Tim Pengajar IF2250

IF2250 – Rekayasa Perangkat Lunak
Pendahuluan

SEMESTER II TAHUN AJARAN 2022/2023

KNOWLEDGE & SOFTWARE ENGINEERING

Sixty years ago no one could have predicted...

- software would enable the creation of new technologies (e.g., genetic engineering and nanotechnology),
- the extension of existing technologies (e.g., telecommunications),
- the radical change in older technologies (e.g., the media);
- software would be the driving force behind the PC revolution;
- software applications would be purchased by consumers using their smart phones;
- software would slowly evolve from a product to a service as “on-demand” software companies deliver just-in-time functionality via a Web browser;
- a software company would become larger and more influential than all industrial-era companies;
- software-driven network would evolve (from library research to consumer shopping /political discourse / the dating habits).

Software

- Software is designed and built by software engineers.
- Software is used by virtually everyone in society.
- Software is pervasive in our commerce, our culture, and our everyday lives.
- Software engineers have a moral obligation to build **reliable** software that does no harm to other people.
- Software engineers view computer software, as being made up of the **programs**, **documents**, and **data** required to design and build the system.
- Software users are only concerned with whether or not software products meet their **expectations** and make their tasks **easier** to complete.

When computer software succeeds?

- when it **meets the needs of the people** who use it,
- when it **performs flawlessly** over a long period of time,
- when it is **easy to modify** and even **easier to use**

it can and does change things for the better.

When software fails?

- when its **users are dissatisfied**,
- when it is **error prone**,
- when it is **difficult to change** and even **harder to use**

bad things can and do happen

Important Questions for Software Engineers

- Why does it take **so long** to get software finished?
- Why are development **costs so high**?
- Why can’t we find all **errors** before we give the software to our customers?
- Why do we spend so much time and effort **maintaining** existing programs?
- Why do we continue to have difficulty in **measuring** progress as software is being developed?

What is software ?

- Definitions:

  Computer programs, procedures, and possibly associated documentation and data pertaining to the operation of a computer system
  *(***IEEE Standard Glossary of Software Engineering Terminology, 1990***)*

Software Characteristics

- Software is both a **product** and a **vehicle** for delivering a product (information).
- Software is **engineered** not manufactured.
- Software does **not wear out**, but it does **deteriorate**.
- Industry is moving toward **component-based software construction**, but most software is still **custom-built**.

Software Application Domains

- **System** software
- **Application** software
- **Engineering** or Scientific Software
- **Embedded** software
- **Product-line** software (includes entertainment software)
- **Web**-Applications
- **Mobile Based** Applications
- **Artificial intelligence** software

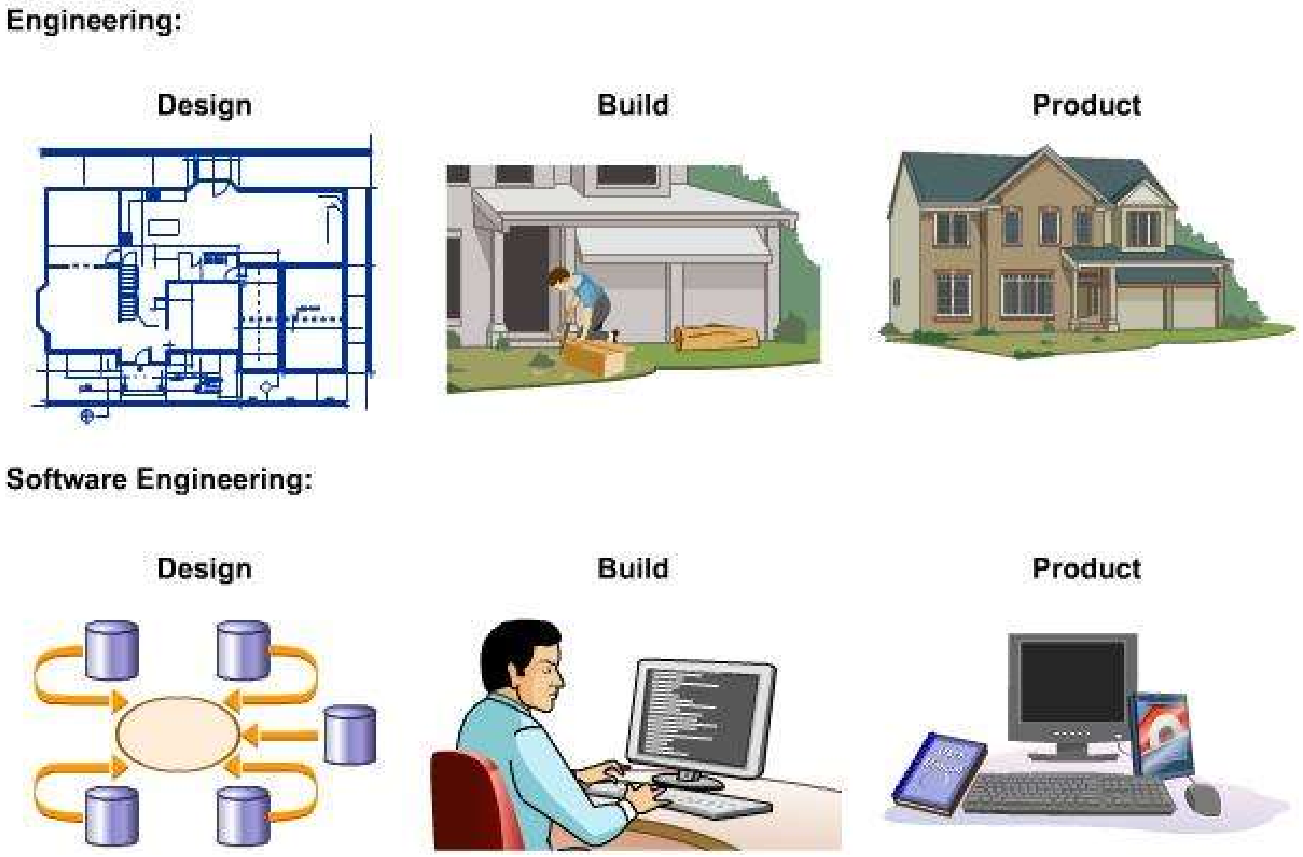
Legacy Software Evolves

- The software must be adapted to meet the needs of new computing environments or technology.
- The software must be enhanced to implement new business requirements.
- The software must be extended to make it interoperable with other more modern systems or databases.
- The software must be re-architected to make it viable within a evolving computing environment.

Software Engineering (I)

- Software engineering is the establishment of sound **engineering principles** in order to obtain **reliable** and **efficient** software in an **economical** manner.
- Software engineering is the application of a **systematic**, **disciplined**, **quantifiable** approach to the **development**, **operation**, and **maintenance** of software.
- Software engineering encompasses a **process**, **management techniques**, **technical methods**, and the **use** of tools.

Software Engineering (2)



Four broad categories of software are evolving to dominate the industry

1. Web-based systems and applications (WebApps)
2. Mobile Applications
3. Cloud computing
4. Product Line Software

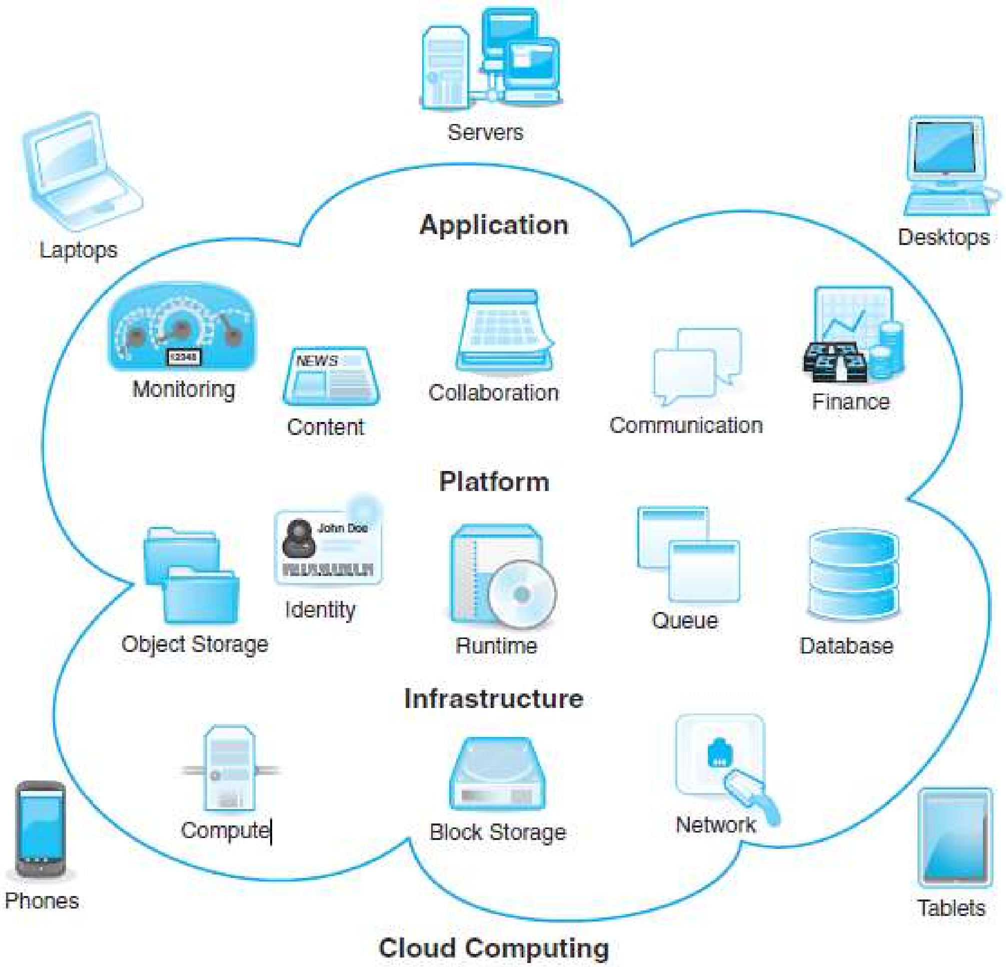
I. Web-based systems and applications

- The augmentation of HTML by development tools (e.g., XML, Java) enabled Web engineers to provide computing capability along with informational content.
- Over the past decade, Semantic Web technologies (Web 3.0) have evolved into sophisticated corporate and consumer applications that encompass "semantic databases [that] provide new functionality that requires Web linking, flexible [data] representation, and external access APIs."
- Sophisticated relational data structures will lead to entirely new WebApps that allow access to disparate information in ways never before possible.

2. Mobile Applications

- The term *app has evolved to connote software that has been specifically designed* to reside on a mobile platform (e.g., iOS, Android, or Windows Mobile).
- encompass a user interface that takes advantage of the unique interaction mechanisms provided by the mobile platform,
- interoperability with Web-based resources

3. Cloud computing

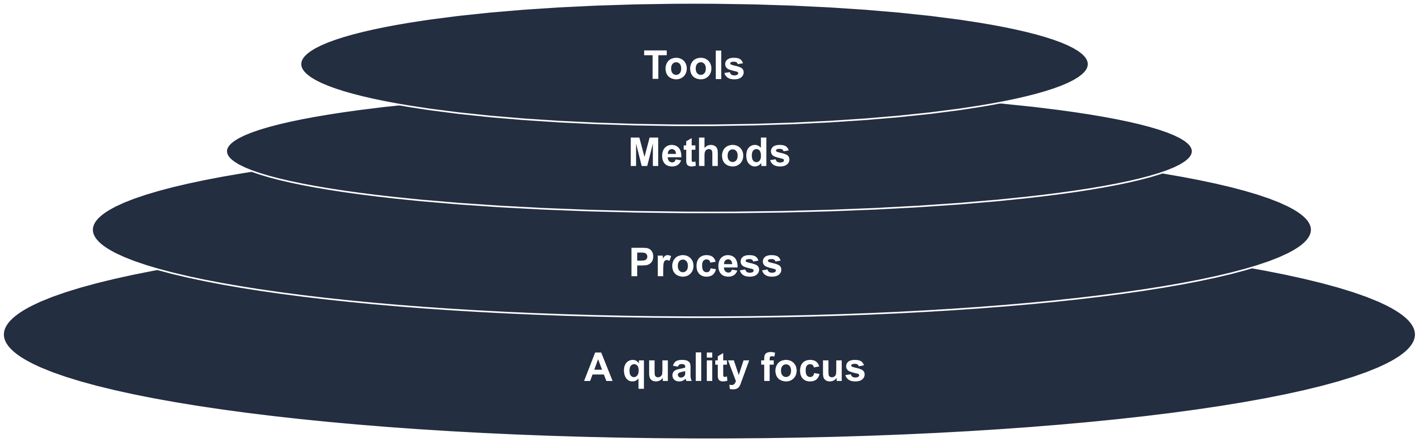


KNOWLEDGE & SOFTWARE ENGINEERING

4. Product Line Software

- The Software Engineering Institute defines a *software product line as "a set of* software-intensive systems that share a common, managed set of features satisfying the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way."
- include requirements, architecture, design patterns, reusable components, test cases, and other software engineering work products

