where elements are added at the end and removed from the front. The operations are Create, which brings an empty list into existence, Cons, which creates a new list with an added member, Length, which evaluates the list size, Head, which evaluates the front element of the list, and Tail, which creates a list by removing the head from its input list. Undefined represents an undefined value of type Elem.

Create → List
Cons (List, Elem) → List
Head (List) → Elem
Length (List) → Integer
Tail (List) → List

Head (Create) = Undefined **exception** (empty list)
Head (Cons (L, v)) = **if** L = Create **then** v **else** Head (L)
Length (Create) = 0
Length (Cons (L, v)) = Length (L) + 1
Tail (Create) = Create
Tail (Cons (L, v)) = **if** L = Create **then** Create **else** Cons (Tail (L), v)

Formal transformations



Formal systems development

- Masalah
  - Butuh keterampilan khusus dan pelatihan untuk menerapkan teknik ini
  - Secara resmi sulit untuk menentukan beberapa aspek dari sistem seperti user interface
- Applicability
  - Sistem kritis terutama kasus keamanan harus dilakukan sebelum sistem ini dimasukkan ke dalam operasi

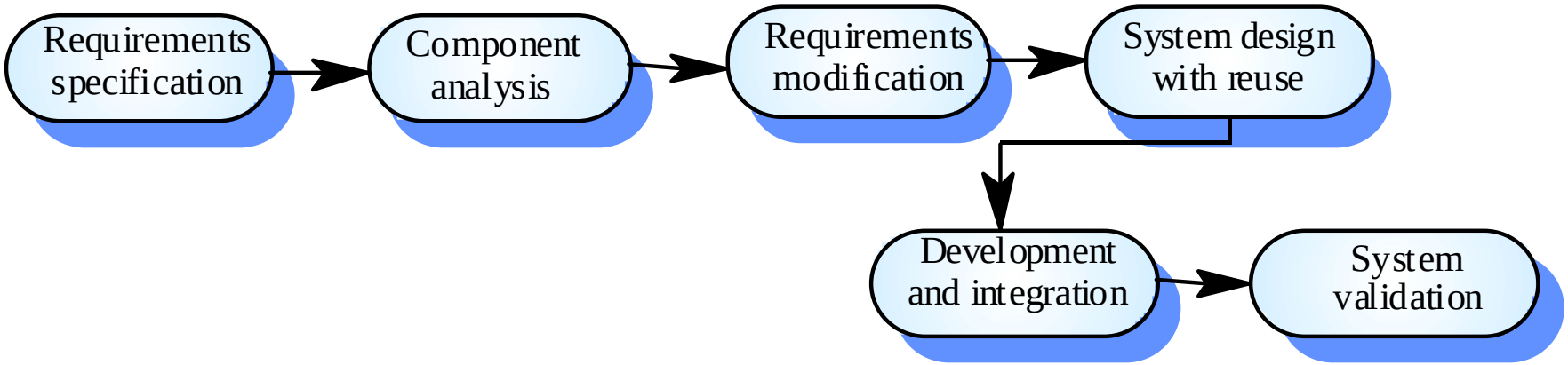
Reuse-oriented development

Component-based software engineering /

Rekayasa Perangkat Lunak Berbasis Komponen

- Berdasarkan penggunaan kembali / reuse sistem yang sistematis di mana sistem terintegrasi dari komponen yang ada or COTS (Commercial-off-the-shelf) systems.
- Tahapan proses
  - Analisis Komponen;
  - Modifikasi Kebutuhan;
  - Perancangan Sistem dengan penggunaan kembali / reuse yang sudah ada;
  - Pengembangan dan integrasi.
- Pendekatan ini mengalami peningkatan sejalan dengan penggunaan komponen standar telah muncul.

Reuse-oriented development



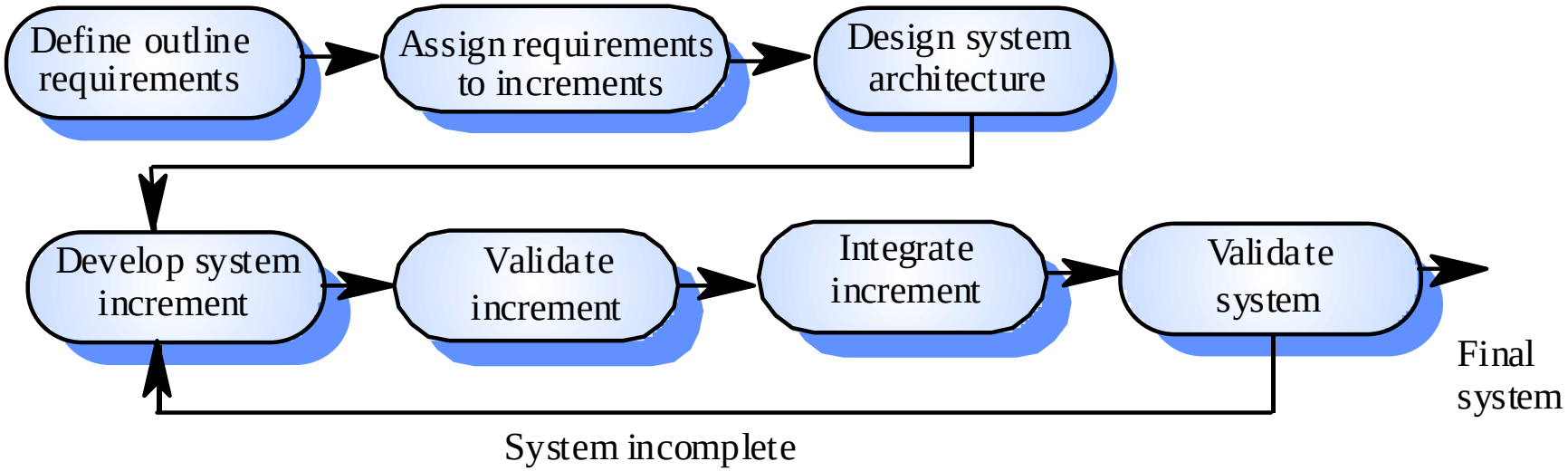
Process iteration

- Kebutuah sistem selalu berkembang dalam proyek sehingga proses iterasi pada tahap-tahap awal selalu dikerjakan ulang bagian dari proses untuk sistem yang besar
- Iterasi dapat diterapkan pada salah satu model proses generik
- Pendekatan
  - Incremental development
  - Spiral development

