Software Process

S2 Teknik Informatika dan Komputer
PENS

What is Software Process?

- When you work to build a product or system, it’s important to go through a series of predictable steps.
- That is a road map that helps you create a timely, high-quality result.
- The road map that you follow is called a “software process.”

The Software Process

- It is important because it provides stability, control, and organization to an activity that can --if left uncontrolled-- become quite chaotic.
- A modern software engineering approach must be “agile.” It must demand only those activities, controls, and work products that are
  - appropriate for the project team and
  - the product that is to be produced.

The Software Process

- A generic process framework for software engineering defines *five framework activities*—**communication, planning, modeling, construction, and deployment.**
- The software process, or sometimes is called as *Software Development Process* or *Software Development Life Cycle*, is often resulted from a long-time best practice process of the company.

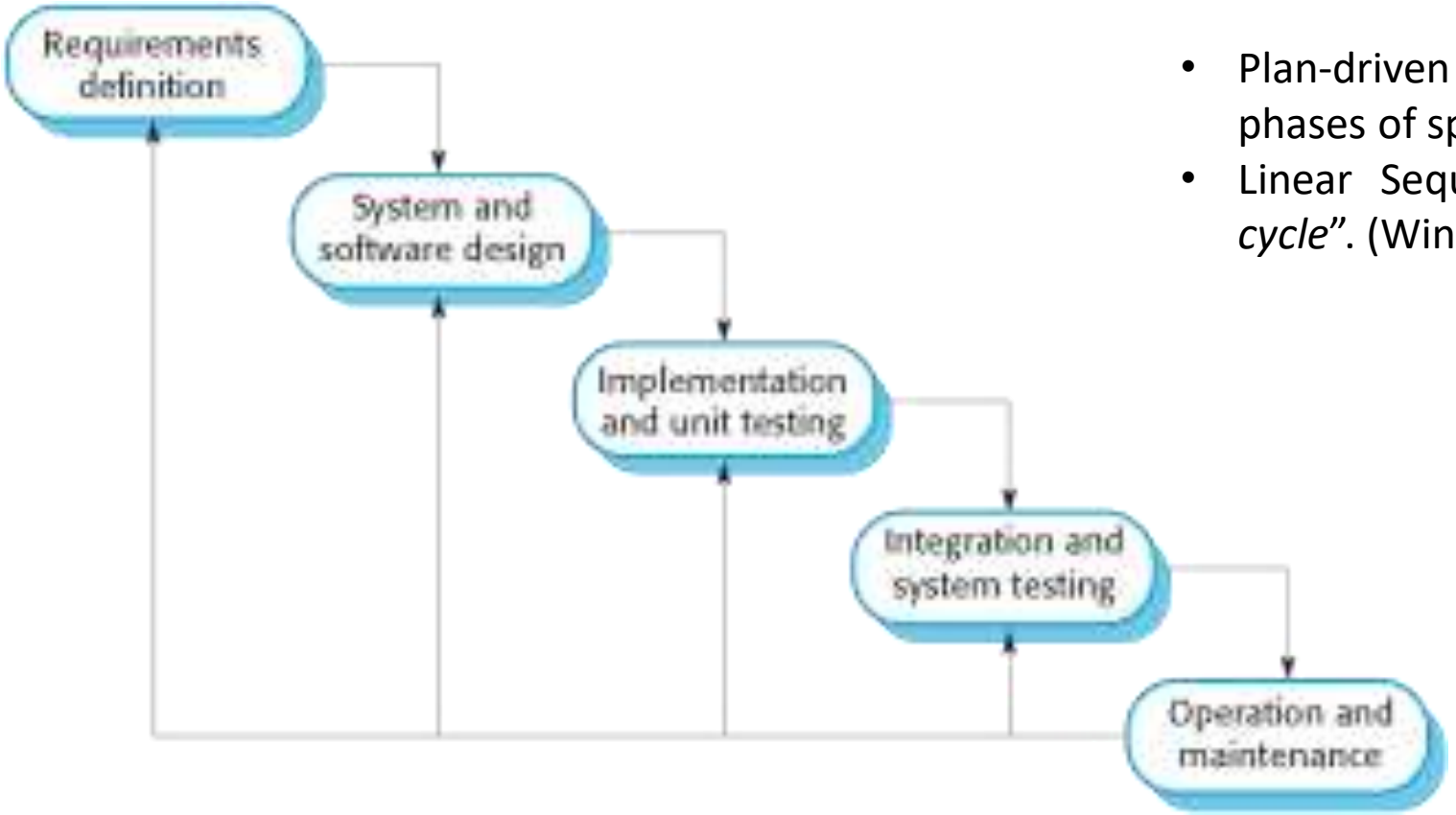
The Software Process

- A structured set of activities required to develop a software system.
- Common activity to all software processes:
  - Specification – defining what the system should do;
  - Design and implementation – defining the organization of the system and implementing the system;
  - Validation – checking that it does what the customer wants;
  - Evolution – changing the system in response to changing customer needs.

Software process models

- A software process model is an abstract representation of a process. It presents a description of a process from some particular perspective.
- Software process models:
  - Waterfall model
  - Evolutionary development
  - Reuse-based development
  - Agile
- Hybrid software process models:
  - Incremental development
  - Spiral development

The waterfall model



- Plan-driven model. Separate and distinct phases of specification and development.
- Linear Sequential Model" or "*classic life cycle*". (Winston Ryoce – 1970)

Waterfall model phases

- There are separate identified phases in the waterfall model:
  - Requirements analysis and definition
  - System and software design
  - Implementation and unit testing
  - Integration and system testing
  - Operation and maintenance
- The main drawback of the waterfall model is the difficulty of accommodating change after the process is underway. In principle, a phase has to be complete before moving onto the next phase.

Requrement Definition

1. Pengumpulan bahan-bahan mengenai kebutuhan-kebutuhan user.
2. Analisa sesuai dengan apa yang diinginkan oleh pengguna.
3. Konsultasi dengan pengguna sistem.
4. Definisikan kebutuhan-kebutuhan yang mungkin dalam sistem yang akan kita buat.

System and Software Design

1. Menghasilkan sebuah arsitektur sistem secara keseluruhan.
2. Desain perangkat lunak (*software*).
3. Fungsi sistem perangkat lunak dalam bentuk yang memungkinkan untuk ditransformasikan kedalam satu atau lebih program yang dapat dijalankan.

Implementation and Unit Testing

1. Desain direalisasikan kedalam bentuk program-program yang terpisah sesuai dengan unit-unitnya.
2. Setelah terbentuk kedalam suatu program, maka dilakukan *testing* atau uji coba terhadap program tersebut.

Integration and System Testing

1. Penyatuan terhadap program-program yang telah diuji pada tahap sebelumnya.
2. Uji coba terakhir terhadap sistem yang telah lengkap sebelum diserahkan kepada pengguna.

Operation and Maintenance

1. Tahapan ini merupakan tahap yang membutuhkan waktu paling lama.
2. Tahap penggunaan sistem oleh pengguna.
3. Dilakukan tahap perawatan atau *maintenance*.

