BINUS UNIVERSITY ONLINE LEARNING

People
Innovation
Excellence

Software Engineering

Topic 1

An Introduction to Software Engineering

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Acknowledgement

These slides have been adapted from Pressman, R.S. (2015). *Software Engineering : A Practioner's Approach. 8th ed*. McGraw-Hill Companies.Inc, Americas, New York. ISBN : 978 1 259 253157. Chapter 12, 13, 14, 15, 16, 17, and 18

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Learning Objectives

LO 1 : Describe the concepts of software process models and the opportunity for potential business project

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Contents

- Software Engineering
- Software Costs
- FAQs about Software Engineering
- What are the Costs of Software Engineering?
- What are Software Engineering Methods?
- What is CASE (Computer-Aided Software Engineering)
- What are the Attributes of Good Software?
- What are the Key Challenges Facing Software Engineering?
- Professional and Ethical Responsibility

OVERVIEW OF SOFTWARE ENGINEERING

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Software Engineering

- The economies of ALL developed nations are dependent on software.
- More and more systems are software controlled
- Software engineering is concerned with theories, methods and tools for professional software development.
- Expenditure on software represents a significant fraction of GNP in all developed countries.

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Software Costs

- Software costs often dominate computer system costs. The costs of software on a PC are often greater than the hardware cost.
- Software costs more to maintain than it does to develop. For systems with a long life, maintenance costs may be several times development costs.
- Software engineering is concerned with cost-effective software development.

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

FAQs about Software Engineering

- What is software?
- What is software engineering?
- What is the difference between software engineering and computer science?
- What is the difference between software engineering and system engineering?
- What is a software process?
- What is a software process model?

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

FAQs about Software Engineering

What is software?

- Computer programs and associated documentation such as requirements, design models and user manuals.
- Software products may be developed for a particular customer or may be developed for a general market.
- Software products may be
  - Generic - developed to be sold to a range of different customers e.g. PC software such as Excel or Word.
  - Bespoke (custom) - developed for a single customer according to their specification.
- New software can be created by developing new programs, configuring generic software systems or reusing existing software.

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

FAQs about Software Engineering

Categories of software

- Web application
- Mobile application
- Cloud computing
- Product Line Software

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

FAQs about Software Engineering

What is software engineering?

- Software engineering is an engineering discipline that is concerned with all aspects of software production.
- Software engineers should adopt a systematic and organised approach to their work and use appropriate tools and techniques depending on the problem to be solved, the development constraints and the resources available.

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

FAQs about Software Engineering

What is the difference between software engineering and computer science?

- Computer science is concerned with theory and fundamentals; software engineering is concerned with the practicalities of developing and delivering useful software.
- Computer science theories are still insufficient to act as a complete underpinning for software engineering (unlike e.g. physics and electrical engineering).

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

FAQs about Software Engineering

What is the difference between software engineering and system engineering?

- System engineering is concerned with all aspects of computer-based systems development including hardware, software and process engineering. Software engineering is part of this process concerned with developing the software infrastructure, control, applications and databases in the system.
- System engineers are involved in system specification, architectural design, integration and deployment.

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

FAQs about Software Engineering

What is a software process?

- A set of activities whose goal is the development or evolution of software.
- Generic activities in all software processes are:
  - Specification - what the system should do and its development constraints
  - Development - production of the software system
  - Validation - checking that the software is what the customer wants
  - Evolution - changing the software in response to changing demands.

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

FAQs about Software Engineering

What is a software process model?

- A simplified representation of a software process, presented from a specific perspective.
- Examples of process perspectives are
  - Workflow perspective - sequence of activities;
  - Data-flow perspective - information flow;
  - Role/action perspective - who does what.
- Generic process models
  - Waterfall;
  - Iterative development;
  - Component-based software engineering.

COST OF SOFTWARE ENGINEERING

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

What are the Costs of Software Engineering?

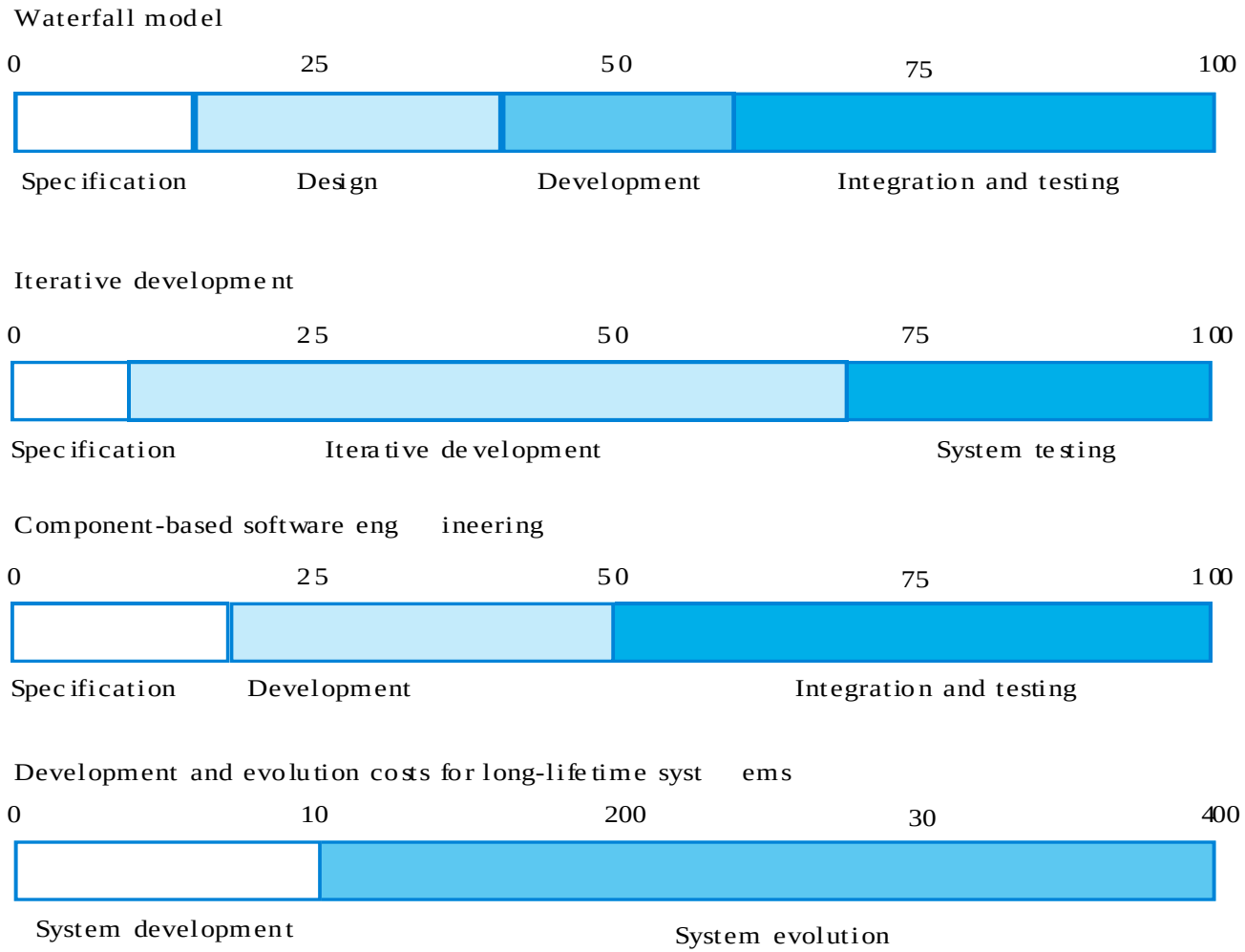
- Roughly 60% of costs are development costs, 40% are testing costs. For custom software, evolution costs often exceed development costs.
- Costs vary depending on the type of system being developed and the requirements of system attributes such as performance and system reliability.
- Distribution of costs depends on the development model that is used.

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

What are the Costs of Software Engineering?

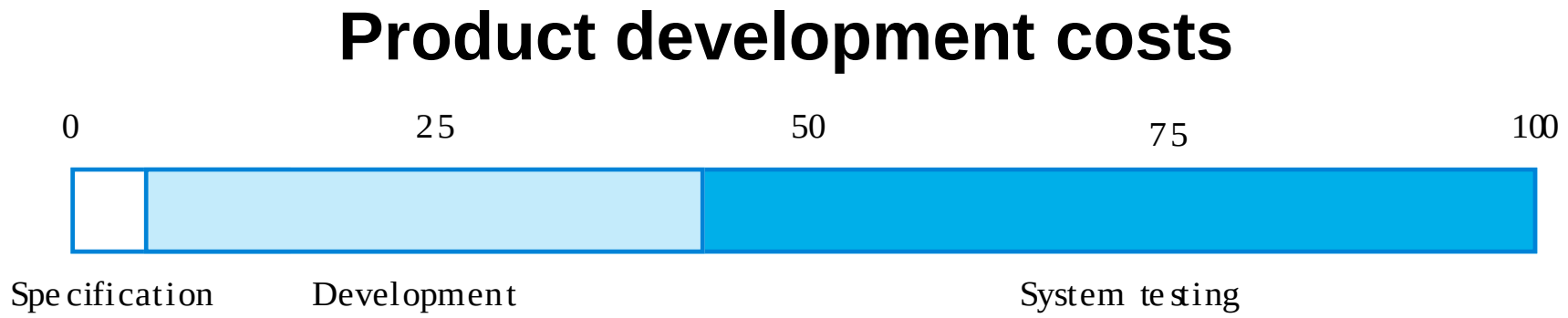
Activity cost distribution



BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

What are the Costs of Software Engineering?



SOFTWARE ENGINEERING METHODS, TOOLS, AND ATTRIBUTES

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

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.

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

What is CASE (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.

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

What are the Attributes of Good Software?

- The software should deliver the required functionality and performance to the user and should be maintainable, dependable and acceptable.
- Maintainability
  - Software must evolve to meet changing needs;
- Dependability
  - Software must be trustworthy;
- Efficiency
  - Software should not make wasteful use of system resources;
- Acceptability
  - Software must accepted by the users for which it was designed. This means it must be understandable, usable and compatible with other systems.

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

What are the Key Challenges Facing Software Engineering?

- Heterogeneity, delivery and trust.
- Heterogeneity
  - Developing techniques for building software that can cope with heterogeneous platforms and execution environments;
- Delivery
  - Developing techniques that lead to faster delivery of software;
- Trust
  - Developing techniques that demonstrate that software can be trusted by its users.

PROFESSIONAL AND ETHICAL RESPONSIBILITY

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Professional and Ethical Responsibility?

- Software engineering involves wider responsibilities than simply the application of technical skills.
- Software engineers must behave in an honest and ethically responsible way if they are to be respected as professionals.
- Ethical behaviour is more than simply upholding the law.

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Professional and Ethical Responsibility

Issues of professional responsibility

- Confidentiality
  - Engineers should normally respect the confidentiality of their employers or clients irrespective of whether or not a formal confidentiality agreement has been signed.
- Competence
  - Engineers should not misrepresent their level of competence. They should not knowingly accept work which is outwith their competence.

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Professional and Ethical Responsibility

Issues of professional responsibility

- Intellectual property rights
  - Engineers should be aware of local laws governing the use of intellectual property such as patents, copyright, etc. They should be careful to ensure that the intellectual property of employers and clients is protected.
- Computer misuse
  - Software engineers should not use their technical skills to misuse other people's computers. Computer misuse ranges from relatively trivial (game playing on an employer's machine, say) to extremely serious (dissemination of viruses).

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

References

- Sommerville, I. (2011), ***Software Engineering 9th ed***, Pearson, Addison Wesley United states of America, ISBN 13 : 978 0 13 705346 9.
- **What can we learn from software engineering and why?, http://www.youtube.com/watch?v=8kG15VoNxhc**
- **How to Reduce Cost of Software Development,** http://www.youtube.com/watch?v=gKeka3zn198

BINUS UNIVERSITY ONLINE LEARNING

People
Innovation
Excellence

Q & A

BINUS
UNIVERSITY
ONLINE
LEARNING

People
Innovation
Excellence

Thank You

BINUS UNIVERSITY ONLINE LEARNING

People
Innovation
Excellence

Software Engineering

Topic 2

Software Process Model

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Acknowledgement

These slides have been adapted from Pressman, R.S. (2015). *Software Engineering : A Practioner's Approach. 8th ed*. McGraw-Hill Companies.Inc, Americas, New York. ISBN : 978 1 259 253157. Chapter 3 , 4, 5 and 6

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Learning Objectives

LO 1 : Describe the concepts of software process models and the opportunity for potential business project

BINUS UNIVERSITY ONLINE LEARNING

People
Innovation
Excellence

Contents

- Software Process Structure
- Process Models
- Agile Development
- Human Aspects of Software Engineering

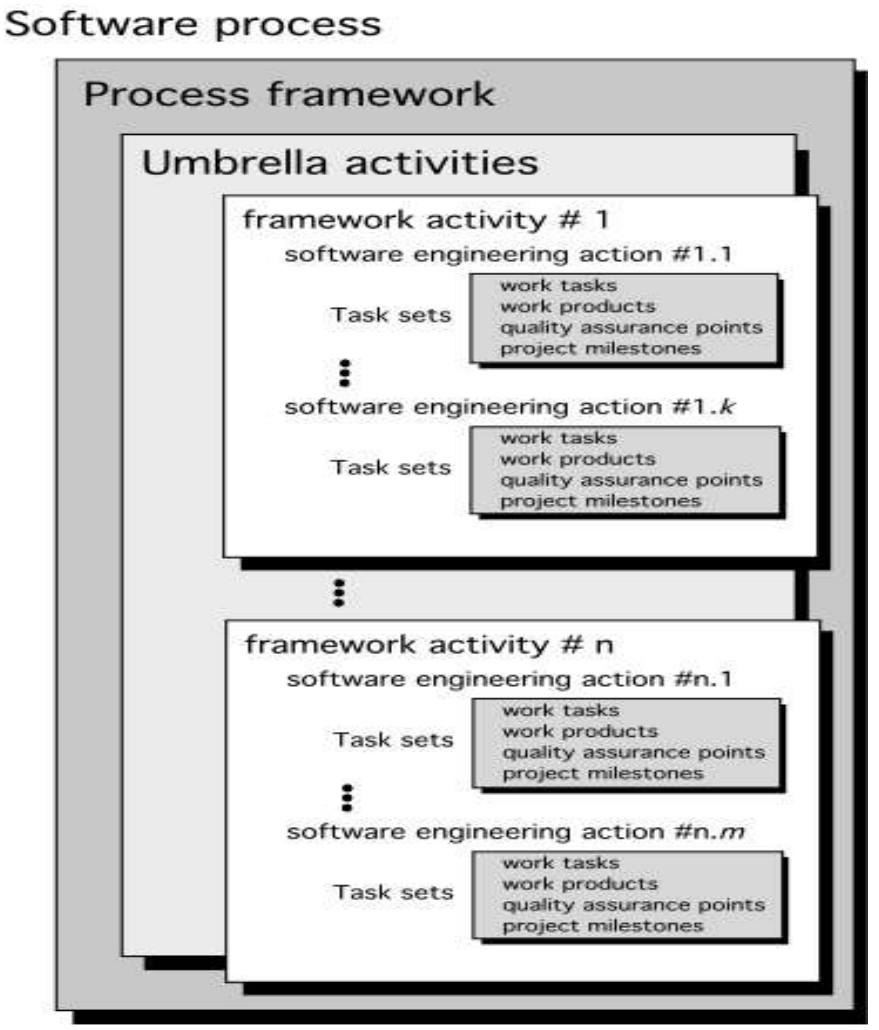
SOFTWARE PROCESS STRUCTURE

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Software Process Structure

Software Process Framework

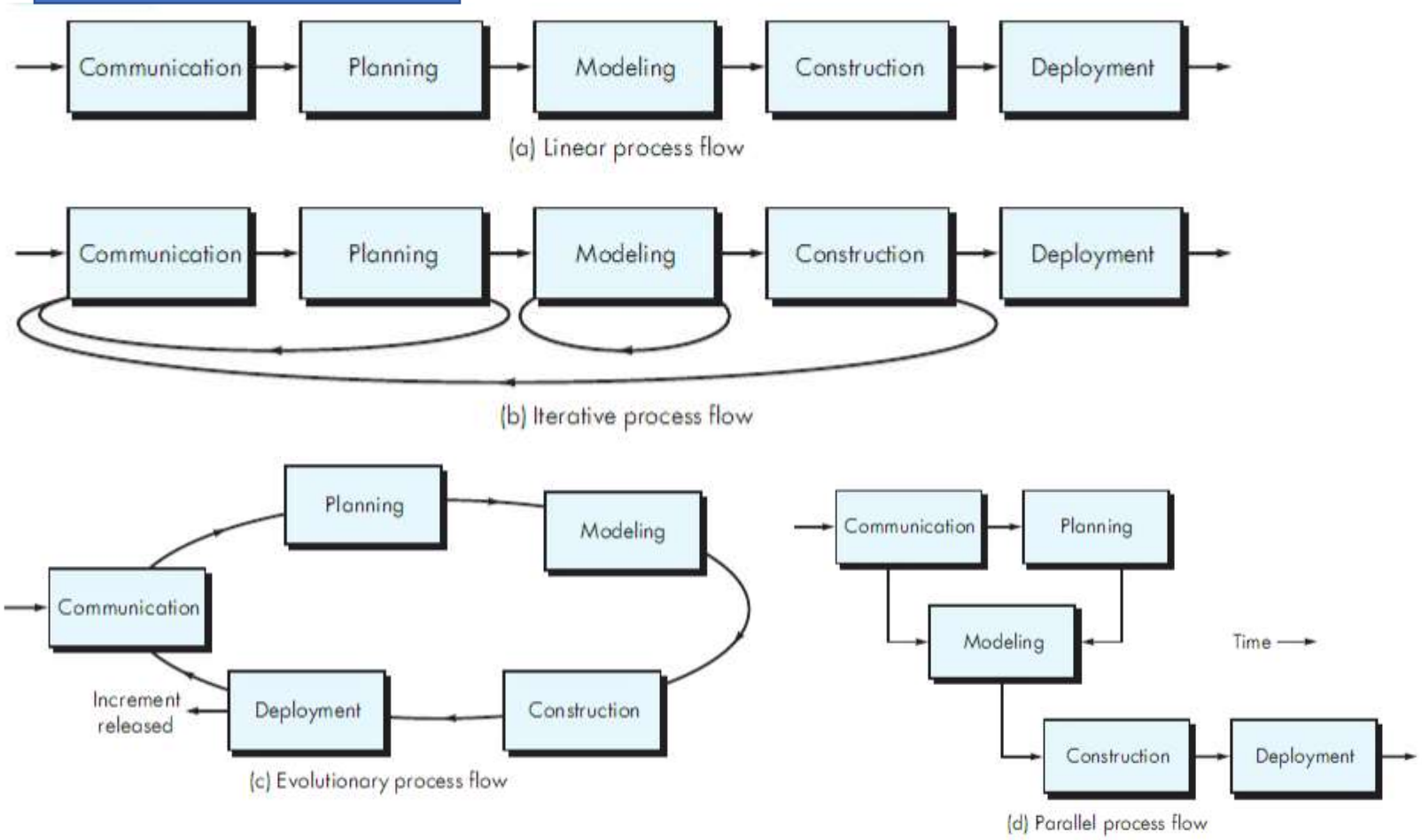


BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Software Process Structure

Process Flow



BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Software Process Structure

Identifying a Task Set

- A task set defines the actual work to be done to accomplish the objectives of a software engineering action.
  - A list of the task to be accomplished
  - A list of the work products to be produced
  - A list of the quality assurance filters to be applied

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Software Process Structure

Process Patterns

- A *process pattern*
  - describes a process-related problem that is encountered during software engineering work,
  - identifies the environment in which the problem has been encountered, and
  - suggests one or more proven solutions to the problem.
- Stated in more general terms, a process pattern provides you with a *template* [Amb98]—a consistent method for describing problem solutions within the context of the software process.

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Software Process Structure

Process Pattern Types

- ***Stage*** *patterns*—defines a problem associated with a framework activity for the process.
- ***Task*** *patterns*—defines a problem associated with a software engineering action or work task and relevant to successful software engineering practice
- ***Phase*** *patterns*—define the sequence of framework activities that occur with the process, even when the overall flow of activities is iterative in nature

PROCESS MODEL

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Process Models

- **Prescriptive process models advocate an orderly approach to software engineering**

That leads to a few questions …

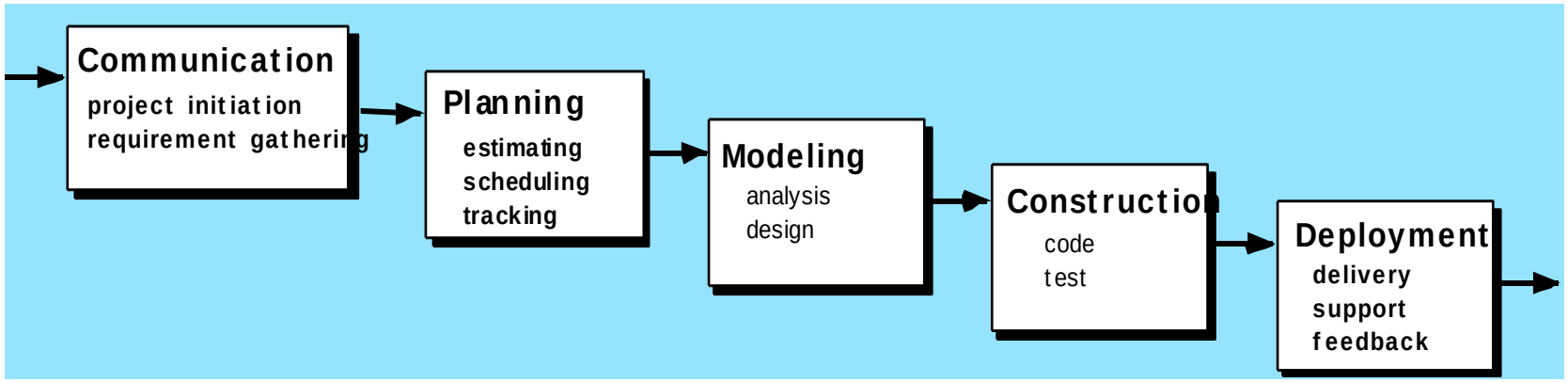
- **If prescriptive process models strive for structure and order, are they inappropriate for a software world that thrives on change?**
- **Yet, if we reject traditional process models (and the order they imply) and replace them with something less structured, do we make it impossible to achieve coordination and coherence in software work?**

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Process Models

The Waterfall Model

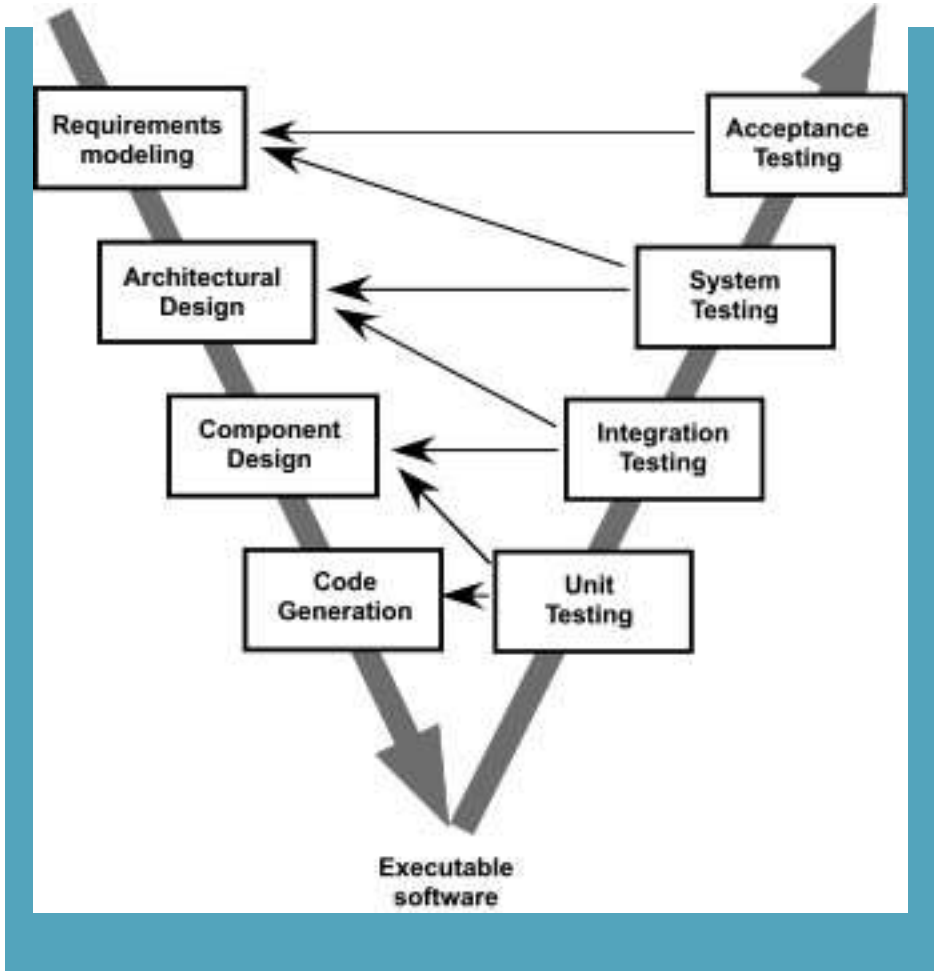


BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Process Models

The V-Model

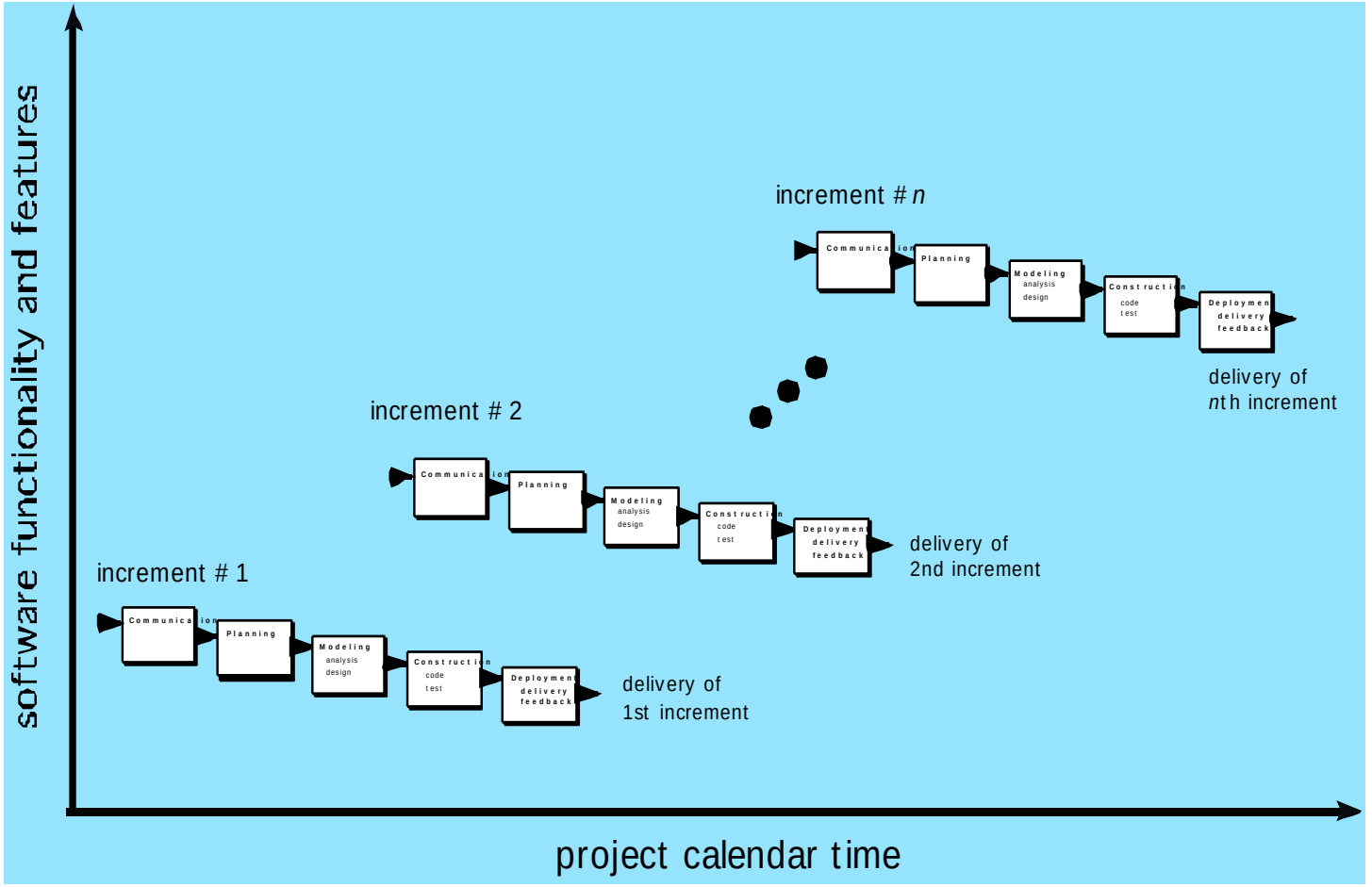


BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Process Models

The Incremental-Model

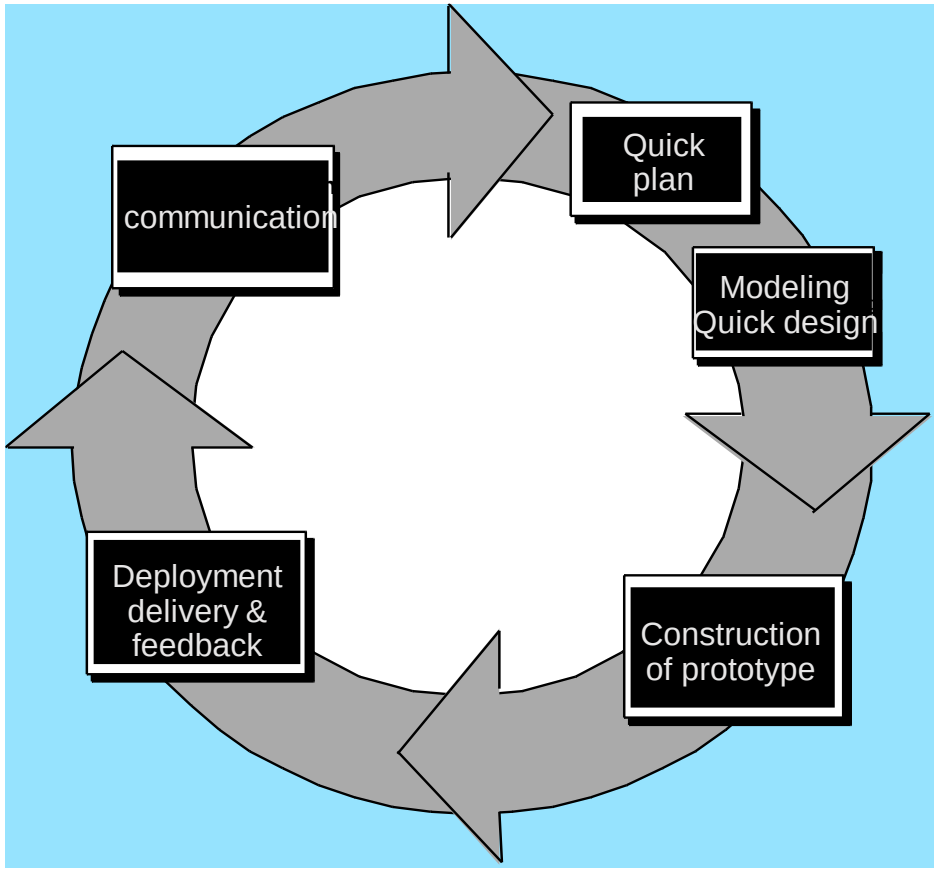


BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Process Models

Evolutionary Models: Prototyping

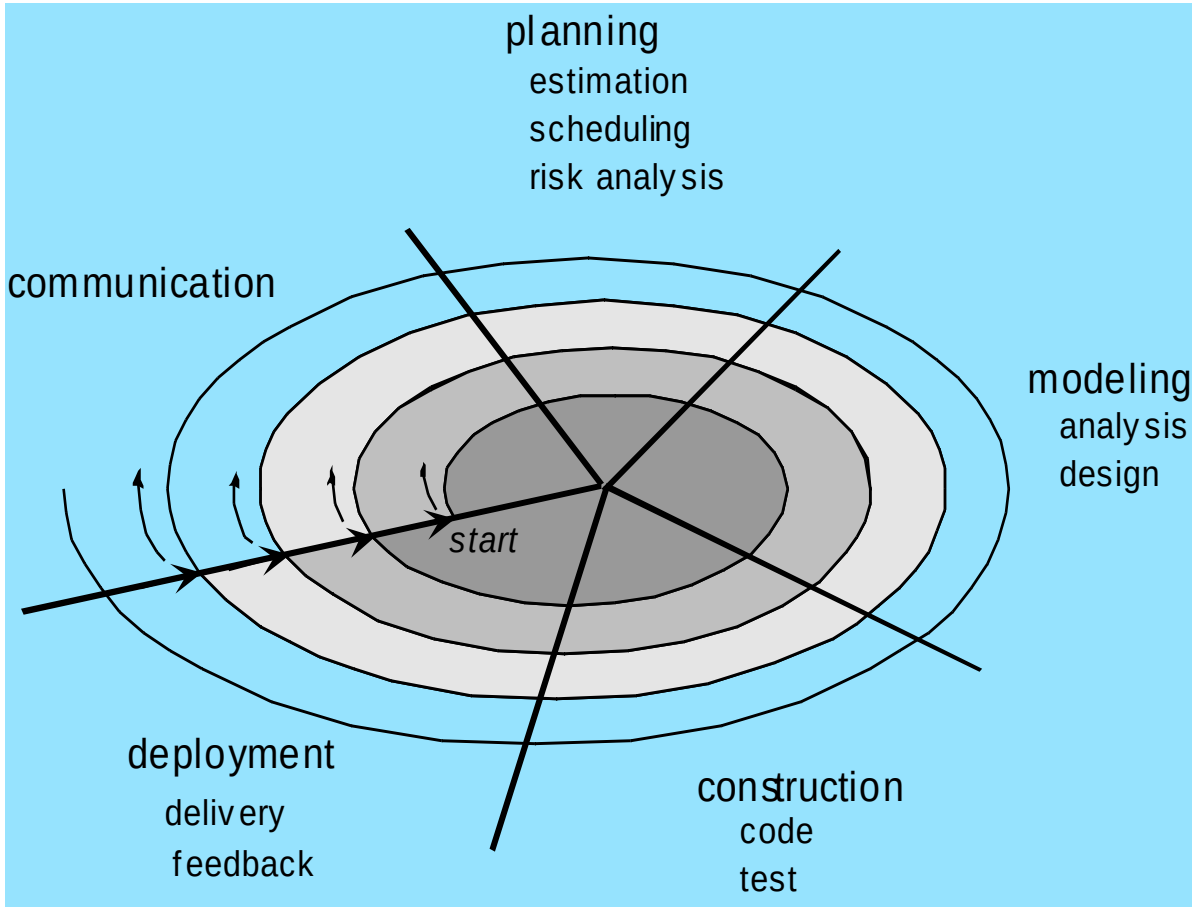


BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Process Models

Evolutionary Models: The Spiral

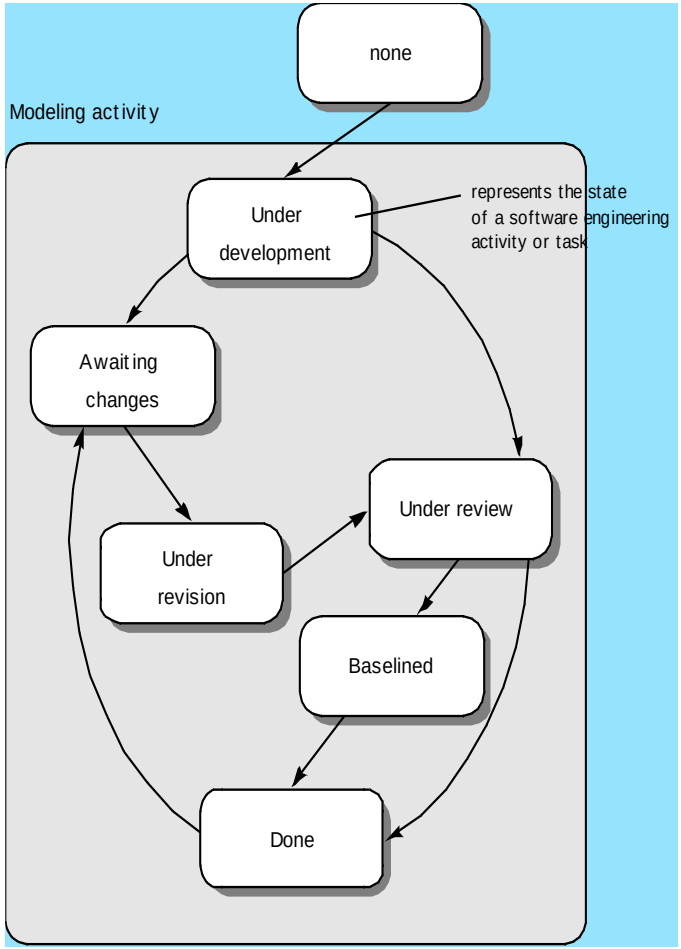


BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Process Models

Evolutionary Models: Concurrent



BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Specialized Process Models

- Component based development—the process to apply when reuse is a development objective
- Formal methods—emphasizes the mathematical specification of requirements
- AOSD—provides a process and methodological approach for defining, specifying, designing, and constructing *aspects*
- Unified Process—a "use-case driven, architecture-centric, iterative and incremental" software process closely aligned with the Unified Modeling Language (UML)

AGILE DEVELOPMENT

BINUS UNIVERSITY ONLINE LEARNING

People
Innovation
Excellence

Agile Development

Manifesto for Agile Software Development

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more

Source: agilemanifesto.org

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Agile Development

What is "Agility"?