Incremental development

- Sistem sebagai pengiriman tunggal, pengem-bangan dan pengiriman dipecah menjadi bertahap dengan setiap kenaikan memberikan bagian dari fungsi yang diperlukan
- Kebutuhan pengguna diprioritaskan dan persyaratan prioritas tertinggi dimasukkan dalam awal increment
- Setelah pengembangan suatu increment dimulai, kebutuhan dibekukan meskipun persyaratan untuk kenaikan nantinya bisa terus berkembang

Incremental development



Manfaat Incremental development

- Nilai pelanggan dapat disampaikan dengan kenaikan masing-masing sehingga fungsionalitas sistem tersedia sebelumnya
- Increment awal bertindak sebagai prototipe untuk membantu mendapatkan persyaratan untuk kenaikan kemudian
- Menurunkan resiko kegagalan proyek secara keseluruhan
- Layanan sistem prioritas tertinggi cenderung menerima pengujian paling banyak

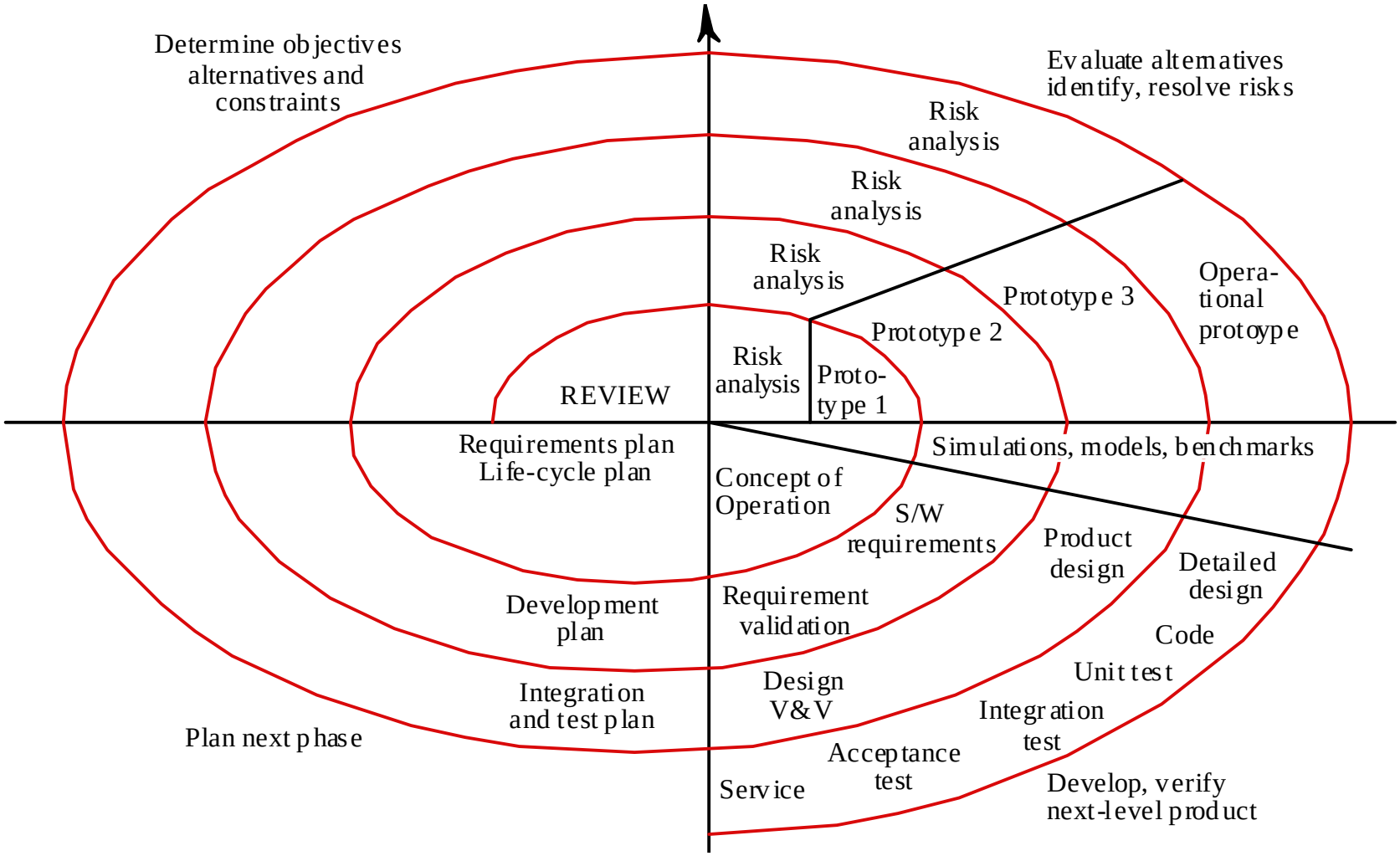
Extreme programming

- Pendekatan baru untuk pengembangan berdasarkan pengembangan dan pengiriman bertahap sangat kecil dari fungsi yang ada
- Mengandalkan kode perbaikan konstan, keterlibatan user dalam tim pengembangan dan pemrograman berpasangan

Spiral development

- Proses digambarkan sebagai spiral bukan sebagai urutan aktivitas dengan backtracking
- Setiap loop dalam spiral merupakan tahap dalam proses.
- Tidak ada fase tetap seperti spesifikasi atau desain - loop dalam spiral dipilih tergantung pada apa yang dibutuhkan.
- Risiko secara eksplisit dinilai dan diselesaikan selama proses.