Waterfall model: Plus and Minuses

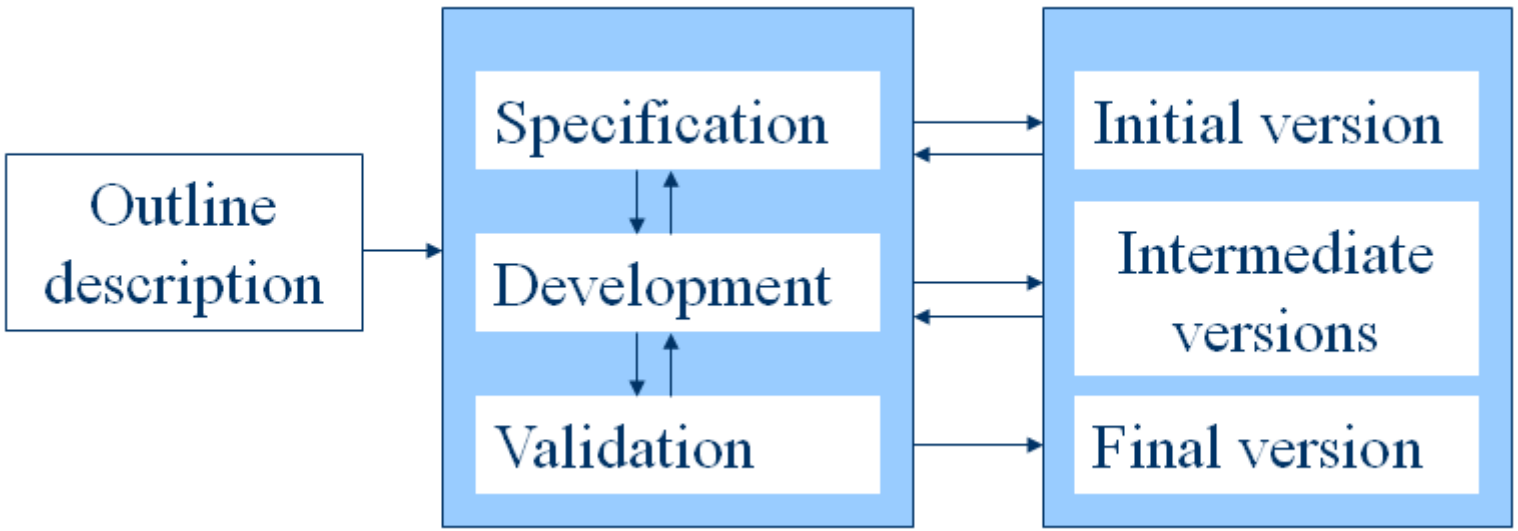
- Advantages of the model:
  - Simple to follow
  - Relatively simple to track progress
  - Good structural design
- Challenges:
  - In practice, often phases overlap
  - Hard to modify and implement changes
  - Need complete requirements from customers to start (the biggest challenge)

Waterfall model problems

- Inflexible partitioning of the project into distinct stages makes it difficult to respond to changing customer requirements.
  - Therefore, this model is only appropriate when the requirements are well-understood and changes will be fairly limited during the design process.
  - Few business systems have stable requirements.
- The waterfall model is mostly used for large systems engineering projects where a system is developed at several sites.
  - In those circumstances, the plan-driven nature of the waterfall model helps coordinate the work.

Evolutionary development

- Develop an initial implementation, expose to users comments, refine until satisfied:



Evolutionary development types

There are two types of evolutionary development:

- Exploratory development
  - Start with requirements that are well defined
  - Add new features when customers propose new requirements
- Throw-away prototyping
  - Objective is to understand customer's requirements (i.e. they often don't know what they want, hence poor requirements to start
  - Use means such as prototyping to focus on poorly understood requirements, redefine requirements as you progress

Evolutionary development: advantages and challenges

- Advantages:
  - Happier customers since you help them define requirements
  - Flexibility in modifying requirements
  - Prototypes are very visual, hence no ambiguities
- Challenges:
  - Hard to trace the “process” due to the ad-hoc nature
  - Systems are often poorly structured
  - Special tools and techniques may be required (for rapid development) that may be incompatible
  - Not cost-effective to produce documents

Reuse-oriented software engineering

- Based on systematic reuse where systems are integrated from existing components or COTS (Commercial-off-the-shelf) systems.
- Process stages
  - Component analysis;
  - Requirements modification;
  - System design with reuse;
  - Development and integration.