Software Engineering – a layered technology



Tools

Methods

Process

A quality focus

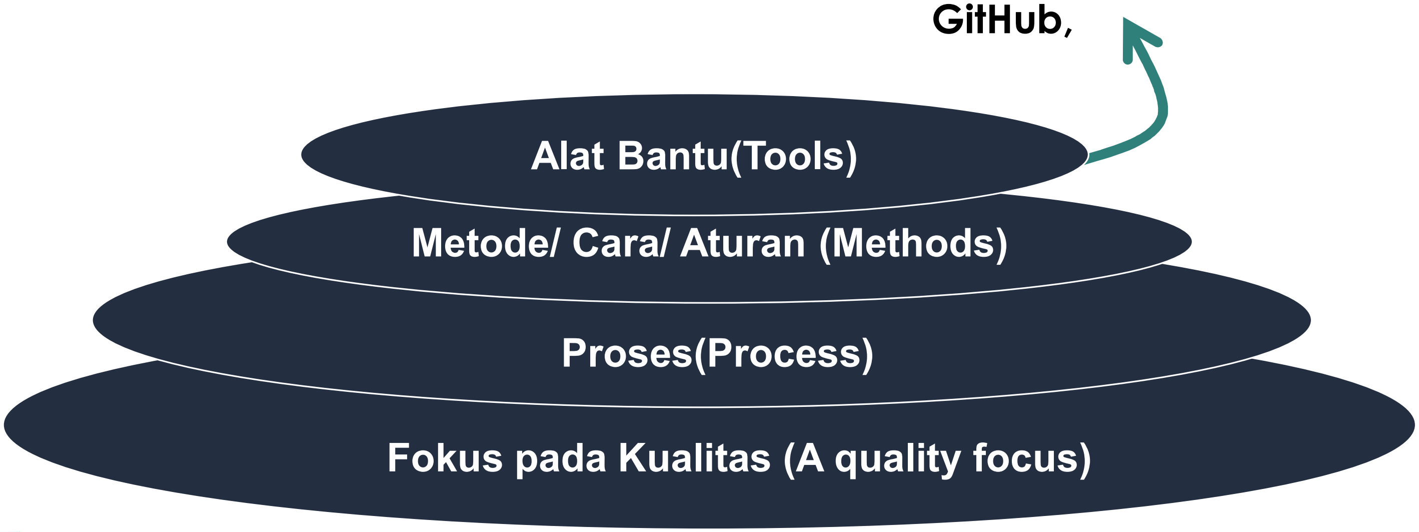
Software Engineering – a layered technology (2)

- The foundation for software engineering is the ***process*** *layer,* defines a framework that must be established for effective delivery of software engineering technology.
- Software engineering ***methods*** *provide the technical how-to's for building* software.
- Software engineering ***tools*** *provide automated or semi-automated support* for the process and the methods
  - *computer-aided software engineering* : e.g Rational Rose; various IDE (Integrated Development Environment) such as: VisualStudio, Eclipse, NetBeans; Software version, such as: CVS, SVN, and GitHub

Lapisan di RPL (I)

Tiap Lapisan tidak bisa berdiri sendiri, masing-masing memiliki ketergantungan antar-lapisan.

- CASE Tool, contoh: Rational Rose,
- Berbagai jenis IDE (Integrated Development Environment) seperti: VisualStudio, Eclipse, NetBeans
- Versi Software, contoh: CVS, SVN, GitHub,

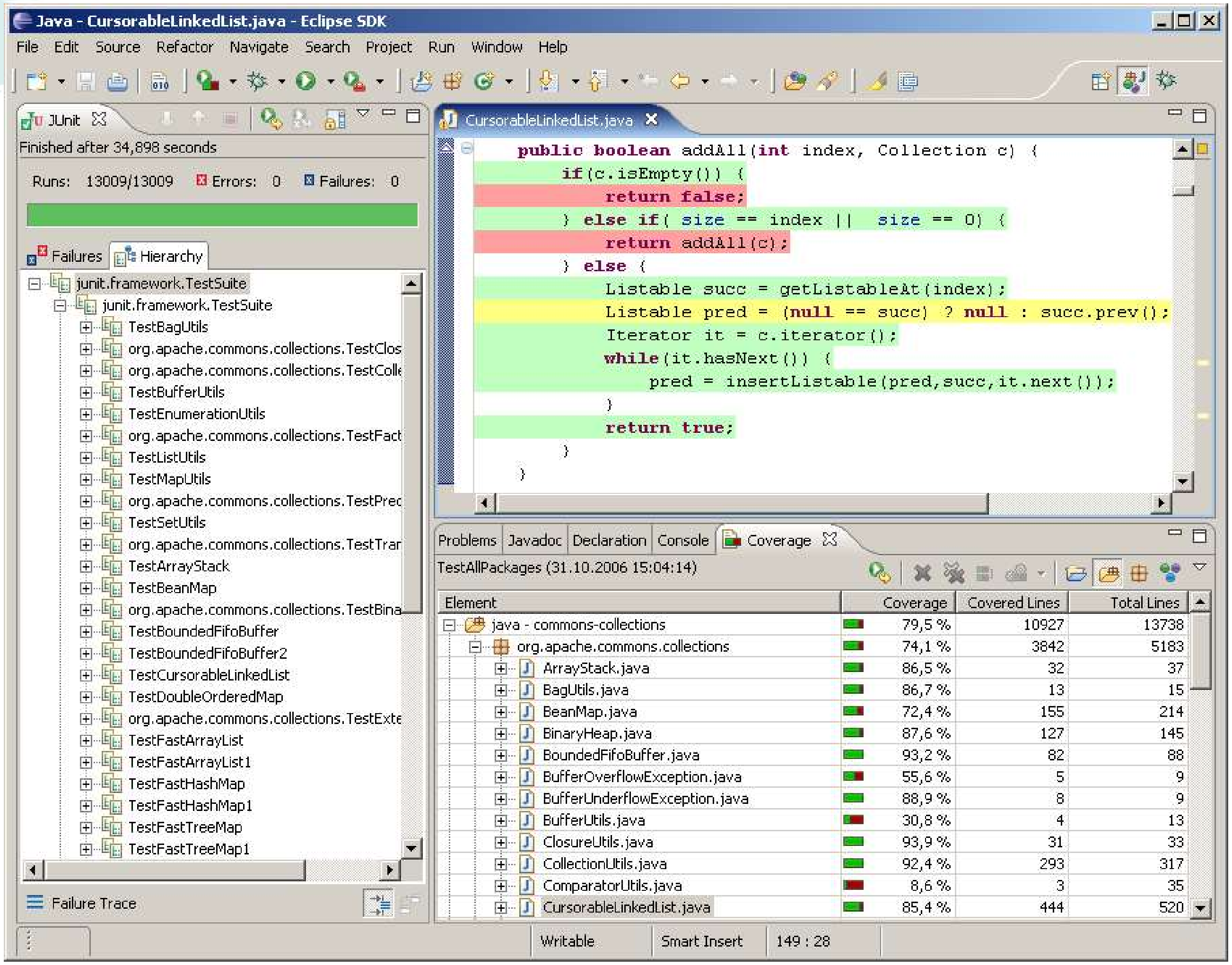

Alat Bantu(Tools)
Metode/ Cara/ Aturan (Methods)
Proses(Process)
Fokus pada Kualitas (A quality focus)

CASE tools (Computer-Aided Software Engineering)

- Software systems that are intended to provide automated support for software process activities
- CASE systems are often used for method support
- Upper-CASE
  - Tools to support the **early process** activities of **requirements** and **design**
- Lower-CASE
  - Tools to support **later activities** such as **programming**, **debugging** and **testing**

**** Software Engineering 7th ed, Ian Sommerville***

Contoh Case Tools untuk source code (IDE Eclipse)


Java - CursorableLinkedList.java - Eclipse SDK
File Edit Source Refactor Navigate Search Project Run Window Help
JUnit
Failures
Hierarchy
Failure Trace
CursorableLinkedList.java
Problems
Javadoc
Declaration
Console
Coverage

Finished after 34,898 seconds

Runs: 13009/13009 Errors: 0 Failures: 0

```
⊟ junit.framework.TestSuite
   ⊟ junit.framework.TestSuite
      ⊞ TestBagUtils
      ⊞ org.apache.commons.collections.TestClos
      ⊞ org.apache.commons.collections.TestColle
      ⊞ TestBufferUtils
      ⊞ TestEnumerationUtils
      ⊞ org.apache.commons.collections.TestFact
      ⊞ TestListUtils
      ⊞ TestMapUtils
      ⊞ org.apache.commons.collections.TestPred
      ⊞ TestSetUtils
      ⊞ org.apache.commons.collections.TestTran
      ⊞ TestArrayStack
      ⊞ TestBeanMap
      ⊞ org.apache.commons.collections.TestBina
      ⊞ TestBoundedFifoBuffer
      ⊞ TestBoundedFifoBuffer2
      ⊞ TestCursorableLinkedList
      ⊞ TestDoubleOrderedMap
      ⊞ org.apache.commons.collections.TestExte
      ⊞ TestFastArrayList
      ⊞ TestFastArrayList1
      ⊞ TestFastHashMap
      ⊞ TestFastHashMap1
      ⊞ TestFastTreeMap
      ⊞ TestFastTreeMap1
```

```java
    public boolean addAll(int index, Collection c) {
        if(c.isEmpty()) {
            return false;
        } else if( size == index ||  size == 0) {
            return addAll(c);
        } else {
            Listable succ = getListableAt(index);
            Listable pred = (null == succ) ? null : succ.prev();
            Iterator it = c.iterator();
            while(it.hasNext()) {
                pred = insertListable(pred,succ,it.next());
            }
            return true;
        }
    }
```

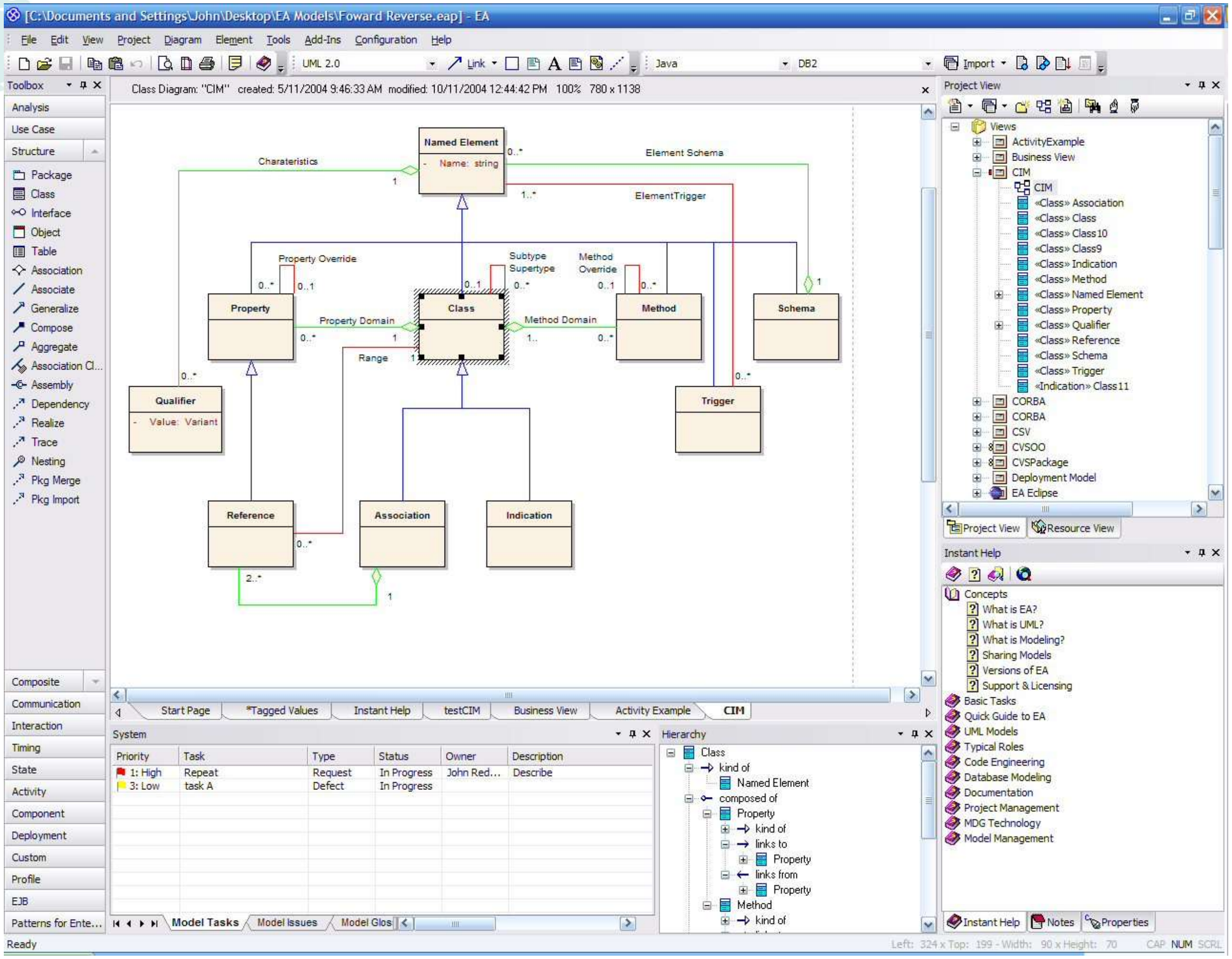
TestAllPackages (31.10.2006 15:04:14)