Software process : Spiral model



Spiral model sectors

- Setting Tujuan
  - Tujuan khusus untuk fase identifikasi
- Penilaian dan Pengurangan Resiko
  - Resiko dinilai dan kegiatan disiapkan untuk mengurangi resiko kunci
- Pengembangan dan Validasi
  - Sebuah model pengembangan untuk sistem terpilih yang dapat menjadi salah satu model generik
- Perencanaan
  - Proyek terakhir di-review dan fase berikutnya dari spiral direncanakan

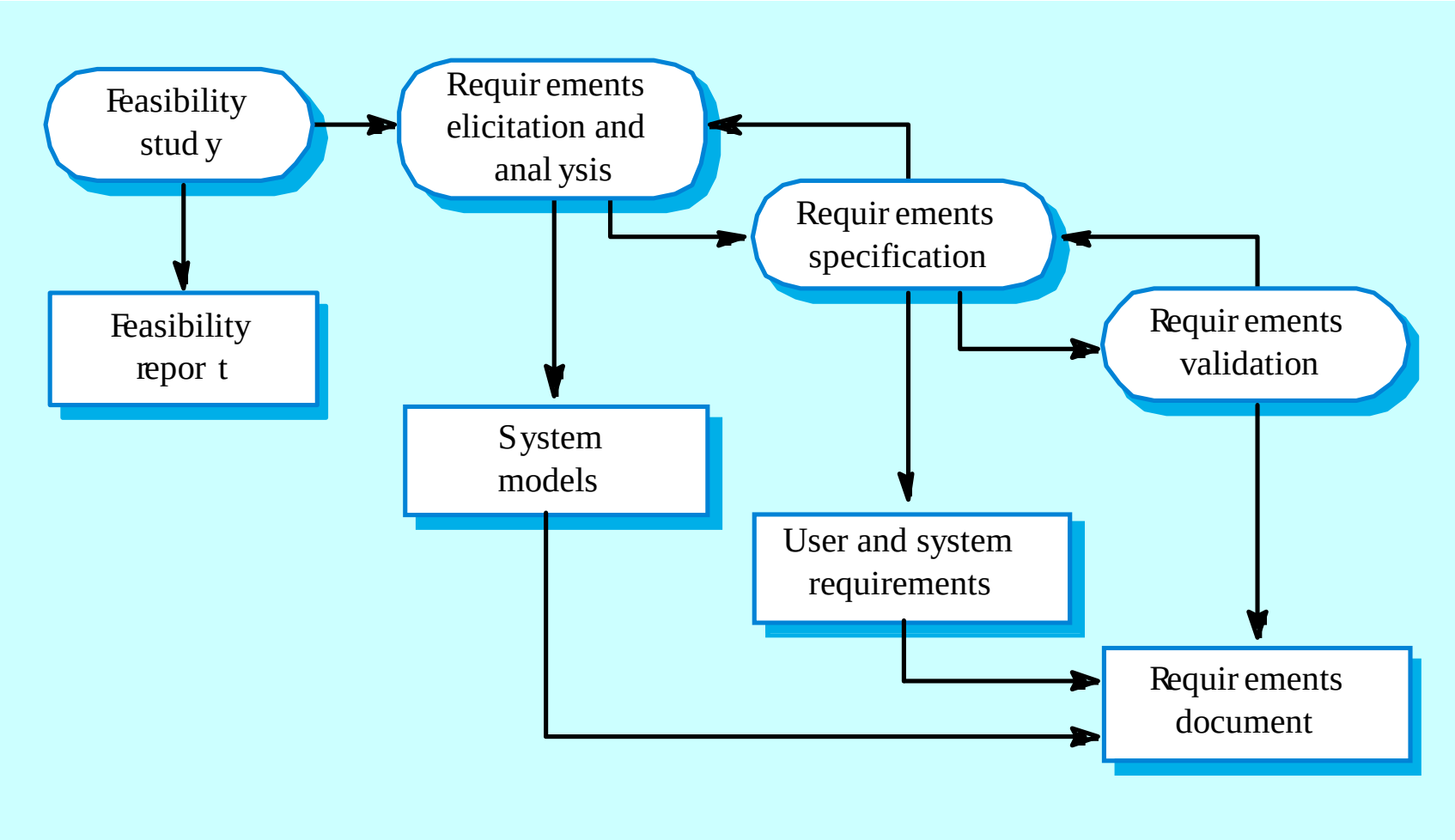
Aktivitas Proses

- Spesifikasi Perangkat Lunak
- Perancangan dan implementasi peranngkat lunak
- Validasi perangkat lunak
- Evolusi perangkat lunak

Spesifikasi Perangkat Lunak

- Proses dibangun dari layanan apa saja yang dibutuhkan dan batasan operasi dan pengembangan sistem
- Kebutuhan rekayasa proses
  - Studi kelayakan
  - Kebutuhan analisis
  - Kebutuhan spesifikasi;
  - Kebutuhan validasi.