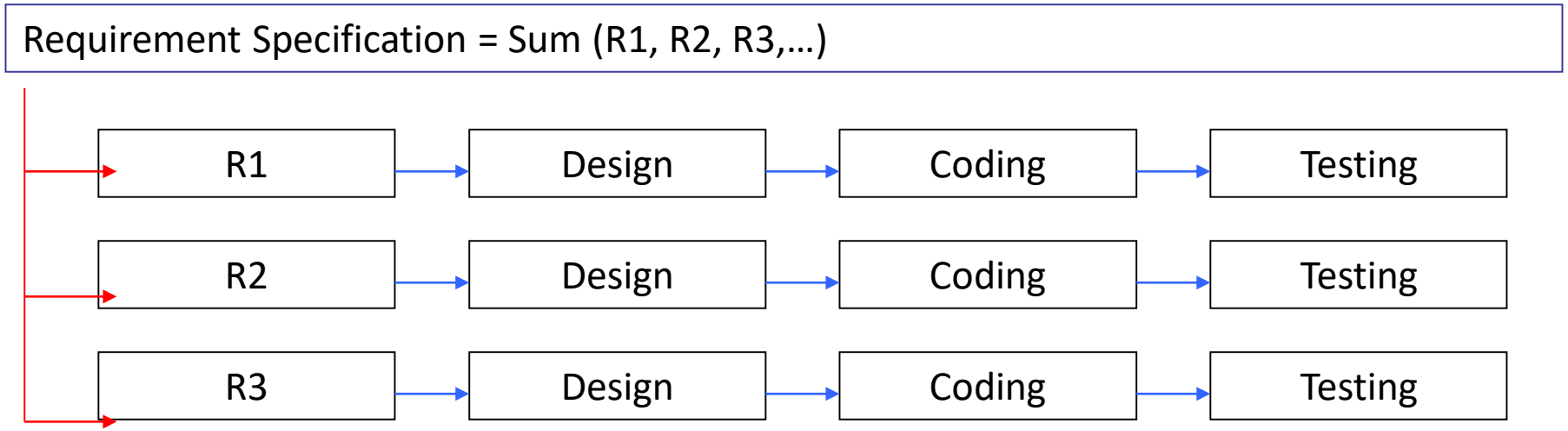
Reuse-oriented software engineering

Incremental development and delivery

- Incremental development
  - Develop the system in increments and evaluate each increment before proceeding to the development of the next increment;
  - Normal approach used in agile methods;
  - Evaluation done by user/customer proxy.
- Incremental delivery
  - Deploy an increment for use by end-users;
  - More realistic evaluation about practical use of software;
  - Difficult to implement for replacement systems as increments have less functionality than the system being replaced.

Incremental development

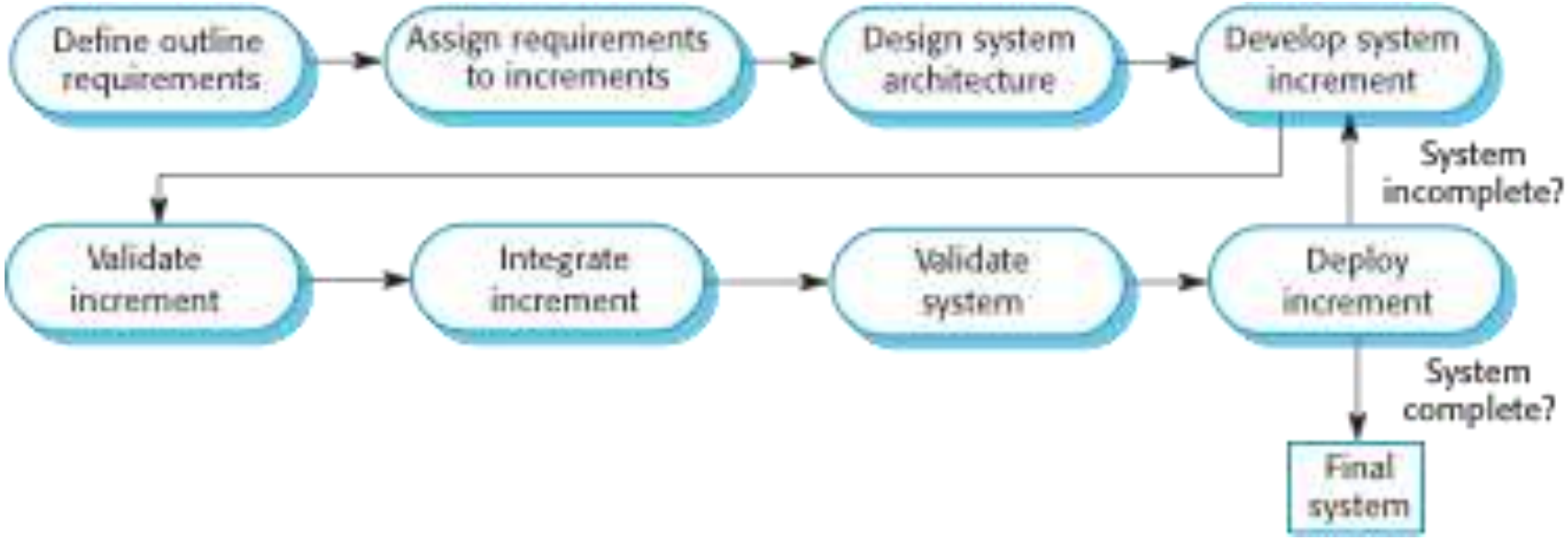
- A hybrid model where the software specification, design, implementation, and testing is broken down into a series of increments which are developed and delivered