Element	Coverage	Covered Lines	Total Lines
⊟ java - commons-collections	79,5 %	10927	13738
⊟ org.apache.commons.collections	74,1 %	3842	5183
⊞ ArrayStack.java	86,5 %	32	37
⊞ BagUtils.java	86,7 %	13	15
⊞ BeanMap.java	72,4 %	155	214
⊞ BinaryHeap.java	87,6 %	127	145
⊞ BoundedFifoBuffer.java	93,2 %	82	88
⊞ BufferOverflowException.java	55,6 %	5	9
⊞ BufferUnderflowException.java	88,9 %	8	9
⊞ BufferUtils.java	30,8 %	4	13
⊞ ClosureUtils.java	93,9 %	31	33
⊞ CollectionUtils.java	92,4 %	293	317
⊞ ComparatorUtils.java	8,6 %	3	35
⊞ CursorableLinkedList.java	85,4 %	444	520

Writable | Smart Insert | 149 : 28

KNOWLEDGE & SOFTWARE ENGINEERING

Contoh Case Tools – untuk diagram (IDE Eclipse)



[C:\Documents and Settings\John\Desktop\EA Models\Foward Reverse.eap] - EA

Class Diagram: "CIM" created: 5/11/2004 9:46:33 AM modified: 10/11/2004 12:44:42 PM 100% 780 x 1138

System

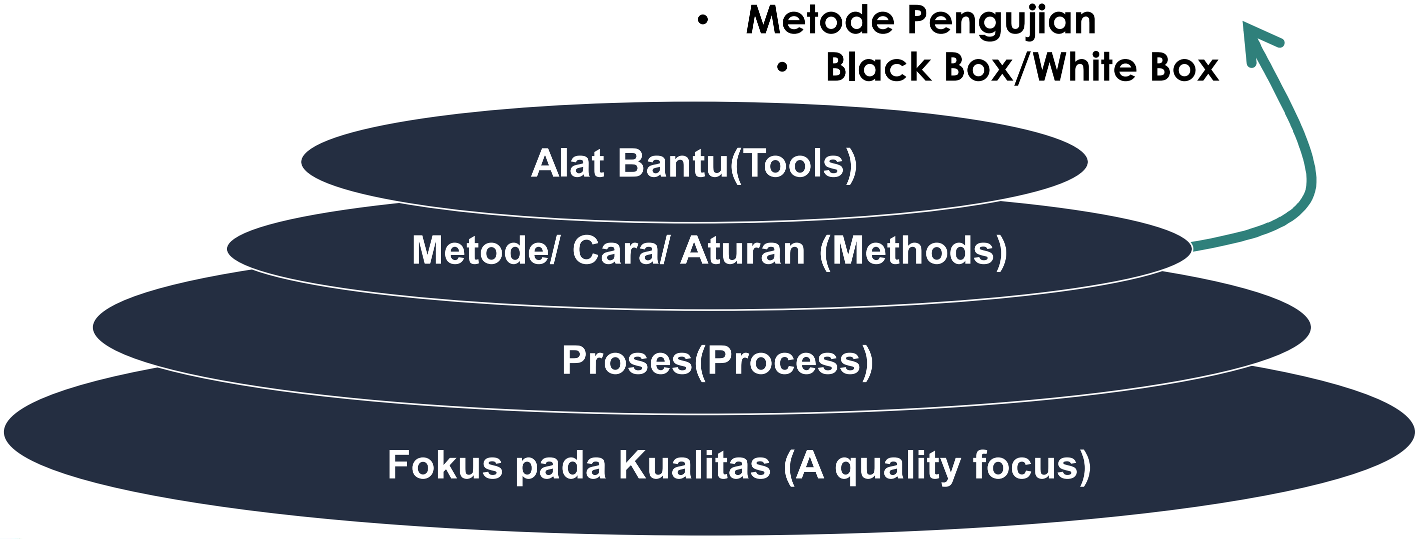
Priority	Task	Type	Status	Owner	Description
🟥 1: High	Repeat	Request	In Progress	John Red...	Describe
🟨 3: Low	task A	Defect	In Progress		

Ready

Left: 324 x Top: 199 - Width: 90 x Height: 70 CAP NUM SCRL

Lapisan di RPL (2)

- Metode Pengumpulan Kebutuhan Pengguna
  - Goal Oriented, Viewpoints, dll
- Metode Analisis
  - Terstruktur/OO
- Metode Perancangan
  - Terstruktur/OO
- Metode Pengujian
  - Black Box/White Box


Alat Bantu(Tools)
Metode/ Cara/ Aturan (Methods)
Proses(Process)
Fokus pada Kualitas (A quality focus)

What are *software engineering methods*?

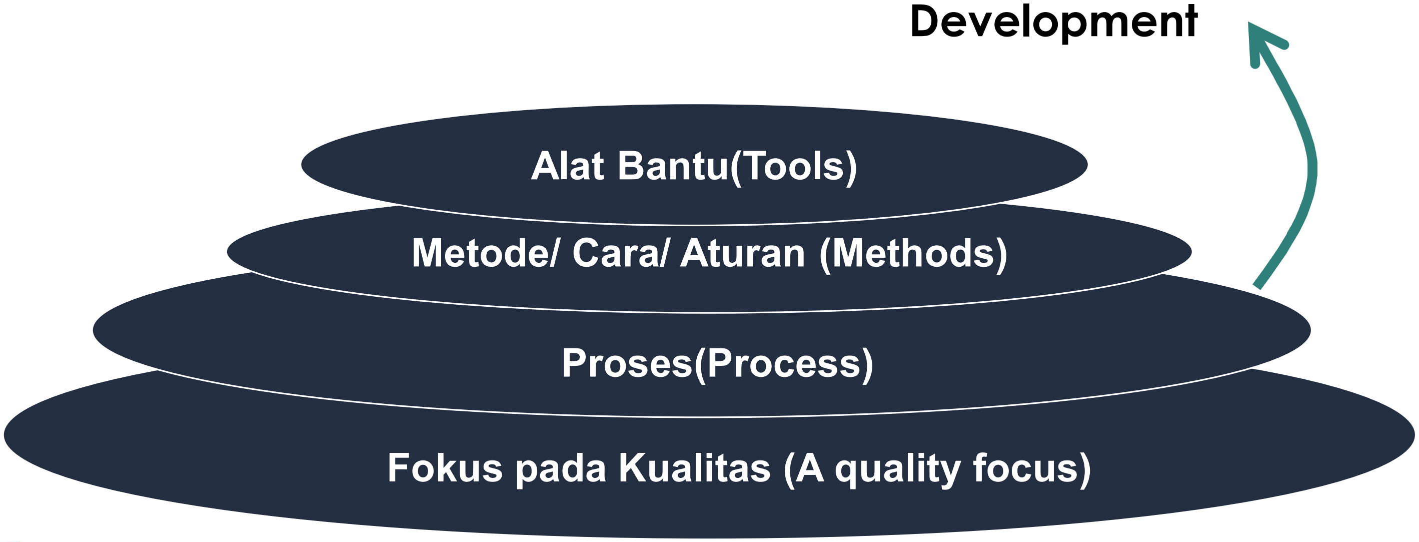
- Structured approaches to software development which include system models, notations, rules, design advice and process guidance.

- Model descriptions
  - Descriptions of graphical models which should be produced
- Rules
  - Constraints applied to system models
- Recommendations
  - Advice on good design practice
- Process guidance
  - What activities to follow

***** Software Engineering 7th ed, Ian Sommerville**

Lapisan di RPL (3)

- **Waterfall Model**
- **Incremental Model/Incremental Process**
- **Spiral model**
- **Agile Development**
- **Rapid Application Development**



Alat Bantu(Tools)

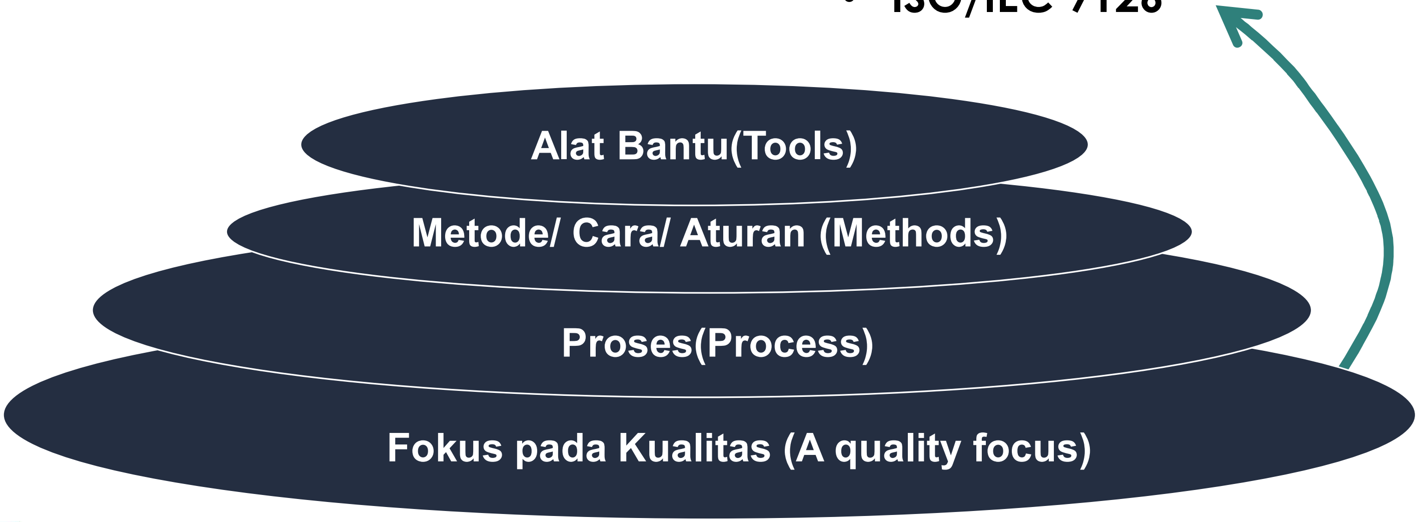
Metode/ Cara/ Aturan (Methods)

Proses(Process)

Fokus pada Kualitas (A quality focus)

Lapisan di RPL (4)

- Six Sigma
- Total Quality Management
- CMM (Capability Maturity Model)
- ISO/IEC 9126


Alat Bantu(Tools)
Metode/ Cara/ Aturan (Methods)
Proses(Process)
Fokus pada Kualitas (A quality focus)

The Software Process

- THE PROCESS FRAMEWORK
- UMBRELLA ACTIVITIES
- PROCESS ADAPTATION

Generic Software Process Framework

- **Communication**
  - System analyst vs User
  - System analyst vs Programmer
- **Planning**
  - Cost, Time, human resources
- **Modeling**
  - Structured approach
  - Object oriented approach
- **Construction**
  - Coding and Testing
- **Deployment**
  - Software delivery to customer

KNOWLEDGE & SOFTWARE ENGINEERING

Umbrella Activities

- **Software project tracking and control**
  - allows the software team to **assess progress** against the project plan and **take** any necessary **action** to maintain the schedule.
- **Risk management**
  - **assesses risks** that may affect the **outcome** of the project or the **quality** of the product.
- **Software quality assurance**
  - defines and conducts the activities required to **ensure** software **quality**.
- **Technical reviews**
  - assesses software engineering work products in an effort to **uncover** and **remove errors** before they are propagated to the next activity.