- **Effective (rapid and adaptive) response to change**
- **Effective communication among all stakeholders**
- **Drawing the customer onto the team**
- **Organizing a team so that it is in control of the work performed**

Yielding …

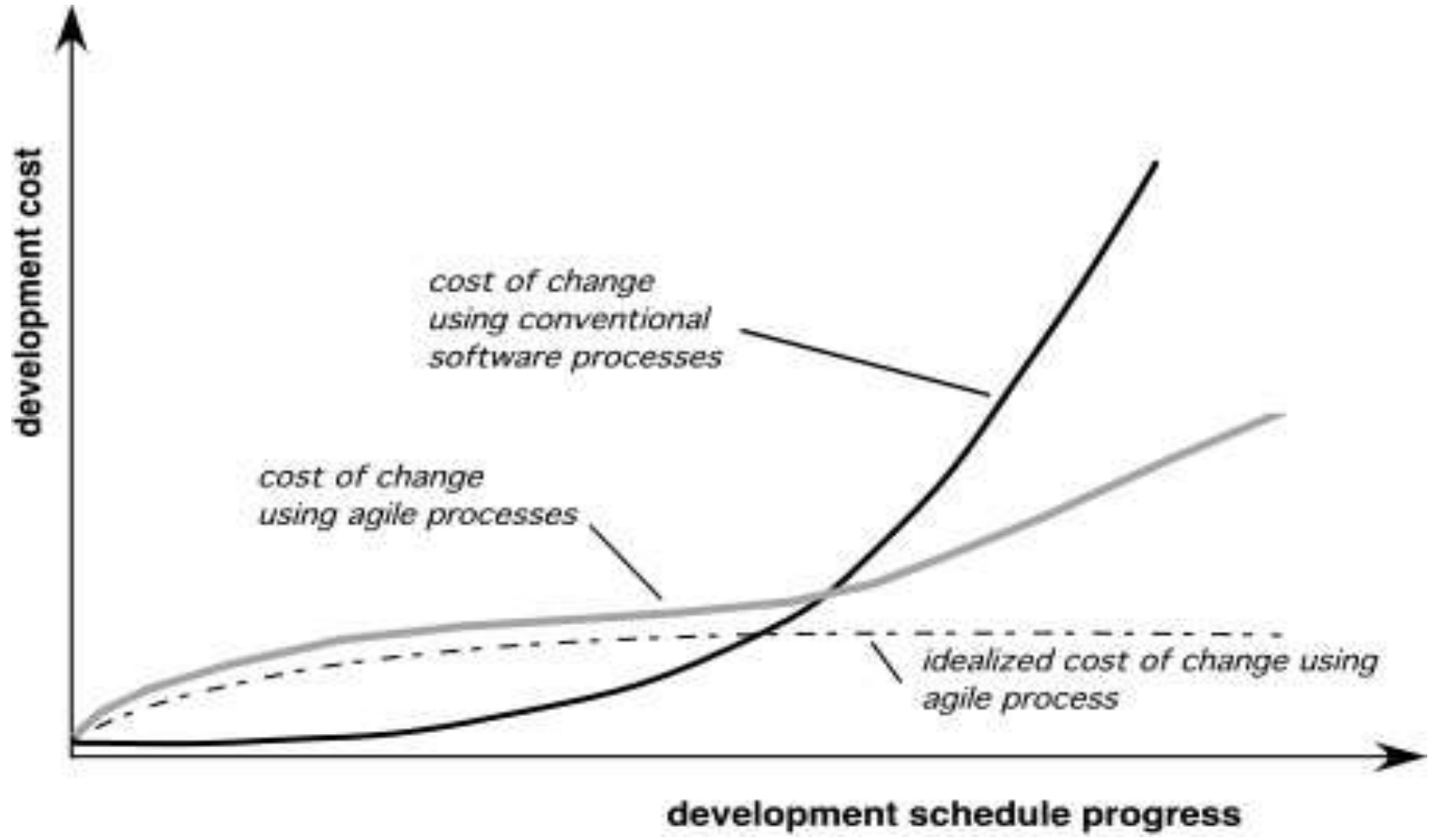
- **Rapid, incremental delivery of software**

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Agile Development

Agility and the Cost of Change



BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Agile Development

Extreme Programming (XP)

- **The most widely used agile process, originally proposed by Kent Beck**
- **XP Planning**
  - **Begins with the creation of "user stories"**
  - **Agile team assesses each story and assigns a cost**
  - **Stories are grouped to for a deliverable increment**
  - **A commitment is made on delivery date**
  - **After the first increment "project velocity" is used to help define subsequent delivery dates for other increments**

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Agile Development

Extreme Programming (XP)

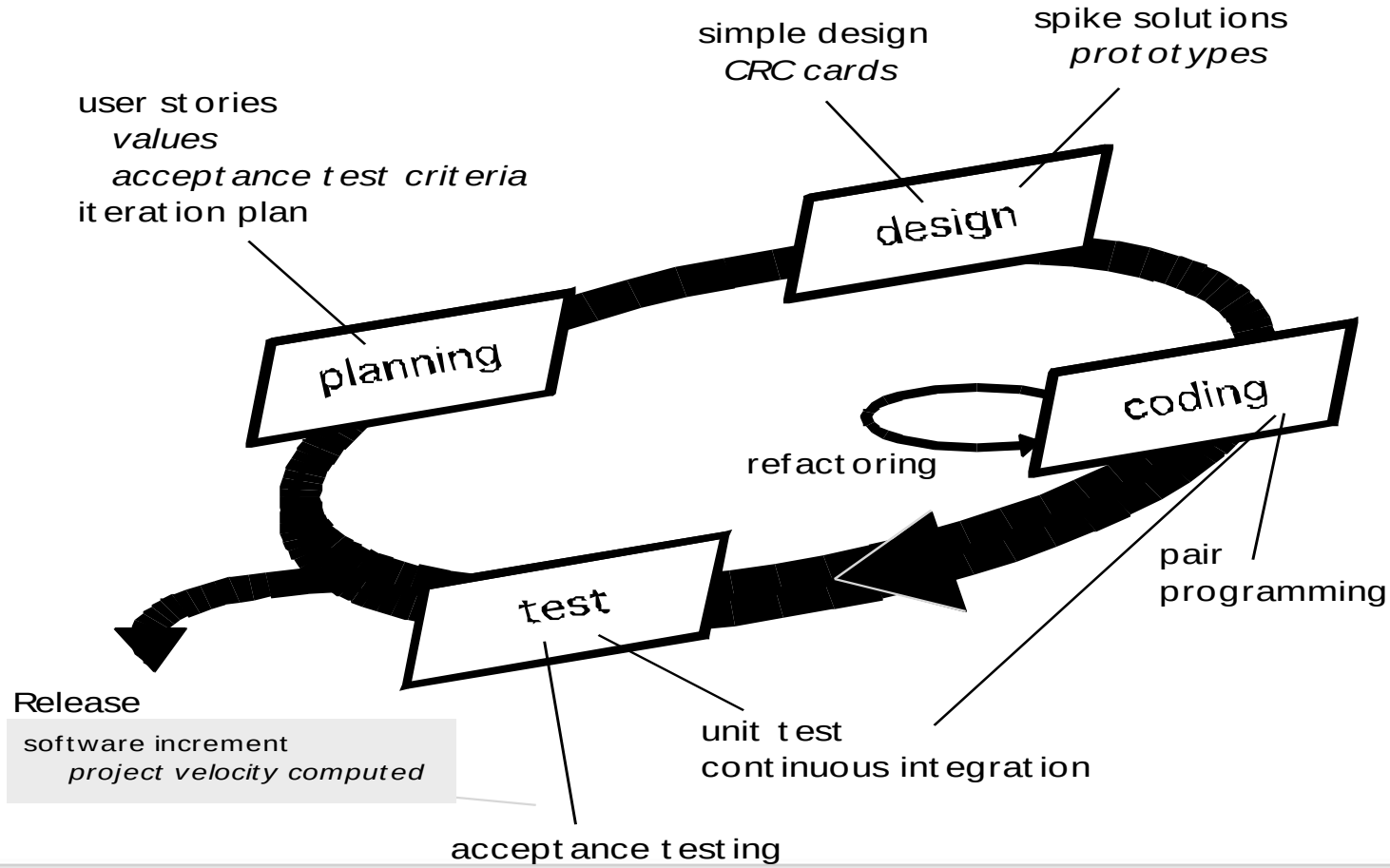
- **XP Design**
  - **Follows the KIS principle**
  - **Encourage the use of CRC cards (see Chapter 8)**
  - **For difficult design problems, suggests the creation of "spike solutions"—a design prototype**
  - **Encourages "refactoring"—an iterative refinement of the internal program design**
- **XP Coding**
  - **Recommends the construction of a unit test for a store *before* coding commences**
  - **Encourages "pair programming"**
- **XP Testing**
  - **All unit tests are executed daily**
  - **"Acceptance tests" are defined by the customer and executed to assess customer visible functionality**

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Agile Development

Extreme Programming (XP)



BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Agile Development

Adaptive Software Development

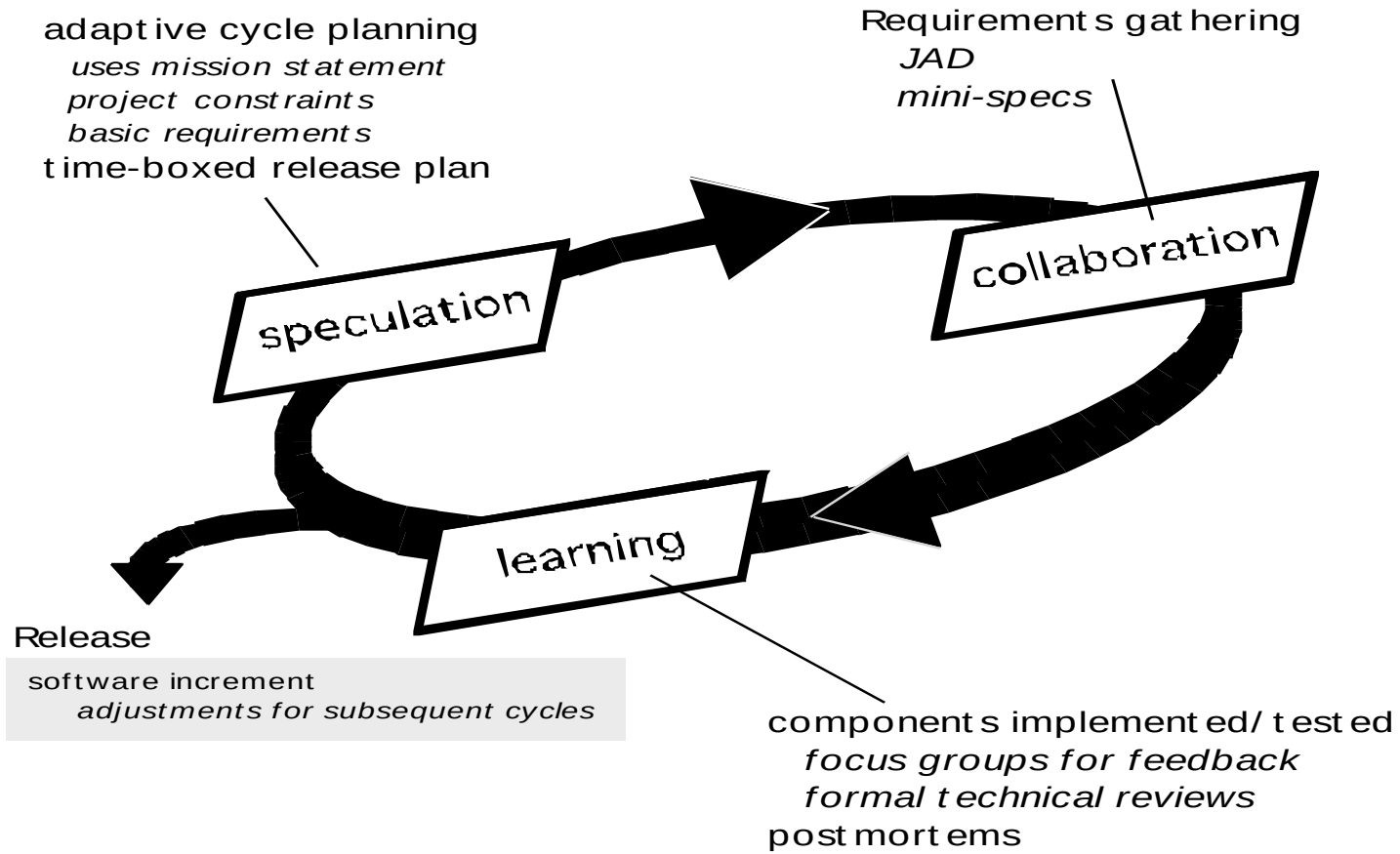
- **Originally proposed by Jim Highsmith**
- **ASD — distinguishing features**
  - **Mission-driven planning**
  - **Component-based focus**
  - **Uses “time-boxing”**
  - **Explicit consideration of risks**
  - **Emphasizes collaboration for requirements gathering**
  - **Emphasizes “learning” throughout the process**

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Agile Development

Adaptive Software Development



BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Agile Development

Dynamic Systems Development Method

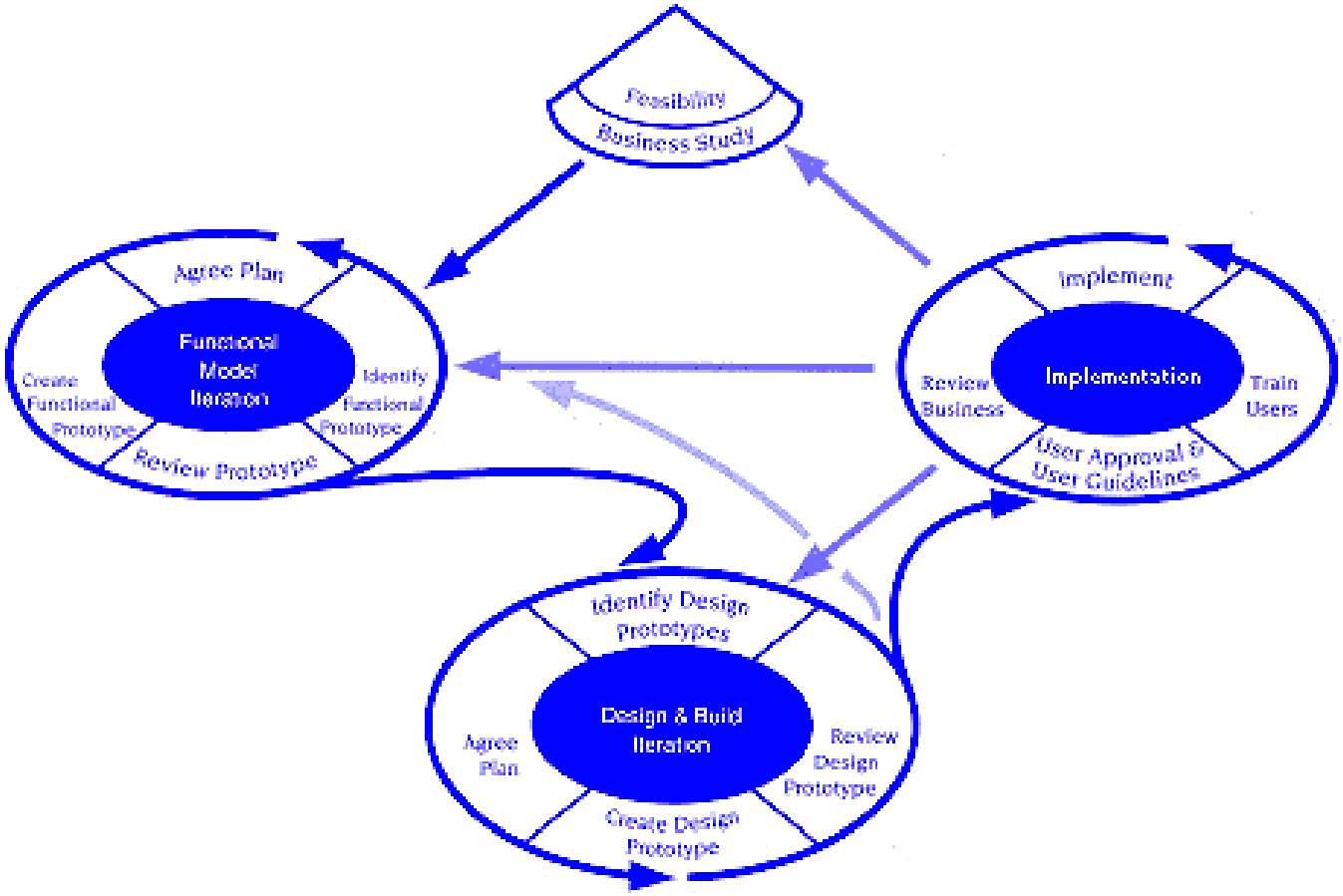
- **Promoted by the DSDM Consortium (www.dsdm.org)**
- **DSDM—distinguishing features**
  - **Similar in most respects to XP and/or ASD**
  - **Nine guiding principles**
    - **Active user involvement is imperative.**
    - **DSDM teams must be empowered to make decisions.**
    - **The focus is on frequent delivery of products.**
    - **Fitness for business purpose is the essential criterion for acceptance of deliverables.**
    - **Iterative and incremental development is necessary to converge on an accurate business solution.**
    - **All changes during development are reversible.**
    - **Requirements are baselined at a high level**
    - **Testing is integrated throughout the life-cycle.**

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Agile Development

Dynamic Systems Development Method



BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Agile Development

Scrum

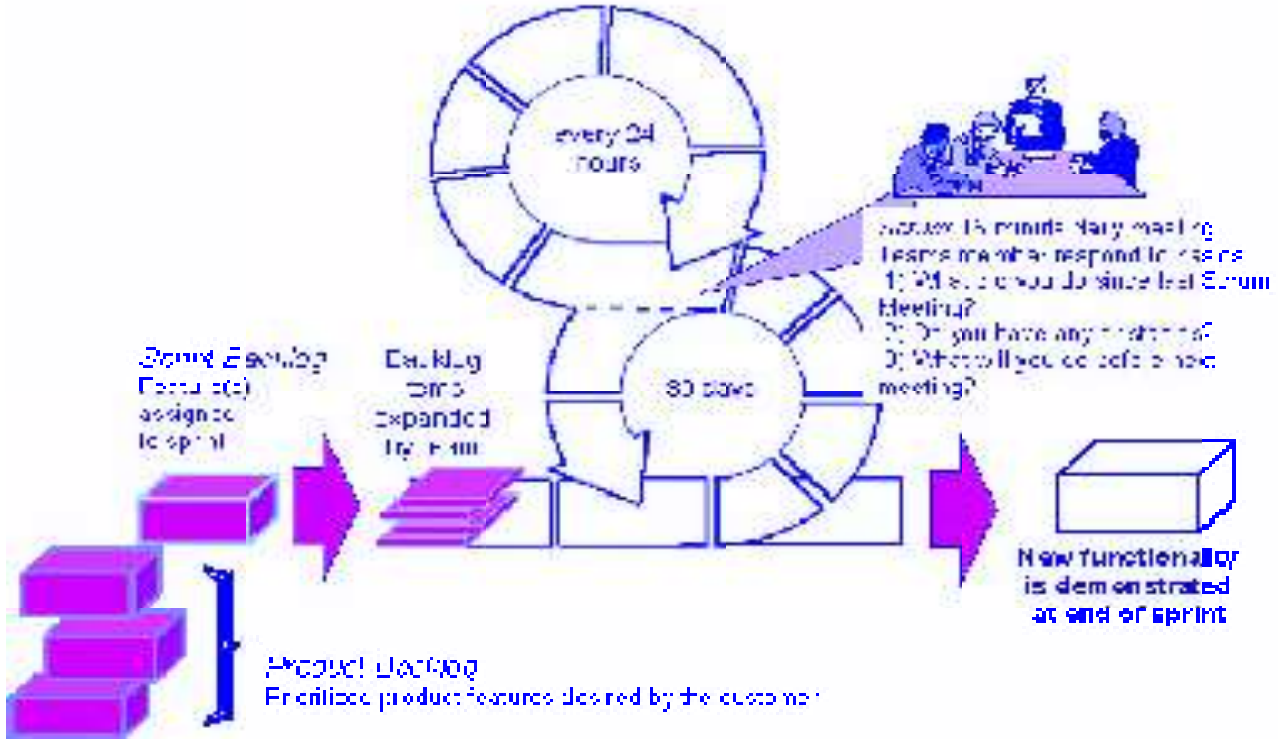
- **Originally proposed by Schwaber and Beedle**
- **Scrum—distinguishing features**
  - **Development work is partitioned into “packets”**
  - **Testing and documentation are on-going as the product is constructed**
  - **Work occurs in “sprints” and is derived from a “backlog” of existing requirements**
  - **Meetings are very short and sometimes conducted without chairs**
  - **“demos” are delivered to the customer with the time-box allocated**

BINUS UNIVERSITY ONLINE LEARNING

People
Innovation
Excellence

Agile Development

Scrum



Scrum Process Flow (used with permission)

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Agile Development

Crystal

- **Proposed by Cockburn and Highsmith**
- **Crystal—distinguishing features**
  - **Actually a family of process models that allow "maneuverability" based on problem characteristics**
  - **Face-to-face communication is emphasized**
  - **Suggests the use of "reflection workshops" to review the work habits of the team**

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Agile Development

Feature Driven Development

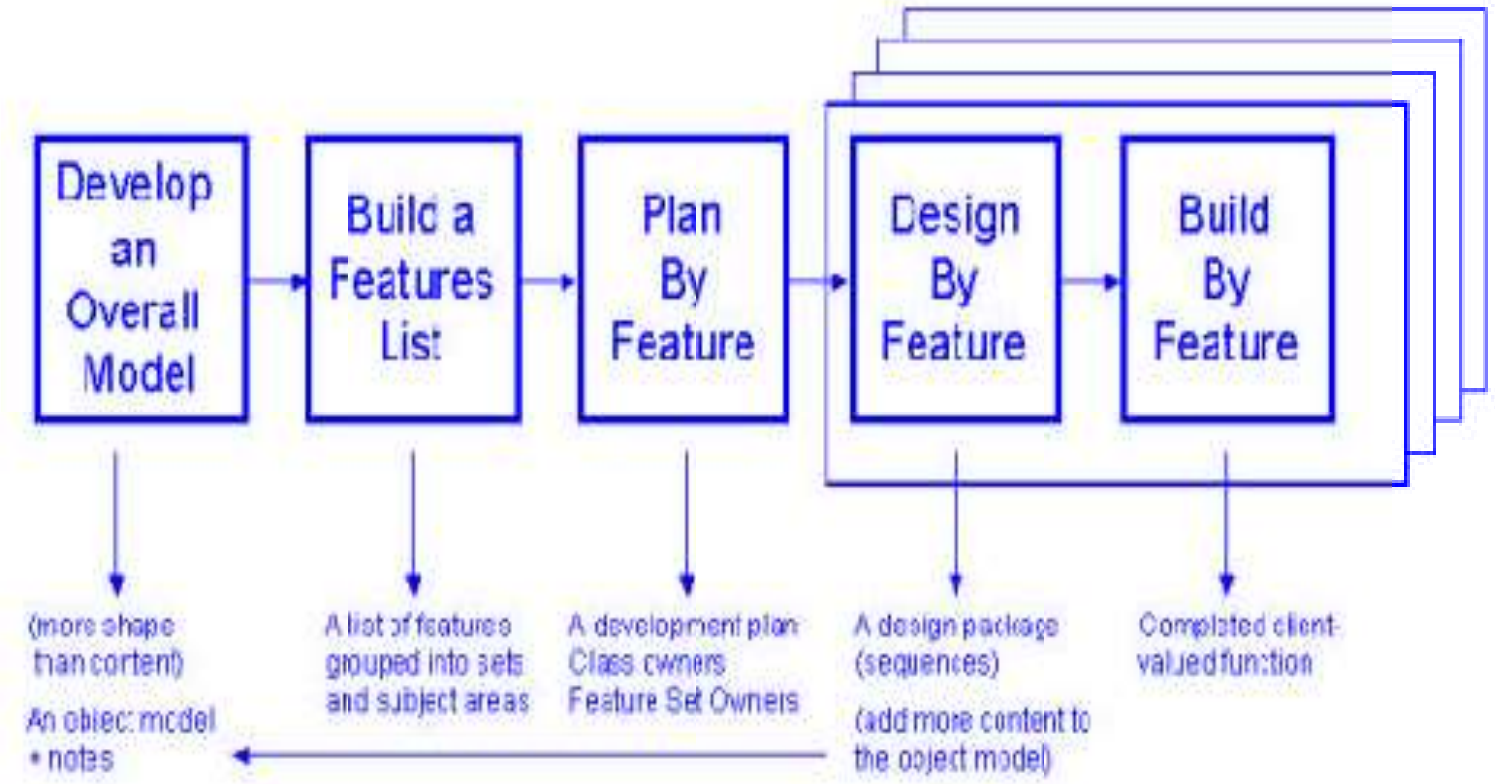
- **Originally proposed by Peter Coad et al**
- **FDD—distinguishing features**
  - **Emphasis is on defining "features"**
    - **a *feature* "is a client-valued function that can be implemented in two weeks or less."**
  - **Uses a feature template**
    - **<action> the <result> <by | for | of | to> a(n) <object>**
  - **A features list is created and "plan by feature" is conducted**
  - **Design and construction merge in FDD**

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Agile Development

Feature Driven Development



Reprinted with permission of Peter Coad

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Agile Development

Agile Modeling

- **Originally proposed by Scott Ambler**
- **Suggests a set of agile modeling principles**
  - **Model with a purpose**
  - **Use multiple models**
  - **Travel light**
  - **Content is more important than representation**
  - **Know the models and the tools you use to create them**
  - **Adapt locally**

HUMAN ASPECT OF SOFTWARE ENGINEERING

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Human Aspects of Software Engineering

Characteristics of Software Engineer

Erdogmus [erd09] identifies seven traits that are present when an individual software engineer exhibits "superprofesional" behavior.

An effective software engineer :

- **has a sense of individual responsibility**
- **has an acute awareness**
- **is brutally honest**
- **exhibits resilience under prerssure**
- **has a heightened sense of fairness**
- **exhibits attention to detail**
- **is pragmatic**

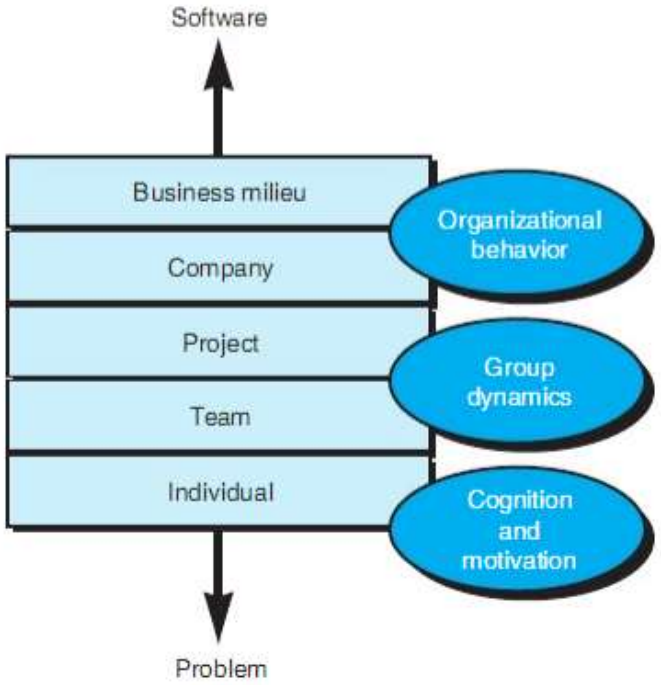
BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Human Aspects of Software Engineering

The Psychology of Software Engineering

In a seminal paper on the psychology of software engineering, Bill curtis and Dave Walz [Cur90] suggest a layered behavioral model for software development.



BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Human Aspects of Software Engineering

- An effective team should foster a sense of trust
- Software engineers on the team should trust the skills and competence of their peers and their managers.
- The team should encourage a sense of improvement by periodically reflecting on its approach to software engineering and looking for ways to improve their work

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Team Structure

Human Aspects of Software Engineering

Constantine [Con93] suggests four "organizational paradigms" for software engineering teams

1. **A closed paradigm; a team along a traditional hierarchy of authority**
2. **A random paradigm; a team loosely and depends on individual initiative of the team members**
3. **An open paradigm; a team in a manner that achieves some of the controls associated with the closed paradigm but also much of the innovation that occurs when using the random paradigm**
4. **A synchronous paradigm; relies on the natural compartmentalization of a problem and organizes team members to work on pieces of the problem with little active communication amonjg themselves**

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

References

- Pressman, R.S. (2015). ***Software Engineering : A Practioner's Approach. 8th ed***. McGraw-Hill Companies.Inc, Americas, New York. ISBN : 978 1 259 253157.
- Manifesto for Agile Software Development, http://agilemanifesto.org/
- Multimedia : Video Water Fall, V model, http://www.youtube.com/watch?v=KaPC0gsEQ68
- Software Engineering Incremental Model, http://www.youtube.com/watch?v=9cBkihYP1rY
- The Strengths and Weaknesses of Extreme Programming, **http://www.youtube.com/watch?v=LkhLZ7_KZ5w**
- Agile project management tutorial: What is agile project managemen, http://www.youtube.com/watch?v=MJR-EgHTA4E

BINUS UNIVERSITY ONLINE LEARNING

People
Innovation
Excellence

Q & A

BINUS
UNIVERSITY
ONLINE
LEARNING

People
Innovation
Excellence

Thank You

BINUS UNIVERSITY ONLINE LEARNING

People
Innovation
Excellence

Software Engineering

Topic 3

Requirement Modelling

BINUS UNIVERSITY ONLINE LEARNING

People
Innovation
Excellence

Acknowledgement

These slides have been adapted from Pressman, R.S. (2015). *Software Engineering : A Practioner's Approach. 8th ed*. McGraw-Hill Companies.Inc, Americas, New York. ISBN : 978 1 259 253157. Chapter 9, 10, 11 and 12

BINUS UNIVERSITY ONLINE LEARNING

People
Innovation
Excellence

Learning Objectives

LO 2 : Explain the software engineering practices and business environment

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Contents

- Requirement Analysis
- Eliciting Requirements
- Developing Use Case
- Negotiating Requirements
- Validating Requirements

REQUIREMENT ENGINEERING

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Requirement Engineering

- Inception—ask a set of questions that establish …
  - basic understanding of the problem
  - the people who want a solution
  - the nature of the solution that is desired, and
  - the effectiveness of preliminary communication and collaboration between the customer and the developer
- Elicitation—elicit requirements from all stakeholders
- Elaboration—create an analysis model that identifies data, function and behavioral requirements
- Negotiation—agree on a deliverable system that is realistic for developers and customers

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Requirement Engineering

- Specification—can be any one (or more) of the following:
  - A written document
  - A set of models
  - A formal mathematical
  - A collection of user scenarios (use-cases)
  - A prototype
- Validation—a review mechanism that looks for
  - errors in content or interpretation
  - areas where clarification may be required
  - missing information
  - inconsistencies (a major problem when large products or systems are engineered)
  - conflicting or unrealistic (unachievable) requirements.
- Requirements management

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Requirement Engineering

Inception

- Identify stakeholders
  - "who else do you think I should talk to?"
- Recognize multiple points of view
- Work toward collaboration
- The first questions
  - Who is behind the request for this work?
  - Who will use the solution?
  - What will be the economic benefit of a successful solution
  - Is there another source for the solution that you need?

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Requirement Engineering

Eliciting Requirements

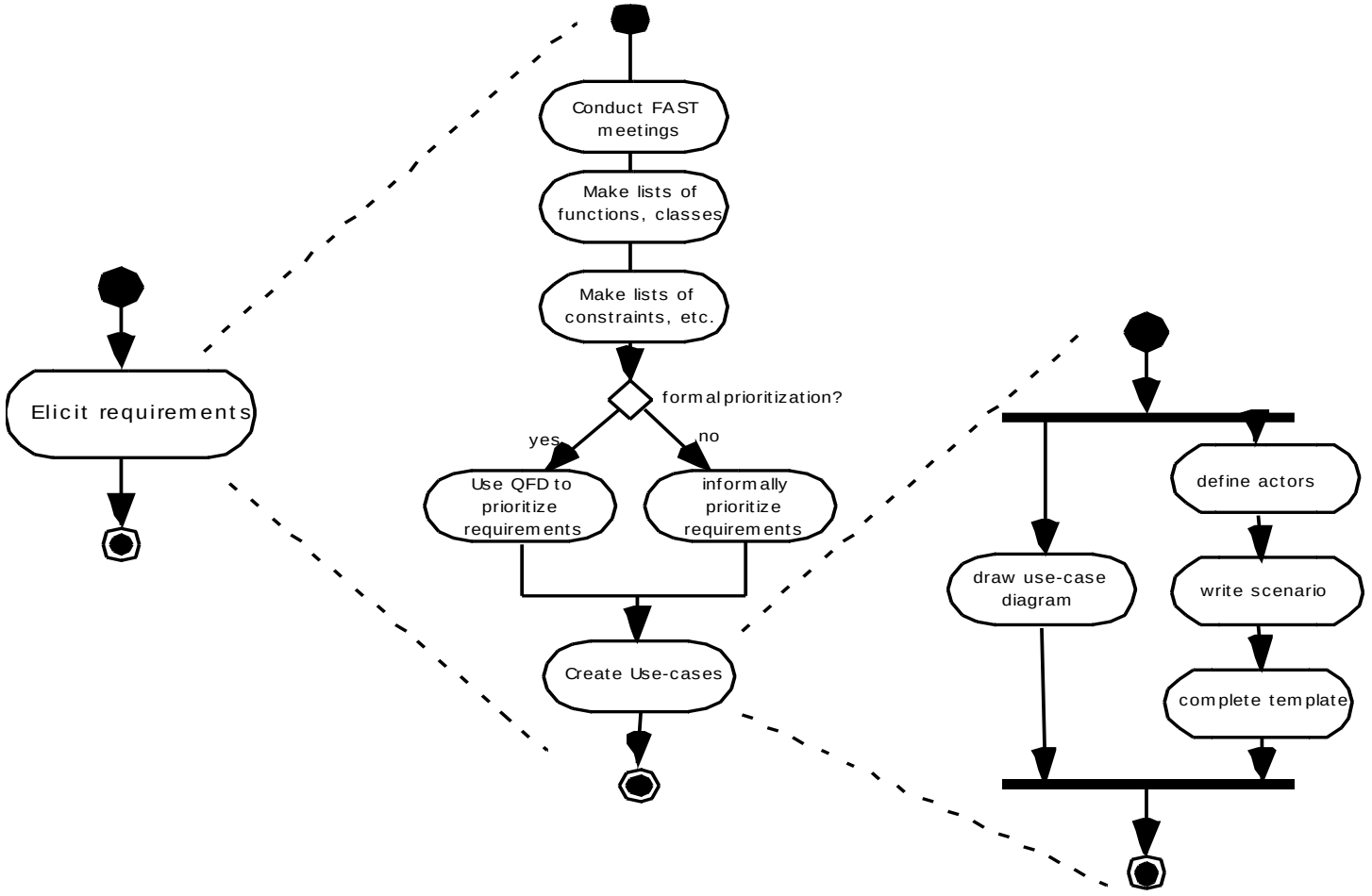
- Meetings are conducted and attended by both software engineers and customers
- Rules for preparation and participation are established
- An agenda is suggested
- A "facilitator" (can be a customer, a developer, or an outsider) controls the meeting
- A "definition mechanism" (can be work sheets, flip charts, or wall stickers or an electronic bulletin board, chat room or virtual forum) is used
- The goal is
  - to identify the problem
  - propose elements of the solution
  - negotiate different approaches, and
  - specify a preliminary set of solution requirements

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Requirement Engineering

Eliciting Requirements



BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Requirement Engineering

Quality Function Deployment

- Function deployment determines the “value” (as perceived by the customer) of each function required of the system
- Information deployment identifies data objects and events
- Task deployment examines the behavior of the system
- Value analysis determines the relative priority of requirements

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Requirement Engineering

Elicitation Work Products

- a statement of need and feasibility.
- a bounded statement of scope for the system or product.
- a list of customers, users, and other stakeholders who participated in requirements elicitation
- a description of the system's technical environment.
- a list of requirements (preferably organized by function) and the domain constraints that apply to each.
- a set of usage scenarios that provide insight into the use of the system or product under different operating conditions.
- any prototypes developed to better define requirements.

BUILDING THE ANALYSIS MODEL

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Building the Analysis Model

Elements of the analysis model :

- Scenario-based elements
  - Functional—processing narratives for software functions
  - Use-case—descriptions of the interaction between an “actor” and the system
- Class-based elements
  - Implied by scenarios
- Behavioral elements
  - State diagram
- Flow-oriented elements
  - Data flow diagram

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Building the Analysis Model

Use-Cases

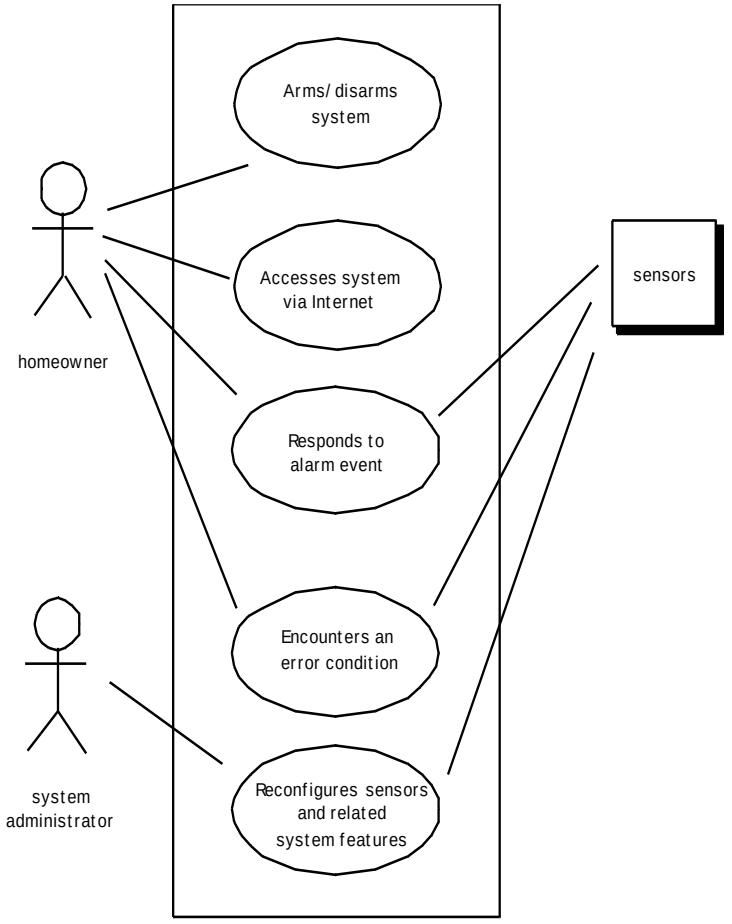
- A collection of user scenarios that describe the thread of usage of a system
- Each scenario is described from the point-of-view of an “actor”—a person or device that interacts with the software in some way
- Each scenario answers the following questions:
  - Who is the primary actor, the secondary actor (s)?
  - What are the actor’s goals?
  - What preconditions should exist before the story begins?
  - What main tasks or functions are performed by the actor?
  - What extensions might be considered as the story is described?
  - What variations in the actor’s interaction are possible?
  - What system information will the actor acquire, produce, or change?
  - Will the actor have to inform the system about changes in the external environment?
  - What information does the actor desire from the system?
  - Does the actor wish to be informed about unexpected changes?

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Building the Analysis Model

Use-Case Diagram



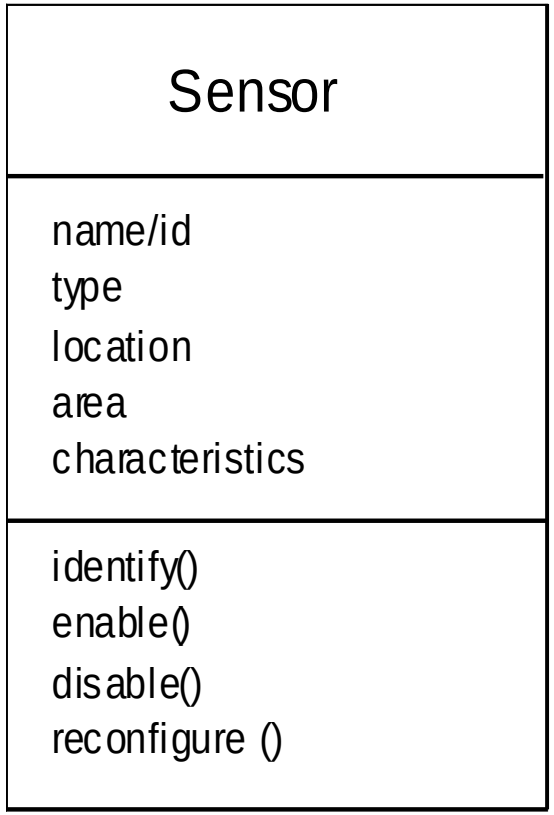
BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Building the Analysis Model

Class Diagram

From the *SafeHome* system …

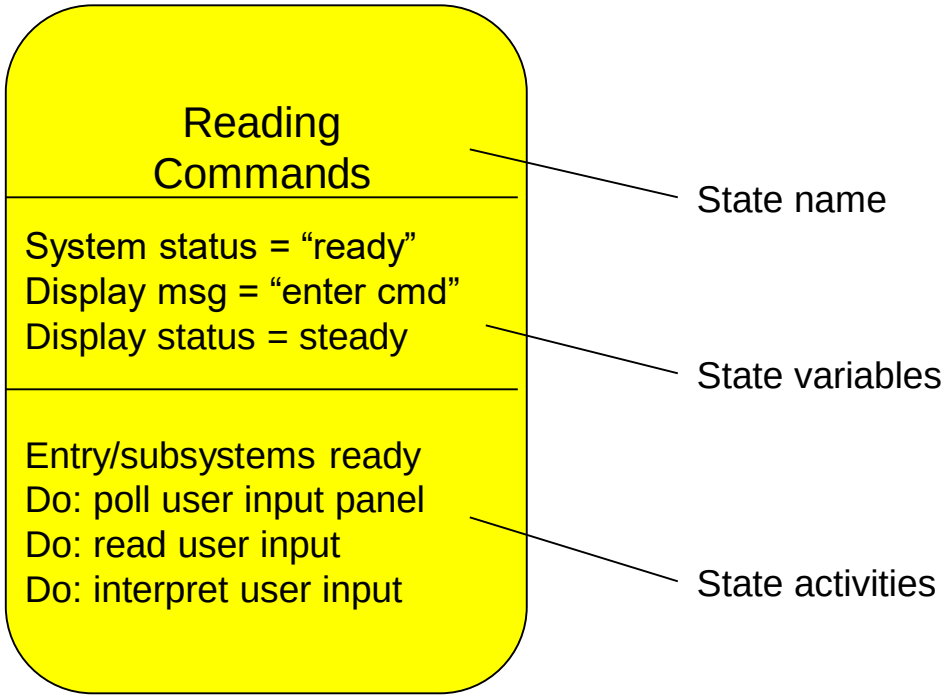


BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Building the Analysis Model

State Diagram



BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Analysis Patterns

Pattern name: A descriptor that captures the essence of the pattern.

Intent: Describes what the pattern accomplishes or represents

Motivation: A scenario that illustrates how the pattern can be used to address the problem.

Forces and context: A description of external issues (forces) that can affect how the pattern is used and also the external issues that will be resolved when the pattern is applied.

Solution: A description of how the pattern is applied to solve the problem with an emphasis on structural and behavioral issues.

Consequences: Addresses what happens when the pattern is applied and what trade-offs exist during its application.

Design: Discusses how the analysis pattern can be achieved through the use of known design patterns.

Known uses: Examples of uses within actual systems.

Related patterns: On e or more analysis patterns that are related to the named pattern because (1) it is commonly used with the named pattern; (2) it is structurally similar to the named pattern; (3) it is a variation of the named pattern.