Kebutuhan Rekayasa Proses



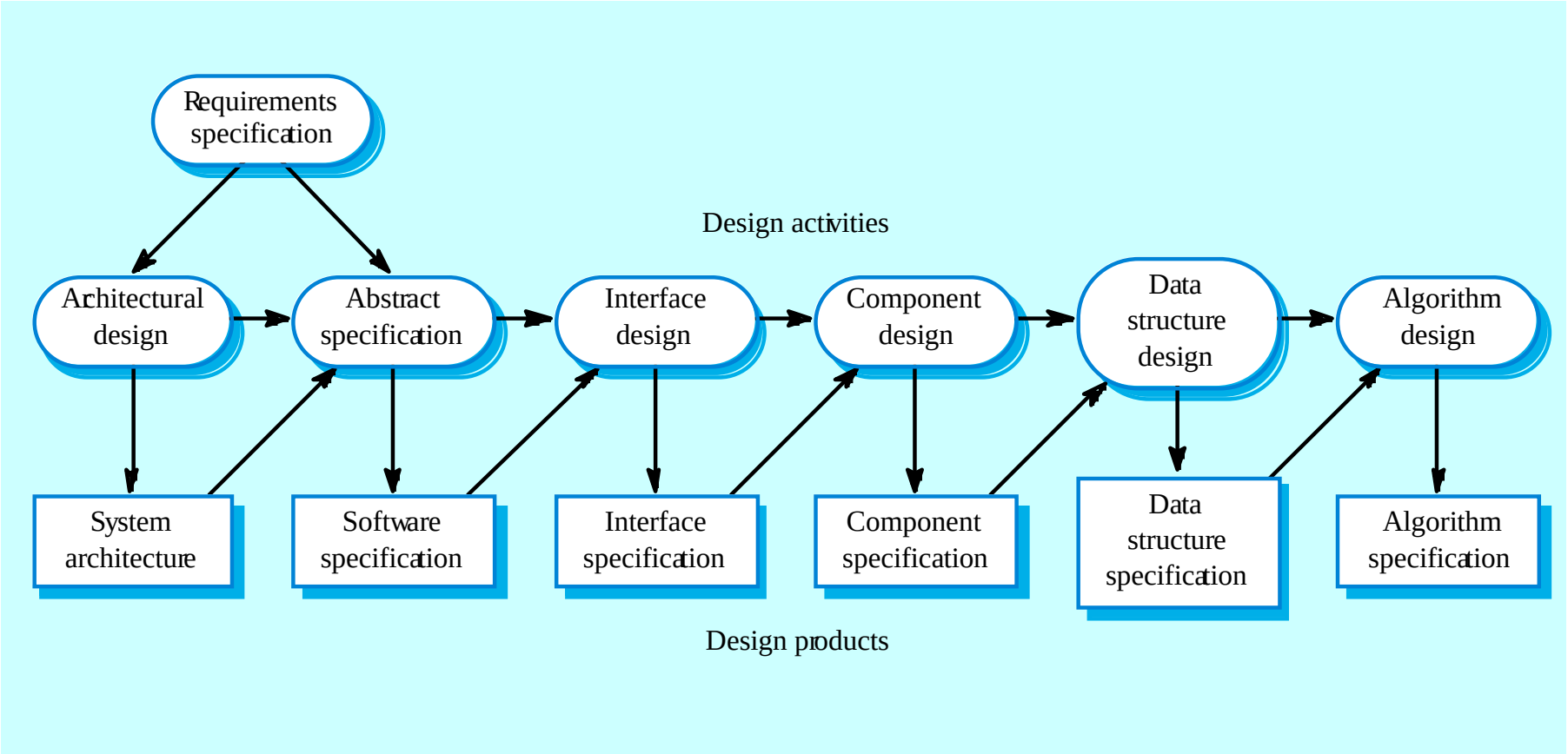
Perancangan dan Implementasi Perangkat Lunak

- Proses konversi spesifikasi sistem ke dalam eksekusi sistem.
- Perancangan perangkat lunak
  - Merancang struktur perangkat lunak yang sesuai dengan spesifikasi
- Implementasi perangkat lunak
  - Translasi struktur ke dalam eksekusi program
- Aktivitas perancangan dan implementasi saling berelasi satu dengan yang lain.

Aktivitas Proses Perancangan

- Perancangan arsitektur
- Spesifikasi Abstrak
- Perancangan Pengantarmukaan
- Perancangan Komponen
- Perancangan Struktur Data
- Perancangan Algoritma