Umbrella Activities

- **Measurement**
  - defines and collects **process**, **project**, and **product measures** that assist the team in delivering software that meets **stakeholders' needs**; can be used in conjunction with all other framework and umbrella activities.
- **Software configuration management**
  - manages the **effects** of **change** throughout the software process.
- **Reusability management**
  - defines **criteria** for work product **reuse** (including software components) and establishes mechanisms to **achieve reusable** components.
- **Work product preparation and production**
  - encompasses the activities required to create work products such as **models**, **documents**, **logs**, **forms**, and **lists**.

Process Adaptation

- The software engineering process should be agile and adaptable
  - to the problem,
  - to the project,
  - to the team, and
  - to the organizational culture
- A process adopted for one project might be significantly different than a process adopted for another project.

The essence of software engineering practice

1. Understand the problem (communication and analysis).
2. Plan a solution (modeling and software design).
3. Carry out the plan (code generation).
4. Examine the result for accuracy (testing and quality assurance).

Software Practice Core Principles

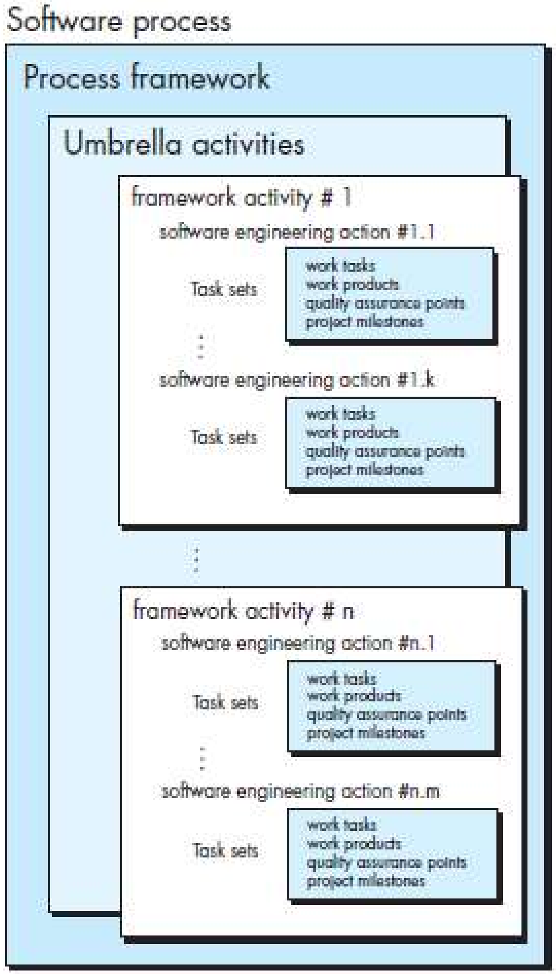
- **The reason it all exist**
  - Software exists **to provide value** to its users
- **Keep it simple stupid (KISS)**
  - Keep the design as **simple as possible**, but not simpler
- **Maintain the vision**
  - **Clear vision** is essential to the success of any software project
- **We produce, others will consume**
  - Always specify, design, and implement knowing that **someone** else **will have to understand** what you **have done** to **carry out** his or her tasks
- **Open to the future**
  - Be **open to future changes**, don't code yourself into a corner
- **Plan for Reuse!**
  - Planning ahead for **reuse** reduces the cost and increases the value of both the reusable components and the systems that require them
- **Think First!**
  - Placing **clear** complete **thought** before any action almost always produces better results

Software Process Structure

PROCESS FLOW

KNOWLEDGE & SOFTWARE ENGINEERING

A Software Process Framework



Software process

Process framework

Umbrella activities

framework activity # 1

software engineering action #1.1

Task sets

work tasks
work products
quality assurance points
project milestones

software engineering action #1.k

Task sets

work tasks
work products
quality assurance points
project milestones

framework activity # n

software engineering action #n.1

Task sets

work tasks
work products
quality assurance points
project milestones

software engineering action #n.m

Task sets

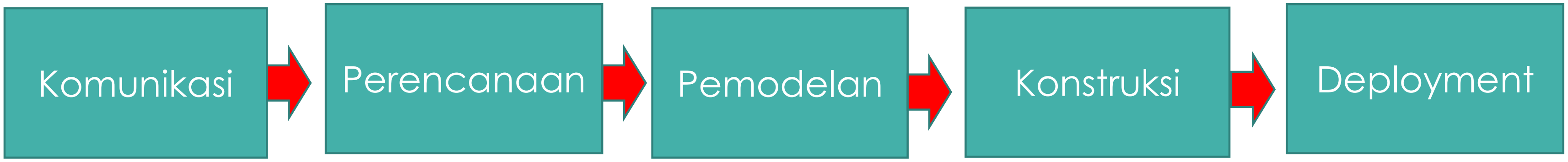
work tasks
work products
quality assurance points
project milestones

One additional aspect of the software process: Process flow

- Process framework:
  - communication,
  - planning,
  - modeling,
  - construction,
  - deployment ,
  - umbrella activities + process flow
- Organized with respect to sequence and time
  - *linear process flow*
  - *iterative process flow*
  - *evolutionary process flow*
  - *parallel process flow*

Process Flow (I)

Alur Proses Linear *(Linear Process Flow)*



Komunikasi → Perencanaan → Pemodelan → Konstruksi → Deployment

Komunikasi
- Inisiasi Proyek
- Pengumpulan Kebutuhan

Perencanaan
- Estimasi
- Penjadwalan
- Tracking

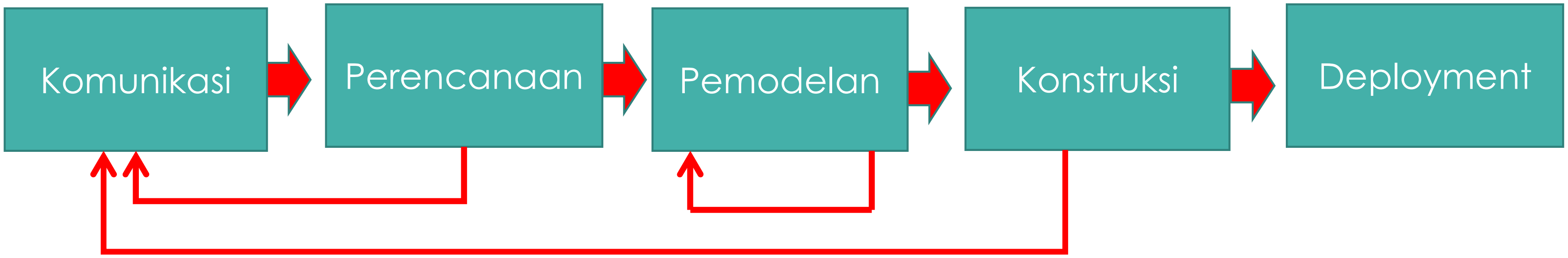
Pemodelan
- Analisis
- Desain

Konstruksi
- Pemrograman
- Pengujian

Deployment
- Delivery
- Dukungan
- Feedback

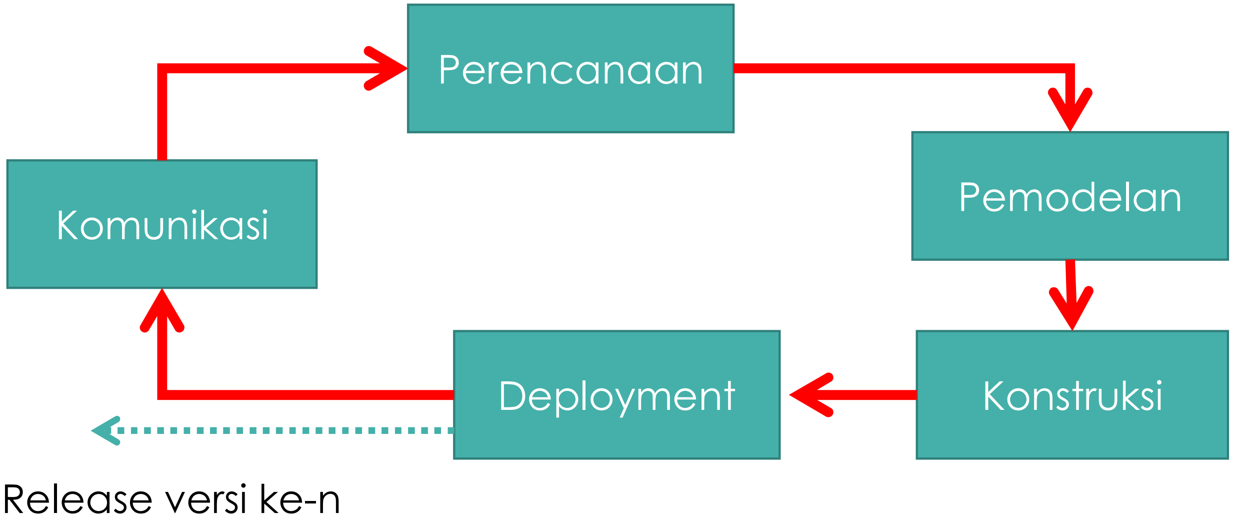
Process Flow (2)

Alur Proses Iteratif *(Iterative Process Flow)*



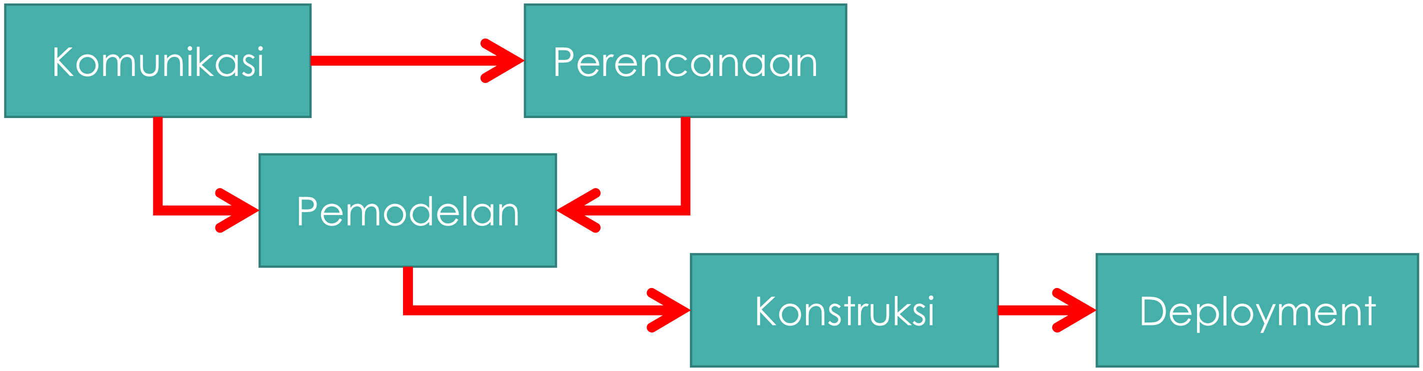
Process Flow (3)

Alur Proses Berevolusi *(Evolutionary process flow)*



Process Flow (4)