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Negotiating Requirements

- Identify the key stakeholders
  - These are the people who will be involved in the negotiation
- Determine each of the stakeholders "win conditions"
  - Win conditions are not always obvious
- Negotiate
  - Work toward a set of requirements that lead to "win-win"

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

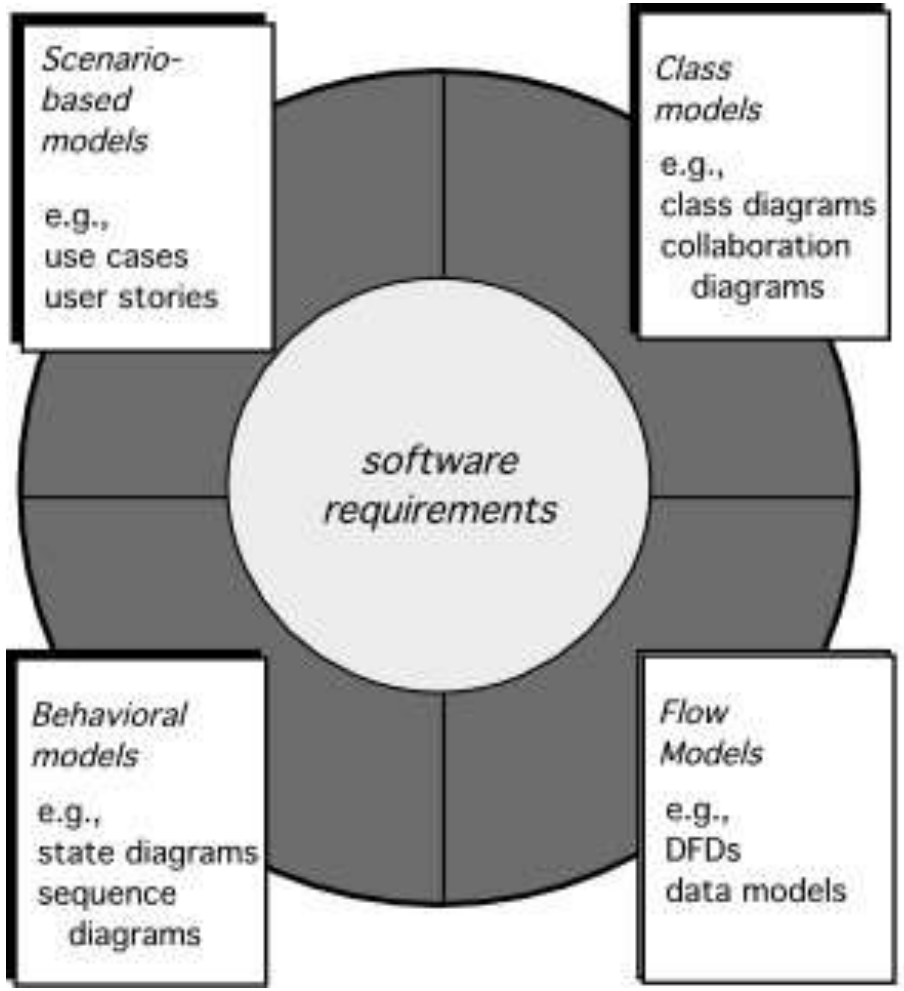
Validating Requirements

- Is each requirement achievable in the technical environment that will house the system or product?
- Is each requirement testable, once implemented?
- Does the requirements model properly reflect the information, function and behavior of the system to be built.
- Has the requirements model been “partitioned” in a way that exposes progressively more detailed information about the system.
- Have requirements patterns been used to simplify the requirements model. Have all patterns been properly validated? Are all patterns consistent with customer requirements?

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Requirement Analysis



REQUIREMENT MODELING

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Scenario-Based Modeling

"[Use-cases] are simply an aid to defining what exists outside the system (actors) and what should be performed by the system (use-cases)."
Ivar Jacobson

(1) What should we write about?

(2) How much should we write about it?

(3) How detailed should we make our description?

(4) How should we organize the description?

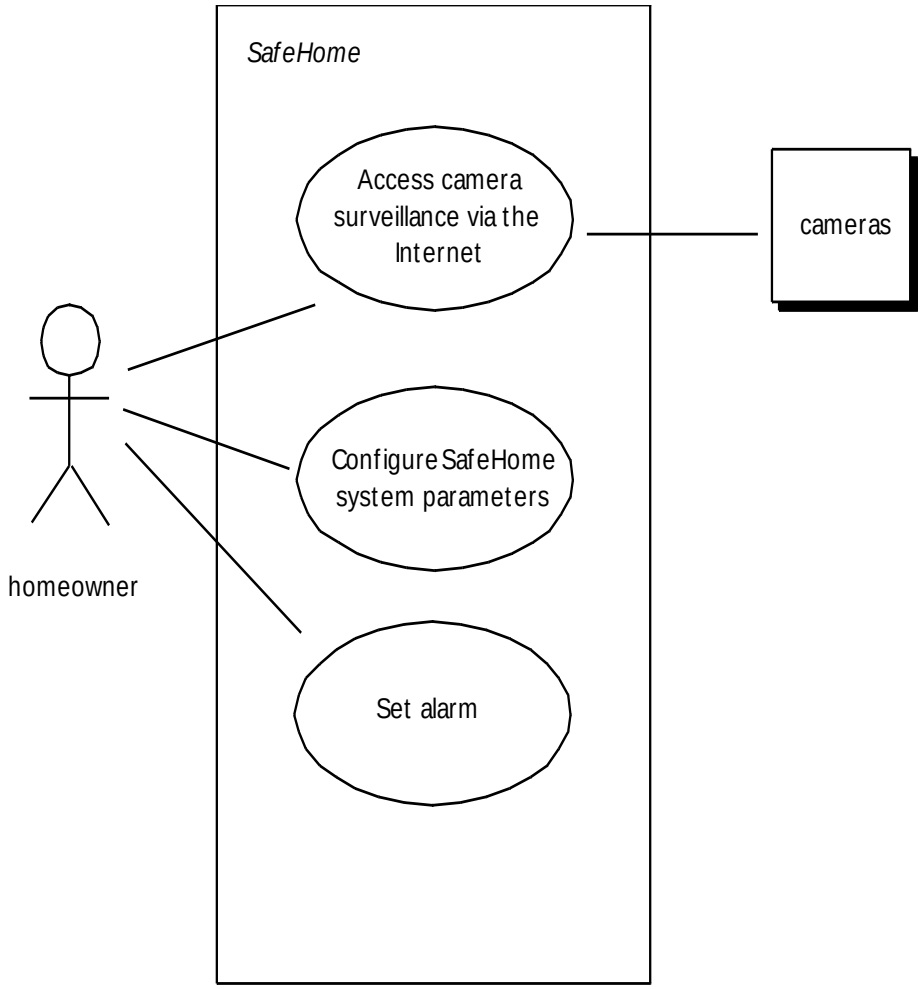
BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Scenario-Based Modeling

Use-Cases

- a scenario that describes a “thread of usage” for a system
- *actors* represent roles people or devices play as the system functions
- *users* can play a number of different roles for a given scenario



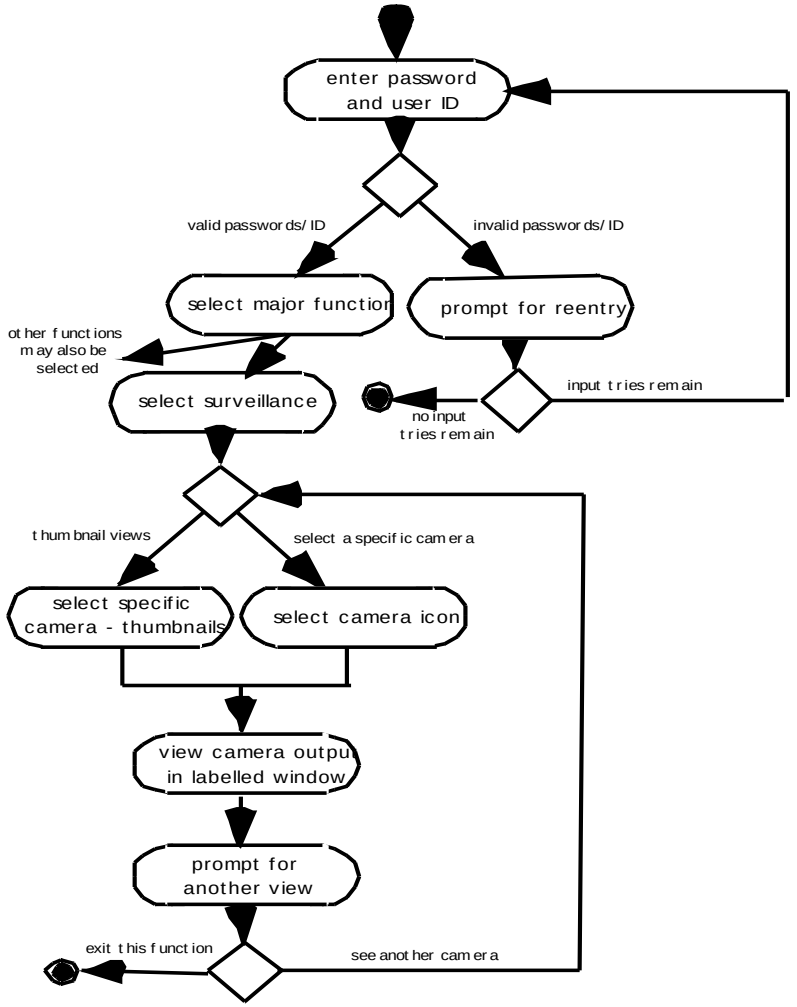
BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Scenario-Based Modeling

Activity Diagram

Supplements the use case by providing a graphical representation of the flow of interaction within a specific scenario



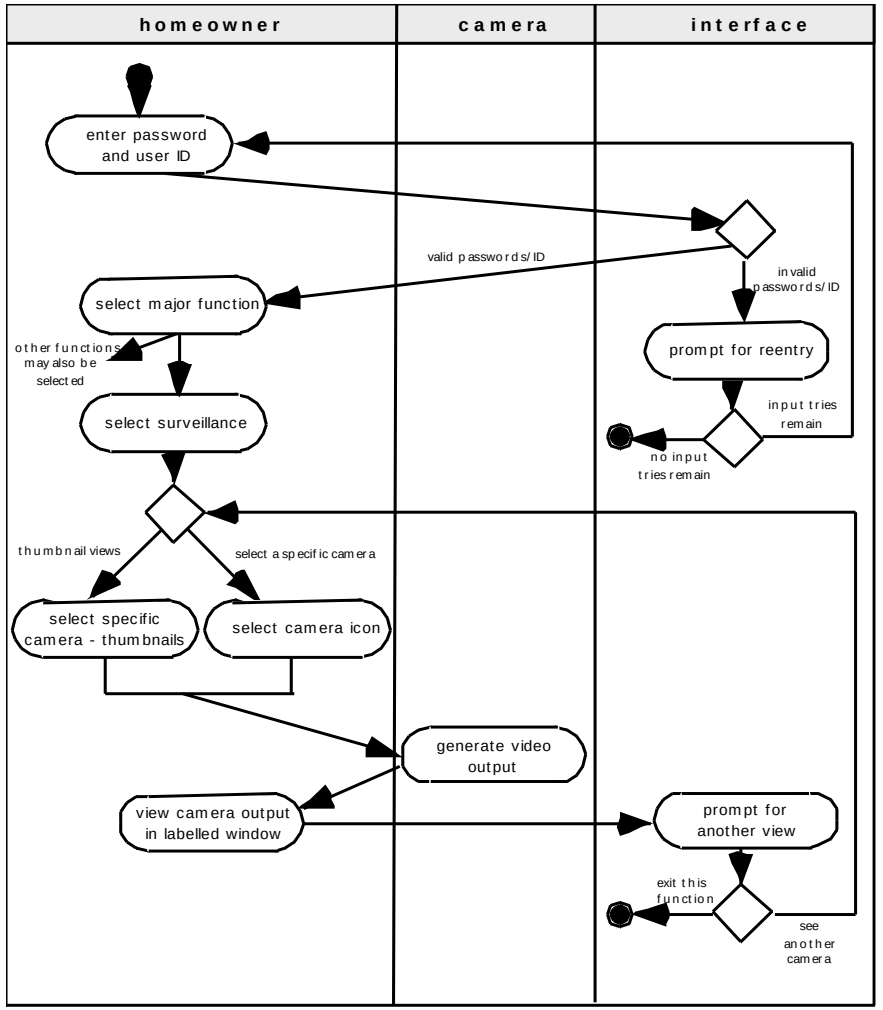
BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Scenario-Based Modeling

Swimlane Diagrams

Allows the modeler to represent the flow of activities described by the use-case and at the same time indicate which actor (if there are multiple actors involved in a specific use-case) or analysis class has responsibility for the action described by an activity rectangle



BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Class-Based Modeling

- Class-based modeling represents:
  - objects that the system will manipulate
  - operations (also called methods or services) that will be applied to the objects to effect the manipulation
  - relationships (some hierarchical) between the objects
  - collaborations that occur between the classes that are defined.
- The elements of a class-based model include classes and objects, attributes, operations, CRC models, collaboration diagrams and packages.

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Class-Based Modeling

Identifying Analysis Classes

- **Examining the usage scenarios developed as part of the requirements model and perform a "grammatical parse" [Abb83]**
  - **Classes are determined by underlining each noun or noun phrase and entering it into a simple table.**
  - **Synonyms should be noted.**
  - **If the class (noun) is required to implement a solution, then it is part of the solution space; otherwise, if a class is necessary only to describe a solution, it is part of the problem space.**

- ***But what should we look for once all of the nouns have been isolated?***

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Class-Based Modeling

Manifestations of Analysis Classes

Analysis classes **manifest themselves in one of the following ways:**

- ***External entities:*** **(e.g., other systems, devices, people) that produce or consume information**
- ***Things*****: (e.g, reports, displays, letters, signals) that are part of the information domain for the problem**
- ***Occurrences or events:*** **(e.g., a property transfer or the completion of a series of robot movements) that occur within the context of system operation**
- ***Roles:*** **(e.g., manager, engineer, salesperson) played by people who interact with the system**
- ***Organizational units:*** **(e.g., division, group, team) that are relevant to an application**
- ***Places:*** **(e.g., manufacturing floor or loading dock) that establish the context of the problem and the overall function**
- ***Structures: (*****e.g., sensors, four-wheeled vehicles, or computers) that define a class of objects or related classes of objects**

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Class-Based Modeling

Defining Attributes

- *Attributes* describe a class that has been selected for inclusion in the analysis model.
  - build two different classes for professional baseball players
    - **For Playing Statistics software:** name, position, batting average, fielding percentage, years played, and games played might be relevant
    - **For Pension Fund software:** average salary, credit toward full vesting, pension plan options chosen, mailing address, and the like.

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Class-Based Modeling

Defining Operations

- Do a grammatical parse of a processing narrative and look at the verbs
- Operations can be divided into four broad categories:
  - (1) operations that manipulate data in some way (e.g., adding, deleting, reformatting, selecting)
  - (2) operations that perform a computation
  - (3) operations that inquire about the state of an object, and
  - (4) operations that monitor an object for the occurrence of a controlling event.

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Class-Based Modeling

CRC Models

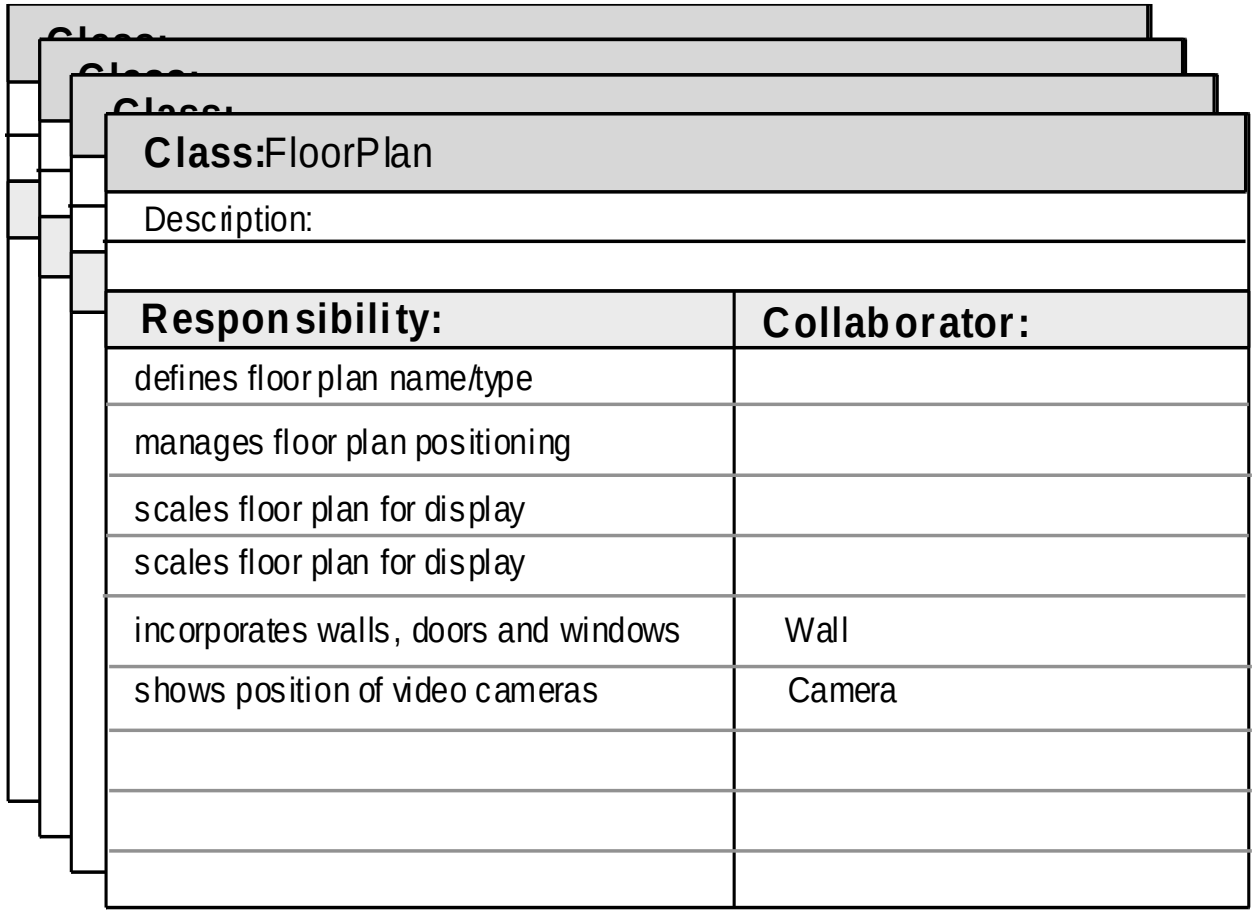
- *Class-responsibility-collaborator (CRC) modeling* [Wir90] provides a simple means for identifying and organizing the classes that are relevant to system or product requirements. Ambler [Amb95] describes CRC modeling in the following way:
  - A CRC model is really a collection of standard index cards that represent classes. The cards are divided into three sections. Along the top of the card you write the name of the class. In the body of the card you list the class responsibilities on the left and the collaborators on the right.

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Class-Based Modeling

CRC Modeling

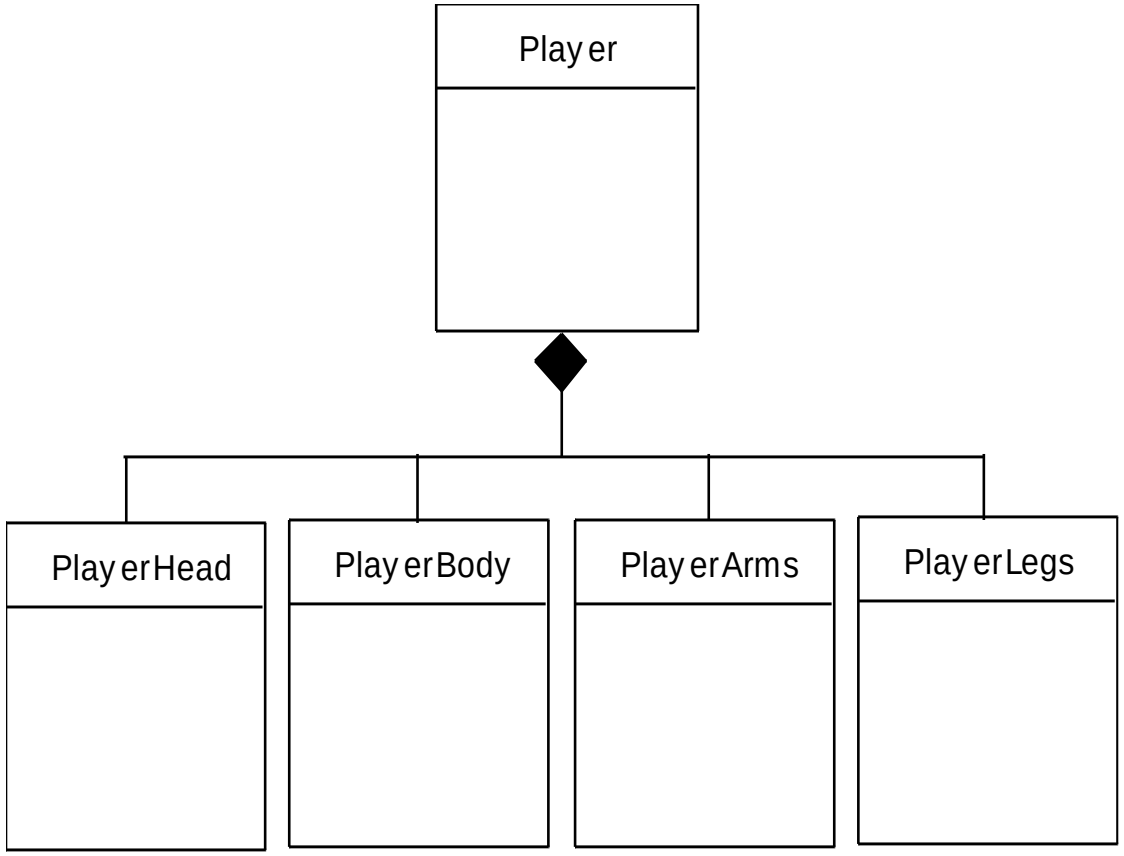


BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Class-Based Modeling

Composite Aggregate Class



BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Class-Based Modeling

Associations and Dependencies

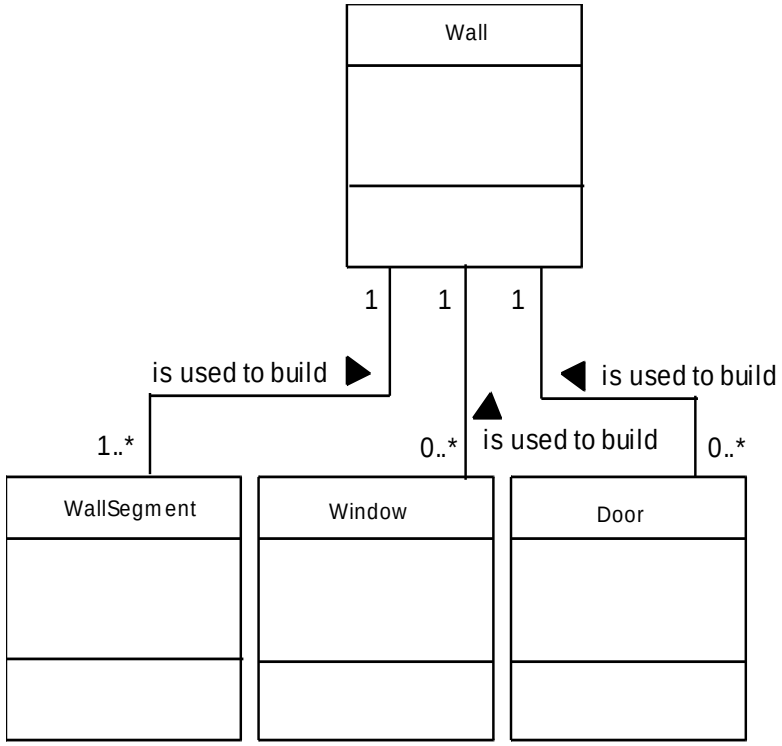
- Two analysis classes are often related to one another in some fashion
  - In UML these relationships are called *associations*
  - Associations can be refined by indicating *multiplicity* (the term *cardinality* is used in data modeling
- In many instances, a client-server relationship exists between two analysis classes.
  - In such cases, a client-class depends on the server-class in some way and a *dependency relationship* is established

BINUS UNIVERSITY ONLINE LEARNING

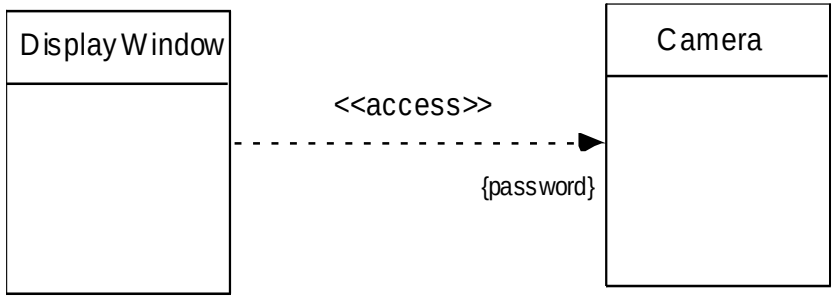
People Innovation Excellence

Class-Based Modeling

Multiplicity



Dependencies



BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Class-Based Modeling

Analysis Packages

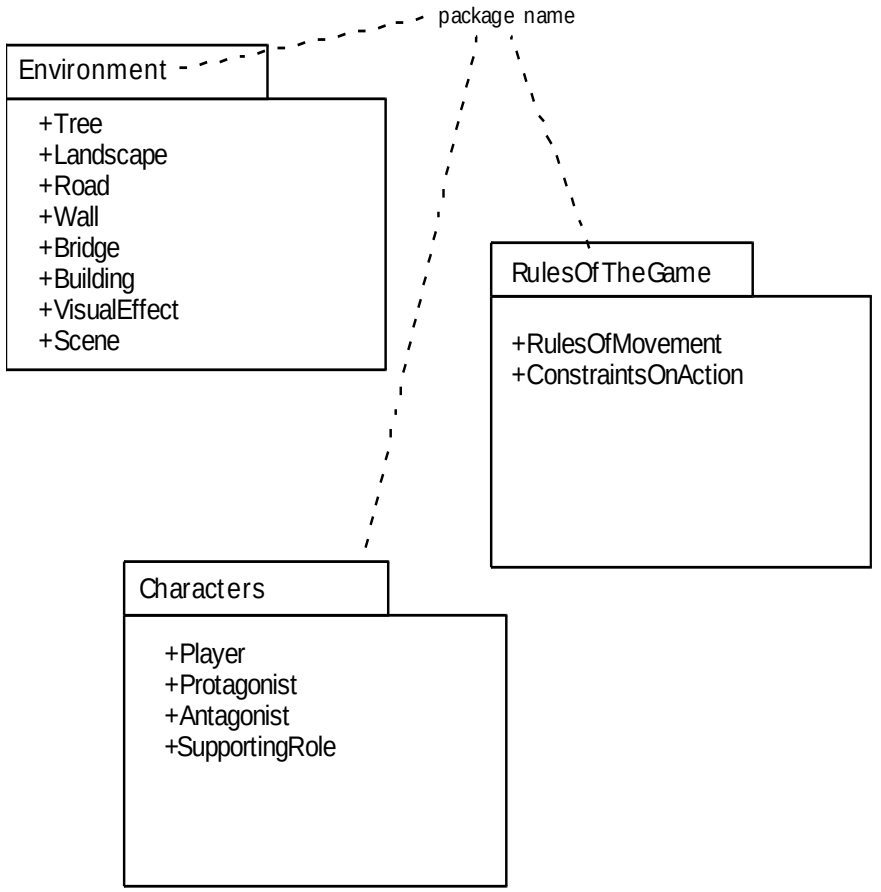
- Various elements of the analysis model (e.g., use-cases, analysis classes) are categorized in a manner that packages them as a grouping
- The plus sign preceding the analysis class name in each package indicates that the classes have public visibility and are therefore accessible from other packages.
- Other symbols can precede an element within a package. A minus sign indicates that an element is hidden from all other packages and a # symbol indicates that an element is accessible only to packages contained within a given package.

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Class-Based Modeling

Analysis Packages



BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Creating A Behavioral Modeling

- The behavioral model indicates how software will respond to external events or stimuli. To create the model, the analyst must perform the following steps:
  - Evaluate all use-cases to fully understand the sequence of interaction within the system.
  - Identify events that drive the interaction sequence and understand how these events relate to specific objects.
  - Create a sequence for each use-case.
  - Build a state diagram for the system.
  - Review the behavioral model to verify accuracy and consistency.

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Creating A Behavioral Modeling

State Representations

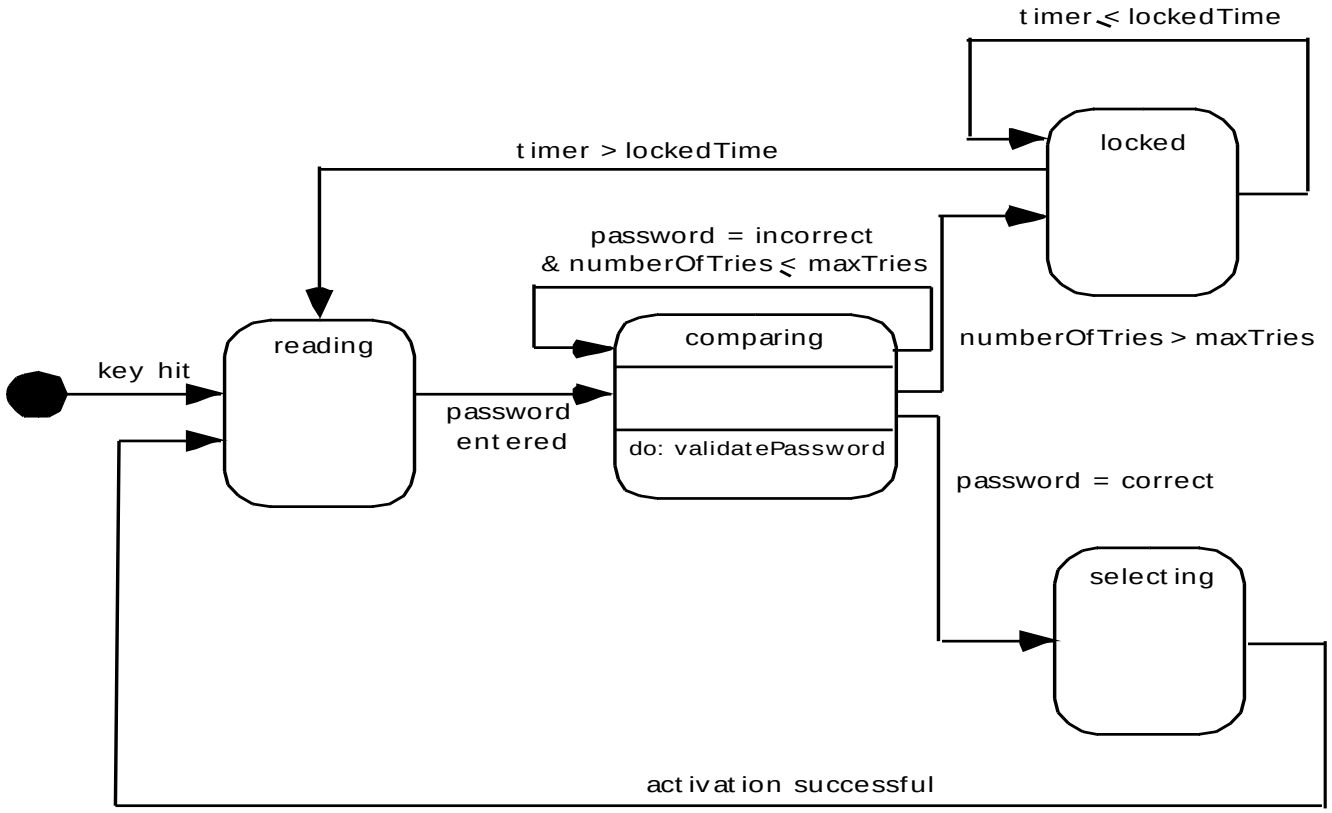
- In the context of behavioral modeling, two different characterizations of states must be considered:
  - the state of each class as the system performs its function and
  - the state of the system as observed from the outside as the system performs its function
- The state of a class takes on both passive and active characteristics [CHA93].
  - A *passive state* is simply the current status of all of an object's attributes.
  - The *active state* of an object indicates the current status of the object as it undergoes a continuing transformation or processing.

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Creating A Behavioral Modeling

State Diagram for the ControlPanel Class



BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Creating A Behavioral Modeling

The States of a System

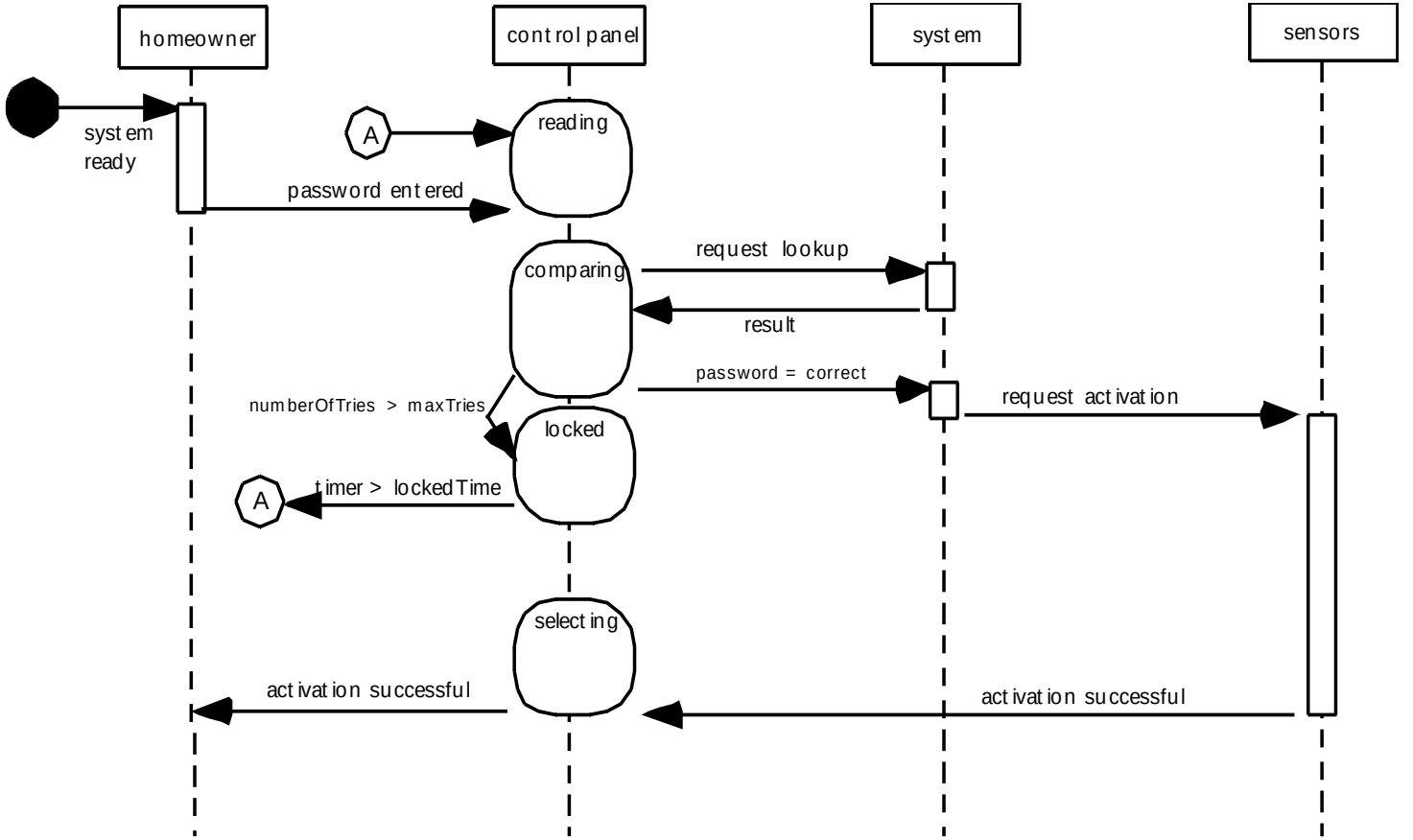
- state—a set of observable circum-stances that characterizes the behavior of a system at a given time
- state transition—the movement from one state to another
- event—an occurrence that causes the system to exhibit some predictable form of behavior
- action—process that occurs as a consequence of making a transition

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Creating A Behavioral Modeling

Sequence Diagram



REQUIREMENT MODELING FOR WEB AND MOBILE APPS

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Requirements Modeling for Web and Mobile Apps

When Do We Perform Analysis?

- In some WebE situations, analysis and design merge. However, **an explicit analysis activity occurs when …**
  - the WebApp to be built is large and/or complex
  - the number of stakeholders is large
  - the number of Web engineers and other contributors is large
  - the goals and objectives (determined during formulation) for the WebApp will effect the business' bottom line
  - the success of the WebApp will have a strong bearing on the success of the business

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Requirements Modeling for Web and Mobile Apps

The Content Model

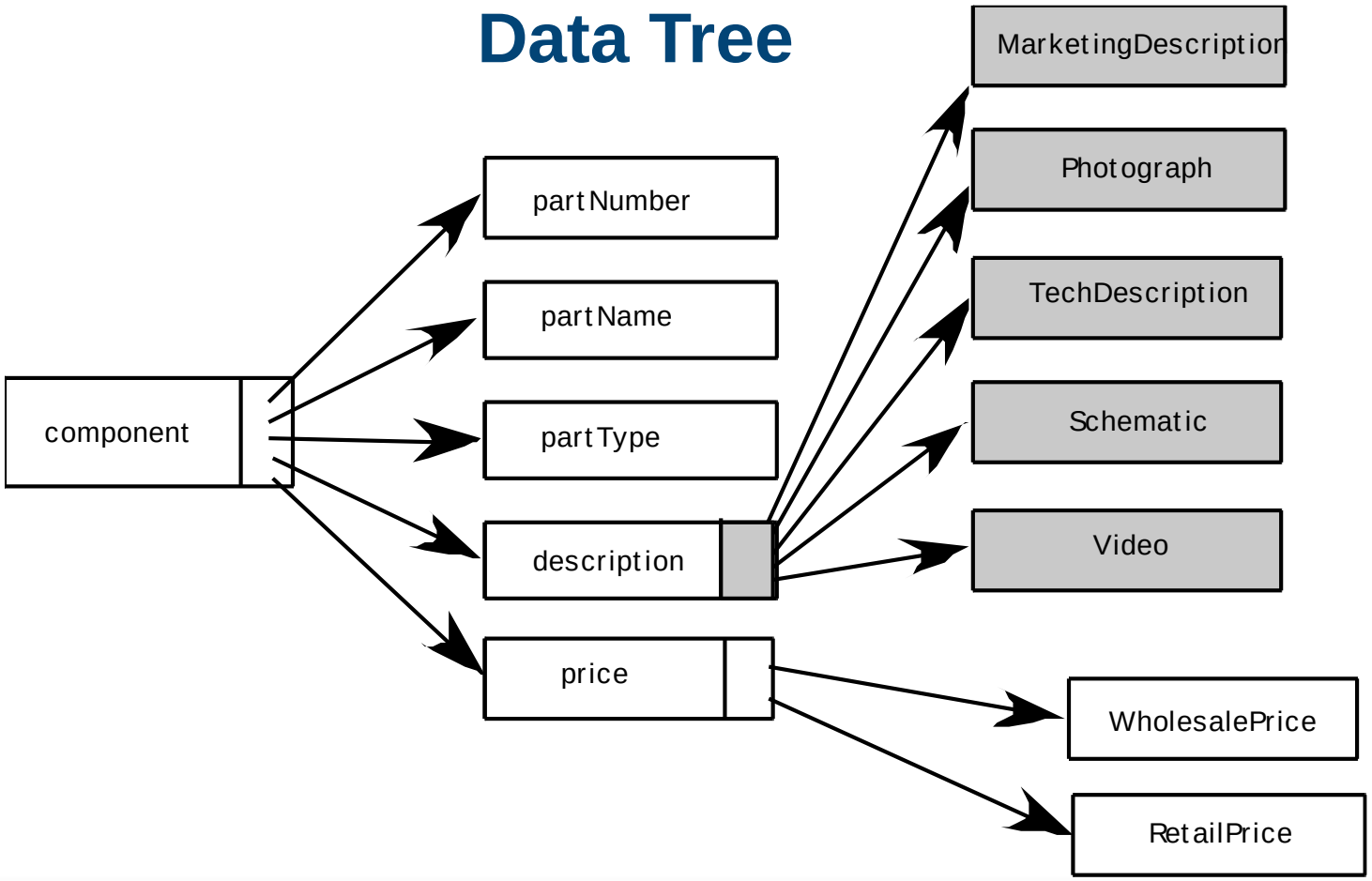
- Content objects are extracted from use-cases
  - examine the scenario description for direct and indirect references to content
- Attributes of each content object are identified
- The relationships among content objects and/or the hierarchy of content maintained by a WebApp
  - Relationships—entity-relationship diagram or UML
  - Hierarchy—data tree or UML

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Requirements Modeling for Web and Mobile Apps

Data Tree



BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Requirements Modeling for Web and Mobile Apps

The Interaction Model

- Composed of four elements:
  - use-cases
  - sequence diagrams
  - state diagrams
  - a user interface prototype

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Requirements Modeling for Web and Mobile Apps

The Functional Model

- The functional model addresses two processing elements of the WebApp
  - user observable functionality that is delivered by the WebApp to end-users
  - the operations contained within analysis classes that implement behaviors associated with the class.
- An activity diagram can be used to represent processing flow

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Requirements Modeling for Web and Mobile Apps

The Configuration Model

- Server-side
  - Server hardware and operating system environment must be specified
  - Interoperability considerations on the server-side must be considered
  - Appropriate interfaces, communication protocols and related collaborative information must be specified
- Client-side
  - Browser configuration issues must be identified
  - Testing requirements should be defined

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

References

- Pressman, R.S. (2015). ***Software Engineering : A Practioner's Approach. 8th ed***. McGraw-Hill Companies.Inc, Americas, New York. ISBN : 978 1 259 253157
- UML Specification, https://www.omg.org/spec/UML/About-UML/
- UML 2.0 Tutorial, **http://www.youtube.com/watch?v=OkC7HKtiZC0**
- Introduction to Model-View-View-Model (MVVM) Pattern, http://www.youtube.com/watch?v=pY60lLotZCg

BINUS UNIVERSITY ONLINE LEARNING

People
Innovation
Excellence

Q & A

BINUS
UNIVERSITY
ONLINE
LEARNING

People
Innovation
Excellence

Thank You

BINUS UNIVERSITY ONLINE LEARNING

People
Innovation
Excellence

Software Engineering

Topic 4

Design Engineering

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Acknowledgement

These slides have been adapted from
Pressman, R.S. (2015). *Software Engineering : A Practioner's Approach. 8th ed*. McGraw-Hill Companies.Inc, Americas, New York. ISBN : 978 1 259 253157. Chapter 12, 13, 14, 15, 16, 17, and 18

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Learning Objectives

LO2 : Explain the software engineering practices and business environment

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Contents

- Software Architecture
- Architectural Design
- Component Level Design
- Interface Design
- Design Evaluation Cycle
- Design Patterns
- Architectural Patterns
- WebApps Design
- MobileApps Design

SOFTWARE ARCHITECTURE

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Software Architecture

The architecture is not the operational software. Rather, it is a representation that enables a software engineer to:

(1) analyze the effectiveness of the design in meeting its stated requirements,

(2) consider architectural alternatives at a stage when making design changes is still relatively easy, and

(3) reduce the risks associated with the construction of the software.