Proses Perancangan Perangkat Lunak



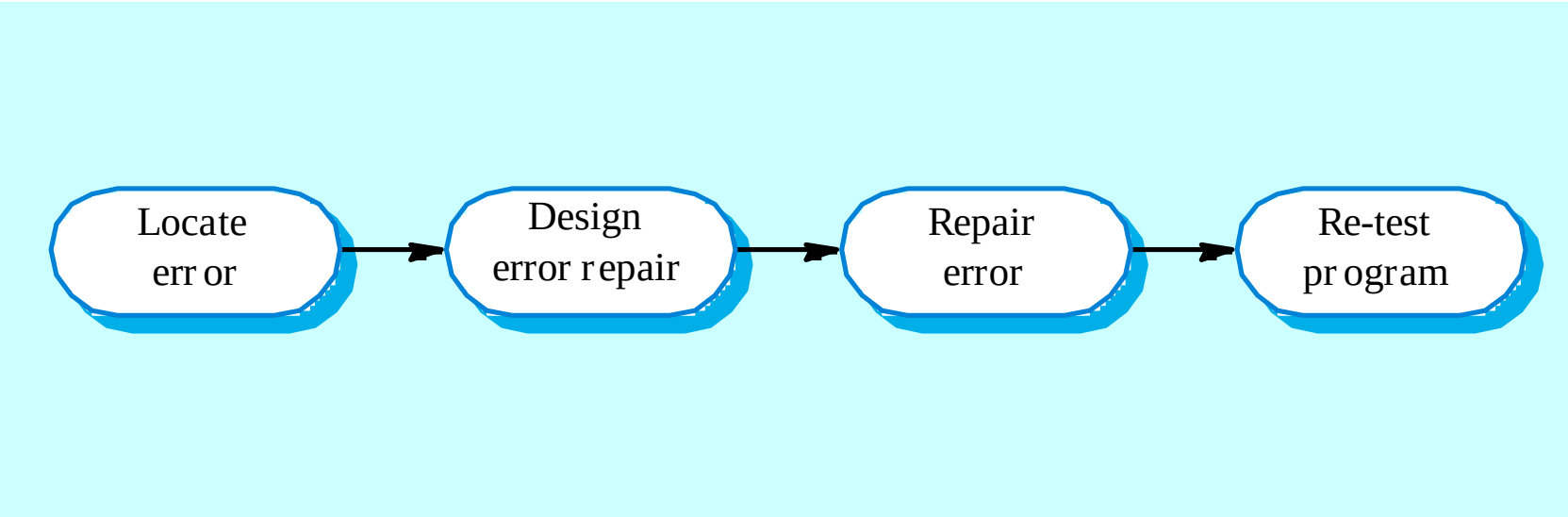
Metode Terstruktur

- Pendekatan sistematis untuk membangun rancangan perangkat lunak
- Perancangan biasanya didokukentasikan dalam suatu set model grafis
- Model grafis yang digunakan :
  - Object model;
  - Sequence model;
  - State transition model;
  - Structural model;
  - Data-flow model.

Pemrograman dan Debugging

- Translasi dari rancangan ke dalam program dan penanganan kesalahan dalam program
- Program merupakan aktivitas personal, dimana proses pemrograman tidak generik
- Pemrogram melakukan beberapa pengujian pada program untuk menemukan kesalahan dalam program dan melakukan proses perbaikan / debugging

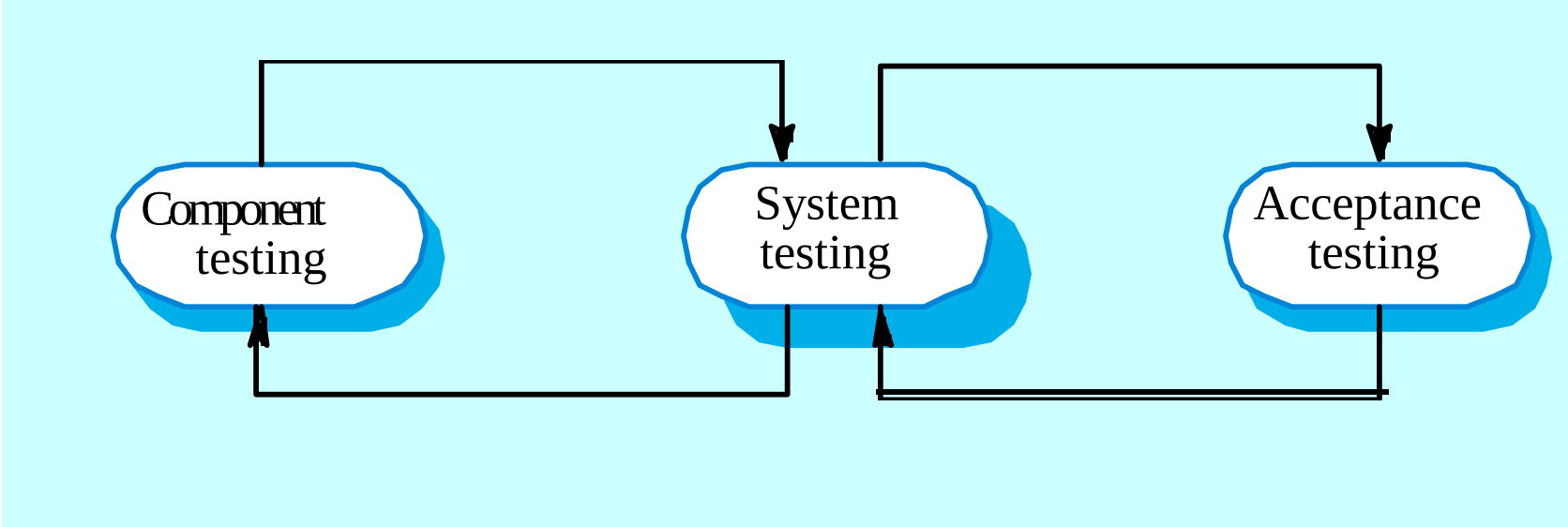
Proses Debugging / Penanganan Kesalahan



Validasi Perangkat Lunak

- Verifikasi dan validasi (V & V) dimaksudkan untuk menunjukkan bahwa sistem sesuai dengan spesifikasi dan memenuhi persyaratan pelanggan sistem
- Terlibat dalam pemeriksaan, peninjauan dan proses dan pengujian sistem
- Pengujian sistem melibatkan eksekusi sistem dengan uji kasus yang berasal dari spesifikasi data sebenarnya yang akan diproses oleh sistem.