Incremental development: Advantages and Challenges

- Advantages:
  - Products delivered incrementally hence faster
  - Lower risk of overall project failure
  - Requirements are implemented based on priority
- Challenges:
  - Relationship between different increments may be cumbersome or non-cohesive
  - Size of each increment and the number of increments may cause challenges

Incremental delivery



Incremental delivery advantages

- Customer value can be delivered with each increment so system functionality is available earlier.
- Early increments act as a prototype to help elicit requirements for later increments.
- Lower risk of overall project failure.
- The highest priority system services tend to receive the most testing.

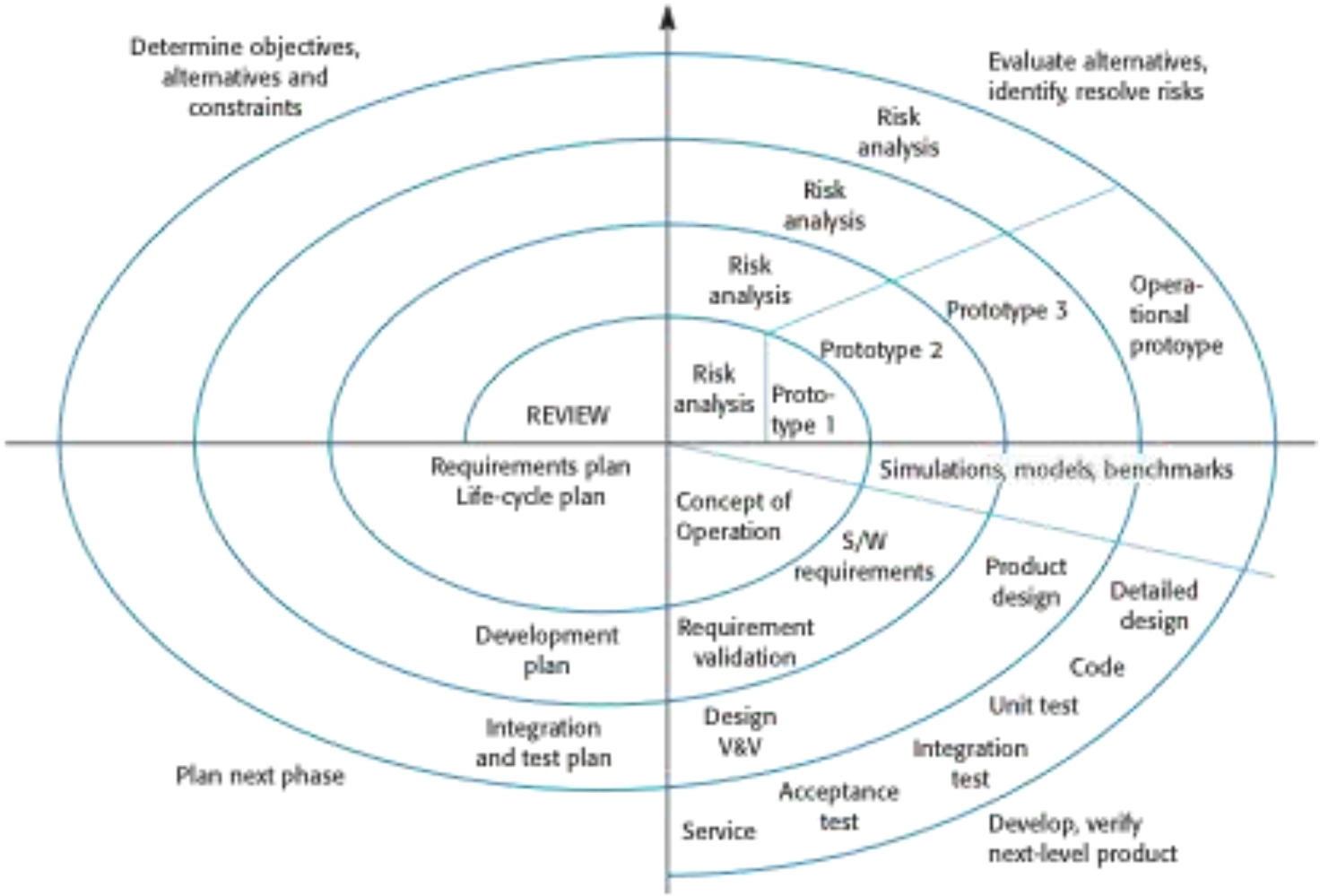
Incremental delivery problems

- Most systems require a set of basic facilities that are used by different parts of the system.
  - As requirements are not defined in detail until an increment is to be implemented, it can be hard to identify common facilities that are needed by all increments.
- The essence of iterative processes is that the specification is developed in conjunction with the software.
  - However, this conflicts with the procurement model of many organizations, where the complete system specification is part of the system development contract.

Boehm’s spiral model

- Process is represented as a spiral rather than as a sequence of activities with backtracking.
- Each loop in the spiral represents a phase in the process.
- No fixed phases such as specification or design - loops in the spiral are chosen depending on what is required.
- Risks are explicitly assessed and resolved throughout the process.

Boehm's spiral model of the software process



Spiral model sectors

- Objective setting
  - Specific objectives for the phase are identified.
- Risk assessment and reduction
  - Risks are assessed and activities put in place to reduce the key risks.
- Development and validation
  - A development model for the system is chosen which can be any of the generic models.
- Planning
  - The project is reviewed and the next phase of the spiral is planned.

Spiral model usage

- Spiral model has been very influential in helping people think about iteration in software processes and introducing the risk-driven approach to development.
- In practice, however, the model is rarely used as published for practical software development.

Spiral development:
Plus & Minuses

- Advantages:
  - Explicit consideration of risks (alternative solutions are evaluated in each cycle)
  - More detailed processes for each development phase
- Disadvantages:
  - Cost
  - Sometime difficult to implement or too time consuming

Component-based SE

- Adalah proses yang menekankan perancangan dan pembangunan software dengan menggunakan komponen software yang sudah ada.

- Terdiri dari 2 bagian yang terjadi secara PARAREL, yaitu:
  1. Domain Engineering
  2. Software Engineering