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Software Architecture

Architectural Styles

Each style describes a system category that encompasses: (1) a **set of components** (e.g., a database, computational modules) that perform a function required by a system, (2) a **set of connectors** that enable “communication, coordination and cooperation” among components, (3) **constraints** that define how components can be integrated to form the system, and (4) **semantic models** that enable a designer to understand the overall properties of a system by analyzing the known properties of its constituent parts.

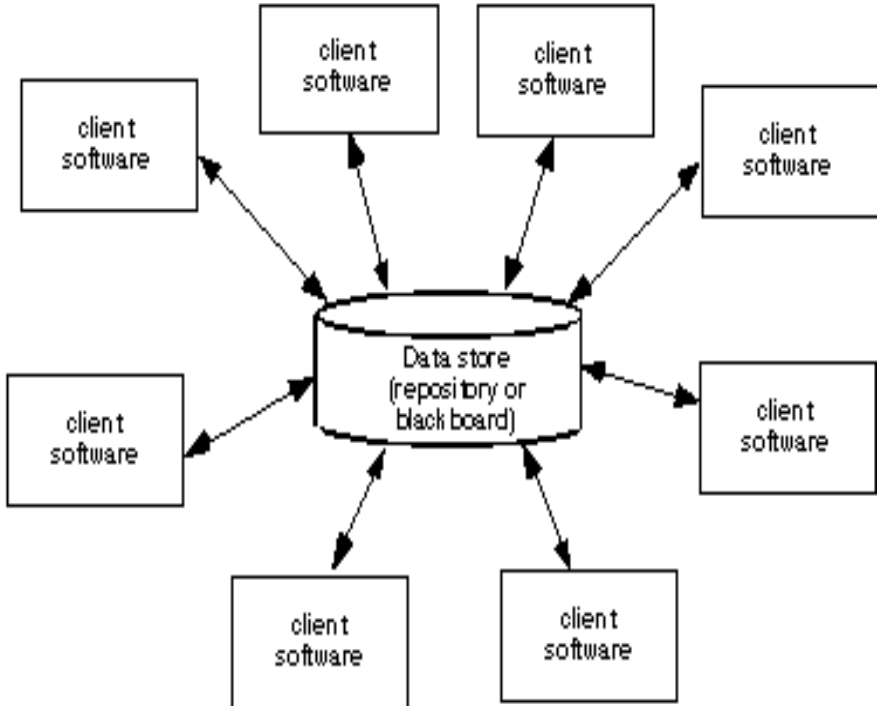
- **Data-centered architectures**
- **Data flow architectures**
- **Call and return architectures**
- **Object-oriented architectures**
- **Layered architectures**

BINUS UNIVERSITY ONLINE LEARNING

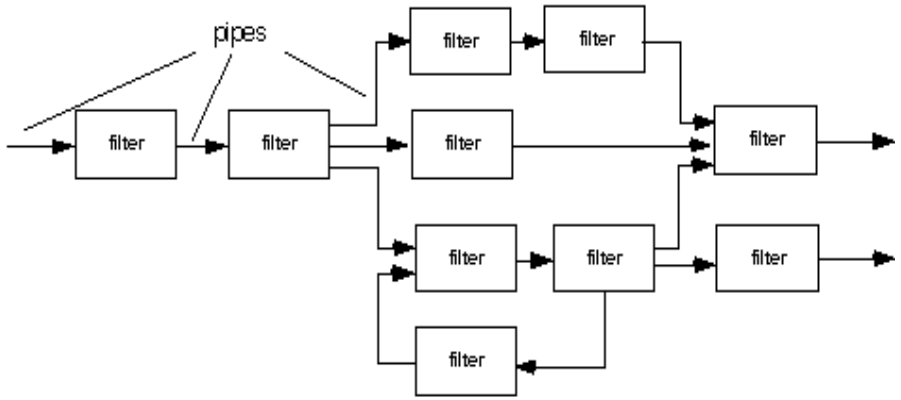
People Innovation Excellence

Software Architecture

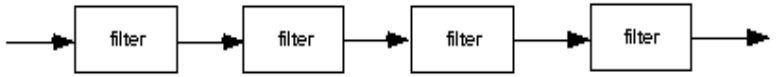
Data Centered Architecture



Data Flow Architecture



(a) pipes and filters



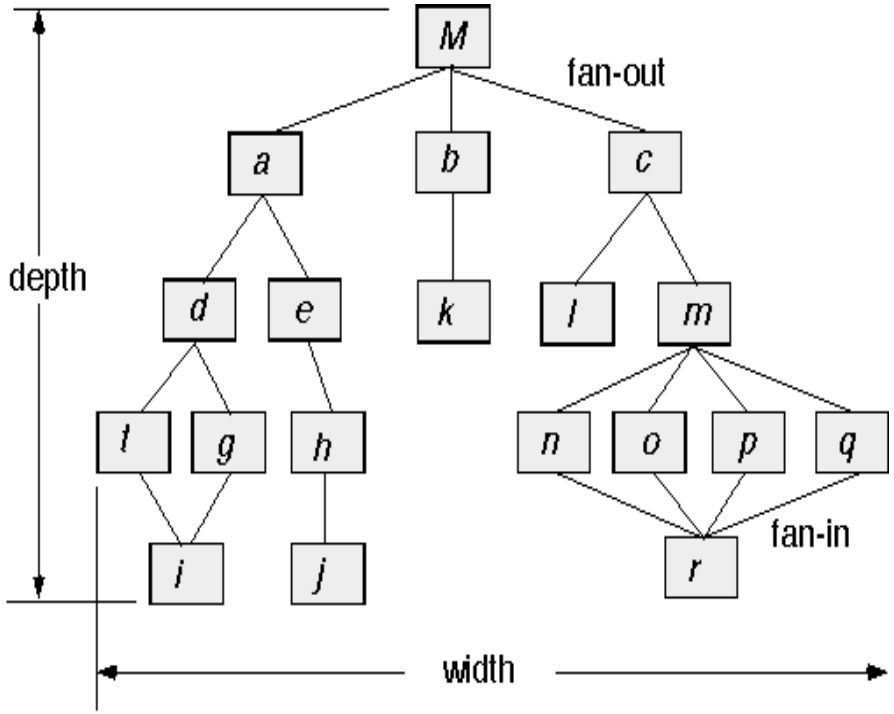
(b) batch sequential

BINUS UNIVERSITY ONLINE LEARNING

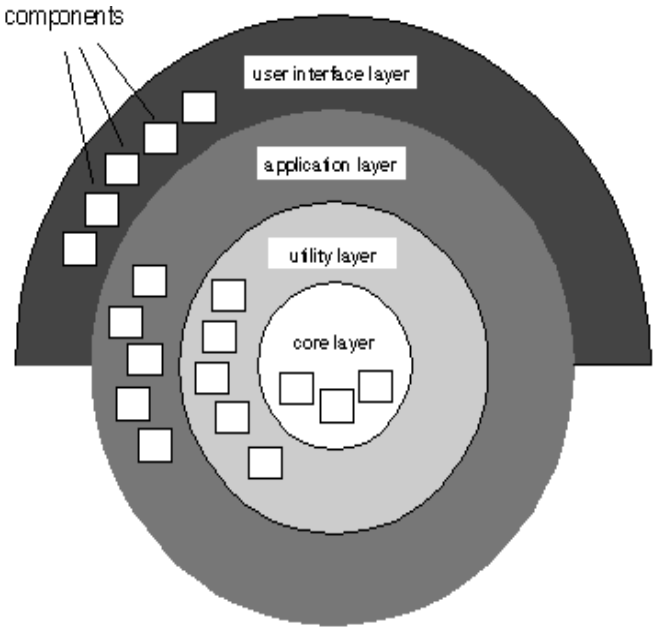
People Innovation Excellence

Software Architecture

Call and Return Architecture



Layered Architecture



ARCHITECTURAL DESIGN

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Architectural Design

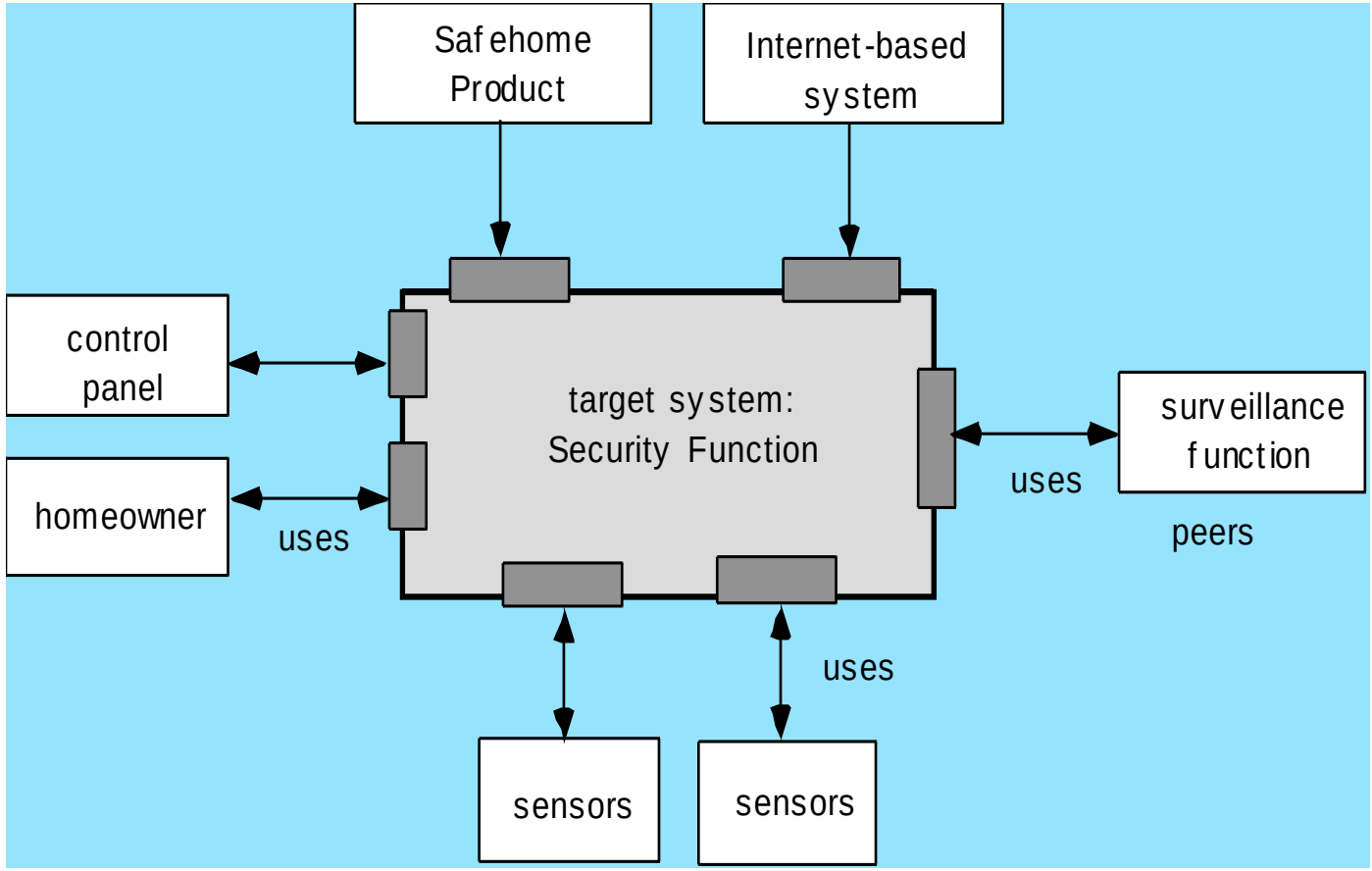
- The software must be placed into context
  - the design should define the external entities (other systems, devices, people) that the software interacts with and the nature of the interaction
- A set of architectural archetypes should be identified
  - An *archetype* is an abstraction (similar to a class) that represents one element of system behavior
- The designer specifies the structure of the system by defining and refining software components that implement each archetype

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Architectural Design

Architectural Context

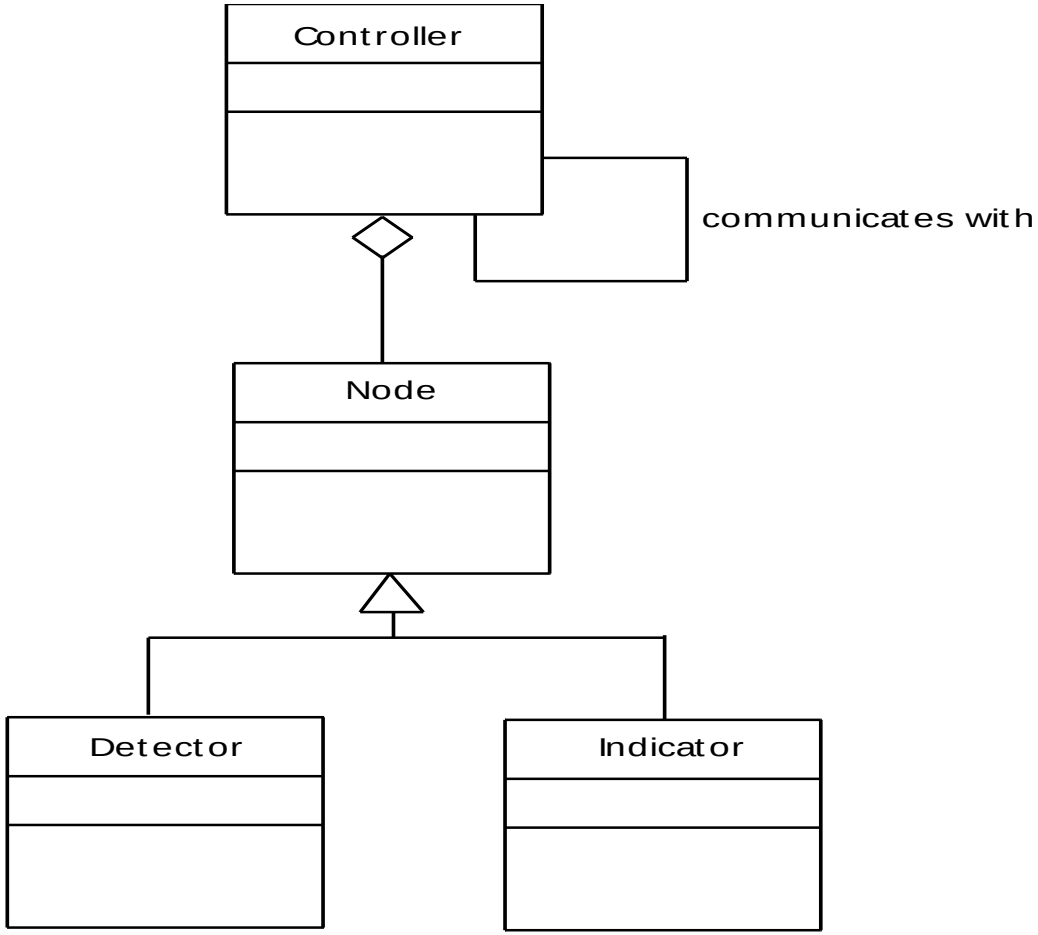


BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Architectural Design

Archetypes

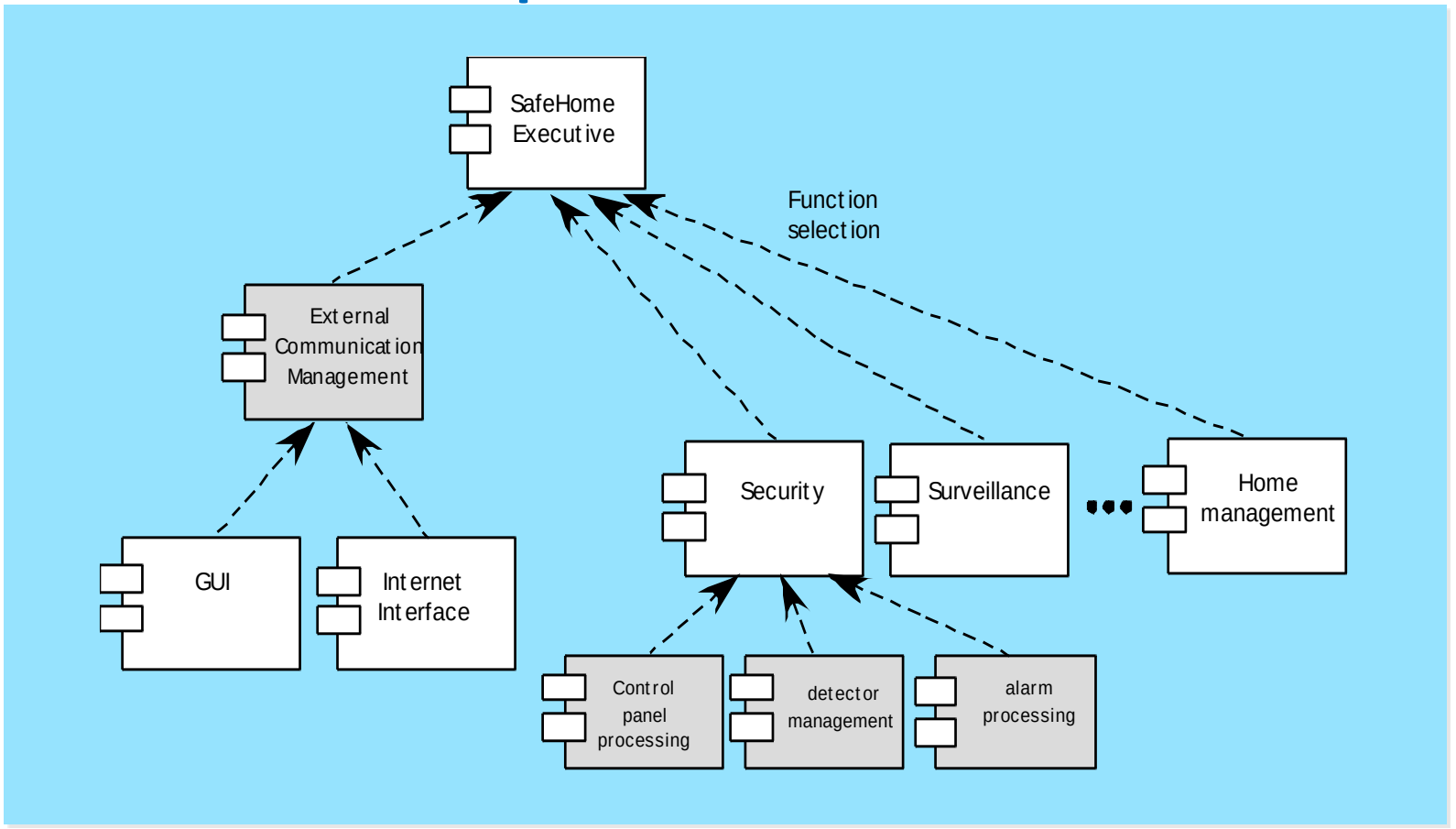


BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Architectural Design

Component Structure

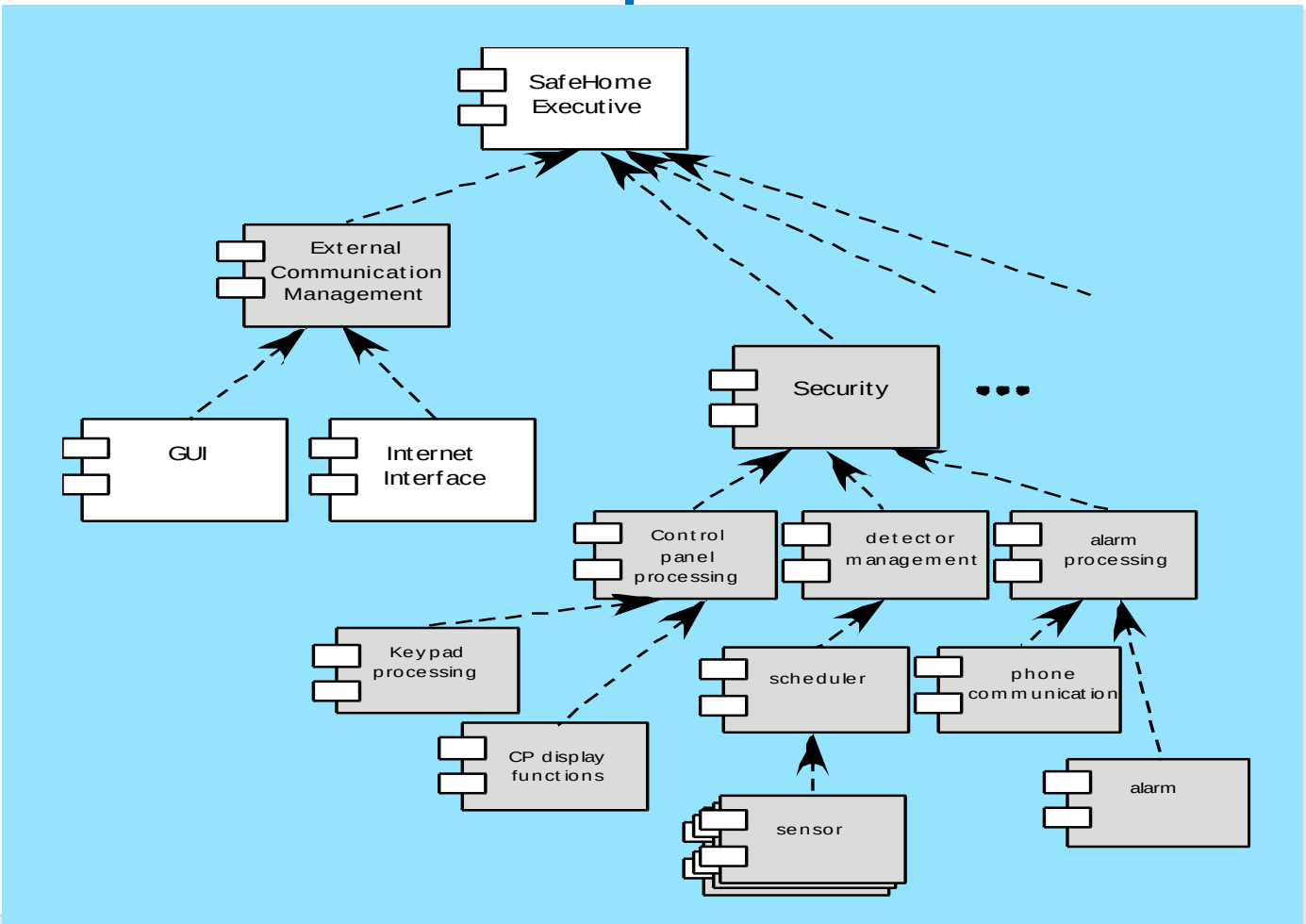


BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Architectural Design

Refined Component Structure



COMPONEN LEVEL DESIGN

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Component Level Design

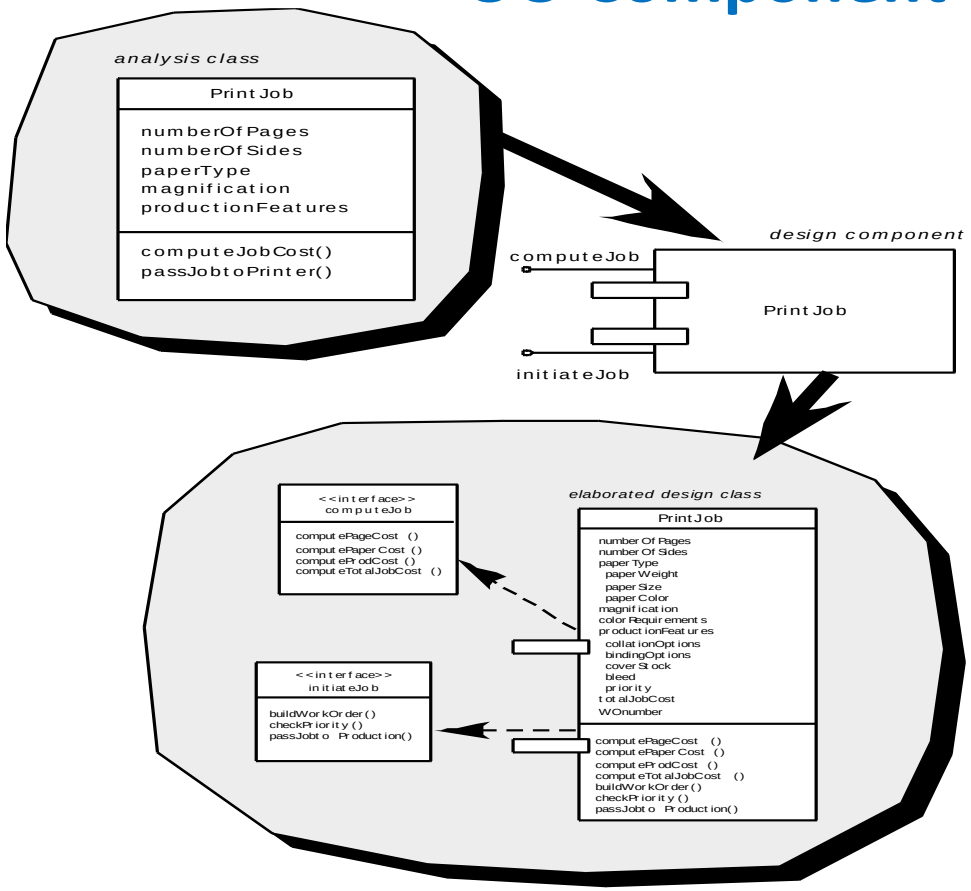
- *OMG Unified Modeling Language Specification* [OMG01] defines a component as
  - "… a modular, deployable, and replaceable part of a system that encapsulates implementation and exposes a set of interfaces."
- *OO view:* a component contains a set of collaborating classes
- *Conventional view:* a component contains processing logic, the internal data structures that are required to implement the processing logic, and an interface that enables the component to be invoked and data to be passed to it.

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Component Level Design

OO Component

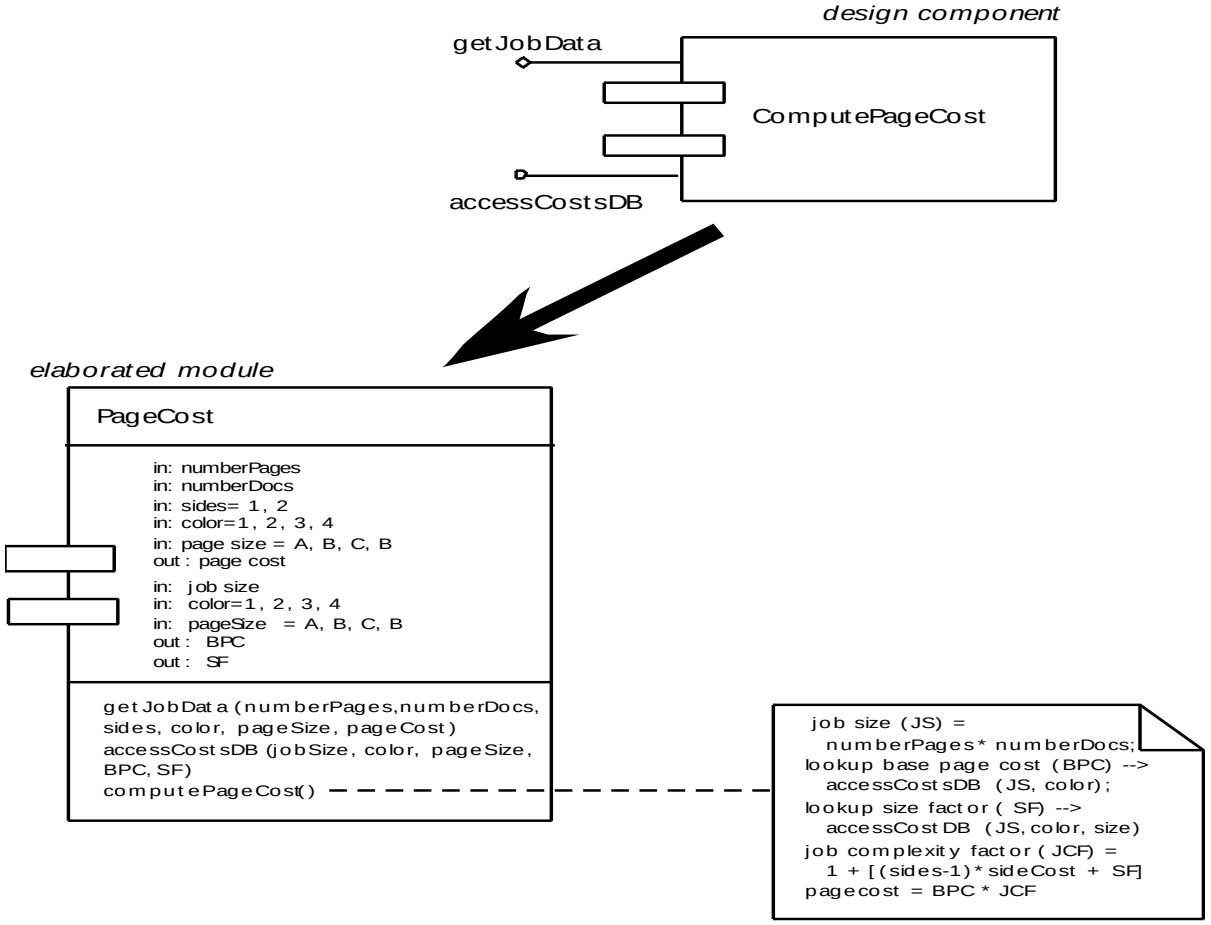


BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Component Level Design

Conventional Component



BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Component Level Design

Cohesion

- **Conventional view:**
  - **the "single-mindedness" of a module**
- **OO view:**
  - ***cohesion* implies that a component or class encapsulates only attributes and operations that are closely related to one another and to the class or component itself**
- **Levels of cohesion**
  - **Functional**
  - **Layer**
  - **Communicational**
  - **Sequential**
  - **Procedural**
  - **Temporal**
  - **utility**

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Component Level Design

Coupling

- **Conventional view:**
  - **The degree to which a component is connected to other components and to the external world**
- **OO view:**
  - **a qualitative measure of the degree to which classes are connected to one another**
- **Level of coupling**
  - **Content**
  - **Common**
  - **Control**
  - **Stamp**
  - **Data**
  - **Routine call**
  - **Type use**
  - **Inclusion or import**
  - **External**

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Component Level Design

Component Design for WebApps

- WebApp component is
  - (1) a well-defined cohesive function that manipulates content or provides computational or data processing for an end-user, or
  - (2) a cohesive package of content and functionality that provides end-user with some required capability.
- Therefore, component-level design for WebApps often incorporates elements of content design and functional design.

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Component Level Design

Component-Based Development

- When faced with the possibility of reuse, the software team asks:
  - Are commercial off-the-shelf (COTS) components available to implement the requirement?
  - Are internally-developed reusable components available to implement the requirement?
  - Are the interfaces for available components compatible within the architecture of the system to be built?
- At the same time, they are faced with the following impediments to reuse ...

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Component Level Design

User Interface Design Models

- User model — a profile of all end users of the system
- Design model — a design realization of the user model
- Mental model (system perception) — the user's mental image of what the interface is
- Implementation model — the interface "look and feel" coupled with supporting information that describe interface syntax and semantics

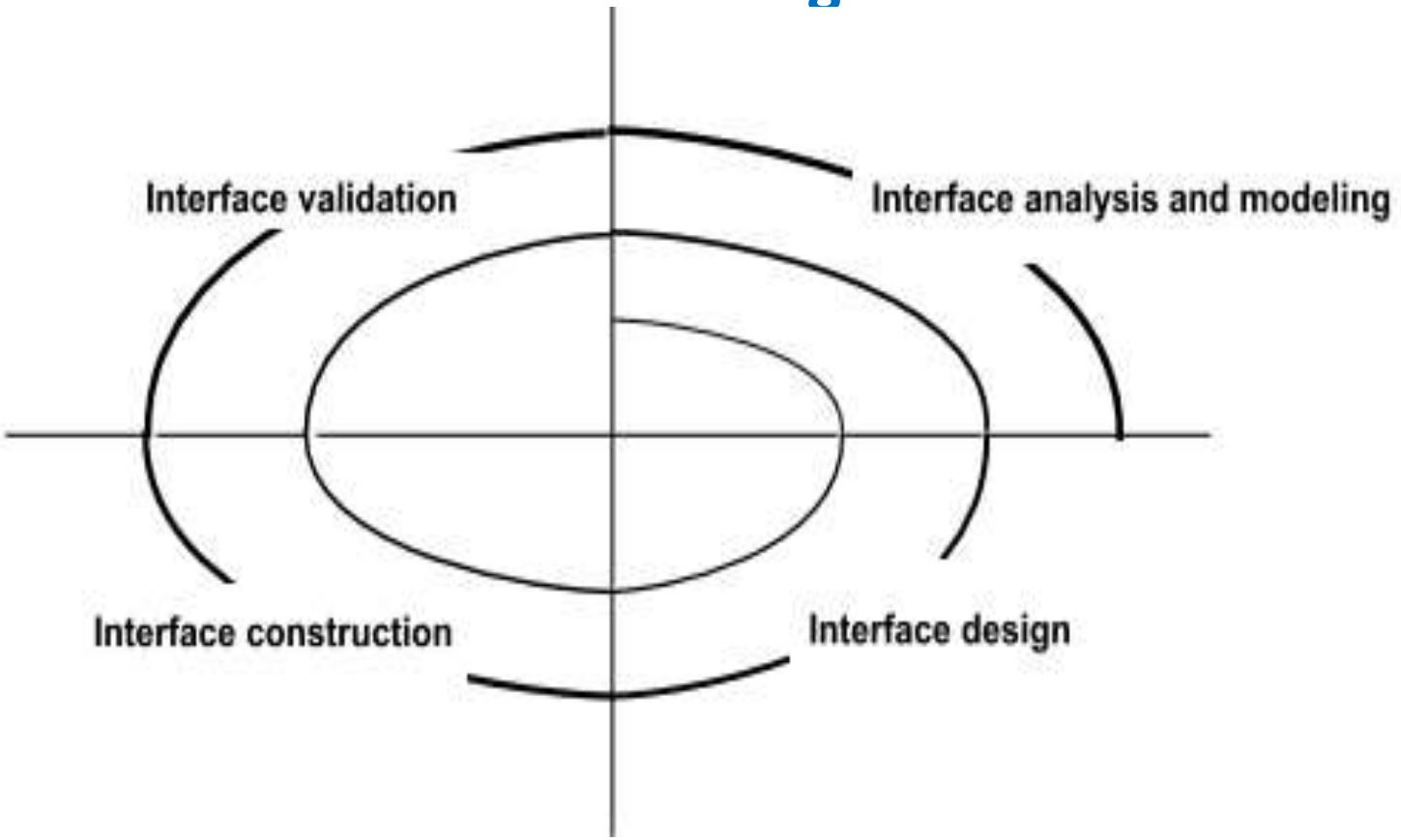
INTERFACE DESIGN

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Interface Design

User Interface Design Process



BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Interface Design

Interface Analysis

- Interface analysis means understanding
  - (1) the people (end-users) who will interact with the system through the interface;
  - (2) the tasks that end-users must perform to do their work,
  - (3) the content that is presented as part of the interface
  - (4) the environment in which these tasks will be conducted.

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Interface Design

Task Analysis and Modeling

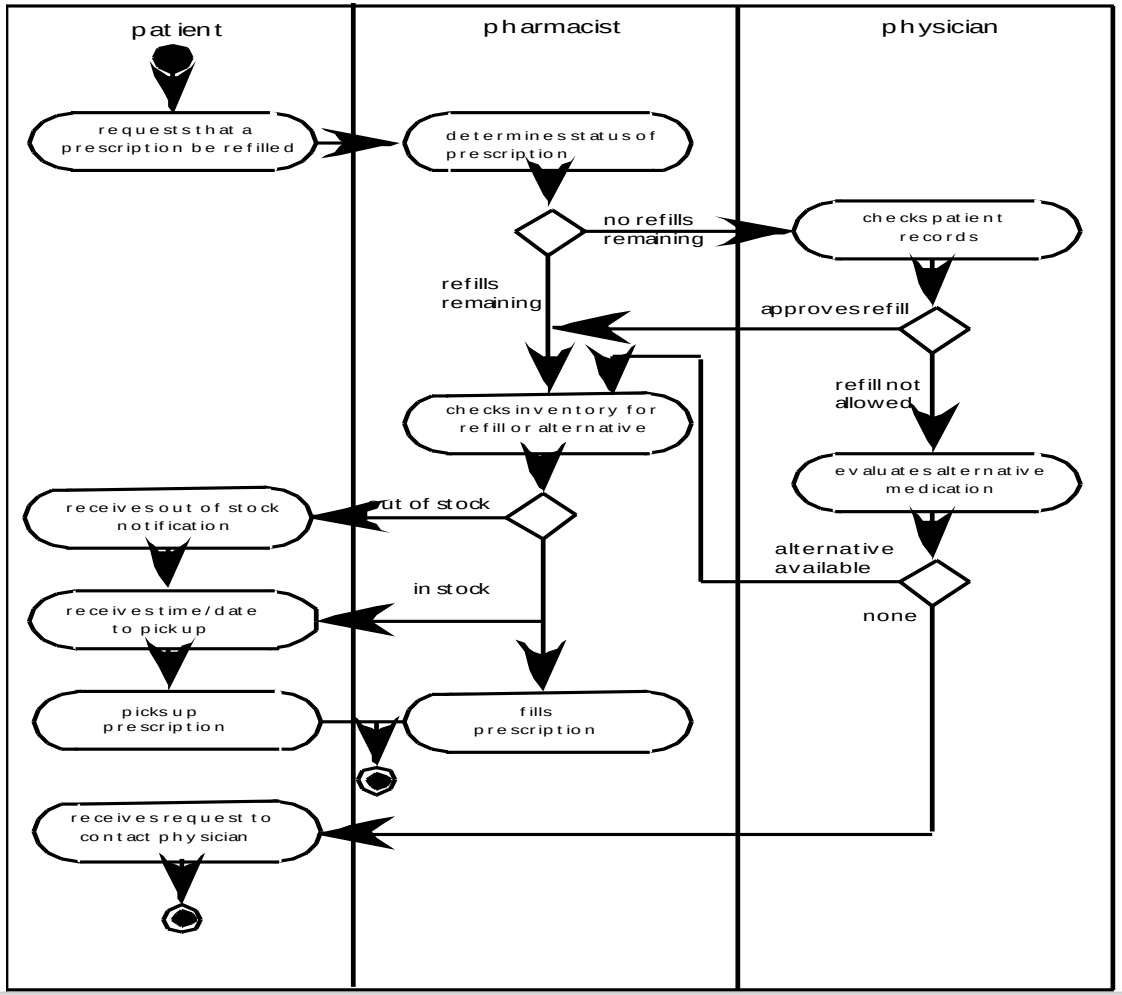
- Answers the following questions …
  - What work will the user perform in specific circumstances?
  - What tasks and subtasks will be performed as the user does the work?
  - What specific problem domain objects will the user manipulate as work is performed?
  - What is the sequence of work tasks—the workflow?
  - What is the hierarchy of tasks?
- Use-cases define basic interaction
- Task elaboration refines interactive tasks
- Object elaboration identifies interface objects (classes)
- Workflow analysis defines how a work process is completed when several people (and roles) are involved

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Interface Design

Swimlane Diagram



BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Interface Design

Interface Design Steps

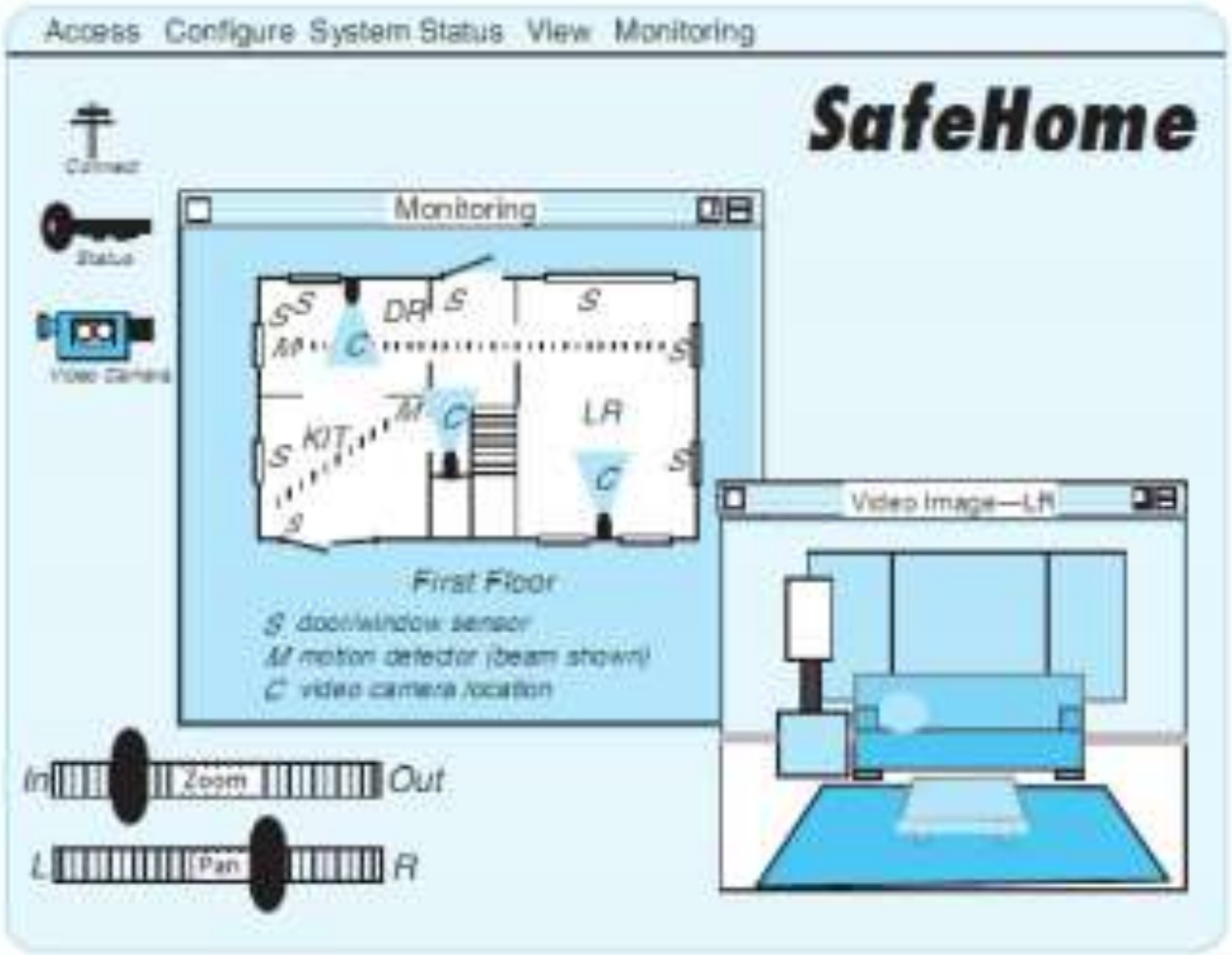
- Using information developed during interface analysis, define interface objects and actions (operations).
- Define events (user actions) that will cause the state of the user interface to change. Model this behavior.
- Depict each interface state as it will actually look to the end-user.
- Indicate how the user interprets the state of the system from information provided through the interface.