Alur Proses Paralel *(Parallel process flow)*

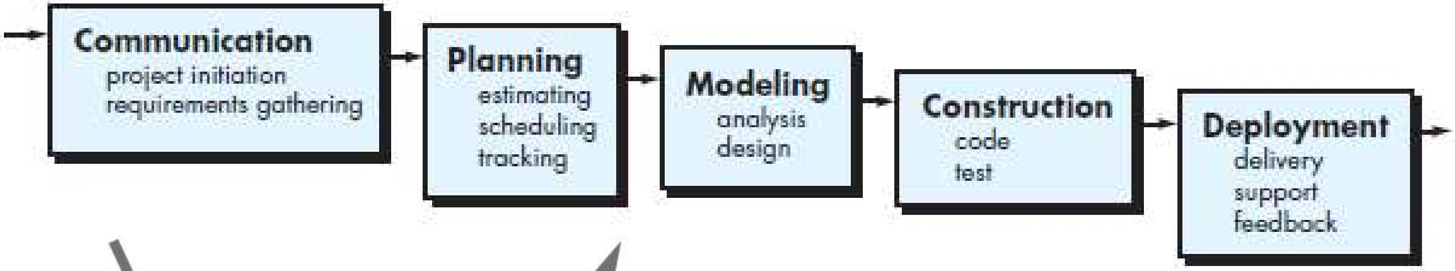


Process Models

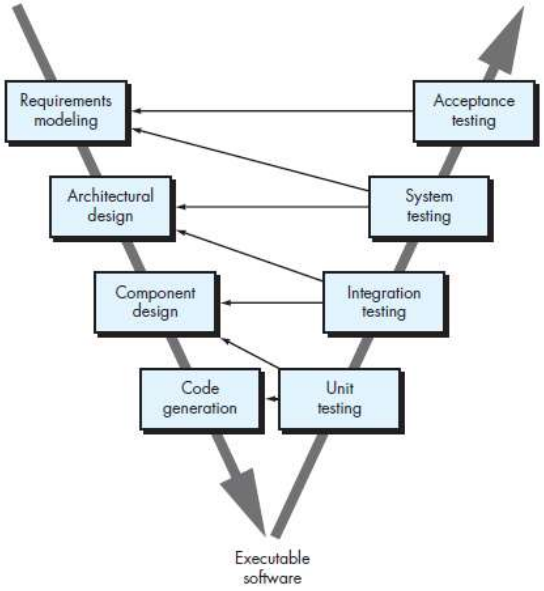
- PRESCRIPTIVE PROCESS MODELS
- SPECIALIZED PROCESS MODELS
- UNIFIED PROCESS

Prescriptive Process Models

- **The Waterfall Model -** *classic life cycle*



- **The V-model**

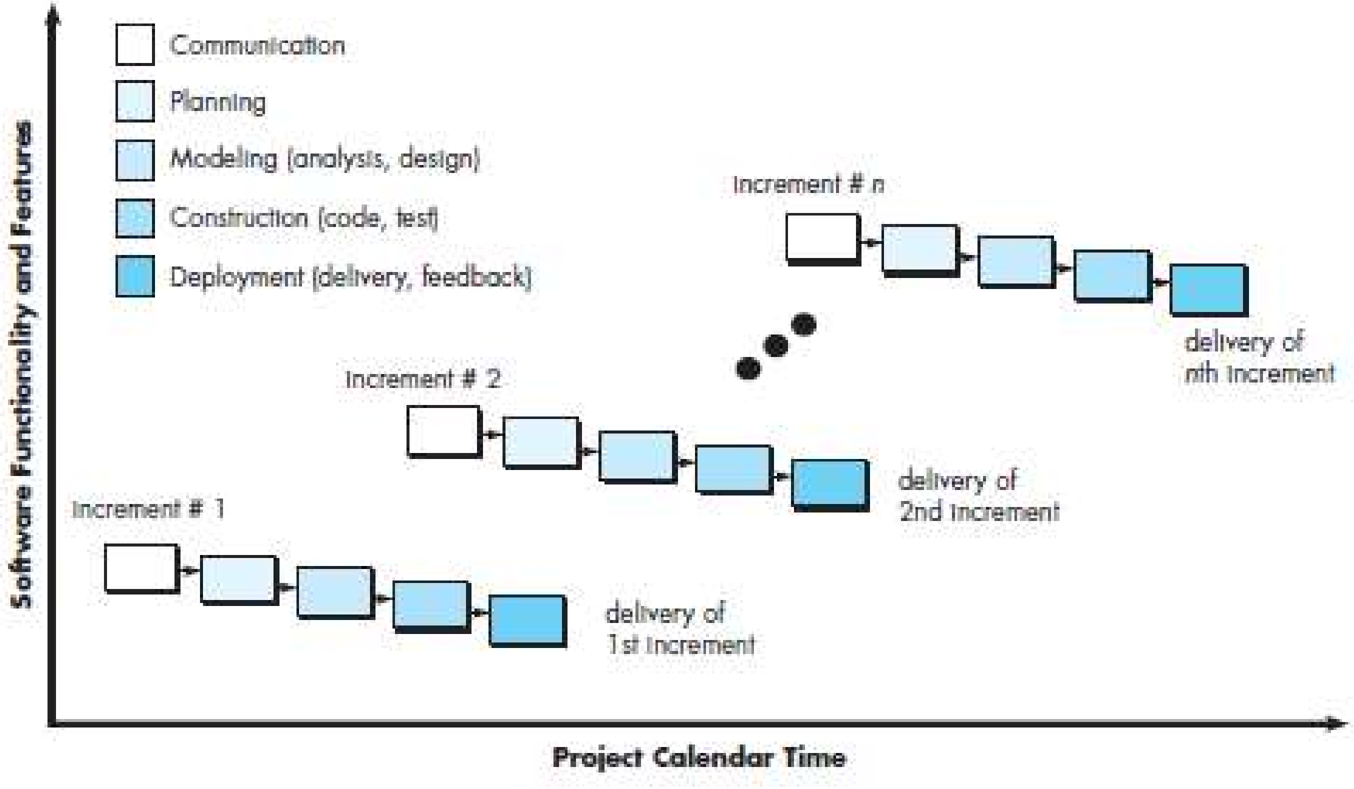


Karakteristik Waterfall

- Proses dijalankan secara sekuensial dari pengumpulan kebutuhan hingga perawatan
- Cocok untuk sistem yang sudah terdefinisi baik atau sistem yang mengutamakan keselamatan (safety)
  - Pengembangan auto-pilot untuk pesawat harus jelas dan lengkap di awal, jadi program harus sudah lengkap tidak bisa hanya sebagian yang di instalasi di pesawat.

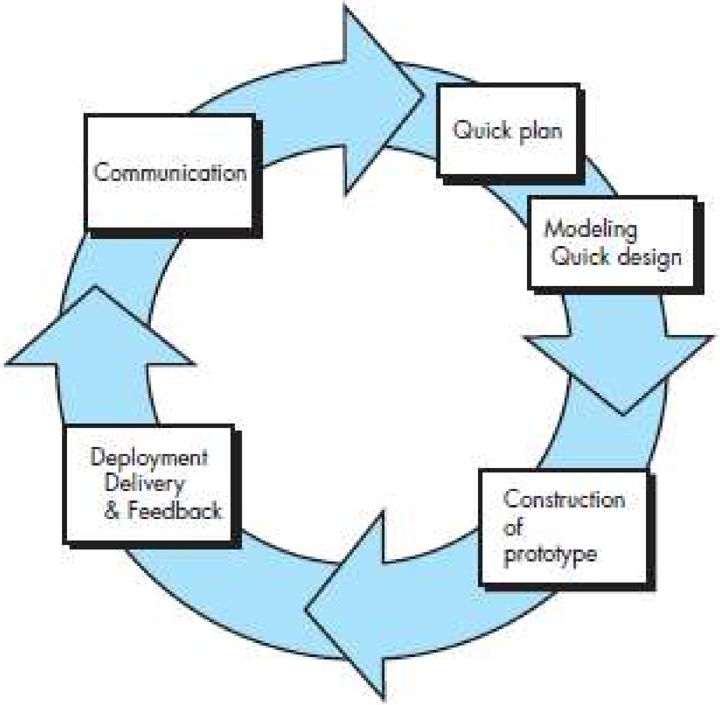
Prescriptive Process Models (2)

- Incremental Process Models



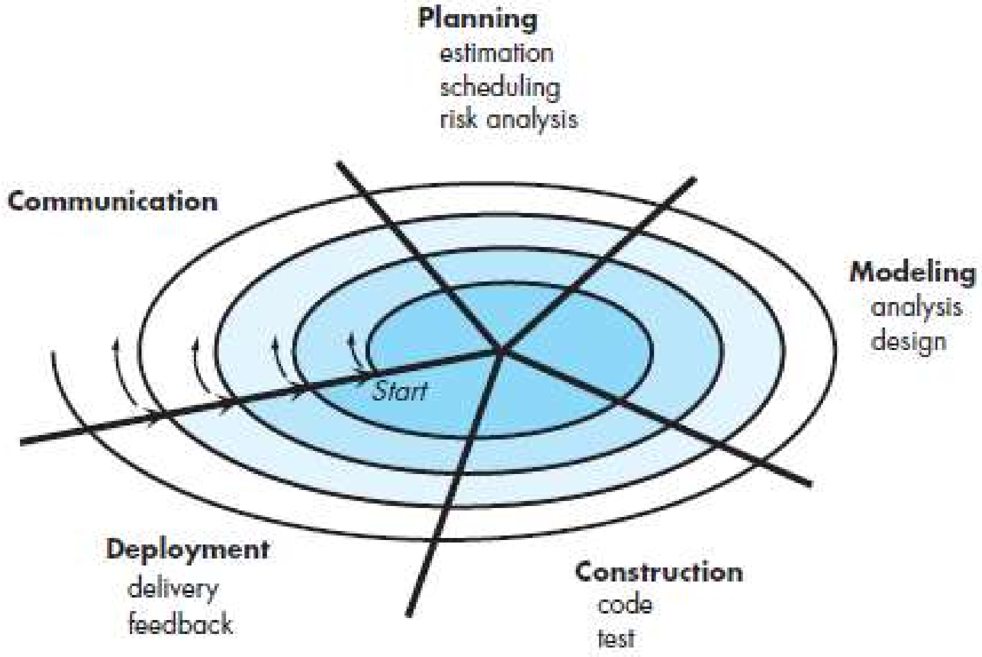
Prescriptive Process Models (3)

- **Evolutionary Process Models – prototyping paradigm**



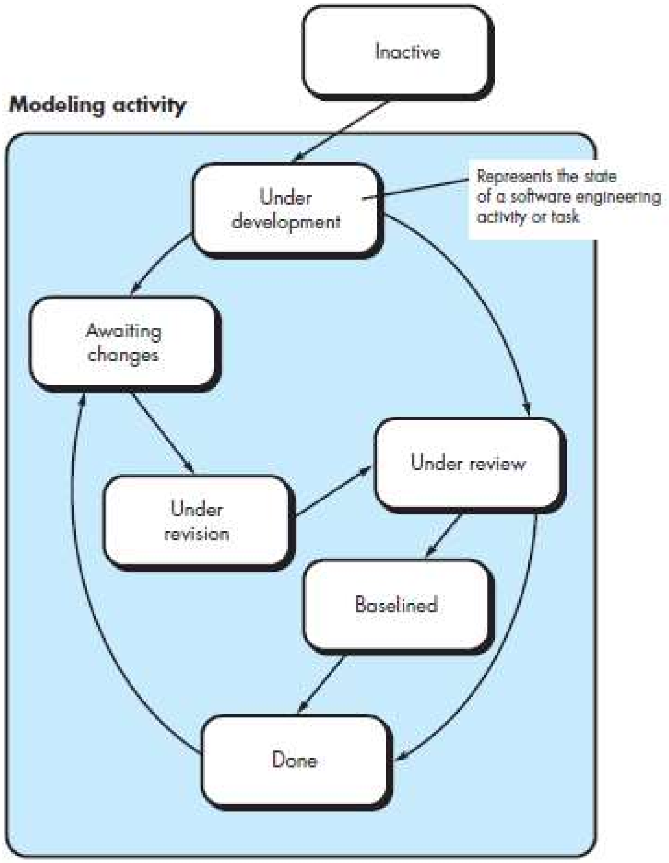
Prescriptive Process Models (4)

- Evolutionary Process Models – the Spiral Model



Prescriptive Process Models (5)

- Concurrent Models



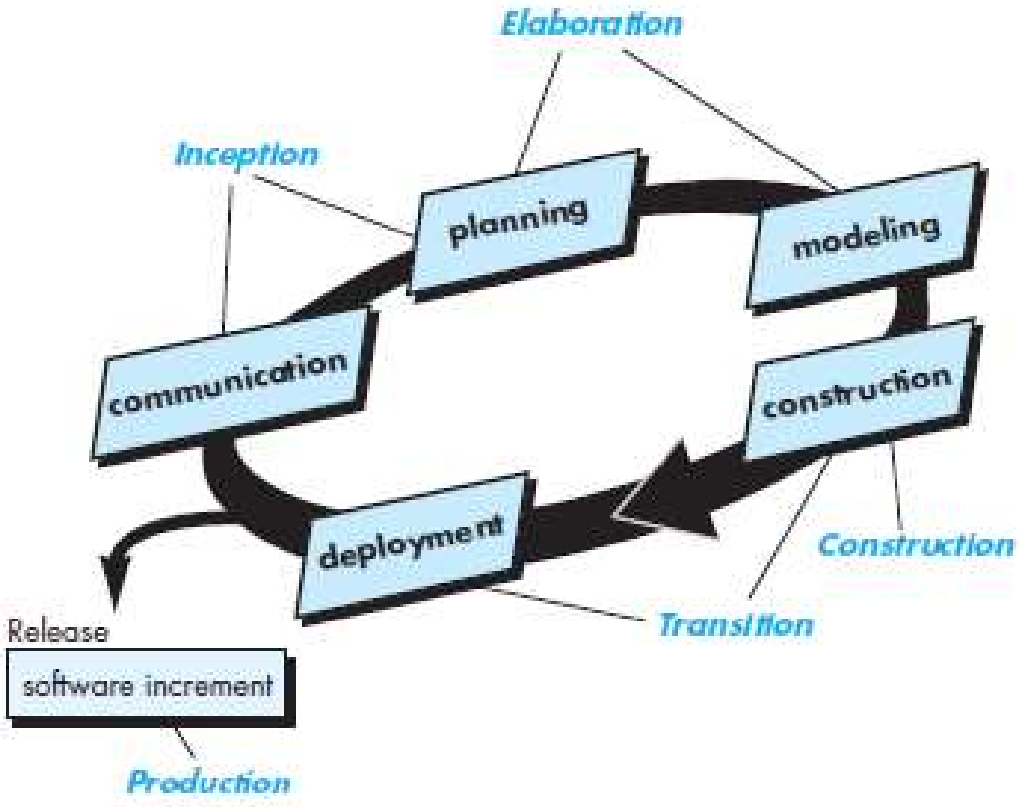
Specialized process models

- **Component-Based Development -** comprises applications from prepackaged software components.
- **The Formal Methods Model**
  - encompasses a set of activities that leads to formal mathematical specification of computer software, enable you to specify, develop, and verify a computer-based system by applying a rigorous, mathematical notation.
  - the formal methods approach has gained adherents among software developers who must build safety-critical software (e.g., developers of aircraft avionics and medical devices) and among developers that would suffer severe economic hardship should software errors occur
- **Aspect-Oriented Software Development**
  - often referred to as *aspect-oriented programming (AOP) or aspect-oriented component engineering*
  - a relatively new software engineering paradigm that provides a process and methodological approach for defining, specifying, designing, and constructing *aspects —"mechanisms beyond subroutines and inheritance for* localizing the expression of a crosscutting concern"