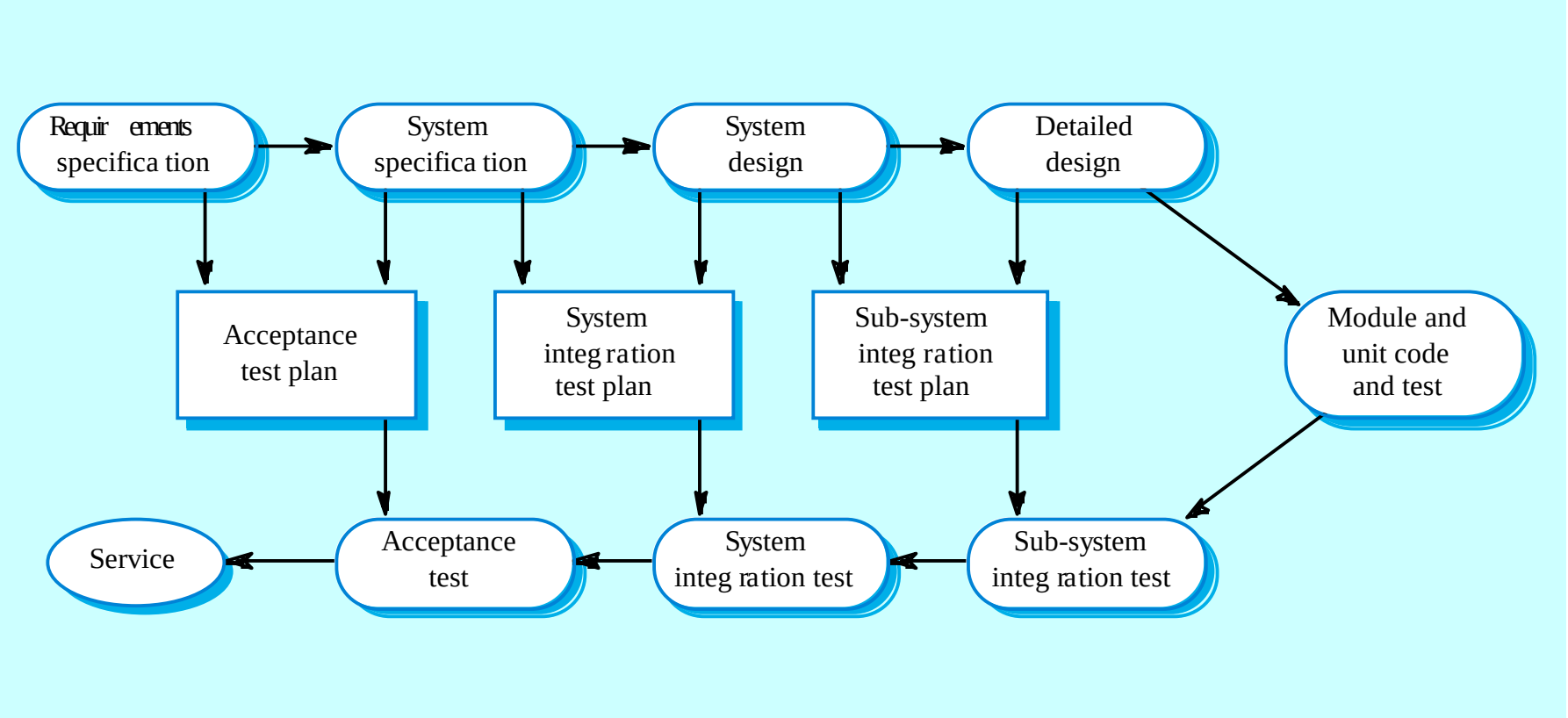
Proses Uji Coba



Tahapan Uji Coba

- Uji Coba Unit atau Komponen
  - Komponen individual diuji secara independen
  - Komponen dapat berupa fungsi atau objek atau kelompok koheren dari suatu entitas
- Uji Coba Sistem
  - Uji coba sistem secara keseluruhan.
  - Uji coba terhadap sifat-sifat yang muncul sangat penting diperhatikan
- Uji Coba Penerimaan
  - Uji coba dengan data pelanggan untuk memeriksa apakah sistem menerima kebutuhan pelanggan

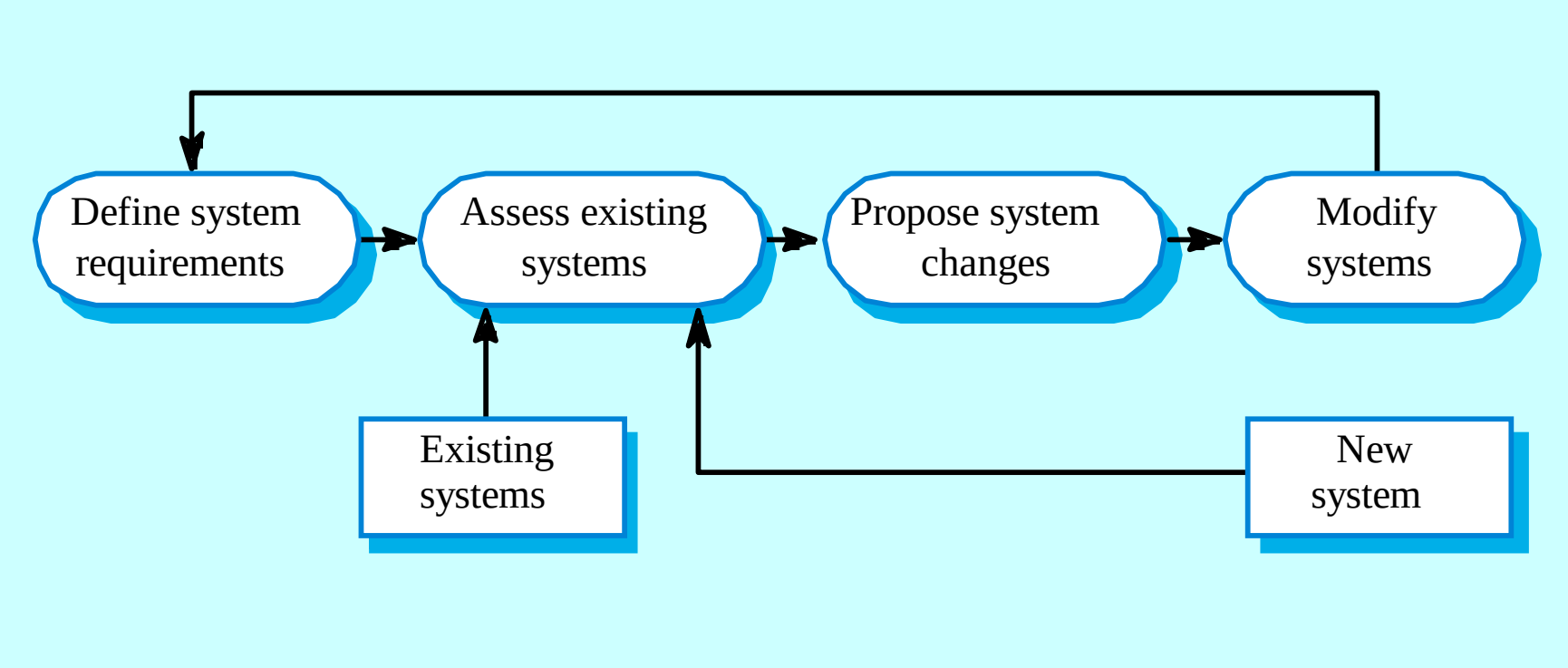
Fase Uji Coba



Evolusi Perangkat Lunak

- Perangkat lunak rentan terhadap perubahan.
- Perubahan keadaan bisnis, biasanya membutuhkan penyesuaian perangkat lunak yang mendukung perubahan tersebut.
- Terdapat garis batas yang tipis antara pengembangan dan evolusi (pemeliharaan) terkait dengan perubahan (walaupun sedikit) menjadi suatu sistem baru yang lebih sempurna (sesuai dengan kebutuhan terkini)