BINUS UNIVERSITY ONLINE LEARNING

People
Innovation
Excellence

Interface Design

Preliminary screen layout



WEB AND MOBILE INTERFACE DESIGN

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

WebApp and Mobile Interface Design

Interface Design Principles - I

- **Anticipation**—A WebApp should be designed so that it anticipates the use's next move.
- **Communication**—The interface should communicate the status of any activity initiated by the user
- **Consistency**—The use of navigation controls, menus, icons, and aesthetics (e.g., color, shape, layout)
- **Controlled autonomy**—The interface should facilitate user movement throughout the WebApp, but it should do so in a manner that enforces navigation conventions that have been established for the application.
- **Efficiency**—The design of the WebApp and its interface should optimize the user's work efficiency, not the efficiency of the Web engineer who designs and builds it or the client-server environment that executes it.

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

WebApp and Mobile Interface Design

Interface Design Principles - II

- **Focus**—The WebApp interface (and the content it presents) should stay focused on the user task(s) at hand.
- **Fitt's Law**—"The time to acquire a target is a function of the distance to and size of the target."
- **Human interface objects**—A vast library of reusable human interface objects has been developed for WebApps.
- **Latency reduction**—The WebApp should use multi-tasking in a way that lets the user proceed with work as if the operation has been completed.
- **Learnability**— A WebApp interface should be designed to minimize learning time, and once learned, to minimize relearning required when the WebApp is revisited.

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

WebApp and Mobile Interface Design

Interface Design Principles-III

- **Maintain work product integrity**—A work product (e.g., a form completed by the user, a user specified list) must be automatically saved so that it will not be lost if an error occurs.
- **Readability**—All information presented through the interface should be readable by young and old.
- **Track state**—When appropriate, the state of the user interaction should be tracked and stored so that a user can logoff and return later to pick up where she left off.
- **Visible navigation**—A well-designed WebApp interface provides "the illusion that users are in the same place, with the work brought to them."

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

WebApp and Mobile Interface Design

Interface Design Workflow-I

- Review information contained in the analysis model and refine as required.
- Develop a rough sketch of the WebApp interface layout.
- Map user objectives into specific interface actions.
- Define a set of user tasks that are associated with each action.
- Storyboard screen images for each interface action.
- Refine interface layout and storyboards using input from aesthetic design.

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

WebApp and Mobile Interface Design

Interface Design Workflow-II

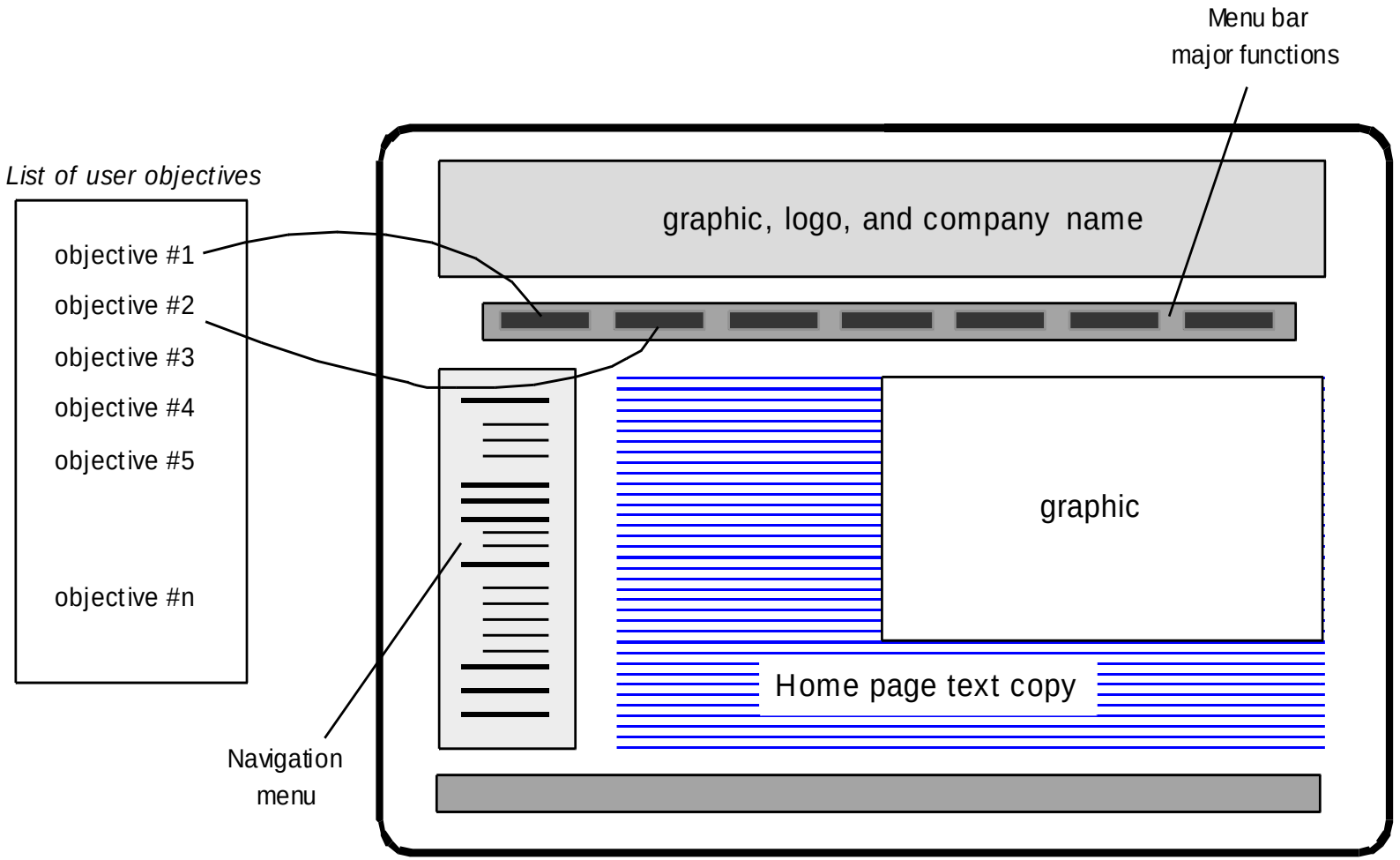
- Identify user interface objects that are required to implement the interface.
- Develop a procedural representation of the user's interaction with the interface.
- Develop a behavioral representation of the interface.
- Describe the interface layout for each state.
- Refine and review the interface design model.

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

WebApp and Mobile Interface Design

The interface design evaluation cycle



DESIGN PATTERNS

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Design Patterns

Effective Patterns

- Coplien [Cop05] characterizes an effective design pattern in the following way:
  - *It solves a problem*: Patterns capture solutions, not just abstract principles or strategies.
  - *It is a proven concept*: Patterns capture solutions with a track record, not theories or speculation.
  - *The solution isn't obvious*: Many problem-solving techniques (such as software design paradigms or methods) try to derive solutions from first principles. The best patterns *generate* a solution to a problem indirectly--a necessary approach for the most difficult problems of design.
  - *It describes a relationship*: Patterns don't just describe modules, but describe deeper system structures and mechanisms.
  - *The pattern has a significant human component (minimize human intervention).* All software serves human comfort or quality of life; the best patterns explicitly appeal to aesthetics and utility.

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Design Patterns

Generative Patterns

- *Generative patterns* describe an important and repeatable aspect of a system and then provide us with a way to build that aspect within a system of forces that are unique to a given context.

- A collection of generative design patterns could be used to “generate” an application or computer-based system whose architecture enables it to adapt to change.

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Design Patterns

Describing a Pattern

- ***Pattern name***—describes the essence of the pattern in a short but expressive name
- ***Problem***—describes the problem that the pattern addresses
- ***Motivation***—provides an example of the problem
- ***Context***—describes the environment in which the problem resides including application domain
- ***Forces***—lists the system of forces that affect the manner in which the problem must be solved; includes a discussion of limitation and constraints that must be considered
- ***Solution***—provides a detailed description of the solution proposed for the problem
- ***Intent***—describes the pattern and what it does
- ***Collaborations***—describes how other patterns contribute to the solution
- ***Consequences***—describes the potential trade-offs that must be considered when the pattern is implemented and the consequences of using the pattern
- ***Implementation***—identifies special issues that should be considered when implementing the pattern
- ***Known uses***—provides examples of actual uses of the design pattern in real applications
- ***Related patterns***—cross-references related design patterns

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Design Patterns

Pattern-Based Design

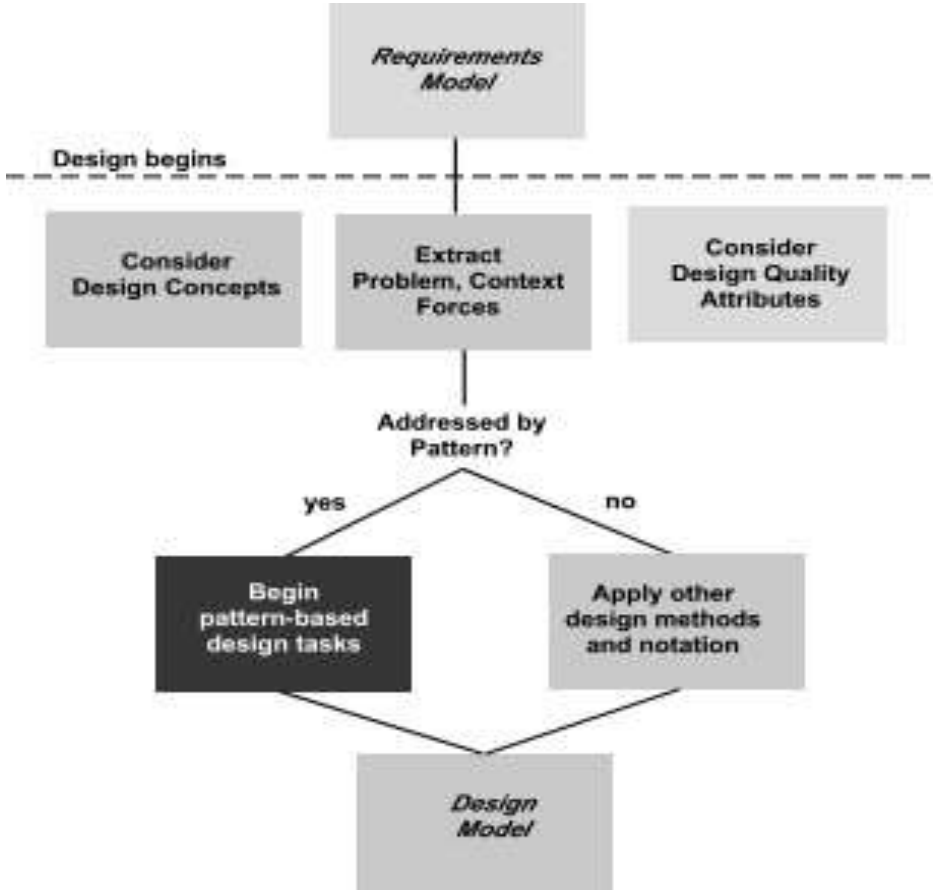
- A software designer begins with a requirements model (either explicit or implied) that presents an abstract representation of the system.
- The requirements model describes the problem set, establishes the context, and identifies the system of forces that hold sway.
- Then …

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Design Patterns

Pattern-based design in context



BINUS UNIVERSITY ONLINE LEARNING

People
Innovation
Excellence

Design Patterns

Pattern Organizing Table

	Database	Application	Implementation	Infrastructure
Data/Content				
Problem statement ...	PatternName(b)		PatternName(e)	
Problem statement ...		PatternName(g)		PatternName(d)
Problem statement ...	PatternName(d)			PatternName(g)
Architecture				
Problem statement ...		PatternName(b)		
Problem statement ...		PatternName(d)		PatternName(g)
Problem statement ...				
Component-level				
Problem statement ...		PatternName(e)	PatternName(d)	
Problem statement ...				PatternName(e)
Problem statement ...		PatternName(d)	PatternName(g)	
User Interface				
Problem statement ...		PatternName(d)	PatternName(g)	
Problem statement ...		PatternName(g)	PatternName(e)	
Problem statement ...		PatternName(e)	PatternName(d)	

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Architectural Patterns

- Example: every house (and every architectural style for houses) employs a **Kitchen** pattern.
- The **Kitchen** pattern and patterns it collaborates with address problems associated with the storage and preparation of food, the tools required to accomplish these tasks, and rules for placement of these tools relative to workflow in the room.
- In addition, the pattern might address problems associated with counter tops, lighting, wall switches, a central island, flooring, and so on.
- Obviously, there is more than a single design for a kitchen, often dictated by the context and system of forces. But every design can be conceived within the context of the ‘solution’ suggested by the **Kitchen** pattern.

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

WebApps Design

Design & WebApp Quality

- Security
  - Rebuff external attacks
  - Exclude unauthorized access
  - Ensure the privacy of users/customers
- Availability
  - the measure of the percentage of time that a WebApp is available for use
- Scalability
  - Can the WebApp and the systems with which it is interfaced handle significant variation in user or transaction volume
- Time to Market

WEBAPPS DESIGN

WebApps Design

Quality Dimensions for End-Users

- *Time*
  - How much has a Web site changed since the last upgrade?
  - How do you highlight the parts that have changed?
- *Structural*
  - How well do all of the parts of the Web site hold together.
  - Are all links inside and outside the Web site working?
  - Do all of the images work?
  - Are there parts of the Web site that are not connected?
- *Content*
  - Does the content of critical pages match what is supposed to be there?
  - Do key phrases exist continually in highly-changeable pages?
  - Do critical pages maintain quality content from version to version?
  - What about dynamically generated HTML pages?

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

WebApps Design

Quality Dimensions for End-Users

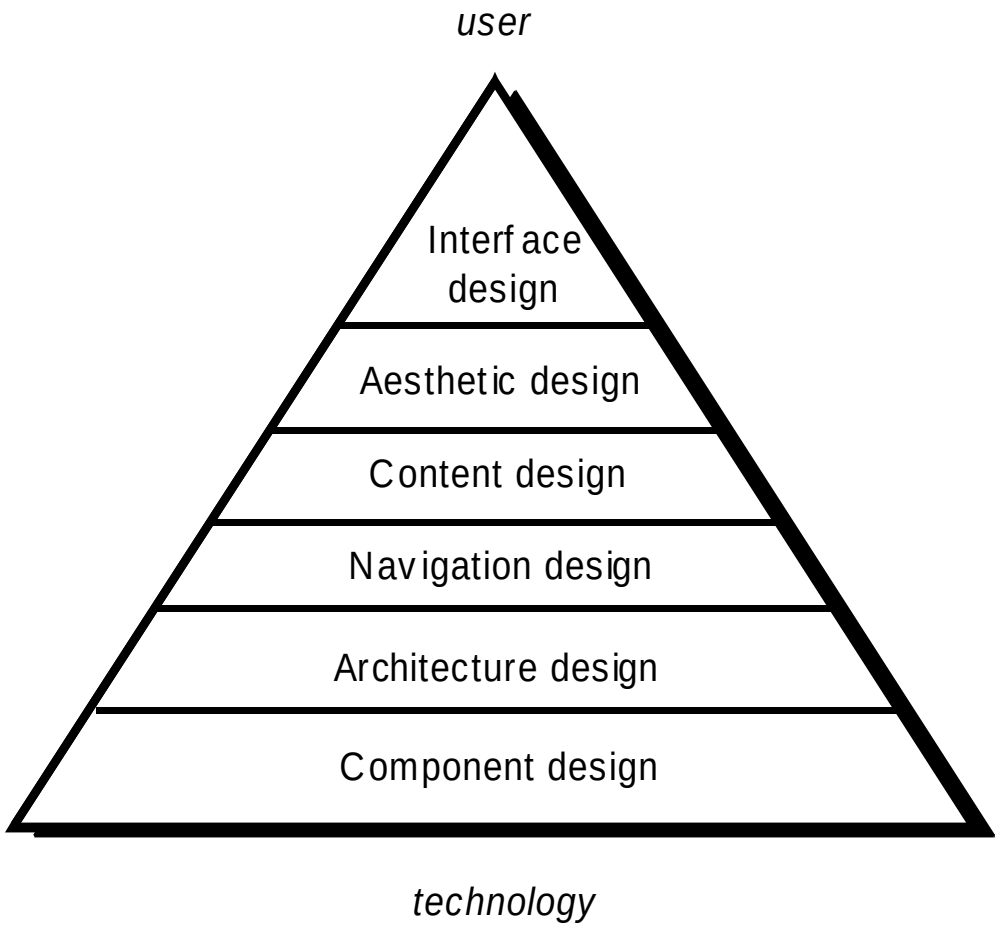
- *Accuracy and Consistency*
  - Are today's copies of the pages downloaded the same as yesterday's? Close enough?
  - Is the data presented accurate enough? How do you know?
- *Response Time and Latency*
  - Does the Web site server respond to a browser request within certain parameters?
  - In an E-commerce context, how is the end to end response time after a SUBMIT?
  - Are there parts of a site that are so slow the user declines to continue working on it?
- *Performance*
  - Is the Browser-Web-Web site-Web-Browser connection quick enough?
  - How does the performance vary by time of day, by load and usage?
  - Is performance adequate for E-commerce applications?

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

WebApps Design

Design Pyramid for WebApps

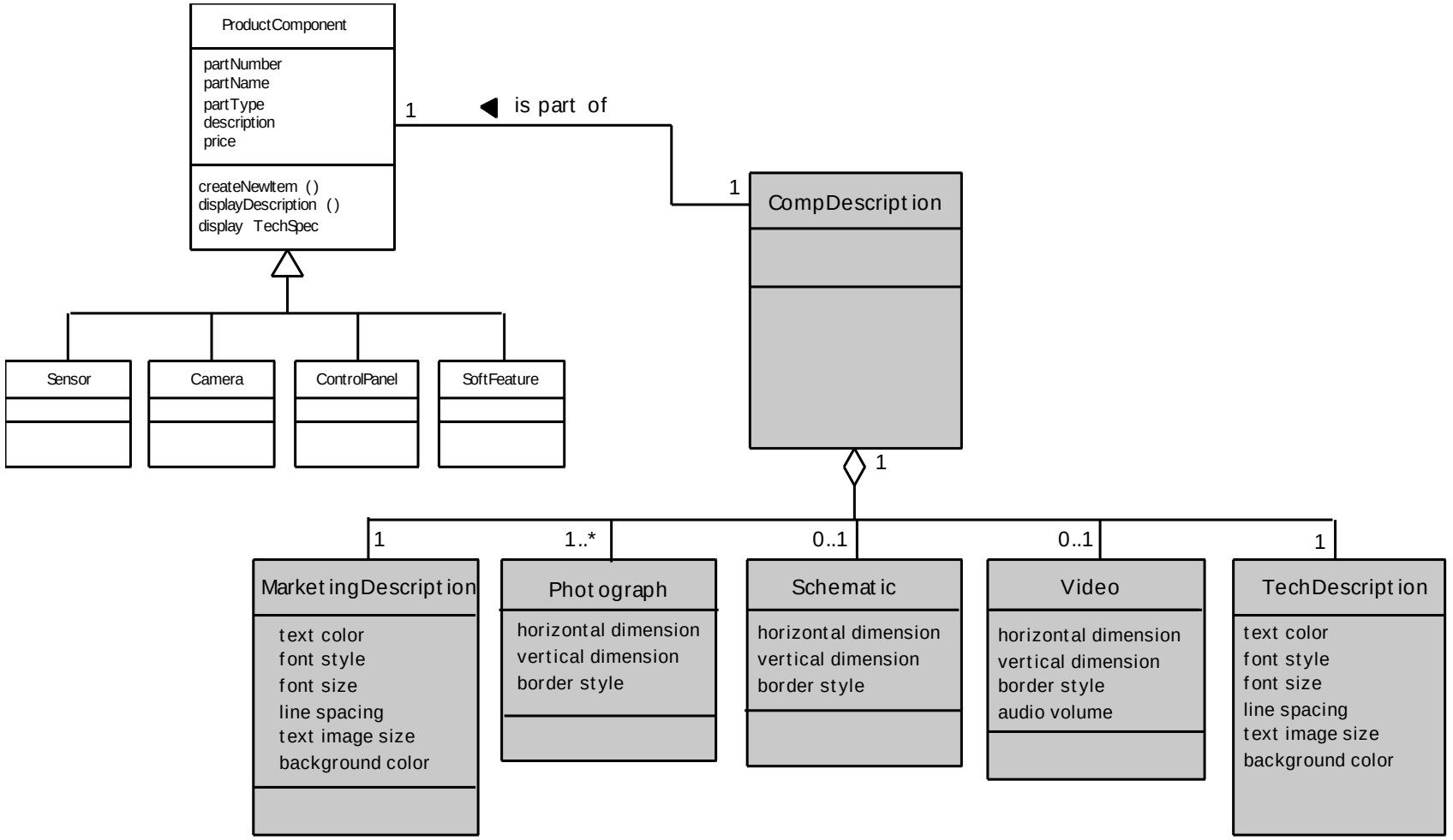


BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

WebApps Design

Design representation of content objects



MOBILE APPS DESIGN

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

MobileApp Design

Development Considerations

- Like all computing devices, mobile platforms are differentiated by the software they deliver – a combination of operating system (e.g., Android or iOS) and a small subset of the hundreds of thousands of MobileApps that provide a very wide range of functionality.
- New tools allow individuals with little formal training to create and sell apps alongside other apps developed by large teams of software developers.

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

MobileApp Design

Technical Considerations

- Multiple hardware and software platforms
- Many development frameworks and programming languages
- Many app store with different rules and tools
- Very short development cycles
- UI limitations and complexities of interaction with sensors and cameras
- Effective use of context
- Power management
- Security and privacy models and policies
- Computational and storage limitations
- Application that depend on external services
- Testing complexity

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

MobileApp Design

Best Practices

- Identify your audience
- Design for context of use
- There is a fine line between simlicity and laziness
- Use the platform as an advantage
- Make scrollbars and selection highlighting more salient
- Increase discoverability of advanced functionality
- Use clear and consistent labels
- Clever icons should never be developed at the expense of user under standing
- Support user expectations for personalization
- Long scrolling forms trump multiple screens on mobile device

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

References

- Pressman, R.S. (2015). *Software Engineering : A Practioner's Approach. 8th ed*. McGraw-Hill Companies.Inc, Americas, New York. ISBN : 978 1 259 253157.
- Guide to the Software Engineering Body of Knowledge, SWEBOK, https://www.computer.org/education/bodies-of-knowledge/software-engineering
- UML Specification, https://www.omg.org/spec/UML/About-UML/
- Introduction to Software Architecture, http://www.youtube.com/watch?v=x30DcBfCJRI
- Component-based game engine, http://www.youtube.com/watch?v=_K4Mc3t9Rtc
- software design pattern, http://www.youtube.com/watch?v=ehGl_V61WJw

BINUS UNIVERSITY ONLINE LEARNING

People
Innovation
Excellence

Q & A

BINUS
UNIVERSITY
ONLINE
LEARNING

People
Innovation
Excellence

Thank You

BINUS UNIVERSITY ONLINE LEARNING

People
Innovation
Excellence

Software Engineering

Topic 5

Software Quality Assurance

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Acknowledgement

These slides have been adapted from Pressman, R.S. (2015). *Software Engineering : A Practioner's Approach. 8th ed*. McGraw-Hill Companies.Inc, Americas, New York. ISBN : 978 1 259 253157. Chapter 19 and 20

BINUS UNIVERSITY ONLINE LEARNING

People
Innovation
Excellence

Learning Objectives

LO 3 : Demonstrate the quality assurances and the potential showcase business project

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Contents

- Software Quality
- The Software Quality Dilemma
- Achieving Software Quality
- Review Techniques
- Defect Amplification
- Review Metrics
- Informal Reviews
- Formal Technical Reviews
- Comment on Quality
- Statistical SQA
- Software Reliability

SOFTWARE QUALITY

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Software Quality

In 2005, *ComputerWorld* [Hil05] lamented that

> **"bad software plagues nearly every organization that uses computers, causing lost work hours during computer downtime, lost or corrupted data, missed sales opportunities, high IT support and maintenance costs, and low customer satisfaction.**

A year later, *InfoWorld* [Fos06] wrote about the

> **"the sorry state of software quality" reporting that the quality problem had not gotten any better.**

Today, software quality remains an issue, but who is to blame?

> **Customers blame developers, arguing that sloppy practices lead to low-quality software.**
>
> **Developers blame customers (and other stakeholders), arguing that irrational delivery dates and a continuing stream of changes force them to deliver software before it has been fully validated.**

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Software Quality

Quality

- The *American Heritage Dictionary* defines *quality* as
  - "a characteristic or attribute of something."
- For software, two kinds of quality may be encountered:
  - Quality of design encompasses requirements, specifications, and the design of the system.
  - Quality of conformance is an issue focused primarily on implementation.
  - User satisfaction = compliant product + good quality + delivery within budget and schedule

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Software Quality

Quality - A Pragmatic View

- The *transcendental view* argues (like Persig) that quality is something that you immediately recognize, but cannot explicitly define.
- The *user view* sees quality in terms of an end-user's specific goals. If a product meets those goals, it exhibits quality.
- The *manufacturer's view* defines quality in terms of the original specification of the product. If the product conforms to the spec, it exhibits quality.
- The *product view* suggests that quality can be tied to inherent characteristics (e.g., functions and features) of a product.
- Finally, the *value-based view* measures quality based on how much a customer is willing to pay for a product. In reality, quality encompasses all of these views and more.

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Software Quality

Software Quality

- Software quality can be defined as:
  - *An effective software process applied in a manner that creates a useful product that provides measurable value for those who produce it and those who use it.*
- This definition has been adapted from [Bes04] and replaces a more manufacturing-oriented view presented in earlier editions of this book.

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Software Quality

Effective Software Process

- An *effective software process* establishes the infrastructure that supports any effort at building a high quality software product.
- The management aspects of process create the checks and balances that help avoid project chaos—a key contributor to poor quality.
- Software engineering practices allow the developer to analyze the problem and design a solid solution—both critical to building high quality software.
- Finally, umbrella activities such as change management and technical reviews have as much to do with quality as any other part of software engineering practice.

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Software Quality

Useful Product

- A *useful product* delivers the content, functions, and features that the end-user desires
- But as important, it delivers these assets in a reliable, error free way.
- A useful product always satisfies those requirements that have been explicitly stated by stakeholders.
- In addition, it satisfies a set of implicit requirements (e.g., ease of use) that are expected of all high quality software.

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Software Quality

Adding Value

- By *adding value for both the producer and user* of a software product, high quality software provides benefits for the software organization and the end-user community.
- The software organization gains added value because high quality software requires less maintenance effort, fewer bug fixes, and reduced customer support.
- The user community gains added value because the application provides a useful capability in a way that expedites some business process.
- The end result is:
  - (1) greater software product revenue,
  - (2) better profitability when an application supports a business process, and/or
  - (3) improved availability of information that is crucial for the business.

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Software Quality

Quality Dimensions

- David Garvin [Gar87]:
  - **Performance Quality**
  - **Feature quality**
  - **Reliability**
  - **Conformance**
  - **Durability**
  - **Serviceability**
  - **Aesthetics**
  - **Perception**

SOFTWARE QUALITY DILEMMA

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

The Software Quality Dilemma

- If you produce a software system that has terrible quality, you lose because no one will want to buy it.
- If on the other hand you spend infinite time, extremely large effort, and huge sums of money to build the absolutely perfect piece of software, then it's going to take so long to complete and it will be so expensive to produce that you'll be out of business anyway.
- Either you missed the market window, or you simply exhausted all your resources.
- So people in industry try to get to that magical middle ground where the product is good enough not to be rejected right away, such as during evaluation, but also not the object of so much perfectionism and so much work that it would take too long or cost too much to complete. [Ven03]

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

The Software Quality Dilemma

"Good Enough" Software

- Good enough software delivers high quality functions and features that end-users desire, but at the same time it delivers other more obscure or specialized functions and features that contain known bugs.
- Arguments *against* "good enough."
  - It is true that "good enough" may work in some application domains and for a few major software companies. After all, if a company has a large marketing budget and can convince enough people to buy version 1.0, it has succeeded in locking them in.
  - If you work for a small company be wary of this philosophy. If you deliver a "good enough" (buggy) product, you risk permanent damage to your company's reputation.
  - You may never get a chance to deliver version 2.0 because bad buzz may cause your sales to plummet and your company to fold.
  - If you work in certain application domains (e.g., real time embedded software, application software that is integrated with hardware can be negligent and open your company to expensive litigation.

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

The Software Quality Dilemma

Cost of Quality

- *Prevention costs* include
  - quality planning
  - formal technical reviews
  - test equipment
  - Training
- *Internal failure costs* include
  - rework
  - repair
  - failure mode analysis
- *External failure costs* are
  - complaint resolution
  - product return and replacement
  - help line support
  - warranty work

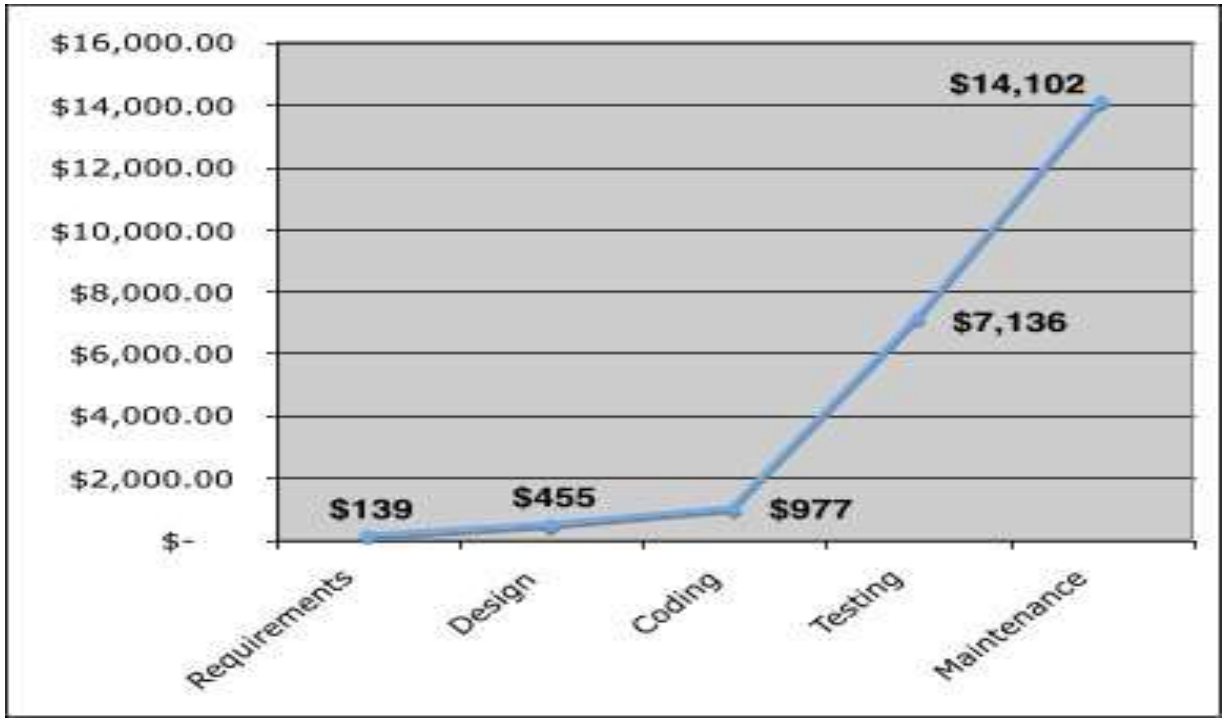
BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

The Software Quality Dilemma

Cost

- The relative costs to find and repair an error or defect increase dramatically as we go from prevention to detection to internal failure to external failure costs.



BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

The Software Quality Dilemma

Quality and Risk

- *"People bet their jobs, their comforts, their safety, their entertainment, their decisions, and their very lives on computer software. It better be right."* SEPA, Chapter 1
- Example:
  - *Throughout the month of November, 2000 at a hospital in Panama, 28 patients received massive overdoses of gamma rays during treatment for a variety of cancers. In the months that followed, five of these patients died from radiation poisoning and 15 others developed serious complications. What caused this tragedy? A software package, developed by a U.S. company, was modified by hospital technicians to compute modified doses of radiation for each patient.*

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

The Software Quality Dilemma

Negligence and Liability

- The story is all too common. A governmental or corporate entity hires a major software developer or consulting company to analyze requirements and then design and construct a software-based “system” to support some major activity.
  - The system might support a major corporate function (e.g., pension management) or some governmental function (e.g., healthcare administration or homeland security).
- Work begins with the best of intentions on both sides, but by the time the system is delivered, things have gone bad.
- The system is late, fails to deliver desired features and functions, is error-prone, and does not meet with customer approval.
- Litigation ensues.

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

The Software Quality Dilemma

Quality and Security

- Gary McGraw comments [Wil05]:
- “Software security relates entirely and completely to quality. You must think about security, reliability, availability, dependability—at the beginning, in the design, architecture, test, and coding phases, all through the software life cycle [process]. Even people aware of the software security problem have focused on late life-cycle stuff. The earlier you find the software problem, the better. And there are two kinds of software problems. One is bugs, which are implementation problems. The other is software flaws—architectural problems in the design. People pay too much attention to bugs and not enough on flaws.”

ACHIEVING SOFTWARE QUALITY

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Achieving Software Quality

- Critical success factors:
  - **Software Engineering Methods**
  - **Project Management Techniques**
  - **Quality Control**
  - **Quality Assurance**

REVIEW TECHNIQUE

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Review Techniques

What Are Reviews?

- a meeting conducted by technical people for technical people
- a technical assessment of a work product created during the software engineering process
- a software quality assurance mechanism
- a training ground
- **Review are not:**
  - A project summary or progress assessment
  - A meeting intended solely to impart information
  - A mechanism for political or personal reprisal!

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

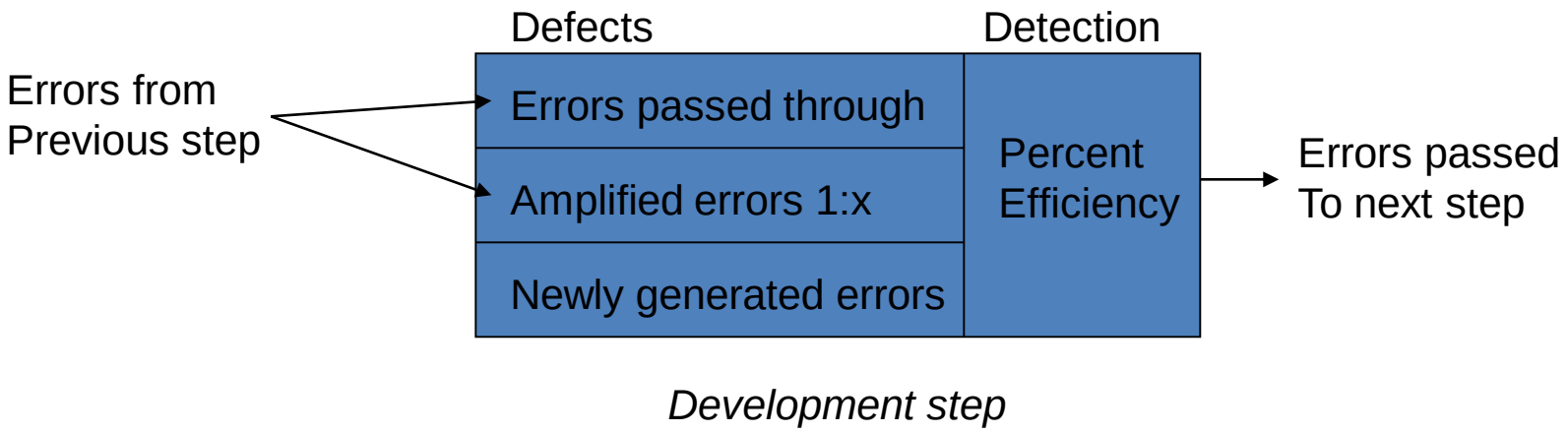
Review Techniques

What Do We Look For?

- Errors and defects
  - *Error*—a quality problem found *before* the software is released to end users
  - *Defect*—a quality problem found only *after* the software has been released to end-users
- We make this distinction because errors and defects have very different economic, business, psychological, and human impact
- However, the temporal distinction made between errors and defects in this book is *not* mainstream thinking

Defect Amplification

- A *defect amplification model* [IBM81] can be used to illustrate the generation and detection of errors during the design and code generation actions of a software process.



BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Review Metrics

- The total review effort and the total number of errors discovered are defined as:
  - $E_{review} = E_p + E_a + E_r$
  - $Err_{tot} = Err_{minor} + Err_{major}$
- *Defect density* represents the errors found per unit of work product reviewed.
  - Defect density = Err_{tot} / WPS
- where …

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Review Metrics

Metrics

- *Preparation effort, E_p*—the effort (in person-hours) required to review a work product prior to the actual review meeting
- *Assessment effort, E_a*— the effort (in person-hours) that is expending during the actual review
- *Rework effort, E_r*— the effort (in person-hours) that is dedicated to the correction of those errors uncovered during the review
- *Work product size, WPS*—a measure of the size of the work product that has been reviewed (e.g., the number of UML models, or the number of document pages, or the number of lines of code)
- *Minor errors found, Err_{minor}*—the number of errors found that can be categorized as minor (requiring less than some pre-specified effort to correct)
- *Major errors found, Err_{major}*— the number of errors found that can be categorized as major (requiring more than some pre-specified effort to correct)

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Review Metrics

An Example—I

- If past history indicates that
  - the average defect density for a requirements model is 0.6 errors per page, and a new requirement model is 32 pages long,
  - a rough estimate suggests that your software team will find about 19 or 20 errors during the review of the document.
  - If you find only 6 errors, you've done an extremely good job in developing the requirements model *or* your review approach was not thorough enough.

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Review Metrics

An Example—II

- The effort required to correct a minor model error (immediately after the review) was found to require 4 person-hours.
- The effort required for a major requirement error was found to be 18 person-hours.
- Examining the review data collected, you find that minor errors occur about 6 times more frequently than major errors. Therefore, you can estimate that the average effort to find and correct a requirements error during review is about 6 person-hours.
- Requirements related errors uncovered during testing require an average of 45 person-hours to find and correct. Using the averages noted, we get:
- Effort saved per error = $E_{testing} - E_{reviews}$
- $45 - 6 = 30$ person-hours/error
- Since 22 errors were found during the review of the requirements model, a saving of about 660 person-hours of testing effort would be achieved. And that's just for requirements-related errors.