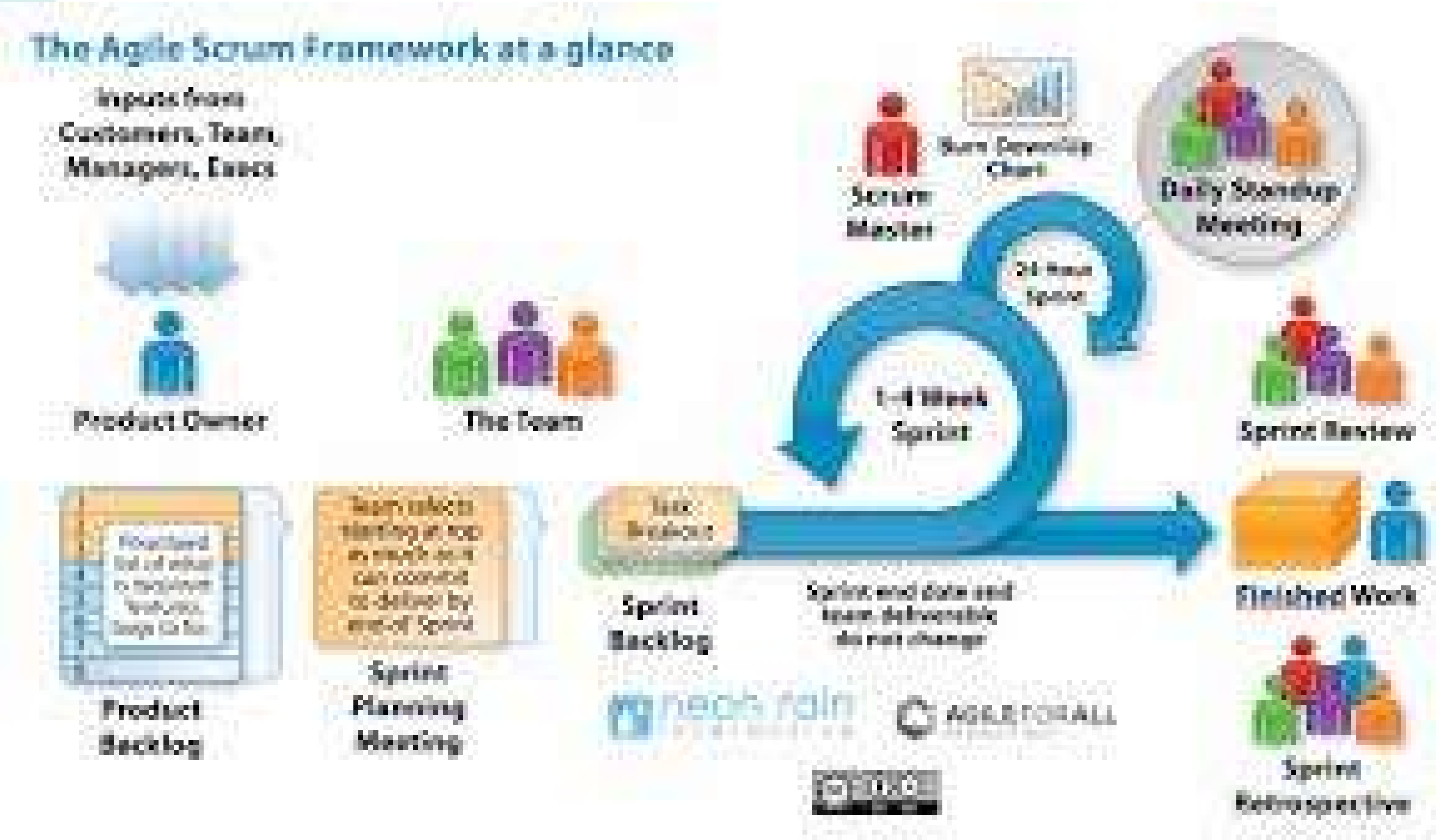
Unified Process

- a "use case driven, architecture-centric, iterative and incremental"
- The result was UML—a *unified modeling*
- *language that contains a robust notation for the modeling and development of* object- oriented systems.

Unified Process (2)



Agile - Scrum


The Agile Scrum Framework at a glance
Inputs from Customers, Team, Managers, Execs
Product Owner
The Team
Prioritized list of what is required: features, bugs to fix...
Product Backlog
Team selects starting at top as much as it can commit to deliver by end of Sprint
Sprint Planning Meeting
Task Breakout
Sprint Backlog
Scrum Master
Burn Down/Up Chart
Daily Standup Meeting
24 Hour Sprint
1-4 Week Sprint
Sprint end date and team deliverable do not change
Sprint Review
Finished Work
Sprint Retrospective
neon rain
AGILEFORALL

System Engineering vs Software Engineering

KNOWLEDGE & SOFTWARE ENGINEERING

System – Definition

Webster's Dictionary

- A set or arrangement of things so related as to form a unity or organic whole
- A set of facts, principles, rules, etc., classified and arranged in an orderly form so as to show a logical plan linking the various parts
- A method or plan of classification or arrangement
- An established way of doing something; method; procedure….
- …..
- ….

Computer-Based Systems [PRE2007]

- *A set or arrangement of elements that are organized to accomplish some predefined goal by **processing information***
- The goal:

  To support some business function or to develop a product that can be sold to **generate business revenue**
- To accomplish the goal, a computer-based system makes use of a variety of system elements

Computer-Based System Elements

- Software
- Hardware
- People
- Data
- Documentation
- Procedures

**** SEPA 6th ed, Roger S. Pressman***

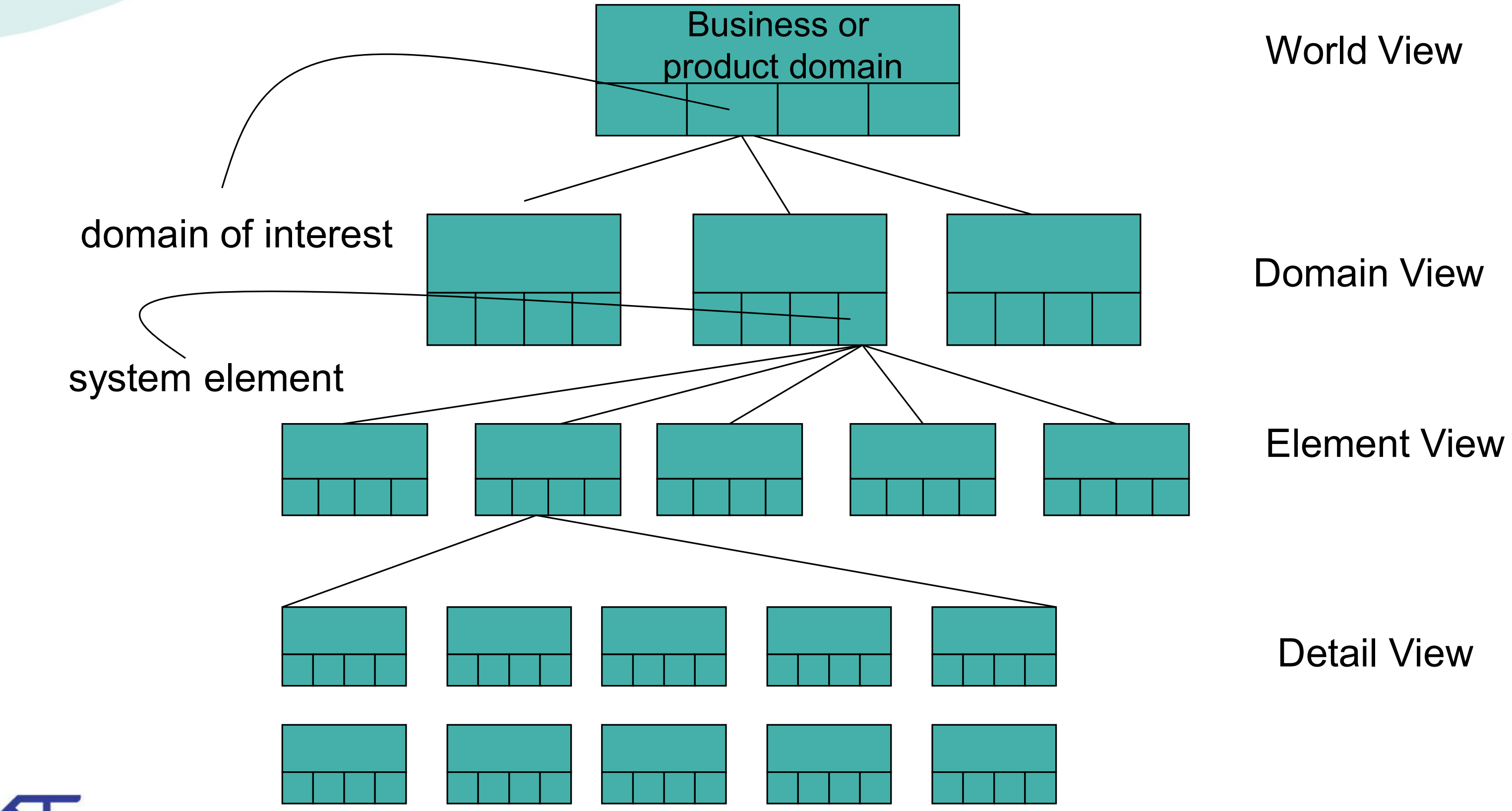
KNOWLEDGE & SOFTWARE ENGINEERING

System Engineering Hierarchy

- World view → WV = $\{D_1, D_2, D_3, \ldots, D_n\}$
  - Composed of a set of domains (D_i) which can be each be a system or system of systems
- Domain view → DV = $\{E_1, E_2, E_3, \ldots, E_m\}$
  - Composed of specific elements (E_j) each of which serves some role in accomplishing the objective and goals fo the domain or component
- Element view → EV = $\{C_1, C_2, C_3, \ldots, C_k\}$
  - Each element is implemented by specifying the technical component (C_k) that achieve the necessary function for an element
- Detail view