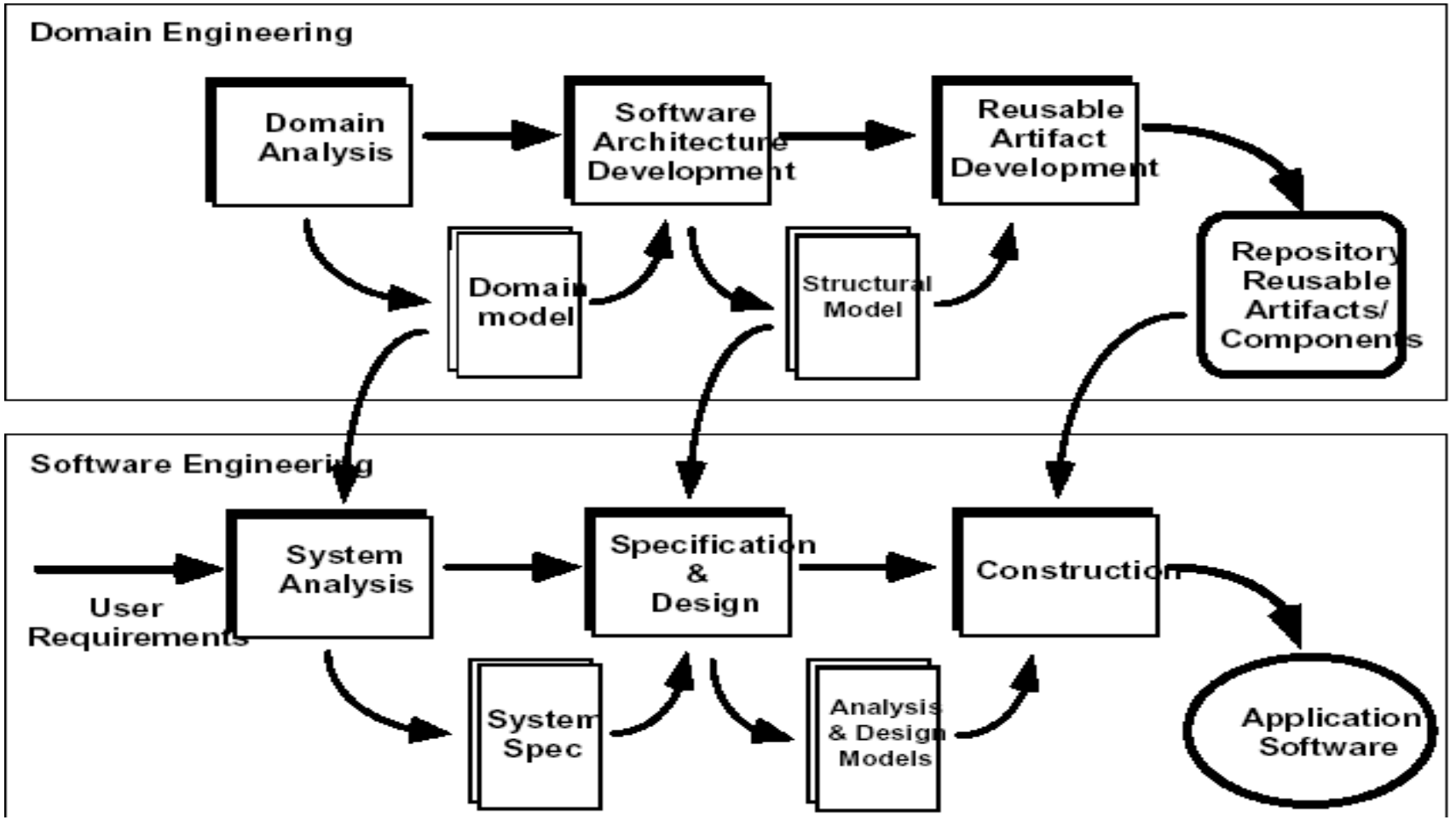
1. Domain Engineering

- Domain engineering menciptakan model domain bagi aplikasi yang akan digunakan untuk menganalisis kebutuhan pengguna.
- Identifikasi, pembangunan, pengelompokan dan pengalokasikan komponen-komponen software supaya bisa digunakan pada sistem yang ada dan yang akan datang.

2. Software Engineering

- Melakukan analisis terhadap domain model yang sudah ditetapkan kemudian menentukan spesifikasi dan merancang berdasarkan model struktur dan spesifikasi sistem.
- Melakukan pembangunan software dengan menggunakan komponen-komponen yang sudah ditetapkan berdasarkan analisis dan rancangan yang dihasilkan sebelumnya hingga akhirnya menghasilkan software.

COMPONENT-BASED SE (Cont)