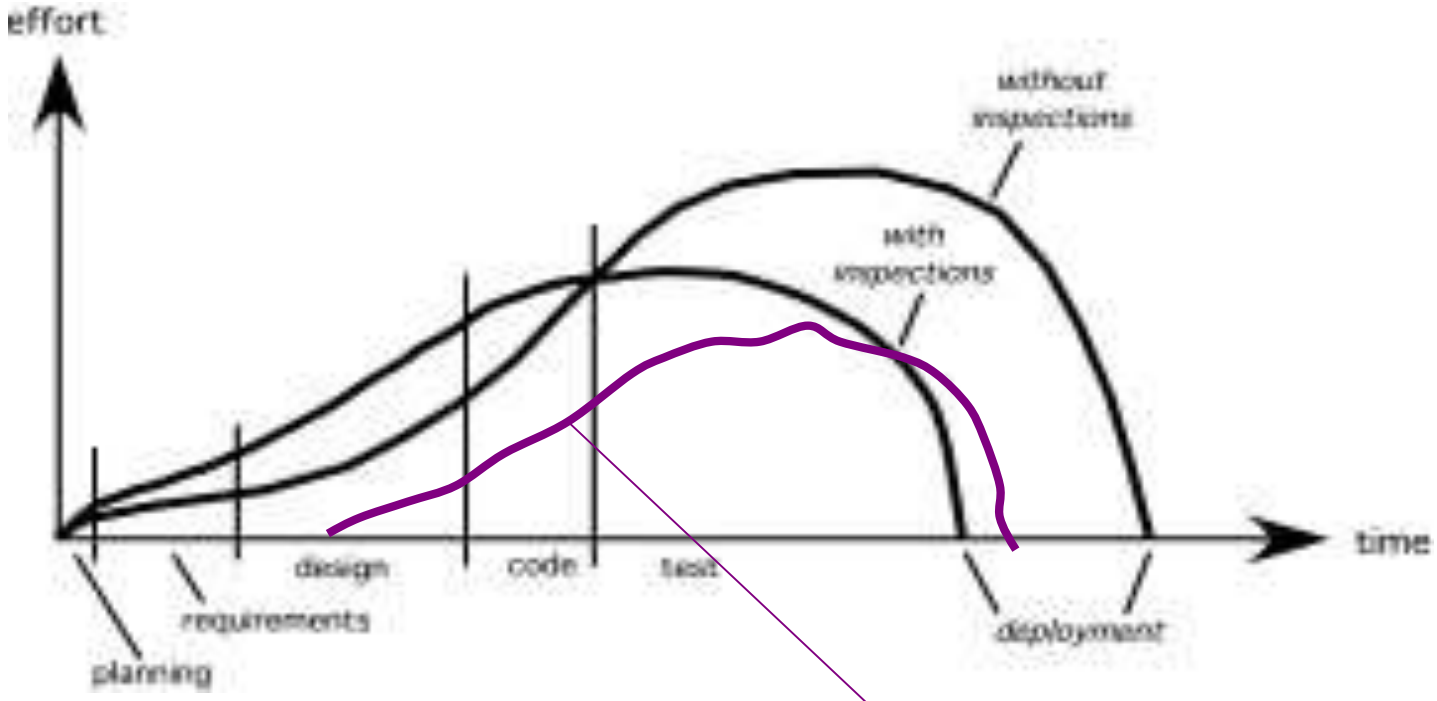
BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Review Metrics

Overall

- Effort expended with and without reviews



BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Informal Reviews

- Informal reviews include:
  - a simple desk check of a software engineering work product with a colleague
  - a casual meeting (involving more than 2 people) for the purpose of reviewing a work product, or
  - the review-oriented aspects of pair programming
- *pair programming* encourages continuous review as a work product (design or code) is created.
  - The benefit is immediate discovery of errors and better work product quality as a consequence.

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Formal Technical Reviews

- The objectives of an FTR are:
  - to uncover errors in function, logic, or implementation for any representation of the software
  - to verify that the software under review meets its requirements
  - to ensure that the software has been represented according to predefined standards
  - to achieve software that is developed in a uniform manner
  - to make projects more manageable
- The FTR is actually a class of reviews that includes *walkthroughs* and *inspections.*

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Formal Technical Reviews

Conducting the Review

- *Review the product, not the producer.*
- *Set an agenda and maintain it.*
- *Limit debate and rebuttal.*
- *Enunciate problem areas, but don't attempt to solve every problem noted.*
- *Take written notes.*
- *Limit the number of participants and insist upon advance preparation.*
- *Develop a checklist for each product that is likely to be reviewed.*
- *Allocate resources and schedule time for FTRs.*
- *Conduct meaningful training for all reviewers.*
- *Review your early reviews.*

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

References

- Pressman, R.S. (2015). ***Software Engineering : A Practioner's Approach. 8th ed***. McGraw-Hill Companies.Inc, Americas, New York. ISBN : 978 1 259 253157.
- **Introduction to software quality assurance,** **http://www.youtube.com/watch?v=5_cTi5xBlYg**
- Lean Six Sigma and IEEE standards for better software engineering**,** http://www.youtube.com/watch?v=oCkPD5YvWqw

BINUS UNIVERSITY ONLINE LEARNING

People
Innovation
Excellence

Q & A

BINUS
UNIVERSITY
ONLINE
LEARNING

People
Innovation
Excellence

Thank You

BINUS UNIVERSITY ONLINE LEARNING

People
Innovation
Excellence

Software Engineering

Topic 6

Application Testing and Security Engineering

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Acknowledgement

These slides have been adapted from Pressman, R.S. (2015). *Software Engineering : A Practioner's Approach. 8th ed*. McGraw-Hill Companies.Inc, Americas, New York. ISBN 978 1 259 253157. Chapter 23, 24, 25, 26, and 27

BINUS UNIVERSITY ONLINE LEARNING

People
Innovation
Excellence

Learning Objectives

LO 3 : Demonstrate the quality assurances and the potential showcase business project

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Contents

- Testing Conventional Applications
- Testing Object Oriented Applications
- Testing Web Applications
- Testing Mobile Applications
- Security Engineering

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Testing Conventional Applications

Software Testing Fundamentals : Testability

- **Operability —it operates cleanly**
- **Observability —the results of each test case are readily observed**
- **Controllability —the degree to which testing can be automated and optimized**
- **Decomposability —testing can be targeted**
- **Simplicity —reduce complex architecture and logic to simplify tests**
- **Stability —few changes are requested during testing**
- **Understandability —of the design**

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Testing Conventional Applications

What is a "Good" Test?

- A good test has a high probability of finding an error
- A good test is not redundant.
- A good test should be "best of breed"
- A good test should be neither too simple nor too complex

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Testing Conventional Applications

Internal and External Views of Testing

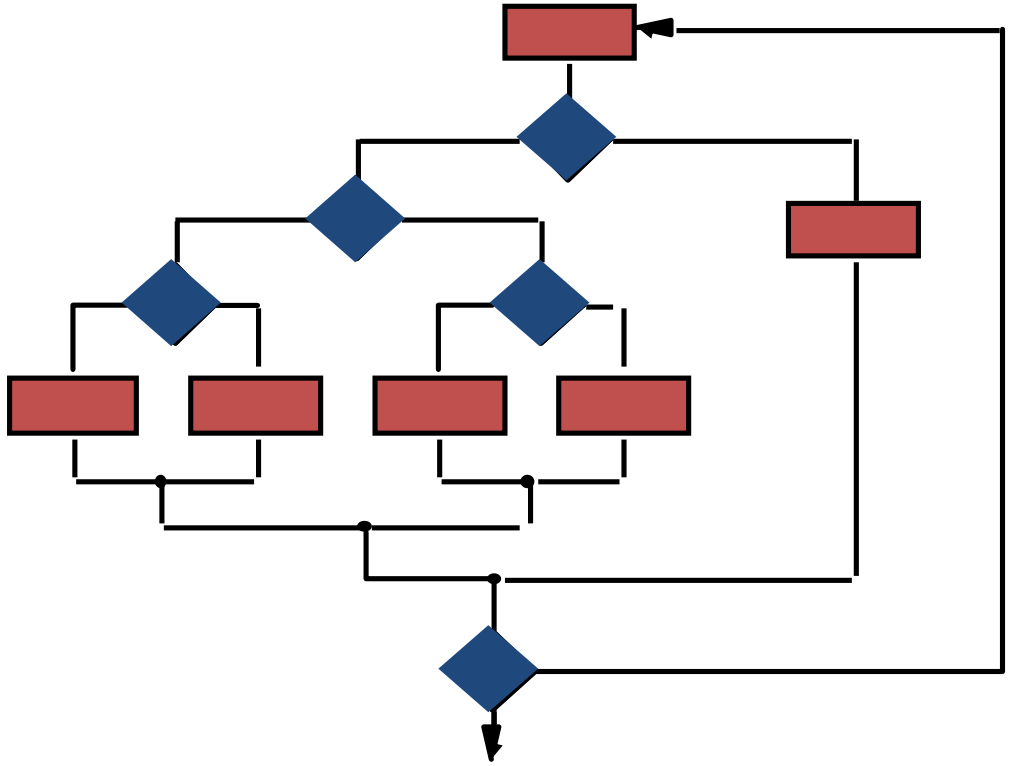
- Any engineered product (and most other things) can be tested in one of two ways:
  - Knowing the specified function that a product has been designed to perform, tests can be conducted that demonstrate each function is fully operational while at the same time searching for errors in each function;
  - Knowing the internal workings of a product, tests can be conducted to ensure that "all gears mesh," that is, internal operations are performed according to specifications and all internal components have been adequately exercised.

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Testing Conventional Applications

Exhaustive Testing



loop < 20 X

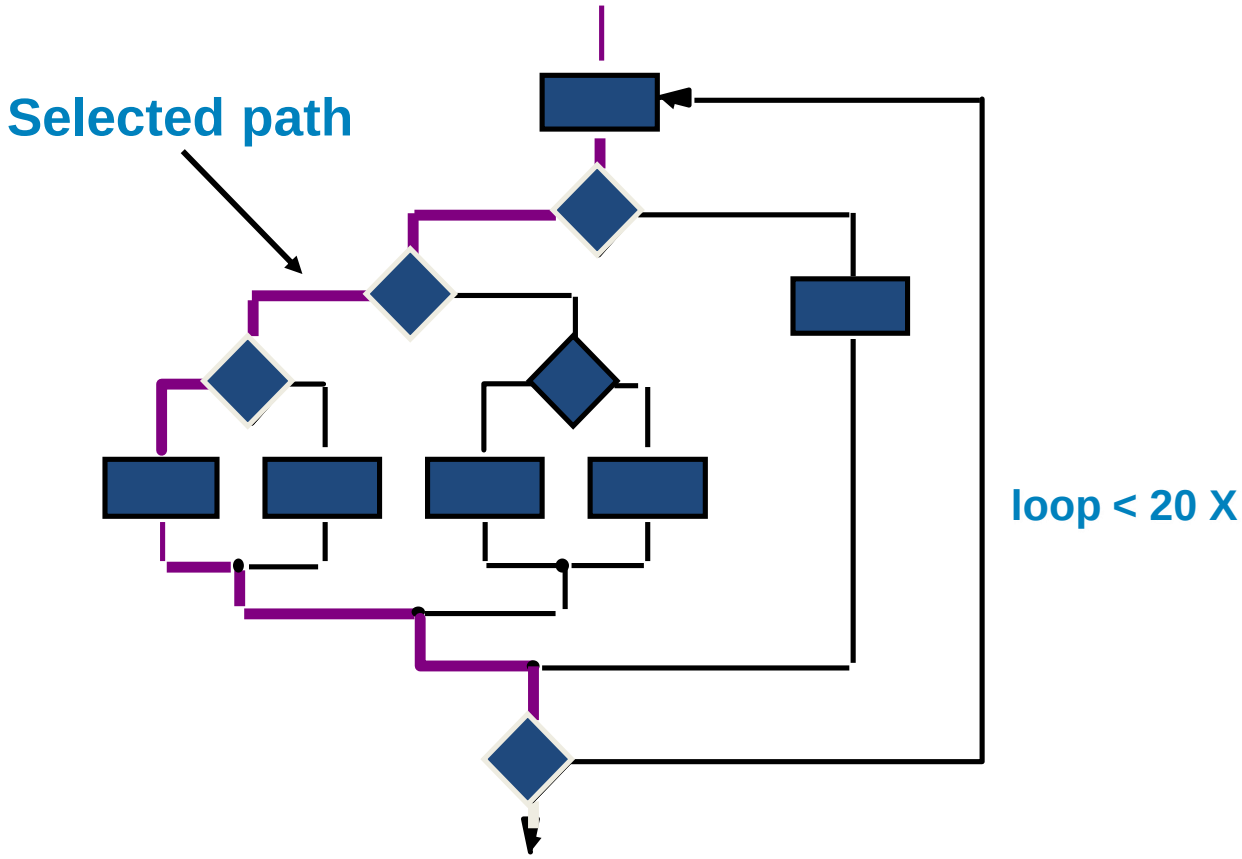
test per millisecond, it would take 3,170 years to

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Testing Conventional Applications

Selective Testing

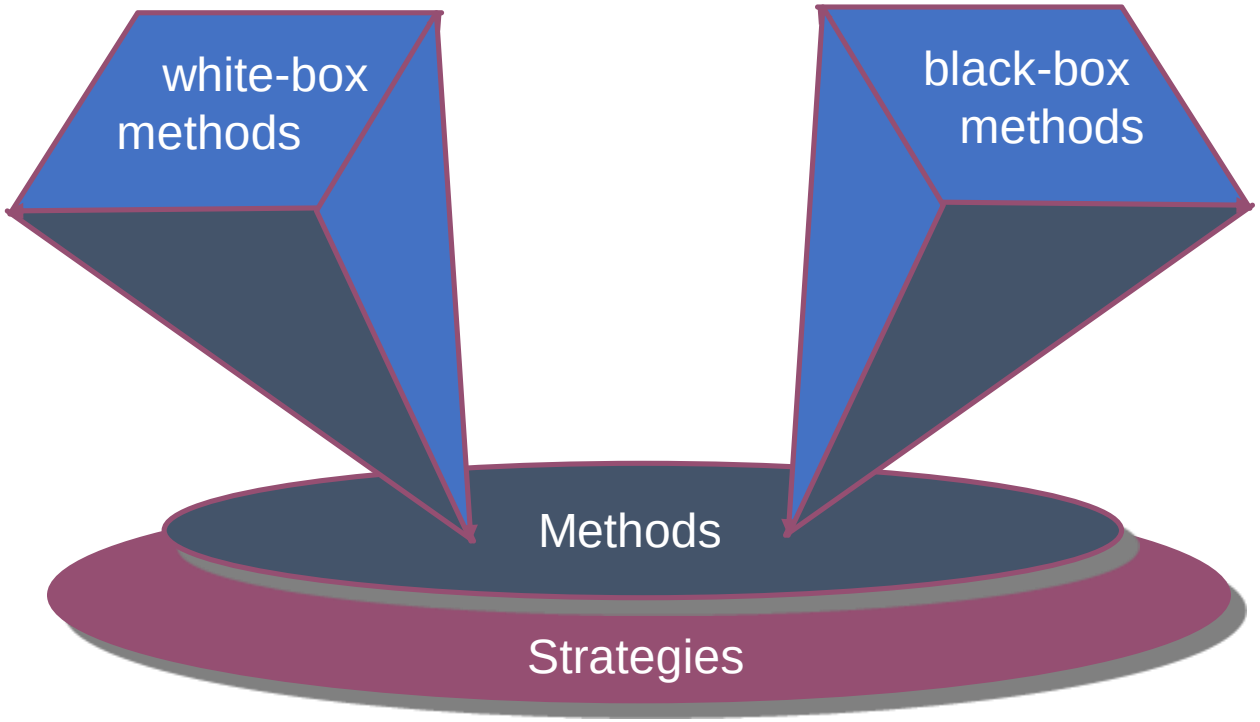


BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Testing Conventional Applications

Software Testing

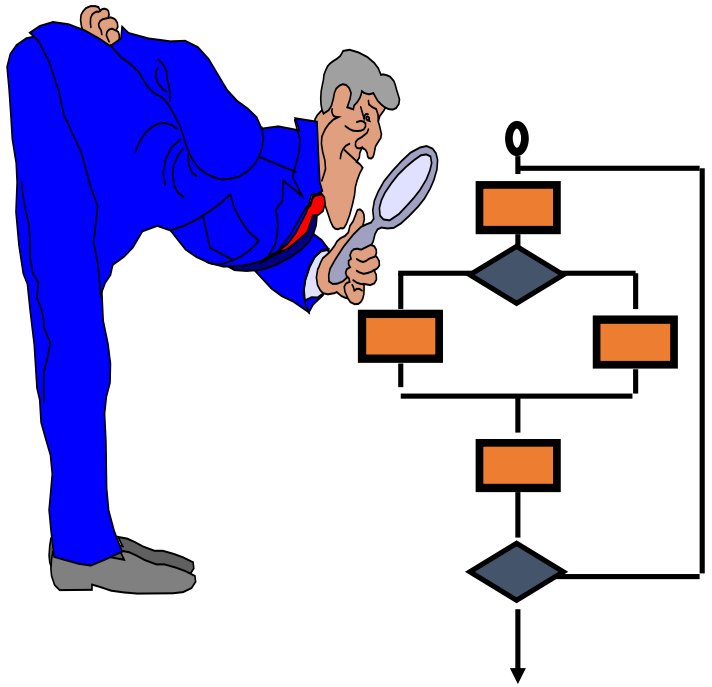


BINUS UNIVERSITY ONLINE LEARNING

People
Innovation
Excellence

Testing Conventional Applications

White-Box Testing



... our goal is to ensure that all statements and conditions have been executed at least once ...

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Testing Conventional Applications

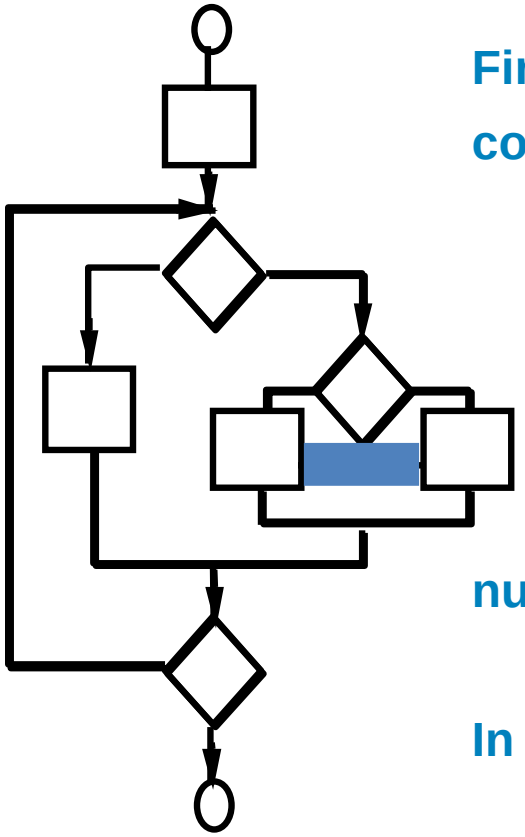
White-Box Testing

- logic errors and incorrect assumptions are inversely proportional to a path's execution probability
- we often believe that a path is not likely to be executed; in fact, reality is often counter intuitive
- typographical errors are random; it's likely that untested paths will contain some

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Testing Conventional Applications



First, we compute the cyclomatic complexity:

or

number of enclosed areas + 1

In this case, V(G) = 4

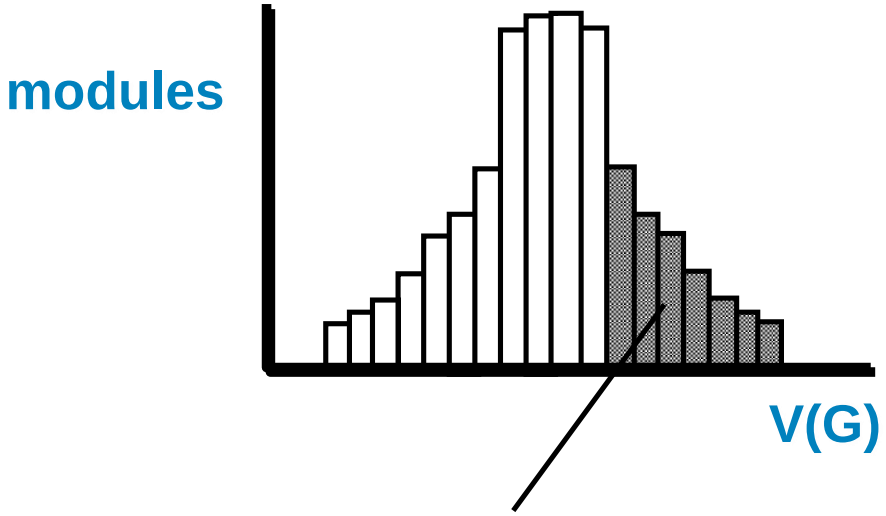
BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Testing Conventional Applications

Cyclomatic Complexity

A number of industry studies have indicated that the higher V(G), the higher the probability or errors.

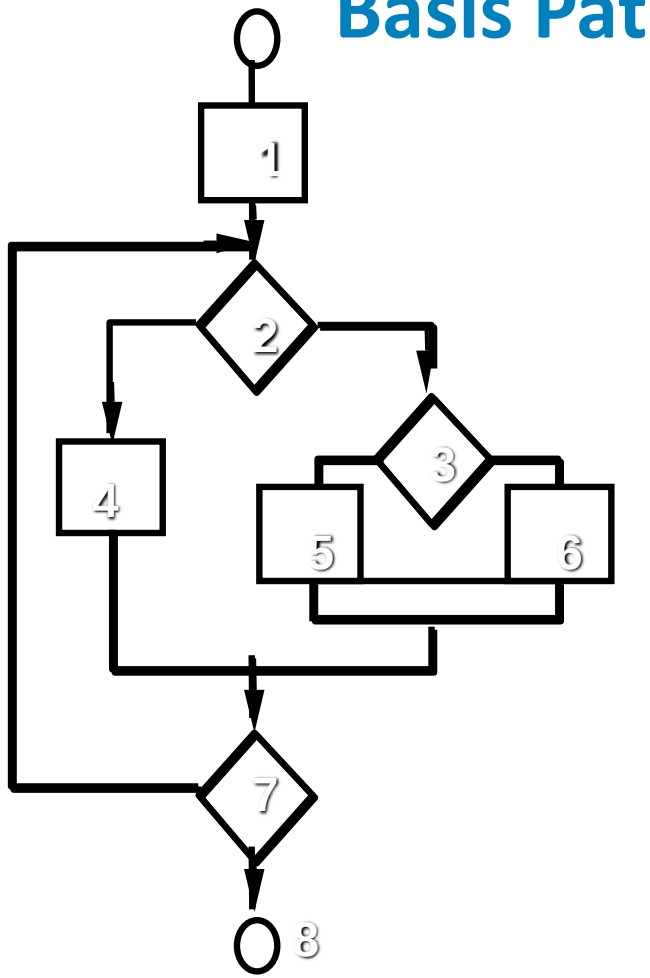


BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Testing Conventional Applications

Basis Path Testing



Next, we derive the independent paths:

Since V(G) = 4, there are four paths

- **Path 1: 1,2,3,6,7,8**
- **Path 2: 1,2,3,5,7,8**
- **Path 3: 1,2,4,7,8**
- **Path 4: 1,2,4,7,2,4,...7,8**

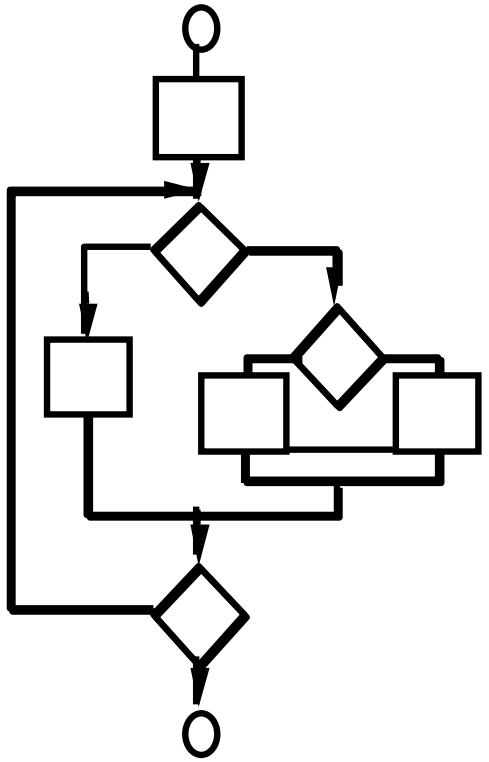
Finally, we derive test cases to exercise these paths.

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Testing Conventional Applications

Basis Path Testing Notes



- you don't need a flow chart, but the picture will help when you trace program paths
- count each simple logical test, compound tests count as 2 or more
- basis path testing should be applied to critical modules

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Testing Conventional Applications

Basis Path Testing - Graph Matrices

- A graph matrix is a square matrix whose size (i.e., number of rows and columns) is equal to the number of nodes on a flow graph
- Each row and column corresponds to an identified node, and matrix entries correspond to connections (an edge) between nodes.
- By adding a *link weight* to each matrix entry, the graph matrix can become a powerful tool for evaluating program control structure during testing

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Testing Conventional Applications

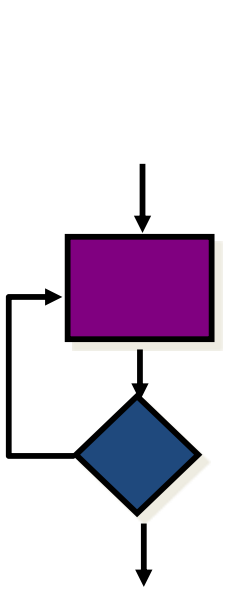
Control Structure Testing

- **Condition testing — a test case design method that exercises the logical conditions contained in a program module**
- **Data flow testing — selects test paths of a program according to the locations of definitions and uses of variables in the program**

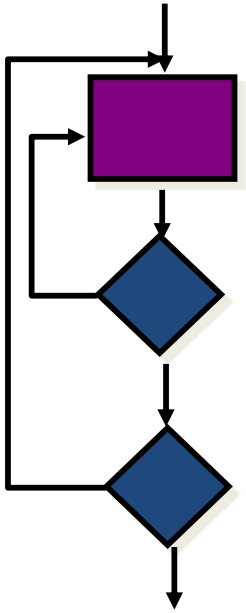
BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

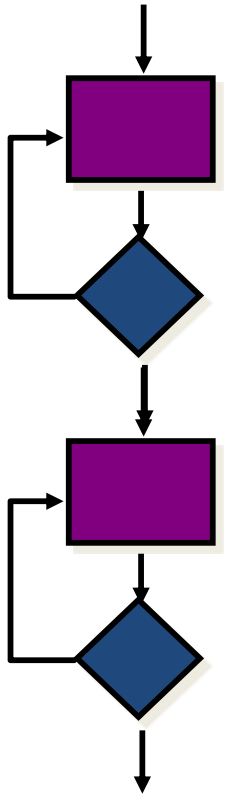
Testing Conventional Applications



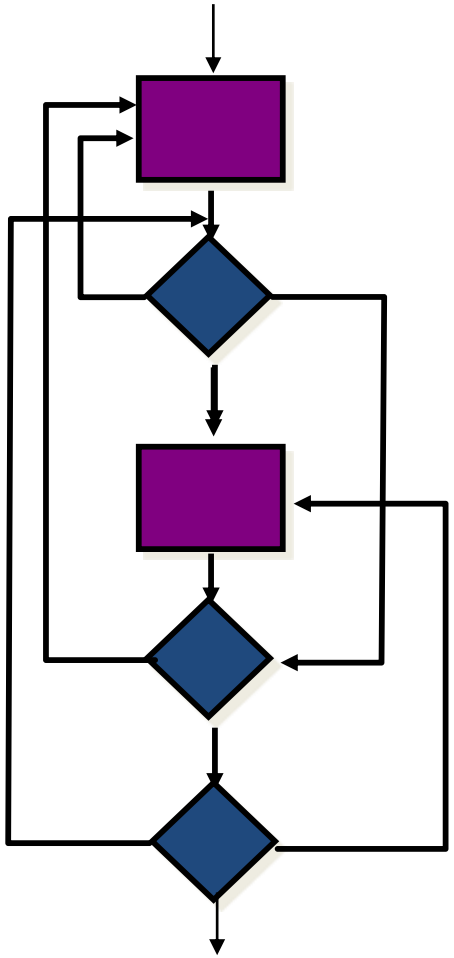
Simple loop



Nested Loops



Concatenated Loops



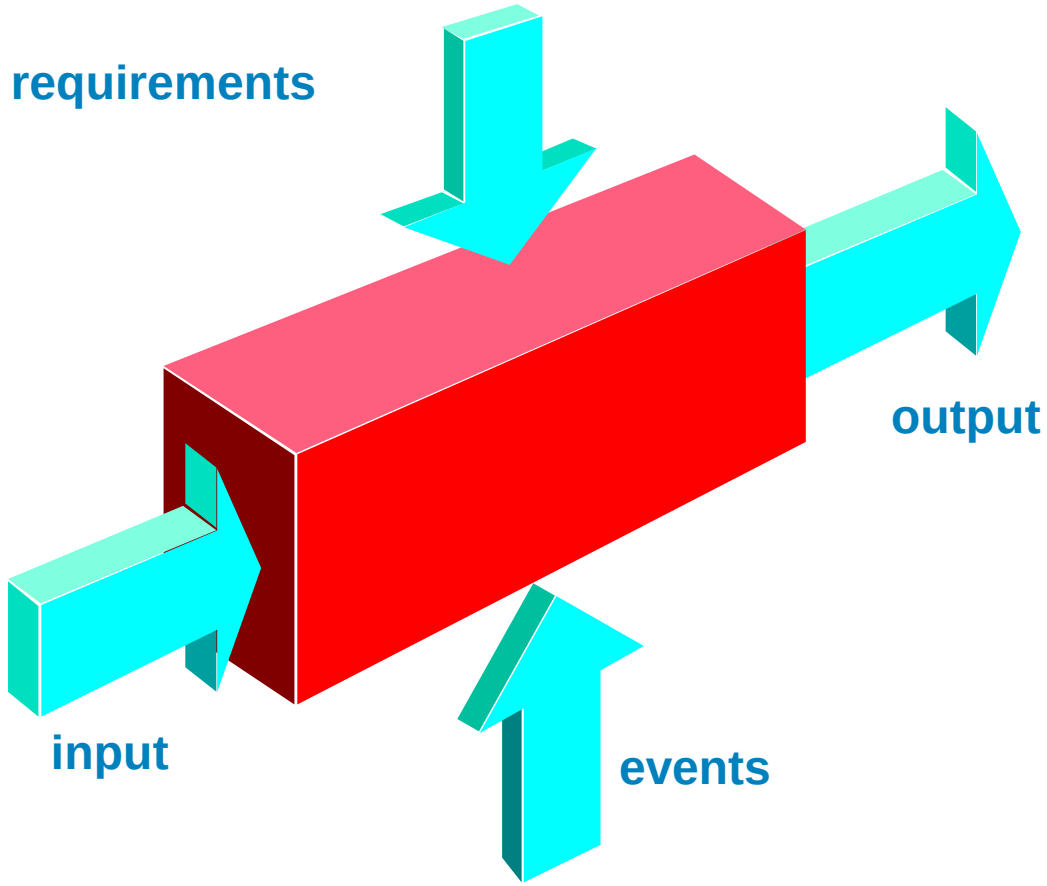
Unstructured Loops

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Testing Conventional Applications

Black-Box Testing



BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Testing Conventional Applications

Black-Box Testing

- How is functional validity tested?
- How is system behavior and performance tested?
- What classes of input will make good test cases?
- Is the system particularly sensitive to certain input values?
- How are the boundaries of a data class isolated?
- What data rates and data volume can the system tolerate?
- What effect will specific combinations of data have on system operation?

BINUS UNIVERSITY ONLINE LEARNING

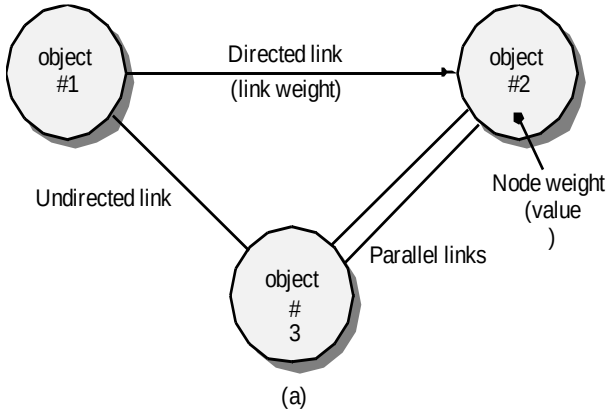
People Innovation Excellence

Testing Conventional Applications

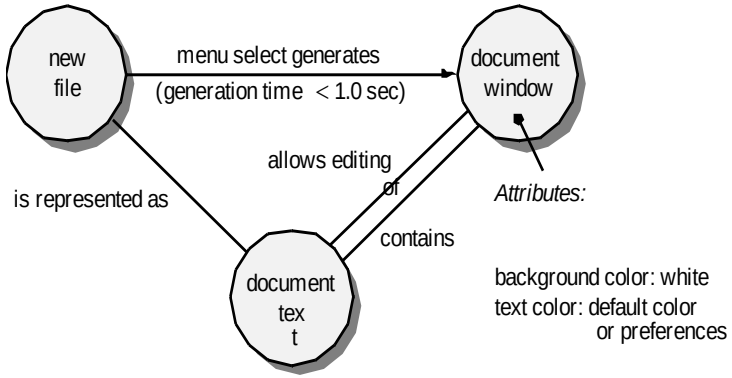
Graph-Based Methods

To understand the objects that are modeled in software and the relationships that connect these objects

In this context, we consider the term "objects" in the broadest possible context. It encompasses data objects, traditional components (modules), and object-oriented elements of computer software.



(a)



(b)

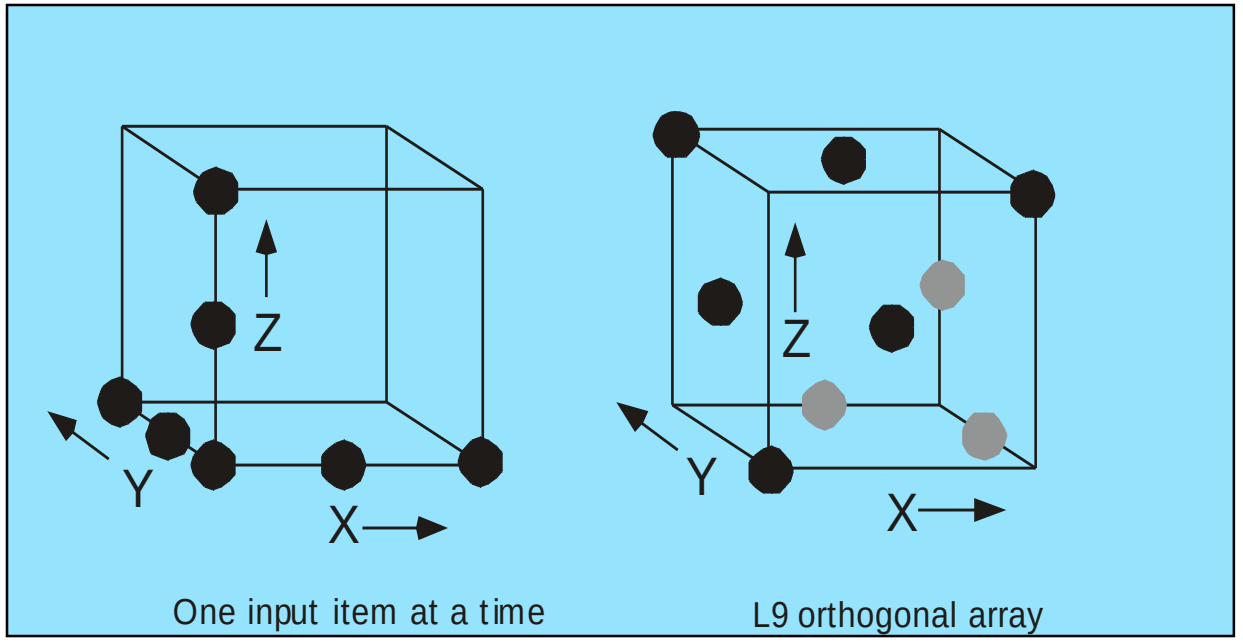
BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Testing Conventional Applications

Orthogonal Array Testing

- **Used when the number of input parameters is small and the values that each of the parameters may take are clearly bounded**



BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Testing Object Oriented Application

- **To adequately test OO systems, three things must be done:**
  - **the definition of testing must be broadened to include error discovery techniques applied to object-oriented analysis and design models**
  - **the strategy for unit and integration testing must change significantly, and**
  - **the design of test cases must account for the unique characteristics of OO software.**

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Testing Object Oriented Application

‘Testing’ OO Models

- The review of OO analysis and design models is especially useful because the same semantic constructs (e.g., classes, attributes, operations, messages) appear at the analysis, design, and code level
- Therefore, a problem in the definition of class attributes that is uncovered during analysis will circumvent side affects that might occur if the problem were not discovered until design or code (or even the next iteration of analysis).

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Testing Object Oriented Application

Correctness of OO Models

- During analysis and design, semantic correctness can be asesssed based on the model's conformance to the real world problem domain.
- If the model accurately reflects the real world (to a level of detail that is appropriate to the stage of development at which the model is reviewed) then it is semantically correct.
- To determine whether the model does, in fact, reflect real world requirements, it should be presented to problem domain experts who will examine the class definitions and hierarchy for omissions and ambiguity.
- Class relationships (instance connections) are evaluated to determine whether they accurately reflect real-world object connections.

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Testing Object Oriented Application

Class Model Consistency

- **Revisit the CRC model and the object-relationship model.**
- **Inspect the description of each CRC index card to determine if a delegated responsibility is part of the collaborator's definition.**
- **Invert the connection to ensure that each collaborator that is asked for service is receiving requests from a reasonable source.**
- **Using the inverted connections examined in the preceding step, determine whether other classes might be required or whether responsibilities are properly grouped among the classes.**
- **Determine whether widely requested responsibilities might be combined into a single responsibility.**

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Testing Object Oriented Application

OO Testing Strategies

- **Unit testing**
  - **the concept of the unit changes**
  - **the smallest testable unit is the encapsulated class**
  - **a single operation can no longer be tested in isolation (the conventional view of unit testing) but rather, as part of a class**
- **Integration Testing**
  - ***Thread-based testing* integrates the set of classes required to respond to one input or event for the system**
  - ***Use-based testing* begins the construction of the system by testing those classes (called *independent classes*) that use very few (if any) of server classes. After the independent classes are tested, the next layer of classes, called *dependent classes***
  - ***Cluster testing* defines a cluster of collaborating classes (determined by examining the CRC and object-relationship model) is exercised by designing test cases that attempt to uncover errors in the collaborations.**

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Testing Object Oriented Application

OOT Methods

Berard [Ber93] proposes the following approach:

1. **Each test case should be uniquely identified and should be explicitly associated with the class to be tested**
2. **The purpose of the test should be stated.**
3. **A list of testing steps should be developed for each test and should contain:**
   a. **a list of specified states for the object that is to be tested**
   b. **a list of messages and operations that will be exercised as a on sequence of the test**
   c. **a list of exceptions that may occur as the object is tested**
   d. **a list of external conditions (i.e., changes in the environment external to the software that must exist in order to properly conduct the test)**
   e. **supplementary information that will aid in understanding or implementing the test.**

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Testing Object Oriented Application

Testing Methods

- **Fault-based testing**
  - **The tester looks for plausible faults (i.e., aspects of the implementation of the system that may result in defects). To determine whether these faults exist, test cases are designed to exercise the design or code.**
- **Class Testing and the Class Hierarchy**
  - **Inheritance does not obviate the need for thorough testing of all derived classes. In fact, it can actually complicate the testing process.**
- **Scenario-Based Test Design**
  - **Scenario-based testing concentrates on what the user does, not what the product does. This means capturing the tasks (via use-cases) that the user has to perform, then applying them and their variants as tests.**

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Testing Object Oriented Application

OOT Methods: Random Testing

- Random testing
  - identify operations applicable to a class
  - define constraints on their use
  - identify a minimum test sequence
    - an operation sequence that defines the minimum life history of the class (object)
  - generate a variety of random (but valid) test sequences
    - exercise other (more complex) class instance life histories

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Testing Object Oriented Application

OOT Methods: Partition Testing

- **Partition Testing**
  - **reduces the number of test cases required to test a class in much the same way as equivalence partitioning for conventional software**
  - **state-based partitioning**
    - **categorize and test operations based on their ability to change the state of a class**
  - **attribute-based partitioning**
    - **categorize and test operations based on the attributes that they use**
  - **category-based partitioning**
    - **categorize and test operations based on the generic function each performs**

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Testing Object Oriented Application

OOT Methods: Inter-Class Testing

- Inter-class testing
  - For each client class, use the list of class operators to generate a series of random test sequences. The operators will send messages to other server classes.
  - For each message that is generated, determine the collaborator class and the corresponding operator in the server object.
  - For each operator in the server object (that has been invoked by messages sent from the client object), determine the messages that it transmits.
  - For each of the messages, determine the next level of operators that are invoked and incorporate these into the test sequence

BINUS UNIVERSITY ONLINE LEARNING

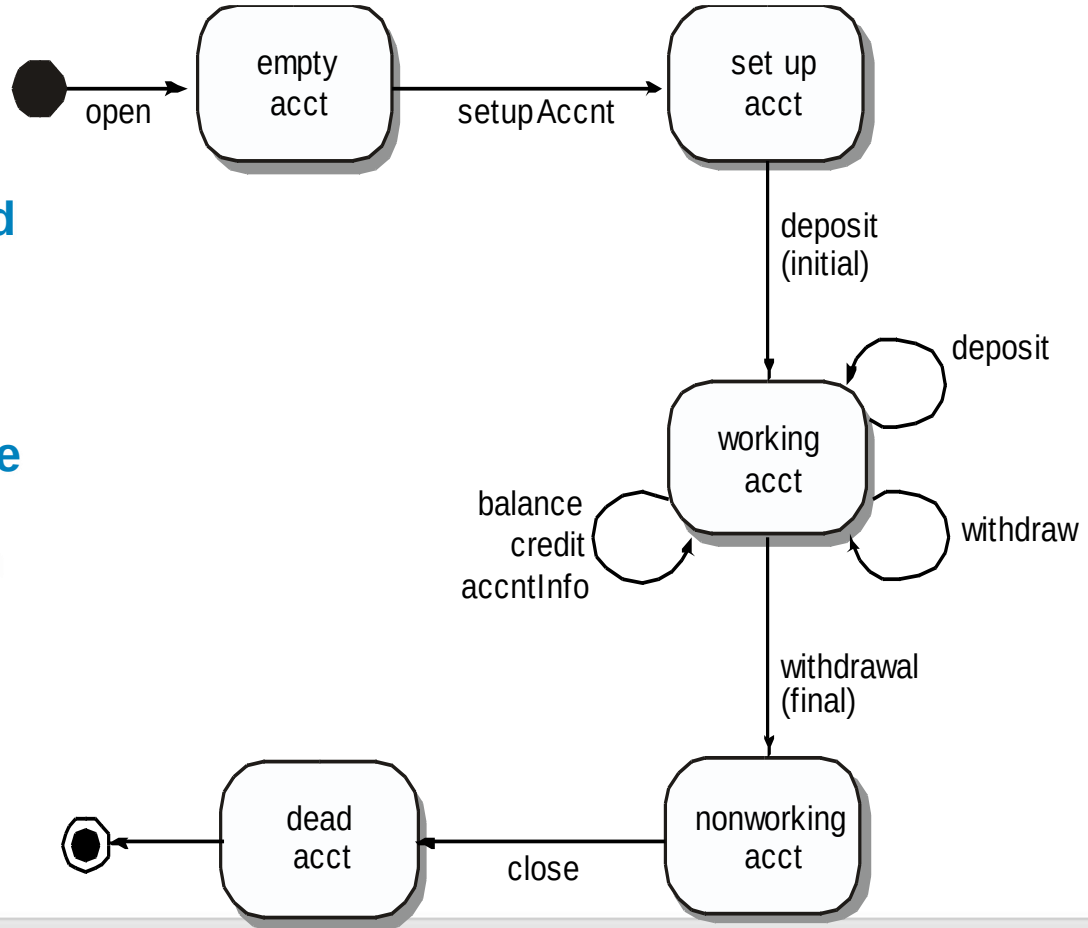
People Innovation Excellence

Testing Object Oriented Application

OOT Methods: Behavior Testing

The tests to be designed should achieve all state coverage [KIR94].