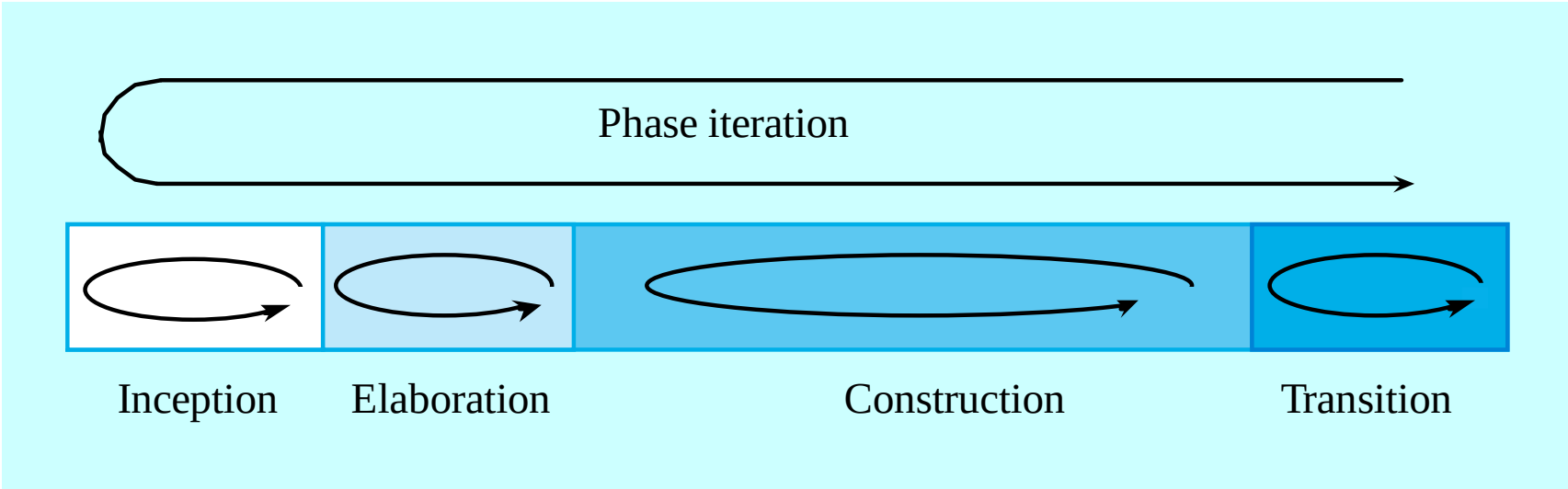
Evolusi Sistem



Rational Unified Process (RUP)

- Suatu model proses modern diturunkan dari UML (Unified Modelling Languange) dan proses-proses yang terkait di dalamnya.
- Terdapat 3 perspektif
  - Perspektif Dinamik, yang menunjukkan fase dari waktu ke waktu
  - Perspektif Statik, yang menunjukkan proses aktivitas
  - Perspektif Praktis, yang menyarankan pemakaian terbaik

Mode Fase RUP



Fase RUP

- Inception / Permulaan
  - Penetapan kasus bisnis untuk sistem.
- Elaboration / Elaborasi-Perluasan
  - Pengembangan dan pemahaman domain masalah dan arsitektur sistem
- Construction / Pembangunan
  - Perancangan sistem, pemrograman dan uji coba
- Transition / Transisi
  - Penyebarluasan sistem di lingkungan operasional

Praktek RUP Yang Baik

- Membangun perangkat lunak secara iteratif
- Mengelola kebutuhan
- Menggunakan arsiterktur berbasis komponen
- Perangkat lunak dengan model visual
- Memverifikasi kualitas perangkat lunak
- Pengendalian terhadapa perubahan perangkat lunak

Aliran Kerja Statik

Workflow	Description
Business modelling	The business processes are modelled using business use cases.
Requirements	Actors who interact with the system are identified and use cases are developed to model the system requirements.
Analysis and design	A design model is created and documented using architectural models, component models, object models and sequence models.
Implementation	The components in the system are implemented and structured into implementation sub-systems. Automatic code generation from design models helps accelerate this process.
Test	Testing is an iterative process that is carried out in conjunction with implementation. System testing follows the completion of the implementation.
Deployment	A product release is created, distributed to users and installed in their workplace.
Configuration and change management	This supporting workflow managed changes to the system (see Chapter 29).
Project management	This supporting workflow manages the system development (see Chapter 5).
Environment	This workflow is concerned with making appropriate software tools available to the software development team.