***** SEPA 6th ed, Roger S. Pressman**

System Engineering Hierarchy

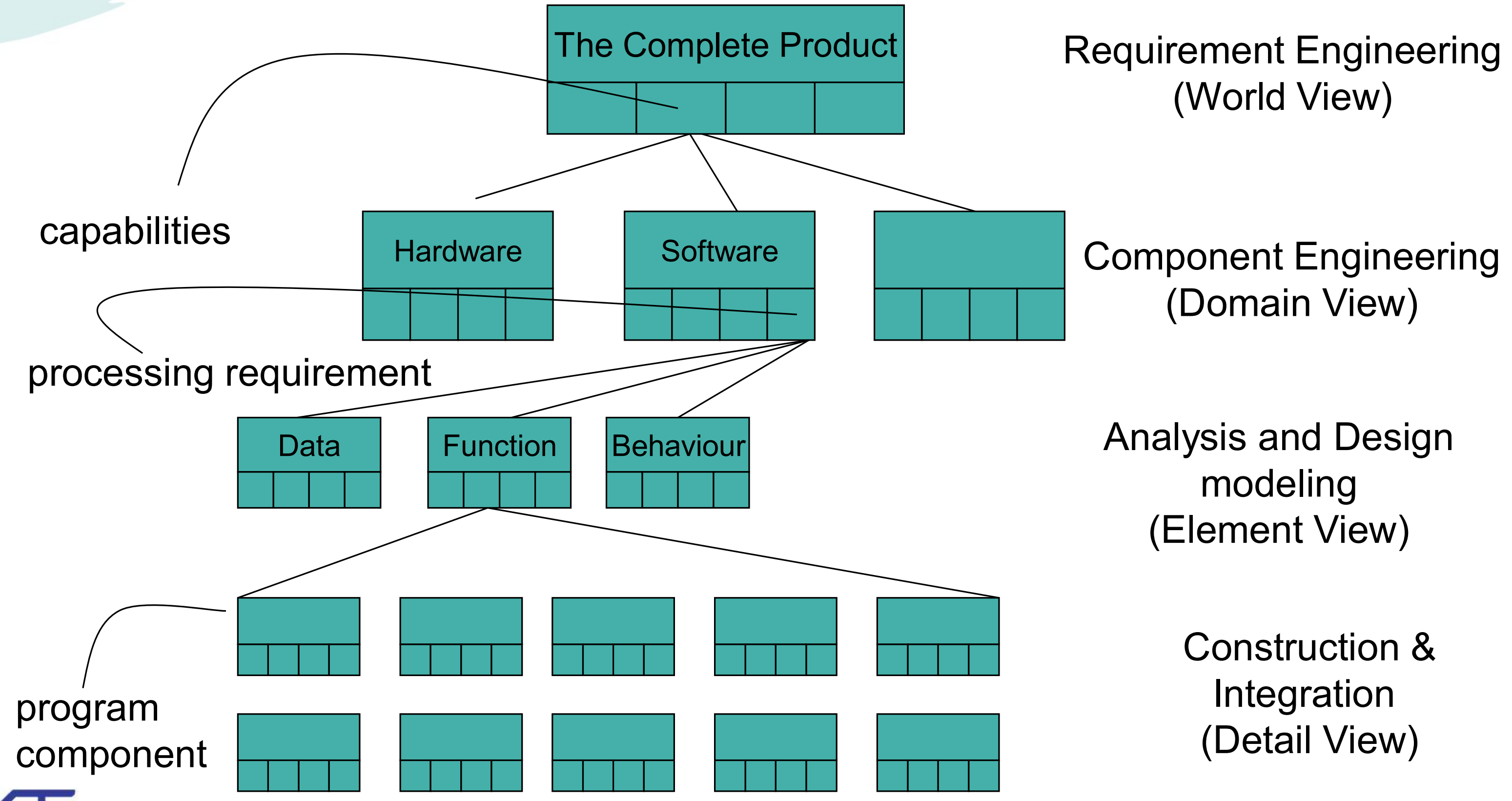


Product Engineering

- Goal
  - to translate the customer's desire for a set of defined capabilities into a working product
- Hirarchy
  - Requirements engineering (world view)
  - Component engineering (domain view)
  - Analysis and Design modeling (element view - software engineers)
  - Construction and Integration (detailed view - software engineers)

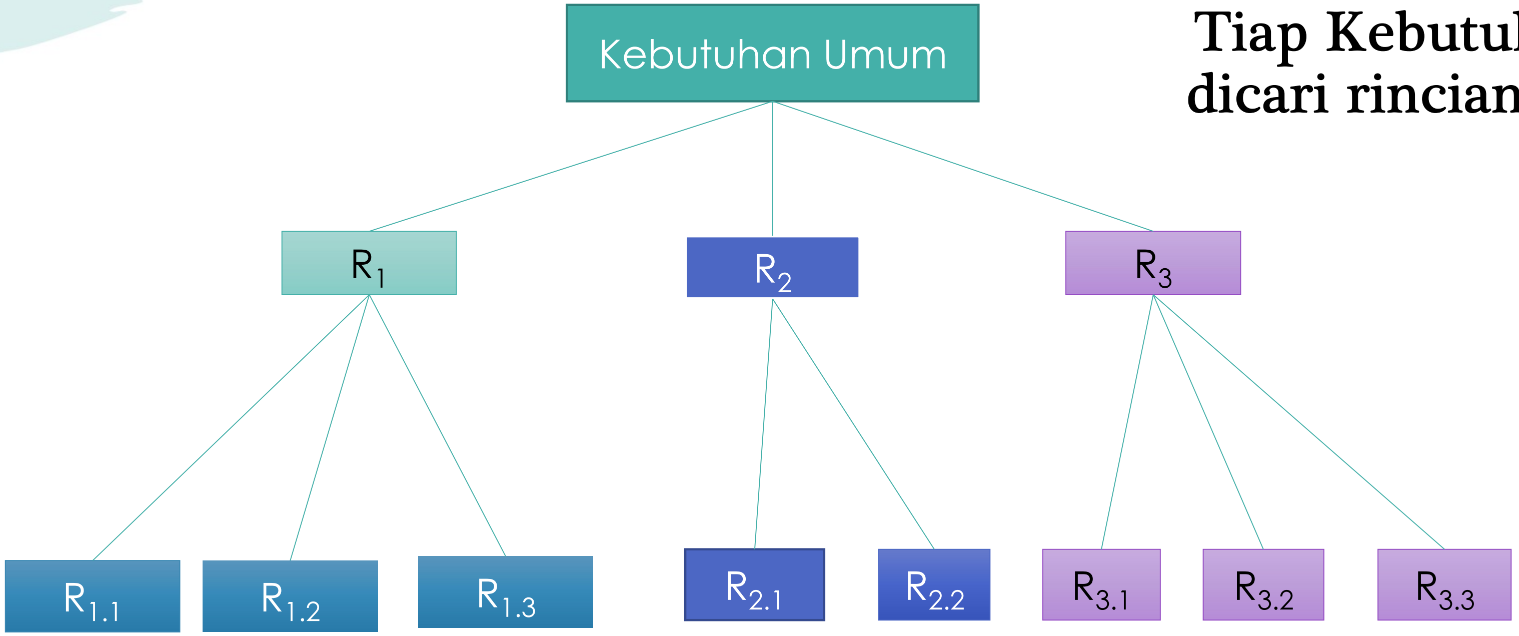
*** SEPA 6th ed, Roger S. Pressman**

The Product Engineering Hierarchy



Aktivitas Pengembangan Perangkat Lunak

Pengumpulan Kebutuhan(Requirements)

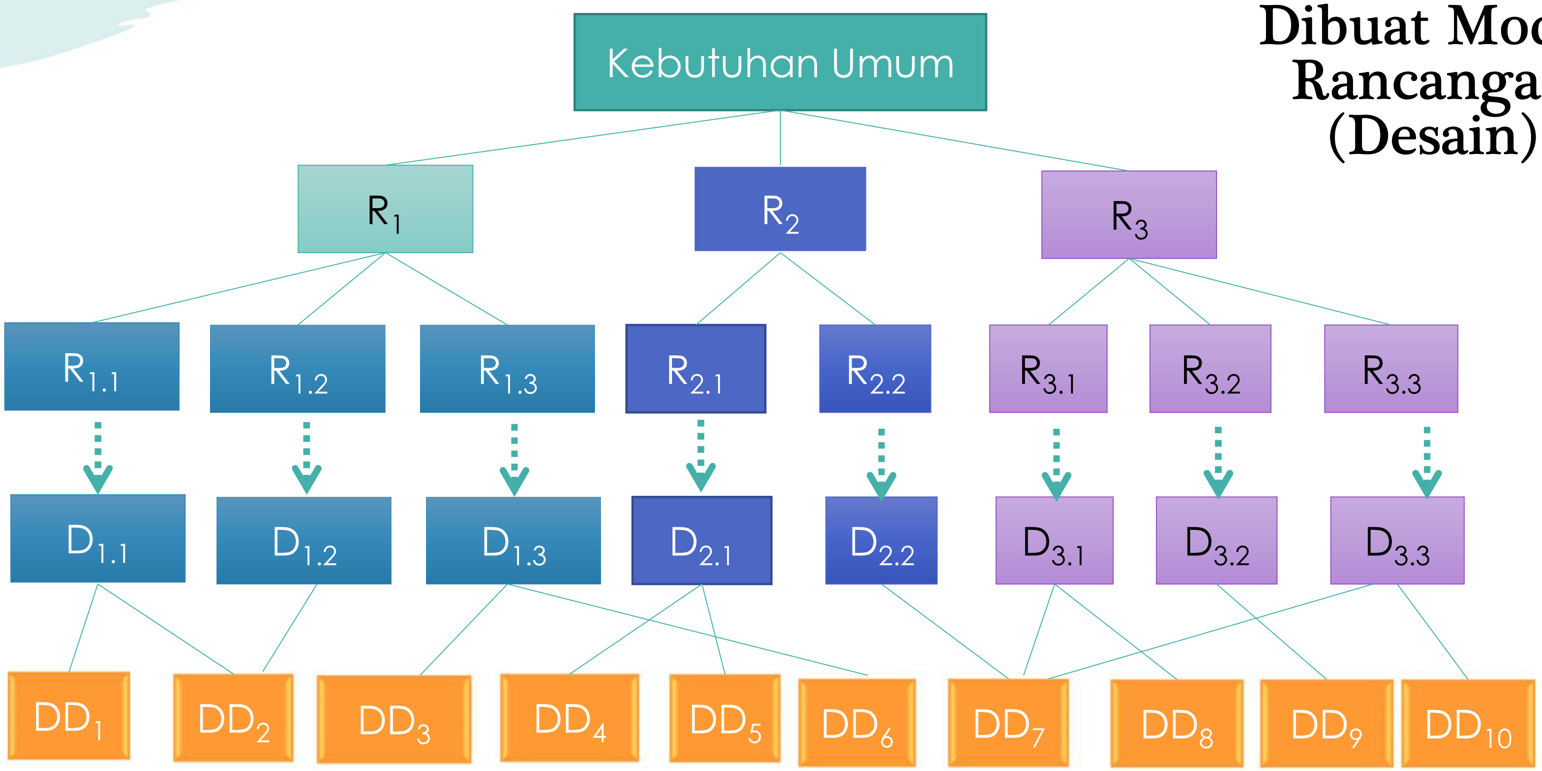


Tiap Kebutuhan dicari rinciannya

Hingga cukup detil!

Tapi sampai kapan kita memecah kebutuhan?

Dari Hasil Pengumpulan Kebutuhan

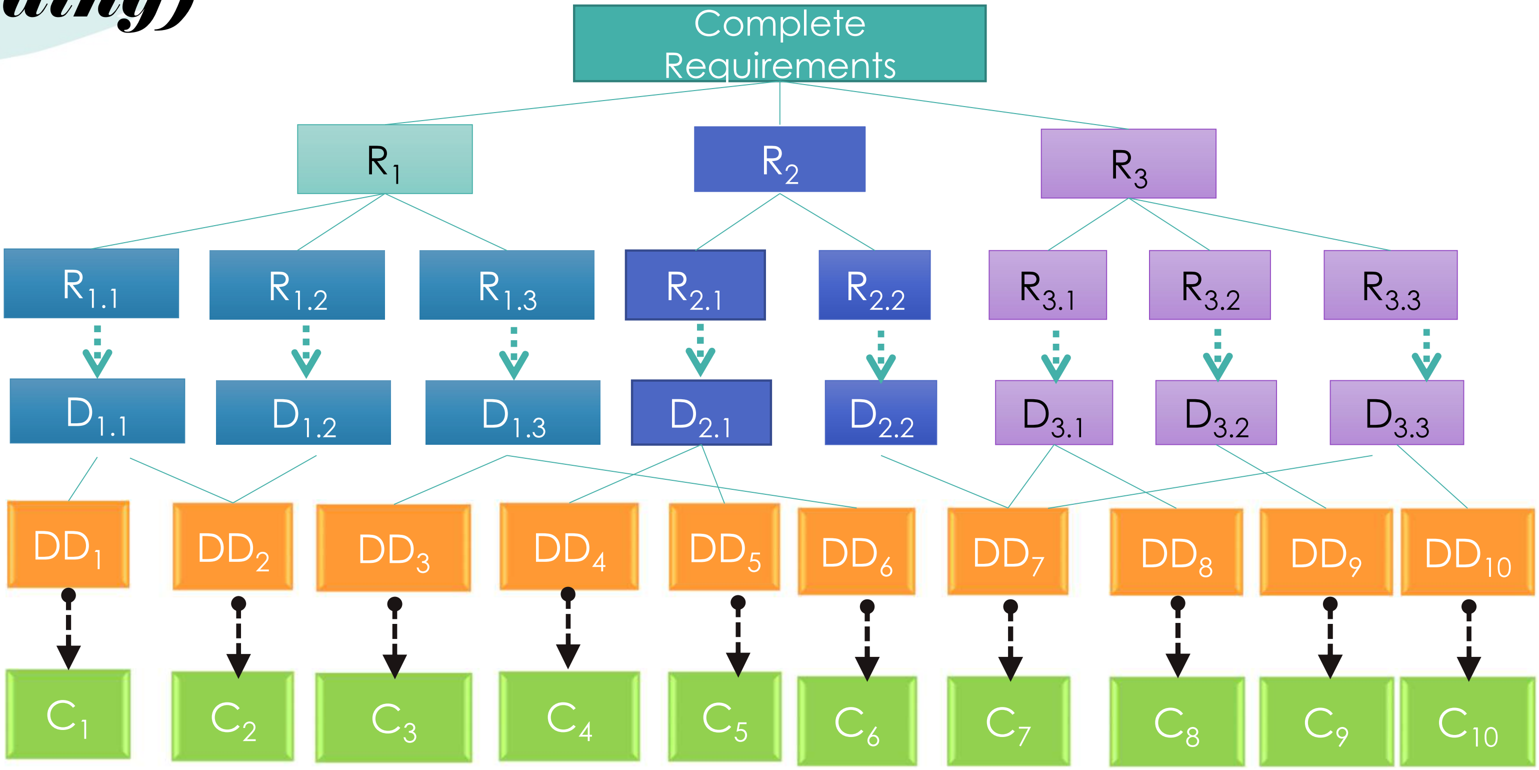


Dibuat Model Rancangan (Desain)