That is, the operation sequences should cause the account class to make transition through all allowable states



Testing Web Applications

Testing Quality Dimensions-I

- ***Content*** **is evaluated at both a syntactic and semantic level.**
  - **syntactic level**
  - **semantic level**
- ***Function*** **is tested for correctness, instability, and general conformance to appropriate implementation standards (e.g.,Java or XML language standards).**
- ***Structure*** **is assessed to ensure that it**
  - **properly delivers WebApp content and function**
  - **is extensible**
  - **can be supported as new content or functionality is added.**
- ***Usability*** **is tested to ensure that each category of user**
  - **is supported by the interface**
  - **can learn and apply all required navigation syntax and semantics**
- ***Navigability*** **is tested to ensure that**
  - **all navigation syntax and semantics are exercised to uncover any navigation errors (e.g., dead links, improper links, erroneous links).**

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Testing Web Applications

Testing Quality Dimensions-II

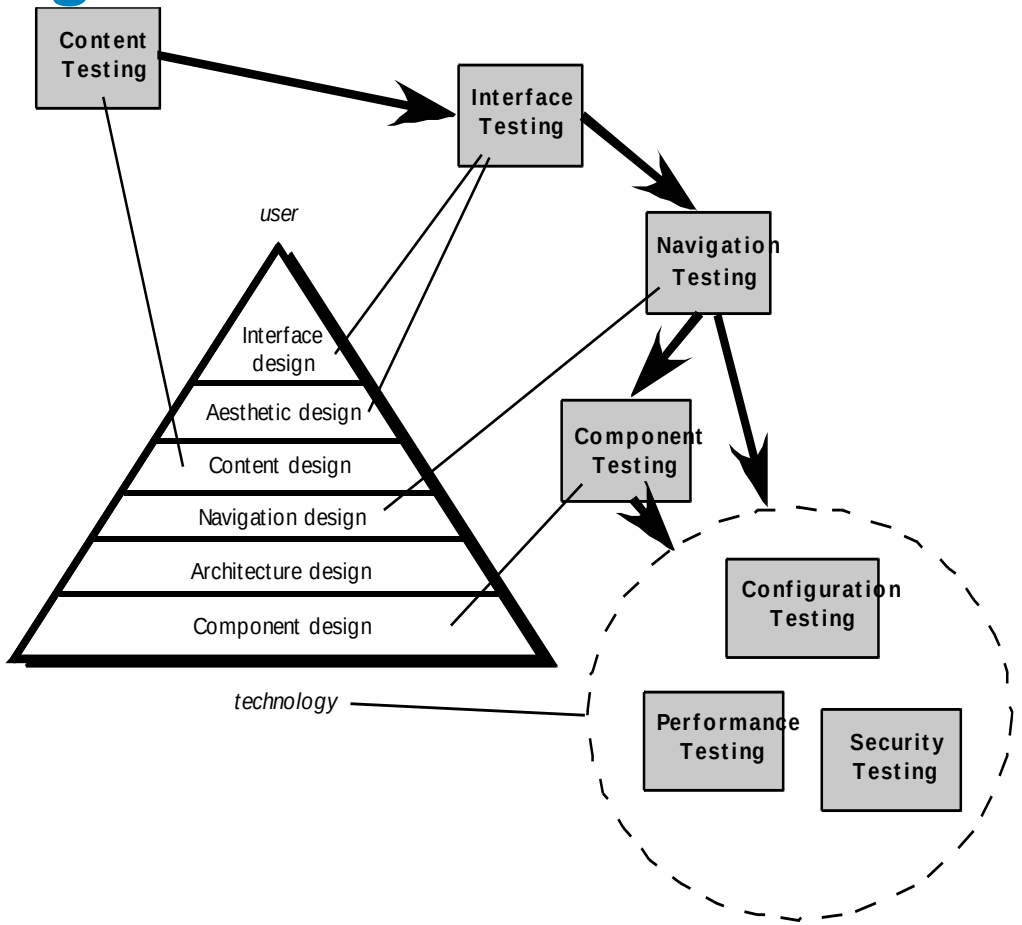
- ***Performance*** **is tested under a variety of operating conditions, configurations, and loading to ensure that**
  - **the system is responsive to user interaction**
  - **the system handles extreme loading without unacceptable operational degradation**
- ***Compatibility*** **is tested by executing the WebApp in a variety of different host configurations on both the client and server sides.**
  - **The intent is to find errors that are specific to a unique host configuration.**
- ***Interoperability*** **is tested to ensure that the WebApp properly interfaces with other applications and/or databases.**
- ***Security*** **is tested by assessing potential vulnerabilities and attempting to exploit each.**
  - **Any successful penetration attempt is deemed a security failure.**

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Testing Web Applications

The Testing Process



BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Testing Web Applications

Content Testing

- Content testing has three important objectives:
  - to uncover syntactic errors (e.g., typos, grammar mistakes) in text-based documents, graphical representations, and other media
  - to uncover semantic errors (i.e., errors in the accuracy or completeness of information) in any content object presented as navigation occurs, and
  - to find errors in the organization or structure of content that is presented to the end-user.

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Testing Web Applications

Assessing Content Semantics

- Is the information factually accurate?
- Is the information concise and to the point?
- Is the layout of the content object easy for the user to understand?
- Can information embedded within a content object be found easily?
- Have proper references been provided for all information derived from other sources?
- Is the information presented consistent internally and consistent with information presented in other content objects?
- Is the content offensive, misleading, or does it open the door to litigation?
- Does the content infringe on existing copyrights or trademarks?
- Does the content contain internal links that supplement existing content? Are the links correct?
- Does the aesthetic style of the content conflict with the aesthetic style of the interface?

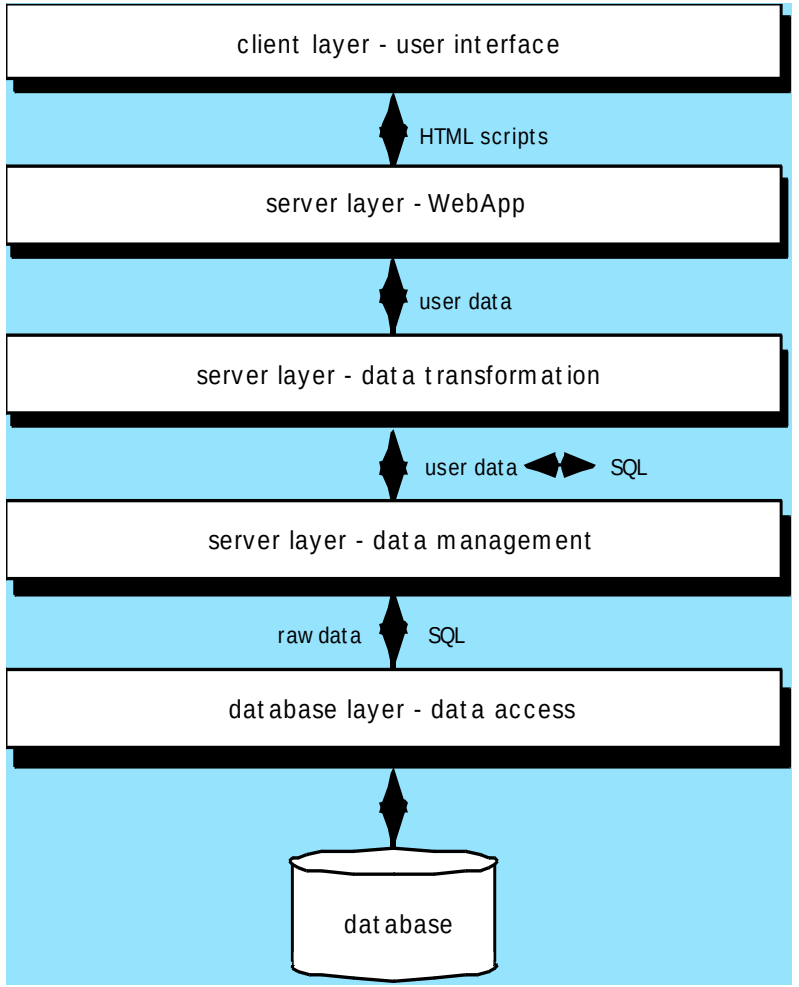
BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Testing Web Applications

Database Testing

Tests are defined for each layer



BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Testing Web Applications

User Interface Testing

- Interface features are tested to ensure that design rules, aesthetics, and related visual content is available for the user without error.
- Individual interface mechanisms are tested in a manner that is analogous to unit testing.
- Each interface mechanism is tested within the context of a use-case or NSU for a specific user category.
- The complete interface is tested against selected use-cases and NSUs to uncover errors in the semantics of the interface.
- The interface is tested within a variety of environments (e.g., browsers) to ensure that it will be compatible.

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Testing Web Applications

Testing Guidelines

- ***Understand the network and device landscape before testing to identify bottlenecks***
- ***Conduct tests in uncontrolled real-world test condition (filed-based testing)***
- ***Select the right automation test tool***
- ***Use the Weighted Device Platform Matrix method to identify the most critical hardware/platform combination to test***
- ***Check the end-to-end functional flow in all possible platforms at least once***
- ***Conduct performance testing, GUI testing, and compatibility testing using actual devices***
- ***Measure performance only in realistic conditions of wireless and user load***

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Testing Web Applications

Testing Strategies

- **Developing a MobileApp testing strategy requires an understanding of both software testing and the challenges that make mobile devices and their network infrastucture unique.**
- **In addition to a thorough knowledge of conventional software testing approach, a MobileApp tester should have a good understanding of telecommunications principles and an awarness of the differences and capabilities of mobile operating systems platforms.**
- **This basic knowledge must be complemented with a thorough understanding of the different types of mobile testing (e.g. MobileApp testing, mobile handset testing, mobile website testing), the use of simulators, test automations tools, and remote data access services (RDA).**

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Testing Mobile Applications

Criteria Testing Tools and Environments

- Object identification
- Security
- Devices
- Functionality
- Emulators and plug-ins
- Connectivity

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Security Engineering

Analyzing Security Requirements

- An important part of building secure systems is anticipating conditions or threats that may be used to damage system resources or render them inaccessible to authorized users.
- This process is called *threat analysis.*
- Once the system assets, vulnerabilities, and threats have been identified, controls can be created to either avoid attacks or mitigate their damage.

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Security Engineering

Analyzing Security Requirements

- **Software security is an essential prerequisite for software integrity, availability, reliability, and safety**
- **It may not possible to create a system that can be defend its assets againt all possible threats, and for that reason, it may be necessary to encourage users to maintain backup copies of critical data, redundant system component, and ensure privacy controls are in place**

BINUS UNIVERSITY ONLINE LEARNING

People
Innovation
Excellence

Security Engineering

Security and Privacy in an Online World

- Social Media
- Mobile Applications
- Cloud Computing
- The Internet of Things

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Security Engineering

Security Risk Analysis

- Identify assets
- Create an architecture overview
- Decompose the application
- Identify threats
- Documented the threats
- Rate the threats

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Case Study

- **In some organization, they have a special QA/QC department who manage and control the testing process. They are responsible for performing the test**
- **The test can involve the related team and stakeholder of the application**
- **The process will start for creating the test plan**
- **The User Acceptance Test (UAT) is the minimum test to ensure the application can be used in the production environment**

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

References

- Pressman, R.S. (2015). ***Software Engineering : A Practioner's Approach. 8th ed***. McGraw-Hill Companies.Inc, Americas, New York. ISBN : 978 1 259 253157.
- **Software Testing, http://www.youtube.com/watch?v=caElFKbceP0&list=PLED41984073D5B532**
- **Conventional sw testing** http://www.testingexcellence.com/conventional-software-testing-on-an-extreme-programming-team/
- **Testing OO SW**

  http://diwww.epfl.ch/researchlgl/research/ongoing/testing.html
- **Web testing**

  http://www.manageengine.com/products/qengine/web-testing.html

BINUS UNIVERSITY ONLINE LEARNING

People
Innovation
Excellence

Q & A

BINUS
UNIVERSITY
ONLINE
LEARNING

People
Innovation
Excellence

Thank You

BINUS UNIVERSITY ONLINE LEARNING

People
Innovation
Excellence

Software Engineering

Topic 7

Software Configuration Management

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Acknowledgement

These slides have been adapted from Pressman, R.S. (2015). *Software Engineering : A Practioner's Approach. 8th ed*. McGraw-Hill Companies.Inc, Americas, New York. ISBN : 978 1 259 253157. Chapter 23, 24, 25, 26 and 27

BINUS UNIVERSITY ONLINE LEARNING

People
Innovation
Excellence

Learning Objectives

LO 4 : Analyze the software project management and the proposed potential business project

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

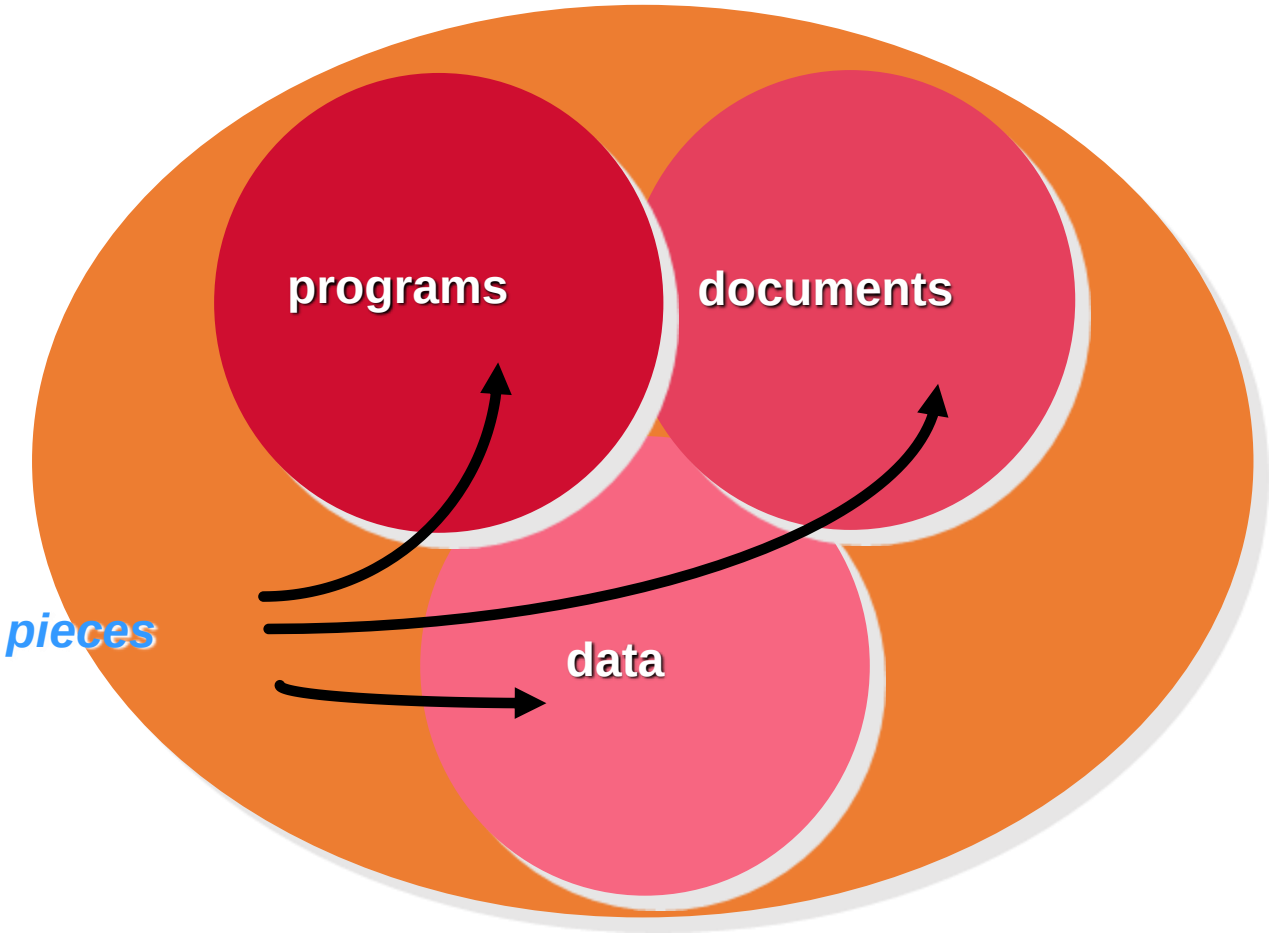
Contents

- **The Software Configuration**
- **SCM Repository**
- **The SCM Process**
- **Configuration Management for Web and Mobile Apps**

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

The Software Configuration



The pieces

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

The Software Configuration

Baselines

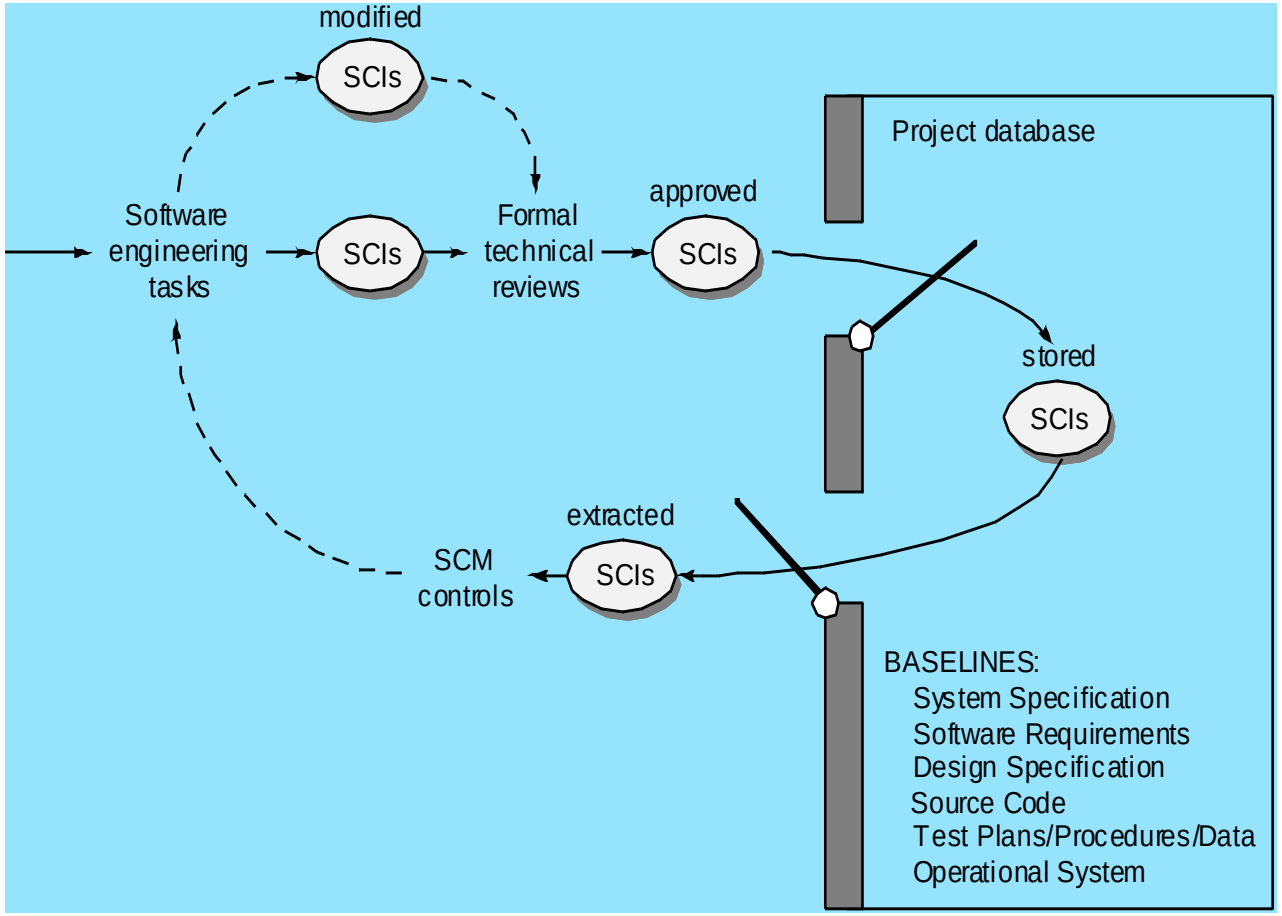
- The IEEE (IEEE Std. No. 610.12-1990) defines a baseline as:
  - A specification or product that has been formally reviewed and agreed upon, that thereafter serves as the basis for further development, and that can be changed only through formal change control procedures.
- a baseline is a milestone in the development of software that is marked by the delivery of one or more software configuration items and the approval of these SCIs that is obtained through a formal technical review

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

The Software Configuration

Baselines

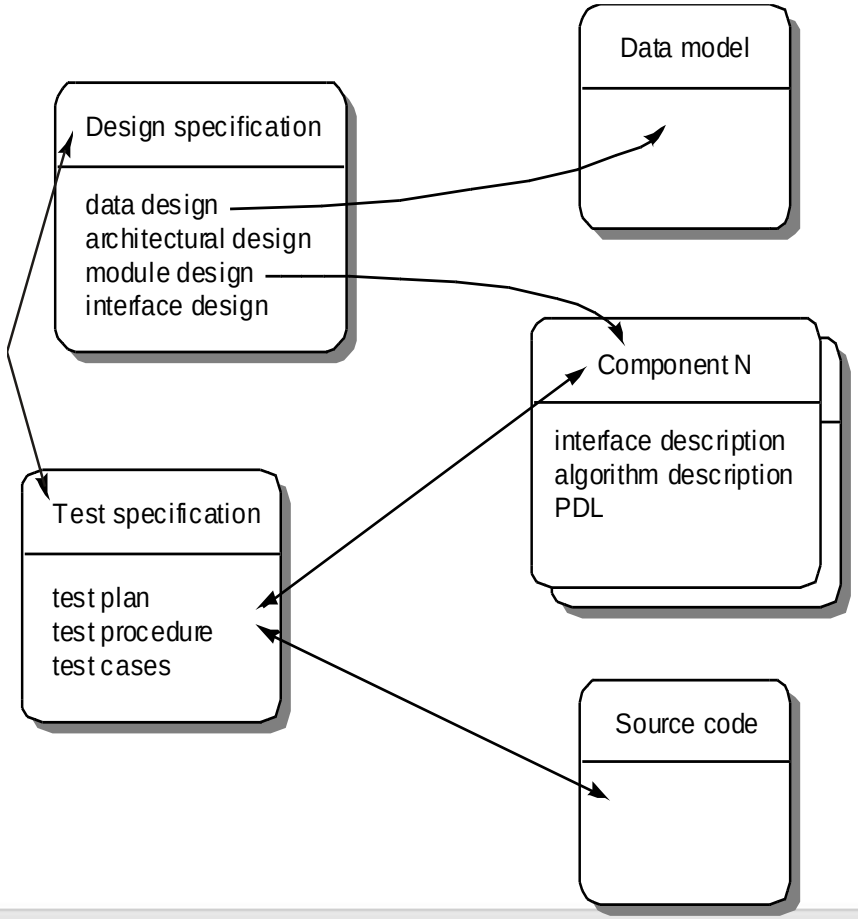


BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

The Software Configuration

Software Configuration Objects



BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

SCM Repository

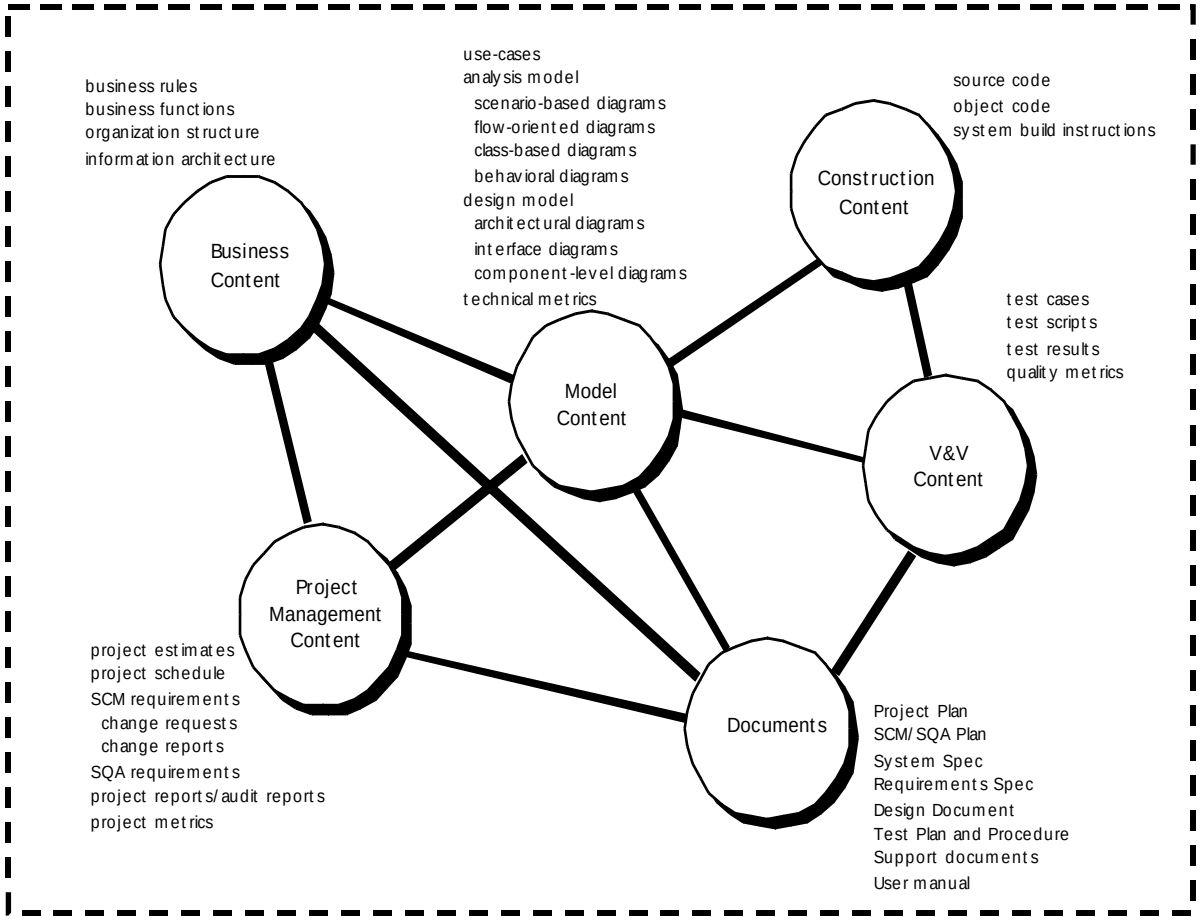
- The SCM repository is the set of mechanisms and data structures that allow a software team to manage change in an effective manner
- The repository performs or precipitates the following functions [For89]:
  - Data integrity
  - Information sharing
  - Tool integration
  - Data integration
  - Methodology enforcement
  - Document standardization

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

The Software Configuration

Repository Content



BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

The Software Configuration

Repository Features

- **Versioning.**
  - saves all of these versions to enable effective management of product releases and to permit developers to go back to previous versions
- **Dependency tracking and change management.**
  - The repository manages a wide variety of relationships among the data elements stored in it.
- **Requirements tracing.**
  - Provides the ability to track all the design and construction components and deliverables that result from a specific requirement specification
- **Configuration management.**
  - Keeps track of a series of configurations representing specific project milestones or production releases. Version management provides the needed versions, and link management keeps track of interdependencies.
- **Audit trails.**
  - establishes additional information about when, why, and by whom changes are made.

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

The Software Configuration

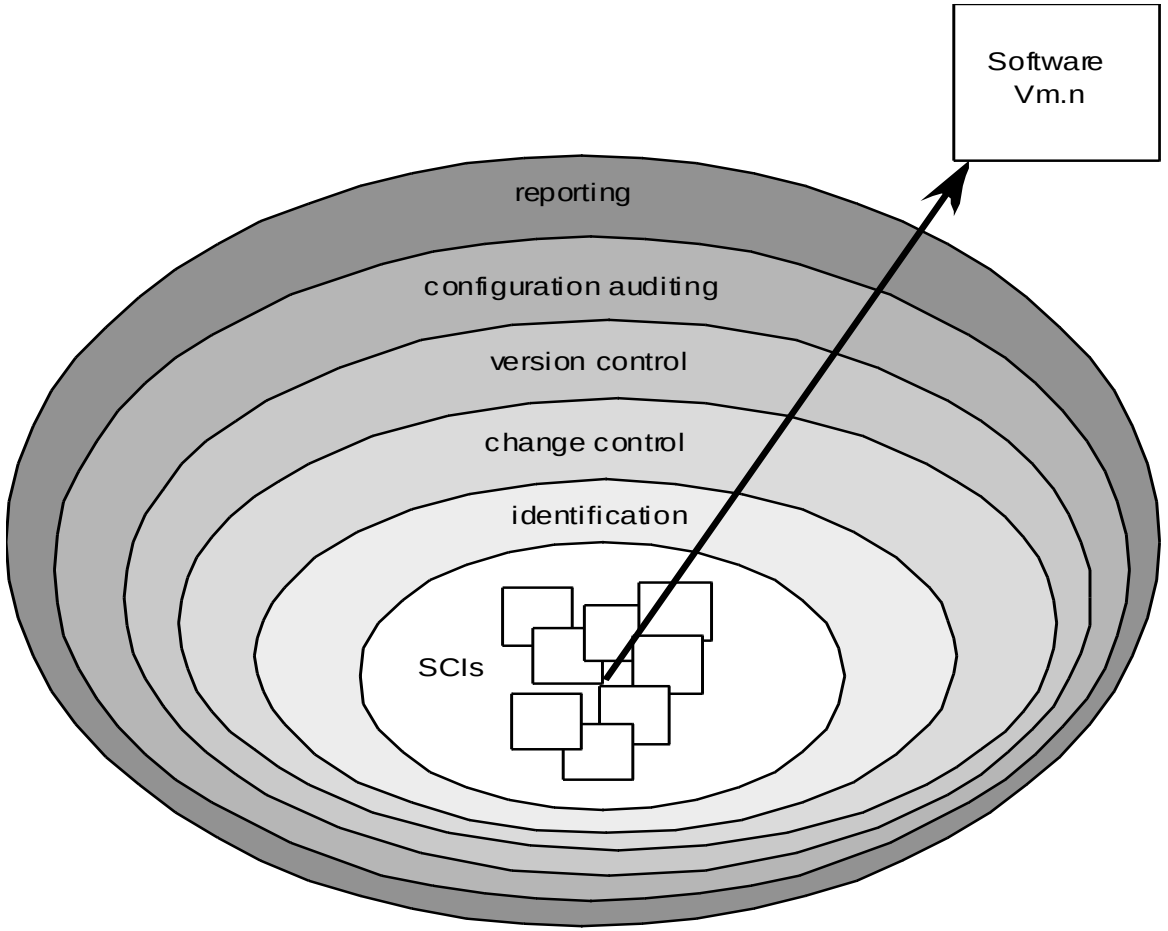
SCM Elements

- *Component elements*—a set of tools coupled within a file management system (e.g., a database) that enables access to and management of each software configuration item.
- *Process elements*—a collection of procedures and tasks that define an effective approach to change management (and related activities) for all constituencies involved in the management, engineering and use of computer software.
- *Construction elements*—a set of tools that automate the construction of software by ensuring that the proper set of validated components (i.e., the correct version) have been assembled.
- *Human elements*—to implement effective SCM, the software team uses a set of tools and process features (encompassing other CM elements)

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

The SCM Process

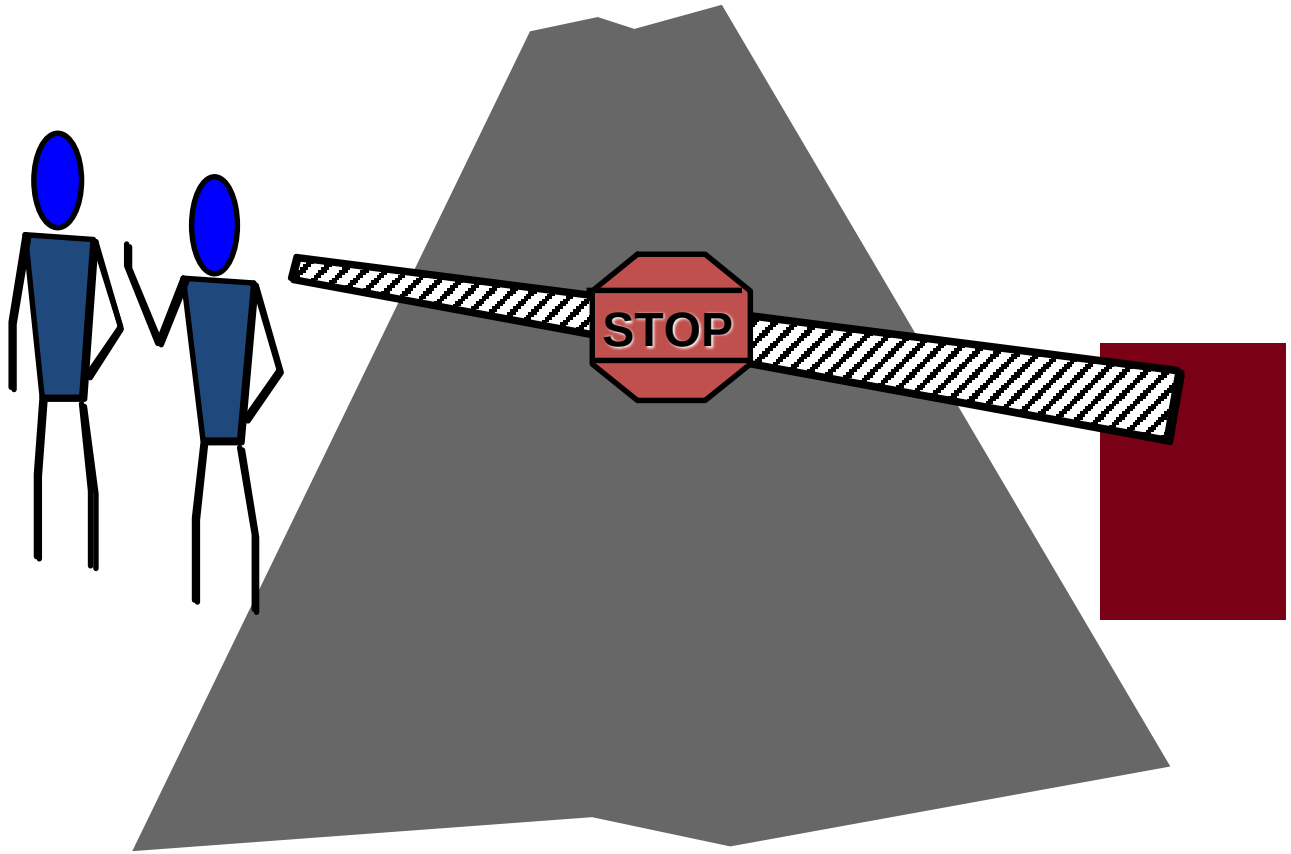


BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

The SCM Process

Change Control

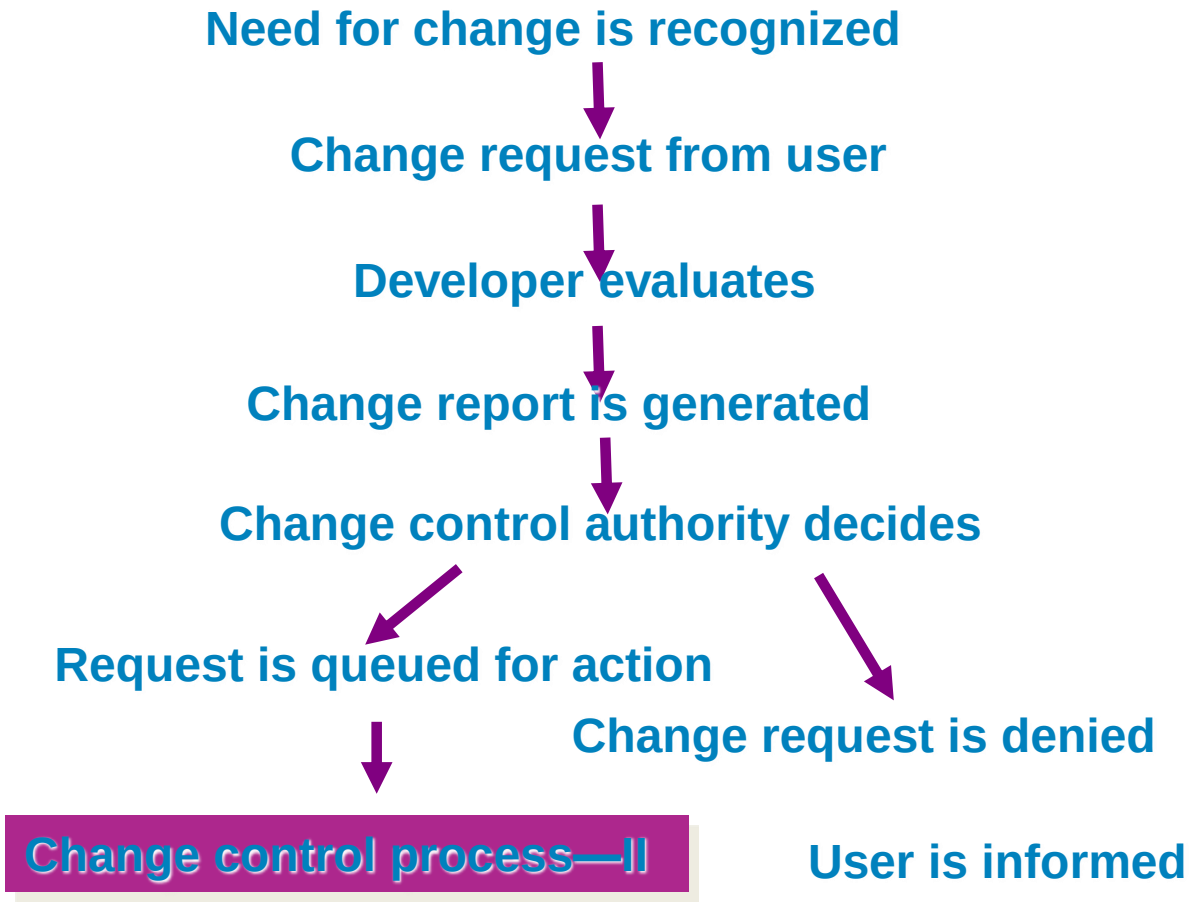


BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

The SCM Process

Change Control Process—I



BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Change Control Process-II

The SCM Process

Request is queued for action

↓

Assign individuals to configuration objects

↓

"Check-out" configuration objects

↓

Make the change

↓

Review (audit)the change

↓

"Check in" the configuration items that have been changed

↓

Establish a baseline for testing

↓

change control process—III

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

The SCM Process

Change Control Process-III

Perform quality assurance and testing activities

↓

"Promote" changes for inclusion in next release (revision)

↓

Rebuild appropriate version of software

↓

Review (audit) the change to all configuration items

↓

Include changes in new version

↓

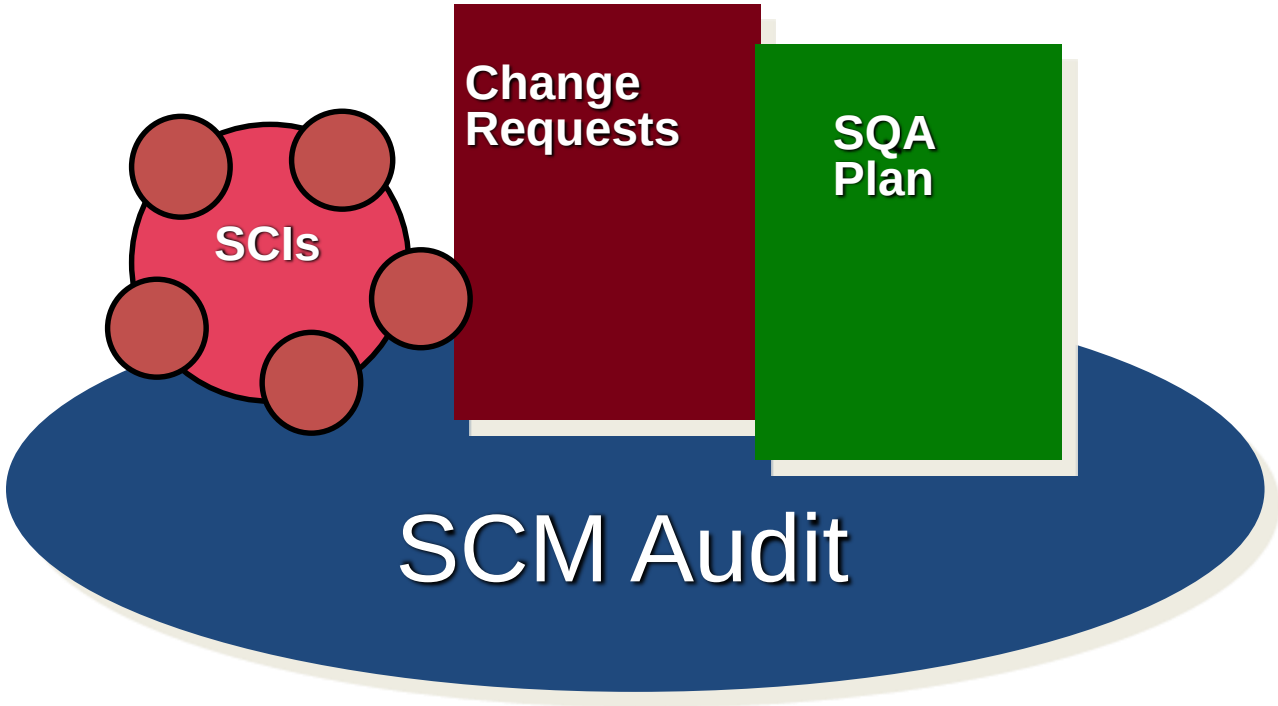
Distribute the new version

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

The SCM Process

Auditing

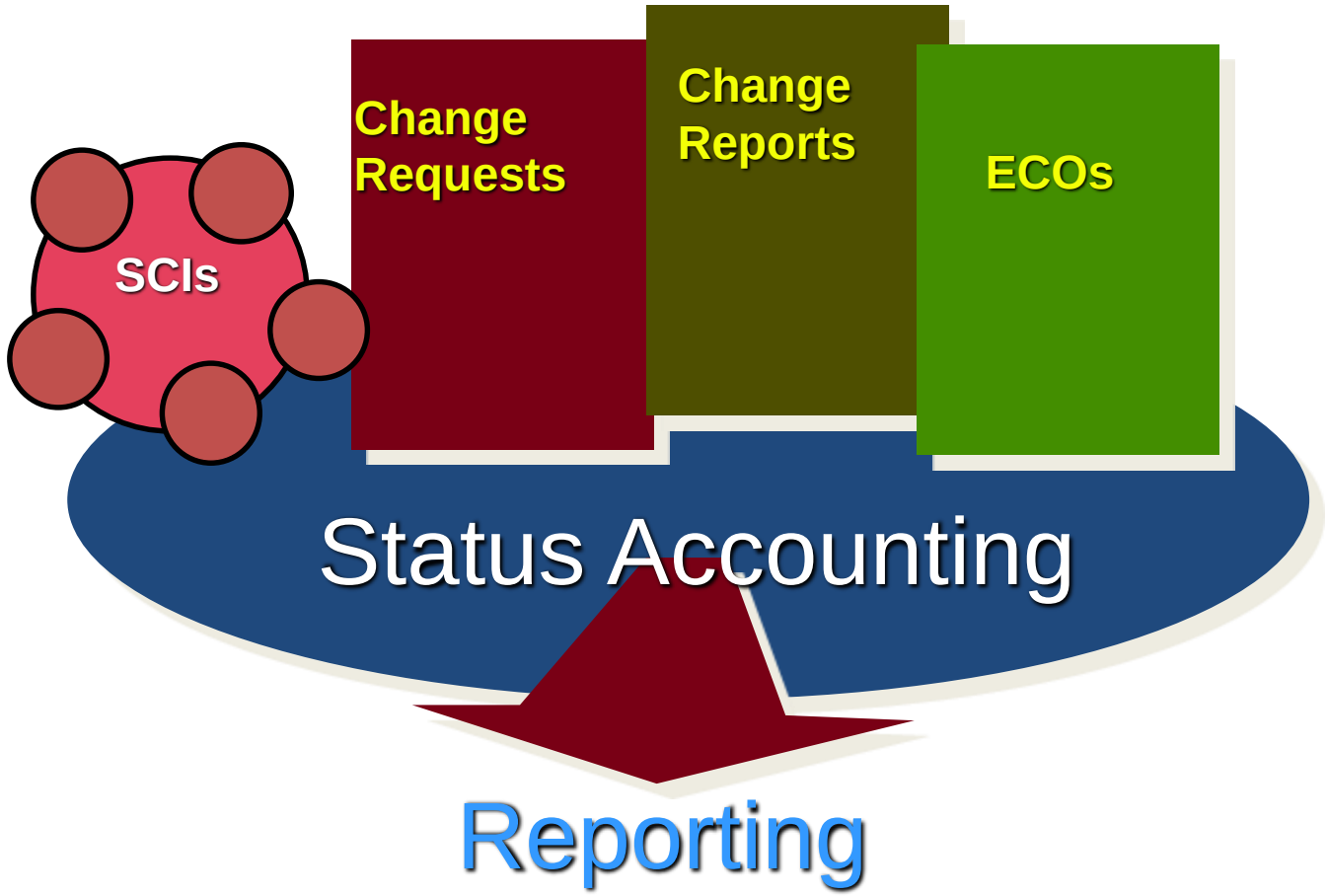


BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

The SCM Process

Status Accounting



BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Configuration Management for Web and MobileApps

- **Content.**
  - A typical WebApp contains a vast array of content—text, graphics, applets, scripts, audio/video files, forms, active page elements, tables, streaming data, and many others.
  - The challenge is to organize this sea of content into a rational set of configuration objects and then establish appropriate configuration control mechanisms for these objects.
- **People.**
  - Because a significant percentage of WebApp development continues to be conducted in an ad hoc manner, any person involved in the WebApp can (and often does) create content.