Computer-Aided Software Engineering (CASE)

- Computer-aided software engineering (CASE) merupakan perangkat lunak yang mendukung pengembangan perangkat lunak dan proses evolusi
- Otomatisasi Aktivitas
  - Editor Grafis untuk pengembangan model sistem
  - Kamus data untuk mengelola perancangan entitas
  - GUI (Graphical User Interface) untuk membangun pengantarmukaan pengguna
  - Debugger untuk mendukung pencarian kesalahan program
  - Translator otomatis untuk men-generate versi baru dari suatu program

Teknologi CASE

- Teknologi CASE telah membawa perbaikan yang signifikan dalam proses perangkat lunak.
- Namun demikian ada beberapa hal yang perlu dipetimbangkan dalam penggunaan CASE :
  - Rekayasa perangkat lunak membutuhkan pemikiran kreatif – tidak selalu dapat diotomatisasi
  - Rekayasa perangkat lunak adalah kegiatan tim dan untuk proyek-proyek besar, banyak waktu yang dihabiskan dalam interaksi tim. Teknologi CASE tidak benar-benar mendukung untuk hal tersebut.

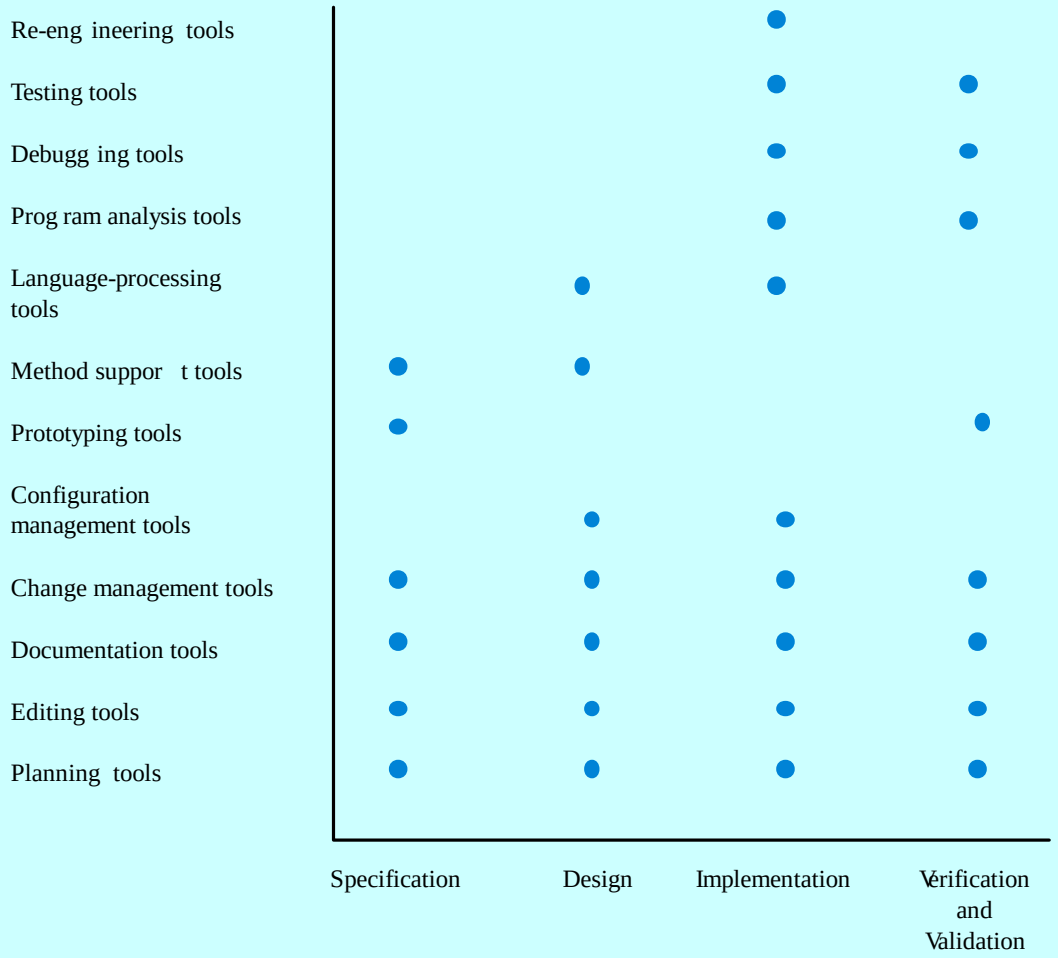
Klasifikasi CASE

- Klasifikasi membantu kita memahami perbedaan tipe perangkat pendukung (tools) CASE dalam mendukung proses aktivitas
- Perspektif Fungsional
  - Perangkat pendukung (tools) diklasifikasikan berdasarkan fungsi spesifik
- Perspektif Proses
  - Perangkat pendukung (tools) diklasifikasikan berdasarkan aktivitas proses yang didukungnya
- Perspektif Integrasi
  - Perangkat pendukung (tools) diklasifikasikan berdasarkan organisasi dan unit yang terintegrasi di dalamnya.

Klasifikasi Perangkat Fungsional

Tool type	Examples
Planning tools	PERT tools, estimation tools, spreadsheets
Editing tools	Text editors, diagram editors, word processors
Change management tools	Requirements traceability tools, change control systems
Configuration management tools	Version management systems, system building tools
Prototyping tools	Very high-level languages, user interface generators
Method-support tools	Design editors, data dictionaries, code generators
Language-processing tools	Compilers, interpreters
Program analysis tools	Cross reference generators, static analysers, dynamic analysers
Testing tools	Test data generators, file comparators
Debugging tools	Interactive debugging systems
Documentation tools	Page layout programs, image editors
Re-engineering tools	Cross-reference systems, program re-structuring systems

Klasifikasi Perangkat Berbasis Aktivitas



Integrasi CASE

- Tools / Perangkat
  - Dukungan proses tugas individual, seperti perancangan, pengecekan konsistensi, text editing dsb.
- Workbenches
  - Dukungan fase proses seperti spesifikasi atau perancangan. Biasanya menggunakan sejumlah perangkat / tools yang terintegrasi
- Environments / Lingkungan
  - Dukungan terhadap semua atau sebagian besar dari proses software secara keseluruhan. Biasanya termasuk terintegrasi beberapa workbenches.

Tools, Workbenches, Environments