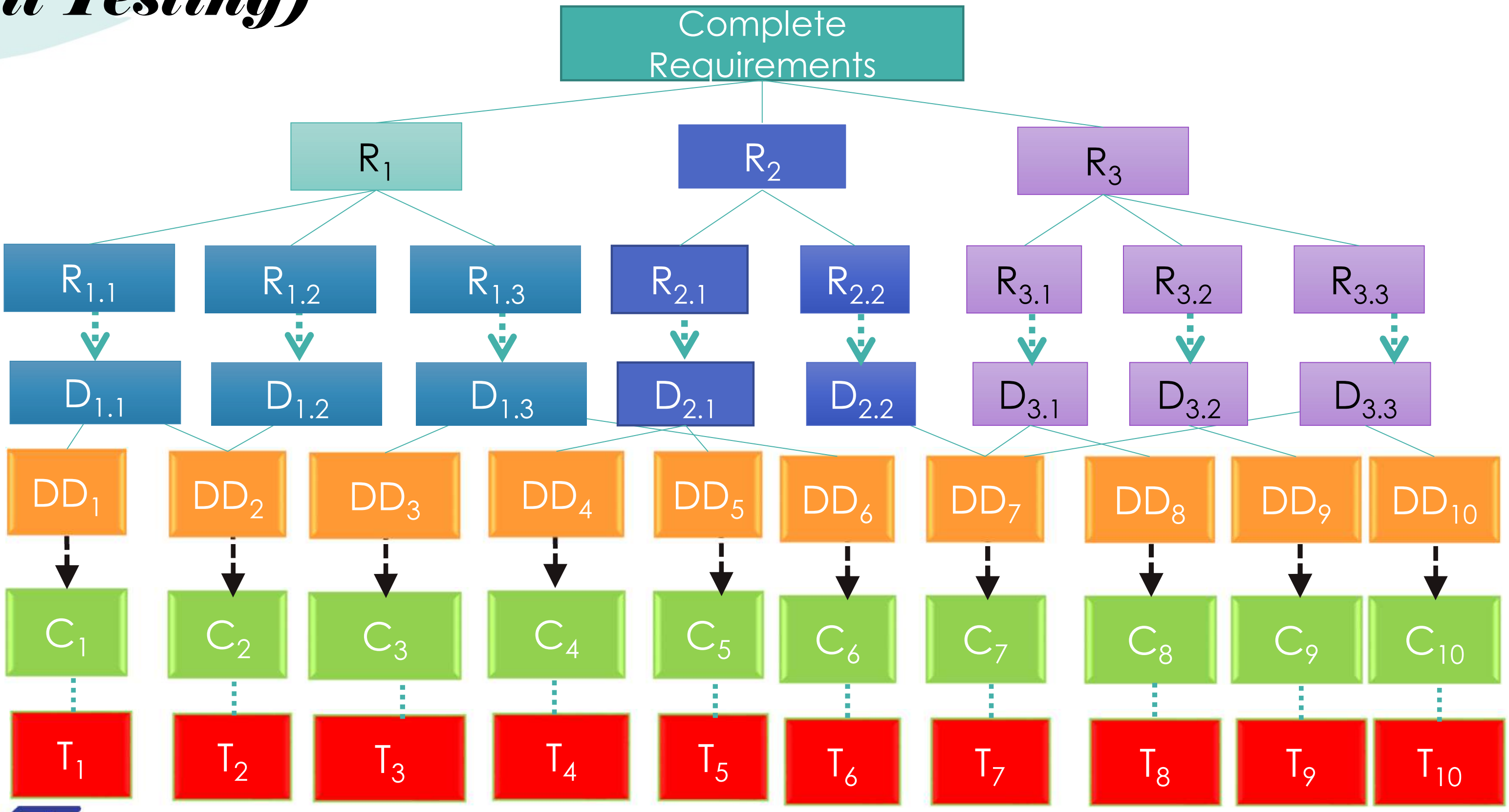
Dari Rancangan Umum (Global) Hingga Lebih Rinci (Detil)

KNOWLEDGE & SOFTWARE ENGINEERING

Dari Perancangan hingga Pemrograman (Coding)

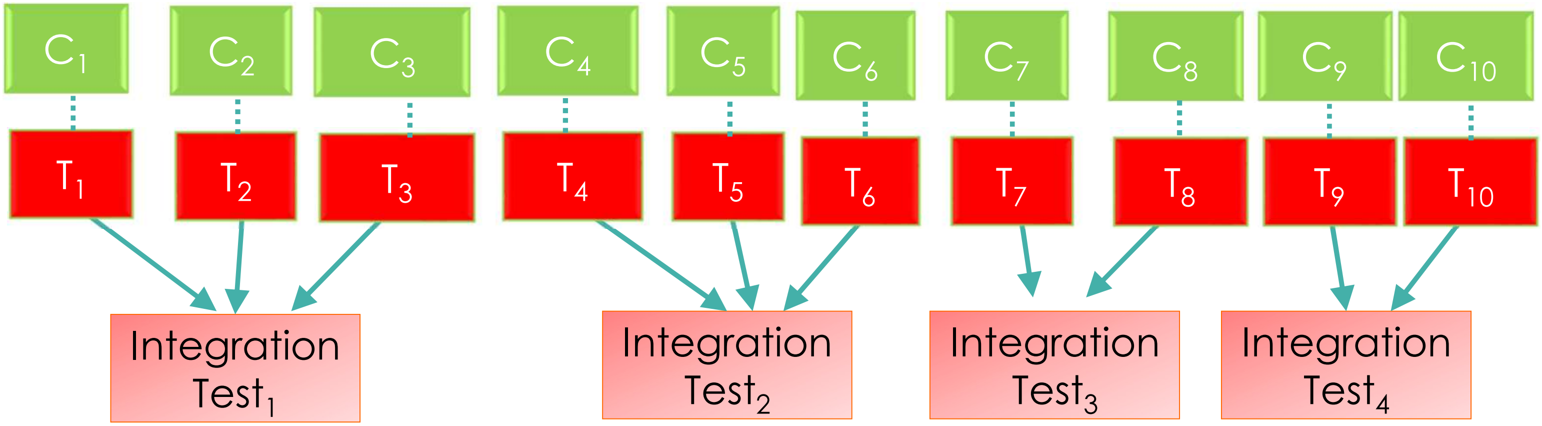


Setiap unit kode program harus diuji (Unit Testing)



KNOWLEDGE & SOFTWARE ENGINEERING

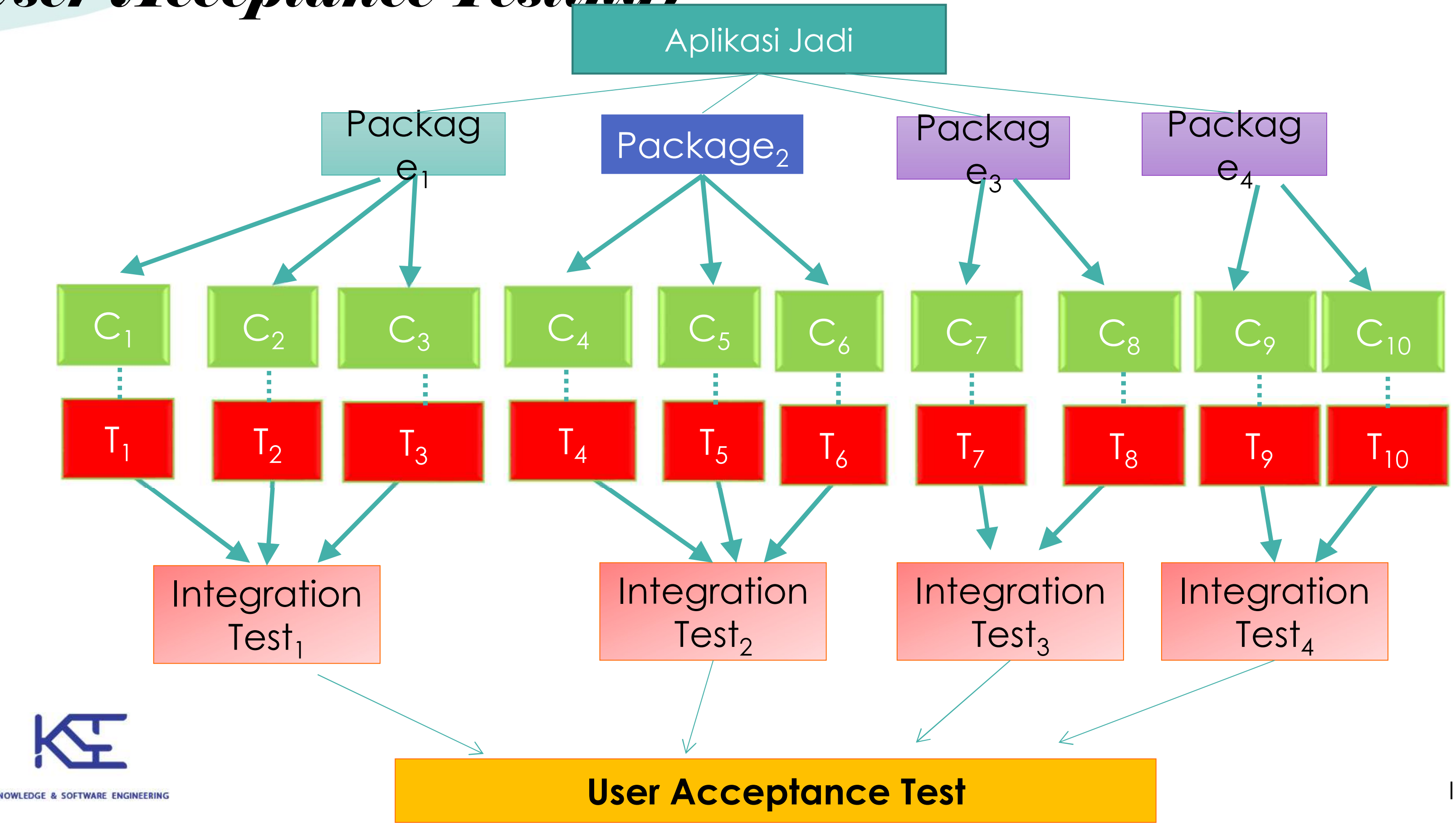
Setiap unit program harus digabung dan hasil penggabungannya di uji kembali (Integration Testing)



Setiap hasil integrasi akan diuji, hingga kita mendapatkan pengujian yang lengkap, artinya semua unit sudah menjadi satu, dan dilakukan pengujian secara keseluruhan

KNOWLEDGE & SOFTWARE ENGINEERING

Pengujian Lengkap di depan calon pengguna disebut Pengujian Penerimaan Pengguna (User Acceptance Testing)


Aplikasi Jadi
Package1
Package2
Package3
Package4
C1
C2
C3
C4
C5
C6
C7
C8
C9
C10
T1
T2
T3
T4
T5
T6
T7
T8
T9
T10
Integration Test1
Integration Test2
Integration Test3
Integration Test4
User Acceptance Test