Kelebihan dan Kekurangan CBSE

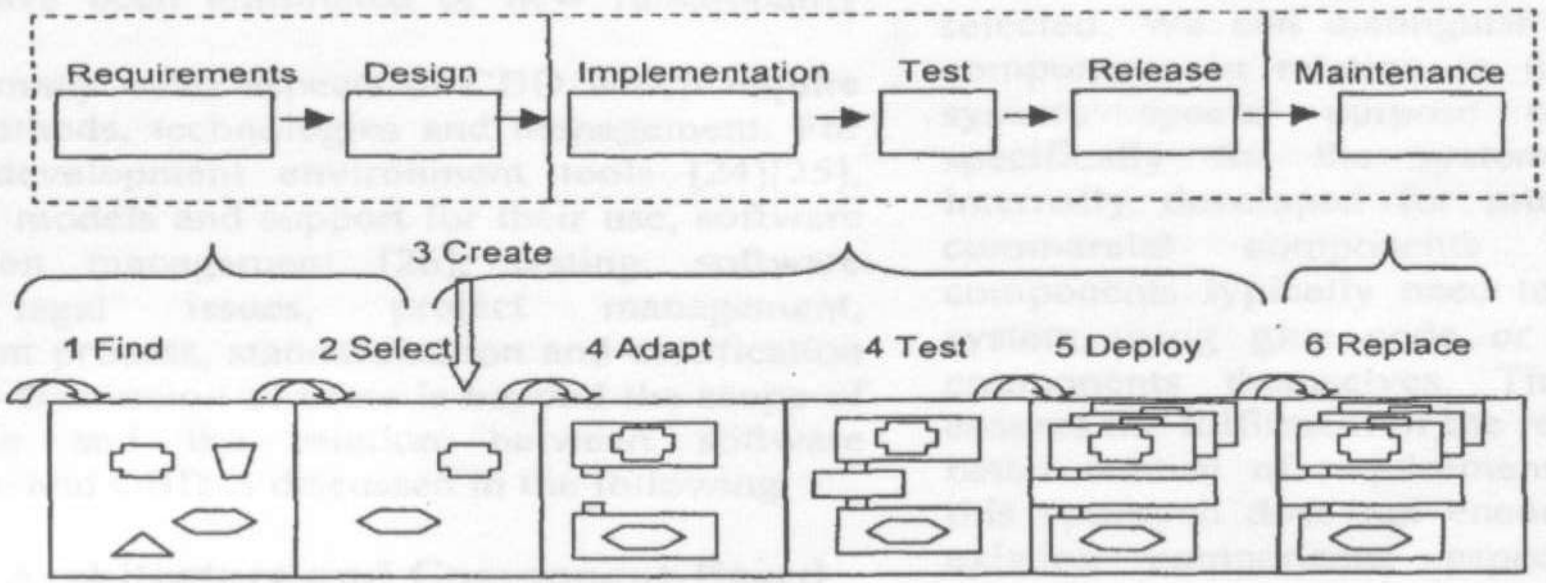
- Kelebihan:
  - Manajemen Kompleksitas
  - Mengurangi Waktu Pembuatan
  - Meningkatkan Produktivitas
  - Meningkatkan Kualitas
- Kekurangan:
  - Pengembangan Komponen terbatas
  - Biaya Pemeliharaan Komponen
  - Keandalan dan Sensitivitas terhadap perubahan

CBSE vs. Traditional

- CBSE Life Cycle is shorter.
- CBSE develops Architecture.
- CBSE is less expensive

CBSE	Waterfall
Find Select	Requirements
	Design
Adapt Test Deploy	Implementation Test Release
Replace	Maintenance

CBSE vs. Traditional (cont)



CBSE vs. Traditional (cont)

- SE dapat memenuhi persyaratan sistem menjadi yang lebih mudah.
- Pemenuhan CBSE persyaratan didasarkan pada komponen yang tersedia
- CBSE tidak memiliki model pengembangan standar seperti UML untuk SE.
- Pemeliharaan jangka panjang sebagian besar tidak diketahui.

IMPLEMENTASI CBSE

- **Binary unit**
  Contoh: *package* di Linux (deb, rpm, sh,dll.), *Win32 application* di Windows (exe), *Java executable* (jar, war, dll.), dan sebagainya

- **Modul atau *script***
  Berisi sekumpulan operasi dan data yang saling terkait, tiap bagiannya berhubungan dengan proses sistem tertentu. Modul yang dimaksud dalam hal ini agak berbeda dengan kelas *object*, dapat dikatakan sebagai servis yang dapat berdiri sendiri.

  Contoh : E-commerce applications & e-catalogue

Key points

- Software processes are the activities involved in producing a software system. Software process models are abstract representations of these processes.
- General process models describe the organization of software processes. Examples of these general models include the ‘waterfall’ model, incremental development, and reuse-oriented development.