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Configuration Management for Web and MobileApps

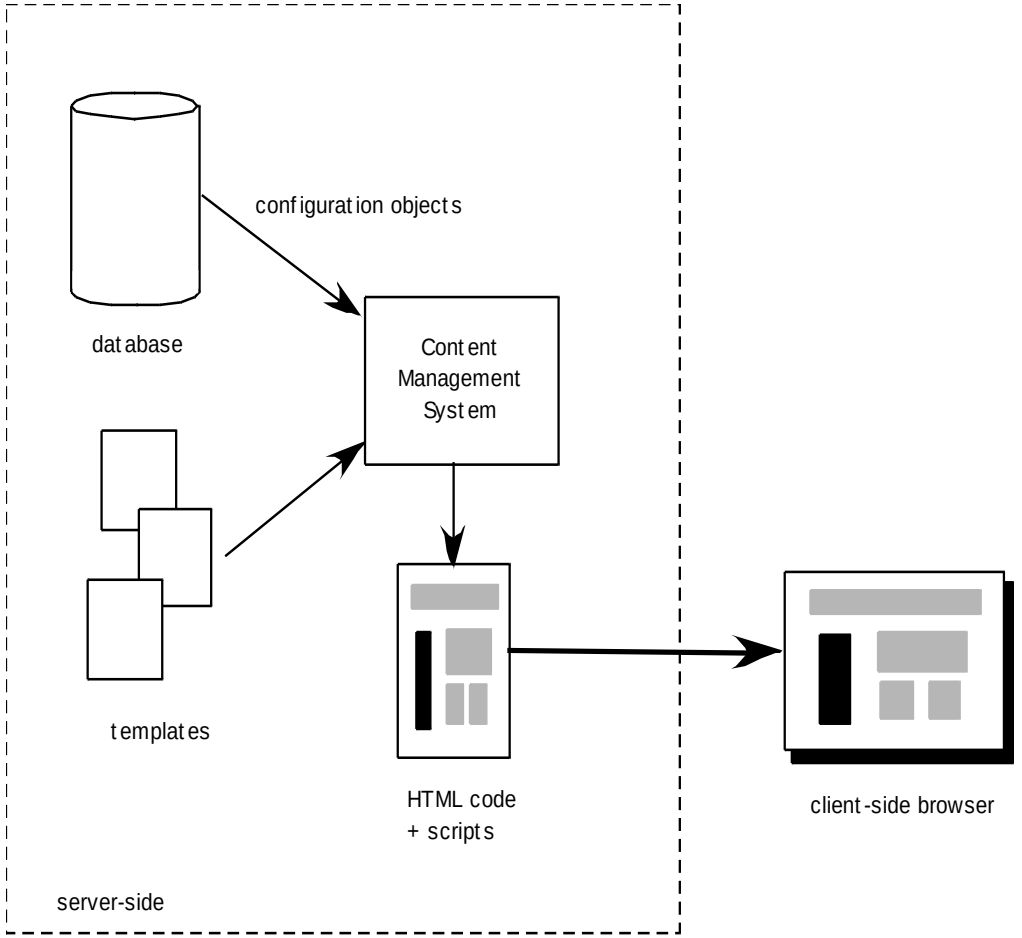
- **Scalability.**
  - As size and complexity grow, small changes can have far-reaching and unintended affects that can be problematic. Therefore, the rigor of configuration control mechanisms should be directly proportional to application scale.
- **Politics.**
  - Who ‘owns’ a WebApp?
  - Who assumes responsibility for the accuracy of the information on the Web site?
  - Who assures that quality control processes have been followed before information is published to the site?
  - Who is responsible for making changes?
  - Who assumes the cost of change?

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Configuration Management for Web and MobileApps

Content Management

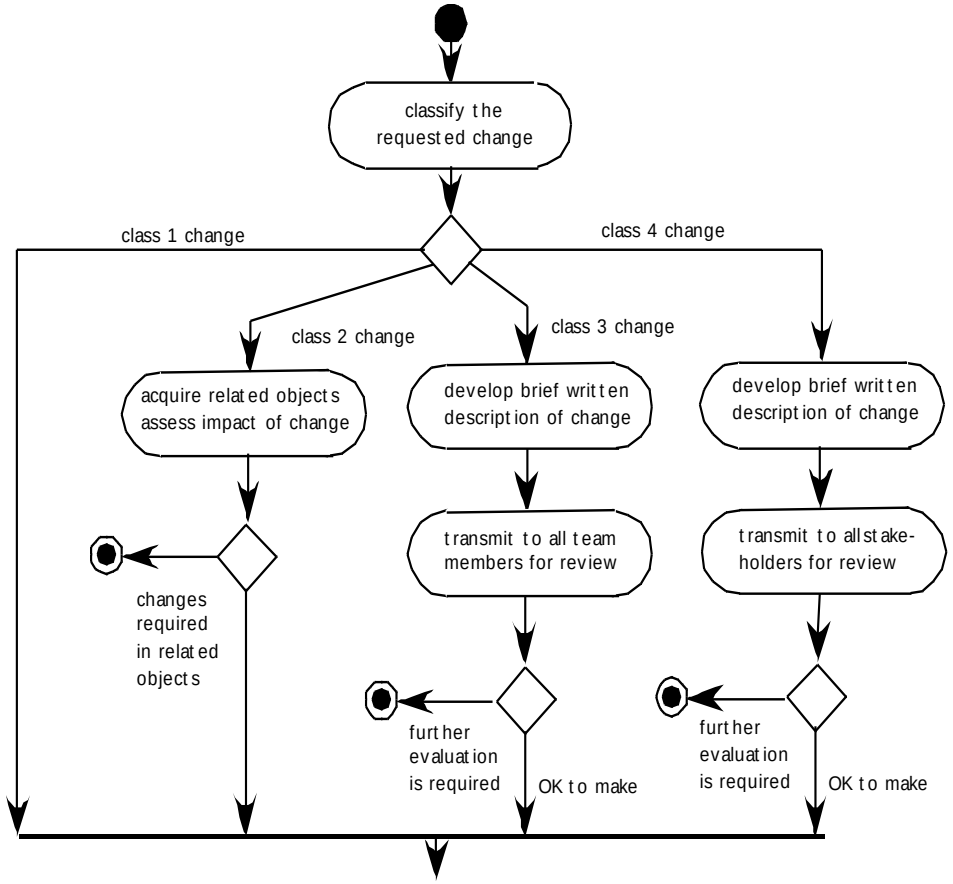


BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Configuration Management for Web and MobileApps

Change Management for WebApps-I

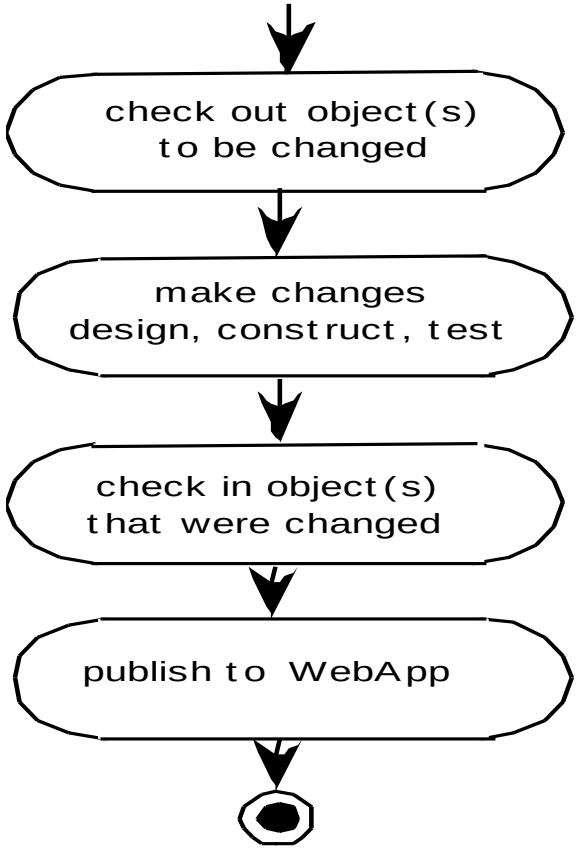


BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Configuration Management for Web and MobileApps

Change Management for WebApps-II



BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Case Study (1)

- Software configuration process is also part of the ITSM (IT Software Management). The popular methodology is ITIL (IT Infrastructure Library)
- In the practical development and operation, the team usually apply the integrated software development tools

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Case Study (2)

- The most popular tools is JIRA, which have the following features:
  - Plan
  - Track
  - Release
  - Report
- In the release software, we can define the release version, status, progress, start date and release date.

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

References

- Pressman, R.S. (2015). ***Software Engineering : A Practioner's Approach. 8th ed***. McGraw-Hill Companies.Inc, Americas, New York. ISBN : 978 1 259 253157.
- Software Testing
  http://www.io.com/~wazmo/qa/
- SW Testing Storm
  http://www.mtsu.edu/~storm
- Conventional sw testing
  http://www.testingexcellence.com/conventional-software-testing-on-an-extreme-programming-team/
- Testing OO SW
  http://diwww.epfl.ch/researchlgl/research/ongoing/testing.html
- Web testing
  http://www.manageengine.com/products/qengine/web-testing.html

BINUS UNIVERSITY ONLINE LEARNING

People
Innovation
Excellence

Q & A

BINUS
UNIVERSITY
ONLINE
LEARNING

People
Innovation
Excellence

Thank You

BINUS UNIVERSITY ONLINE LEARNING

People
Innovation
Excellence

Software Engineering

Topic 8

Software Project Management & Software Metrics

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Acknowledgement

These slides have been adapted from Pressman, R.S. (2015). *Software Engineering : A Practioner's Approach. 8th ed*. McGraw-Hill Companies.Inc, Americas, New York. ISBN : 978 1 259 253157. Chapter 30, 31 and 32

BINUS UNIVERSITY ONLINE LEARNING

People
Innovation
Excellence

Learning Objectives

LO 4 : Analyze the software project management and the proposed potential business project

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Contents

- Team Coordination & Communication
- Problem Decomposition
- A Framework for Product Metrics
- Metrics for the Requirements Model
- Metrics for the Design Model
- Design Metrics for WebApps
- Code Metrics
- Metrics for Testing
- Maintenance Metrics
- Metrics in the Process and Project Domain
- Software Measurement
- Metrics for Software Quality

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Team Coordination and Communication

- *Formal, impersonal approaches* include software engineering documents and work products (including source code), technical memos, project milestones, schedules, and project control tools, change requests and related documentation, error tracking reports, and repository data.
- *Formal, interpersonal procedures* focus on quality assurance activities applied to software engineering work products. These include status review meetings and design and code inspections.
- *Informal, interpersonal procedures* include group meetings for information dissemination and problem solving and "collocation of requirements and development staff."
- *Electronic communication* encompasses electronic mail, electronic bulletin boards, and by extension, video-based conferencing systems.
- *Interpersonal networking* includes informal discussions with team members and those outside the project who may have experience or insight that can assist team members.

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Problem Decomposition

- Sometimes called *partitioning* or *problem elaboration*
- Once scope is defined …
  - It is decomposed into constituent functions
  - It is decomposed into user-visible data objects

  or

  - It is decomposed into a set of problem classes
- Decomposition process continues until all functions or problem classes have been defined

Problem Decomposition

The Process

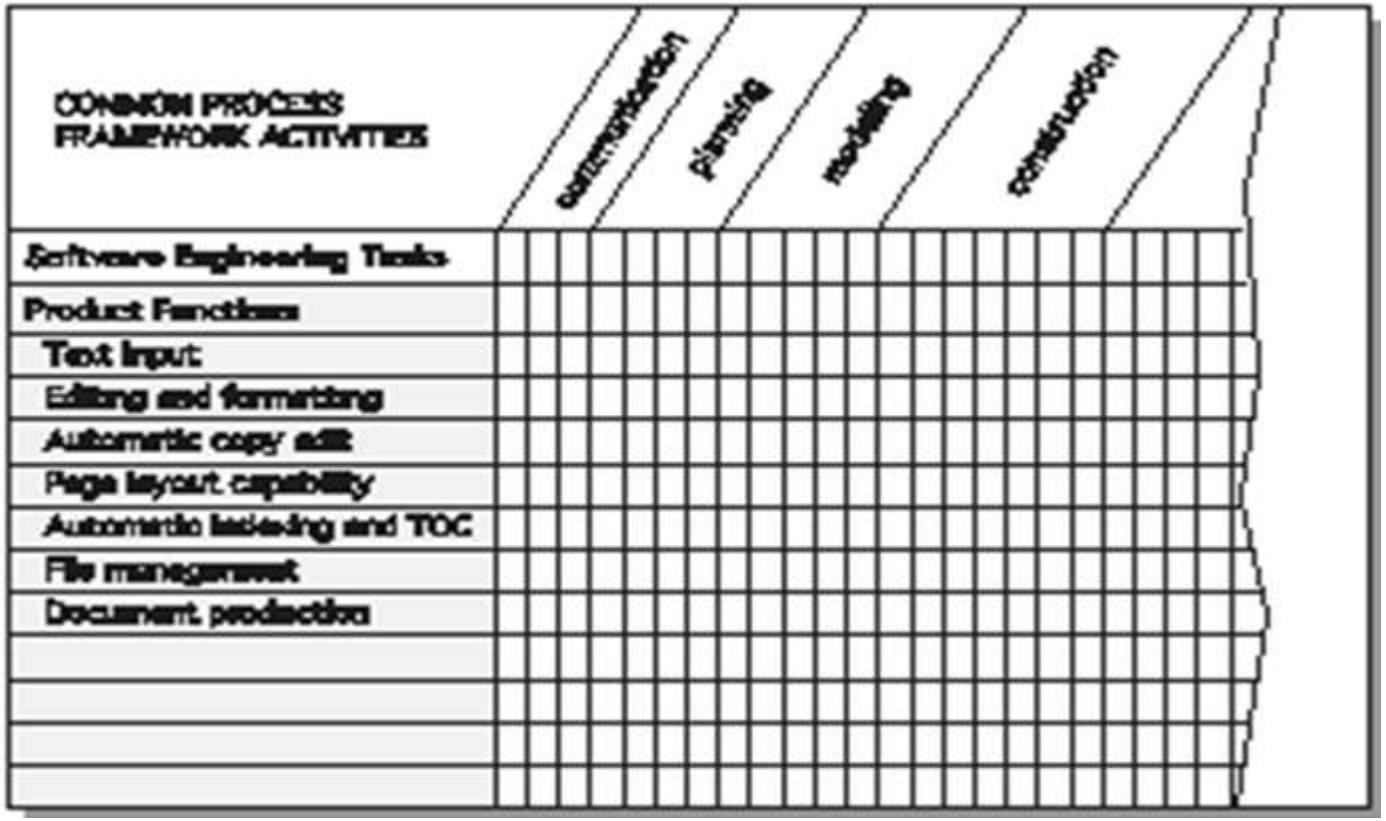
- Once a process framework has been established
  - Consider project characteristics
  - Determine the degree of rigor required
  - Define a task set for each software engineering activity
    - Task set =
      - Software engineering tasks
      - Work products
      - Quality assurance points
      - Milestones

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Problem Decomposition

Melding the Problem and the Process

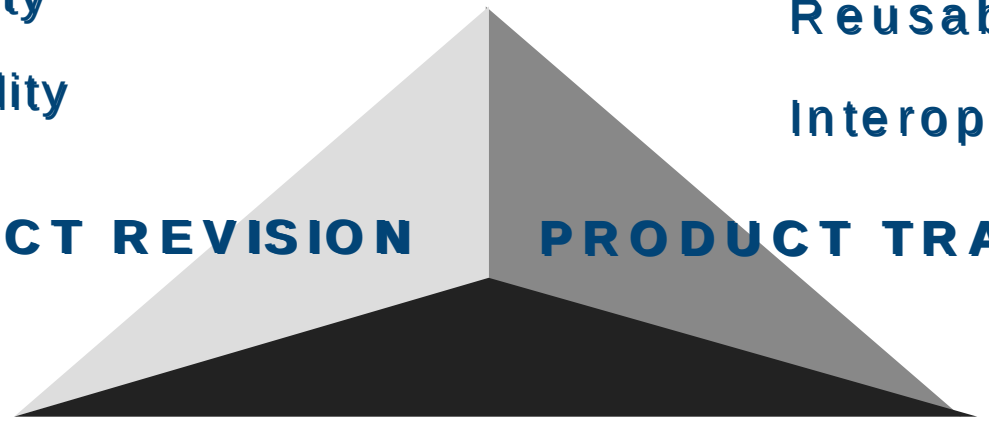


BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

A Framework for Product Metrics

McCall's Triangle of Quality



BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

A Framework for Product Metrics

Goal-Oriented Software Measurement

- A *measure* provides a quantitative indication of the extent, amount, dimension, capacity, or size of some attribute of a product or process
- The IEEE glossary defines a *metric* as “a quantitative measure of the degree to which a system, component, or process possesses a given attribute.”
- An *indicator* is a metric or combination of metrics that provide insight into the software process, a software project, or the product itself

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

A Framework for Product Metrics

Metrics Attributes

- *Simple and computable.* It should be relatively easy to learn how to derive the metric, and its computation should not demand inordinate effort or time
- *Empirically and intuitively persuasive.* The metric should satisfy the engineer's intuitive notions about the product attribute under consideration
- *Consistent and objective.* The metric should always yield results that are unambiguous.
- *Consistent in its use of units and dimensions.* The mathematical computation of the metric should use measures that do not lead to bizarre combinations of unit.
- *Programming language independent.* Metrics should be based on the analysis model, the design model, or the structure of the program itself.
- *Effective mechanism for quality feedback.* That is, the metric should provide a software engineer with information that can lead to a higher quality end product

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

A Framework for Product Metrics

Metrics Attributes

- *Simple and computable.* It should be relatively easy to learn how to derive the metric, and its computation should not demand inordinate effort or time
- *Empirically and intuitively persuasive.* The metric should satisfy the engineer's intuitive notions about the product attribute under consideration
- *Consistent and objective.* The metric should always yield results that are unambiguous.
- *Consistent in its use of units and dimensions.* The mathematical computation of the metric should use measures that do not lead to bizarre combinations of unit.
- *Programming language independent.* Metrics should be based on the analysis model, the design model, or the structure of the program itself.
- *Effective mechanism for quality feedback.* That is, the metric should provide a software engineer with information that can lead to a higher quality end product

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Metrics for the Requirements Model

- **Function-based metrics**: use the function point as a normalizing factor or as a measure of the “size” of the specification
- **Specification metrics**: used as an indication of quality by measuring number of requirements by type

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Metrics for the Requirements Model

Function-Based Metrics

- The *function point metric* (FP), first proposed by Albrecht [ALB79], can be used effectively as a means for measuring the functionality delivered by a system.
- Function points are derived using an empirical relationship based on countable (direct) measures of software's information domain and assessments of software complexity
- Information domain values are defined in the following manner:
  - number of external inputs (EIs)
  - number of external outputs (EOs)
  - number of external inquiries (EQs)
  - number of internal logical files (ILFs)
  - Number of external interface files (EIFs)

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Metrics for the Requirements Model

Function Points

Information Domain Value	Count	Weighting factor				
			simple	average	complex	
External Inputs (EIs)		3	3	4	6	=
External Outputs (EOs)		3	4	5	7	=
External Inquiries (EQs)		3	3	4	6	=
Internal Logical Files (ILFs)		3	7	10	15	=
External Interface Files (EIFs)		3	5	7	10	=
Count total →						

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Metrics for the Design Model

Architectural Design Metrics

- Architectural design metrics
  - Structural complexity = g(fan-out)
  - Data complexity = f(input & output variables, fan-out)
  - System complexity = h(structural & data complexity)
- HK metric: architectural complexity as a function of fan-in and fan-out
- Morphology metrics: a function of the number of modules and the number of interfaces between modules

Metrics for OO Design-I

Whitmire [Whi97] describes nine distinct and measurable characteristics of an OO design:

- **Size**
  - Size is defined in terms of four views: population, volume, length, and functionality
- **Complexity**
  - How classes of an OO design are interrelated to one another
- **Coupling**
  - The physical connections between elements of the OO design
- **Sufficiency**
  - "the degree to which an abstraction possesses the features required of it, or the degree to which a design component possesses features in its abstraction, from the point of view of the current application."

Metrics for OO Design-II

- **Completeness**
  - An indirect implication about the degree to which the abstraction or design component can be reused
- **Cohesion**
  - The degree to which all operations working together to achieve a single, well-defined purpose
- **Primitiveness**
  - Applied to both operations and classes, the degree to which an operation is atomic
- **Similarity**
  - The degree to which two or more classes are similar in terms of their structure, function, behavior, or purpose
- **Volatility**
  - Measures the likelihood that a change will occur

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Metrics for the Design Model

Class-Oriented Metrics

Proposed by Chidamber and Kemerer [Chi94]:

- weighted methods per class
- depth of the inheritance tree
- number of children
- coupling between object classes
- response for a class
- lack of cohesion in methods

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Metrics for the Design Model

Class-Oriented Metrics

Proposed by Lorenz and Kidd [Lor94]:

- class size
- number of operations overridden by a subclass
- number of operations added by a subclass
- specialization index

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Metrics for the Design Model

Class-Oriented Metrics

The MOOD Metrics Suite [Har98b]:

- Method inheritance factor
- Coupling factor
- Polymorphism factor

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Metrics for the Design Model

Class-Oriented Metrics

Proposed by Lorenz and Kidd [Lor94]:

- average operation size
- operation complexity
- average number of parameters per operation

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Metrics for the Design Model

Component-Level Design Metrics

- Cohesion metrics: a function of data objects and the locus of their definition
- Coupling metrics: a function of input and output parameters, global variables, and modules called
- Complexity metrics: hundreds have been proposed (e.g., cyclomatic complexity)

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Metrics for the Design Model

Interface Design Metrics

- **Layout appropriateness**: a function of layout entities, the geographic position and the "cost" of making transitions among entities

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Design Metrics for Web and Mobile Apps

- Does the user interface promote usability?
- Are the aesthetics of the WebApp appropriate for the application domain and pleasing to the user?
- Is the content designed in a manner that imparts the most information with the least effort?
- Is navigation efficient and straightforward?
- Has the WebApp architecture been designed to accommodate the special goals and objectives of WebApp users, the structure of content and functionality, and the flow of navigation required to use the system effectively?
- Are components designed in a manner that reduces procedural complexity and enhances the correctness, reliability and performance?

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Code Metrics

- **Halstead's Software Science**: a comprehensive collection of metrics all predicated on the number (count and occurrence) of operators and operands within a component or program
  - It should be noted that Halstead's "laws" have generated substantial controversy, and many believe that the underlying theory has flaws. However, experimental verification for selected programming languages has been performed (e.g. [FEL89]).

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Metrics for Testing

- Testing effort can also be estimated using metrics derived from Halstead measures
- Binder [Bin94] suggests a broad array of design metrics that have a direct influence on the "testability" of an OO system.
  - Lack of cohesion in methods (LCOM).
  - Percent public and protected (PAP).
  - Public access to data members (PAD).
  - Number of root classes (NOR).
  - Fan-in (FIN).
  - Number of children (NOC) and depth of the inheritance tree (DIT).

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Maintenance Metrics

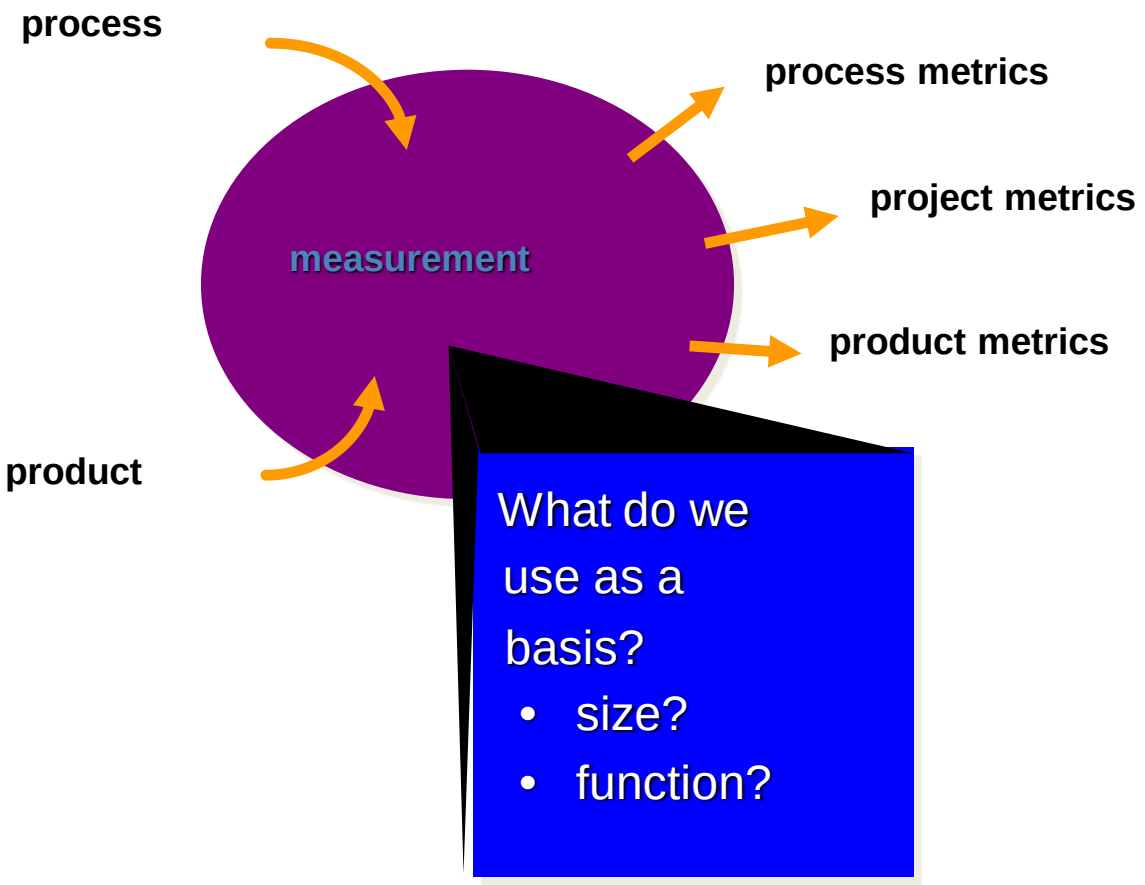
- IEEE Std. 982.1-1988 [IEE94] suggests a *software maturity index* (SMI) that provides an indication of the stability of a software product (based on changes that occur for each release of the product). The following information is determined:
  - M_T = the number of modules in the current release
  - F_c = the number of modules in the current release that have been changed
  - F_a = the number of modules in the current release that have been added
  - F_d = the number of modules from the preceding release that were deleted in the current release
- The software maturity index is computed in the following manner:
  - SMI = $[M_T - (F_a + F_c + F_d)]/M_T$
- As SMI approaches 1.0, the product begins to stabilize.

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Metrics in the Process and Project Domain

A Good Manager Measures



BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Metrics in the Process and Project Domain

Process Measurement

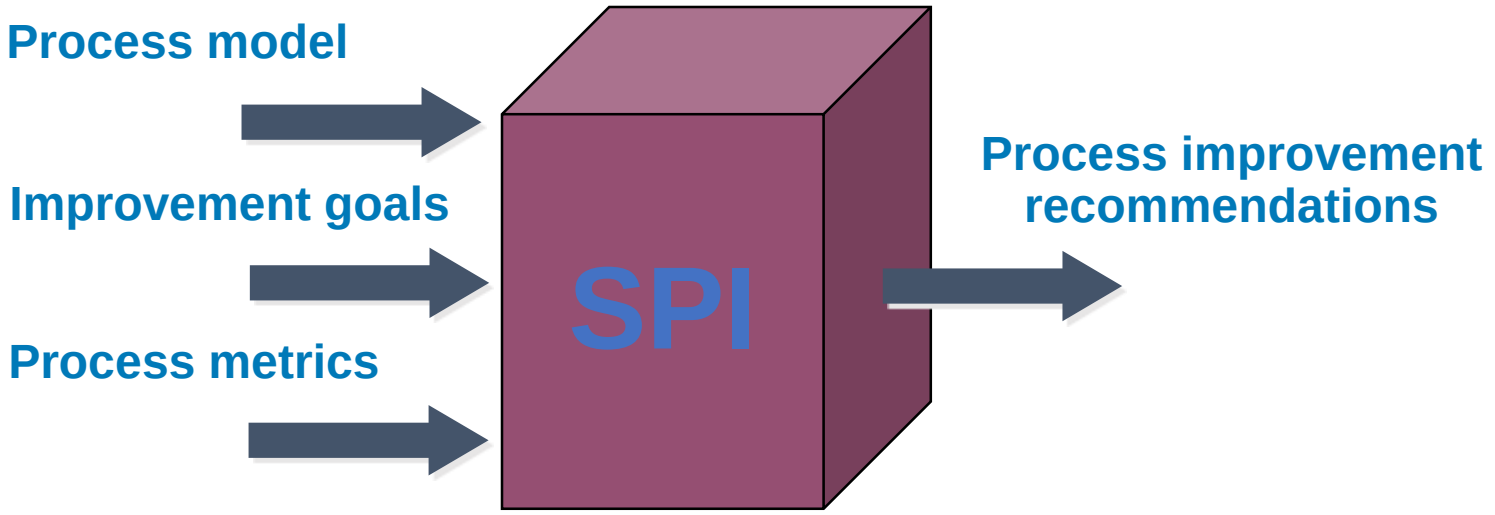
- We measure the efficacy of a software process indirectly.
  - That is, we derive a set of metrics based on the outcomes that can be derived from the process.
  - Outcomes include
    - measures of errors uncovered before release of the software
    - defects delivered to and reported by end-users
    - work products delivered (productivity)
    - human effort expended
    - calendar time expended
    - schedule conformance
    - other measures.
- We also derive process metrics by measuring the characteristics of specific software engineering tasks.

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Metrics in the Process and Project Domain

Software Process Improvement



BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Metrics in the Process and Project Domain

Process Metrics

- **Quality-related**
  - focus on quality of work products and deliverables
- **Productivity-related**
  - Production of work-products related to effort expended
- **Statistical SQA data**
  - error categorization & analysis
- **Defect removal efficiency**
  - propagation of errors from process activity to activity
- **Reuse data**
  - The number of components produced and their degree of reusability

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Metrics in the Process and Project Domain

Project Metrics

- used to minimize the development schedule by making the adjustments necessary to avoid delays and mitigate potential problems and risks
- used to assess product quality on an ongoing basis and, when necessary, modify the technical approach to improve quality.
- every project should measure:
  - *inputs*—measures of the resources (e.g., people, tools) required to do the work.
  - *outputs*—measures of the deliverables or work products created during the software engineering process.
  - *results*—measures that indicate the effectiveness of the deliverables.

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Metrics in the Process and Project Domain

Typical Project Metrics

- Effort/time per software engineering task
- Errors uncovered per review hour
- Scheduled vs. actual milestone dates
- Changes (number) and their characteristics
- Distribution of effort on software engineering tasks

Typical Size-Oriented Metrics

- errors per KLOC (thousand lines of code)
- defects per KLOC
- $ per LOC
- pages of documentation per KLOC
- errors per person-month
- errors per review hour
- LOC per person-month
- $ per page of documentation

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Typical Function-Oriented Metrics

- errors per FP (thousand lines of code)
- defects per FP
- $ per FP
- pages of documentation per FP
- FP per person-month

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Software Measurement

Comparing LOC and FP

Programming Language	LOC per Function point avg.	median	low	high
Ada	154	-	104	205
Assembler	337	315	91	694
C	162	109	33	704
C++	66	53	29	178
COBOL	77	77	14	400
Java	63	53	77	-
JavaScript	58	63	42	75
Perl	60	-	-	-
PL/1	78	67	22	263
Powerbuilder	32	31	11	105
SAS	40	41	33	49
Smalltalk	26	19	10	55
SQL	40	37	7	110
Visual Basic	47	42	16	158

Representative values developed by QSM

Software Measurement

Why Opt for FP?

- Programming language independent
- Used readily countable characteristics that are determined early in the software process
- Does not “penalize” inventive (short) implementations that use fewer LOC that other more clumsy versions
- Makes it easier to measure the impact of reusable components

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Software Measurement

Object-Oriented Metrics

- Number of scenario scripts (use-cases)
- Number of support classes (required to implement the system but are not immediately related to the problem domain)
- Average number of support classes per key class (analysis class)
- Number of subsystems (an aggregation of classes that support a function that is visible to the end-user of a system)

Software Measurement

WebApp Project Metrics

- Number of static Web pages (the end-user has no control over the content displayed on the page)
- Number of dynamic Web pages (end-user actions result in customized content displayed on the page)
- Number of internal page links (internal page links are pointers that provide a hyperlink to some other Web page within the WebApp)
- Number of persistent data objects
- Number of external systems interfaced
- Number of static content objects
- Number of dynamic content objects
- Number of executable functions

BINUS UNIVERSITY ONLINE LEARNING

People
Innovation
Excellence

Metrics for Software Quality

Measuring Quality

- Correctness — the degree to which a program operates according to specification
- Maintainability —the degree to which a program is amenable to change
- Integrity —the degree to which a program is impervious to outside attack
- Usability —the degree to which a program is easy to use

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Metrics for Software Quality

Defect Removal Efficiency

$$DRE = E\ /(E + D)$$

where:

***E* is the number of errors found before delivery of the software to the end-user**

***D* is the number of defects found after delivery.**

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Case Study

In the practical software development, the example of the software measurements are commonly used:

- TPS (Transaction Per Second)
- Throughput
- Availability of the system, example: 99% availability per months
- The monitoring tools is needed effectively to monitor the measurement

BINUS UNIVERSITY ONLINE LEARNING

People
Innovation
Excellence

Case Study

Please give the other examples the measurement and target in your organization software development.

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

References

- Pressman, R.S. (2015). ***Software Engineering : A Practioner's Approach. 8th ed***. McGraw-Hill Companies.Inc, Americas, New York. ISBN : 978 1 259 253157.
- Project Management

  http://www.pricesystems.com/resources/mf_risks_remedies_facts.asp
- Project Portfolio Management

  http://www.daptiv.com/index.htm
- SW Metrics

  http://www.spc.ca/resources/metrics/
- Function Point Measurement

  http://www.functionpoints.com
- SW Metrics service Estimation
  http://www.charismatek.com/_public4/html/services/pdf/service_estimate.pdf

BINUS UNIVERSITY ONLINE LEARNING

People
Innovation
Excellence

Q & A

BINUS
UNIVERSITY
ONLINE
LEARNING

People
Innovation
Excellence

Thank You

BINUS UNIVERSITY ONLINE LEARNING

People
Innovation
Excellence

Software Engineering

Topic 9

Estimation for Software Project and Project Scheduling

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Acknowledgement

These slides have been adapted from Pressman, R.S. (2015). *Software Engineering : A Practioner's Approach. 8th ed*. McGraw-Hill Companies.Inc, Americas, New York. ISBN : 978 1 259 253157. Chapter 33 and 34

BINUS UNIVERSITY ONLINE LEARNING

People
Innovation
Excellence

Learning Objectives

LO 4 : Analyze the software project management

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Contents

- Software Project Planning
- Software Scope
- Resources
- Project Estimation
- Empirical Estimation Models
- Estimation for OO Projects
- Estimation for Agile Projects
- The Make-Buy Decision
- Project Scheduling
- Earned Value Analysis (EVA)

BINUS UNIVERSITY ONLINE LEARNING

People
Innovation
Excellence

Software Project Planning

The overall goal of project planning is to establish a pragmatic strategy for controlling, tracking, and monitoring a complex technical project.

Why?

So the end result gets done on time, with quality!

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Software Project Planning

Project Planning Task Set-I

- Establish project scope
- Determine feasibility
- Analyze risks
  - Risk analysis is considered
- Define required resources
  - Determine require human resources
  - Define reusable software resources
  - Identify environmental resources

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Software Project Planning

Project Planning Task Set-II

- Estimate cost and effort
  - Decompose the problem
  - Develop two or more estimates using size, function points, process tasks or use-cases
  - Reconcile the estimates
- Develop a project schedule
  - Scheduling is considered
    - Establish a meaningful task set
    - Define a task network
    - Use scheduling tools to develop a timeline chart
    - Define schedule tracking mechanisms

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Software Project Planning

Estimation

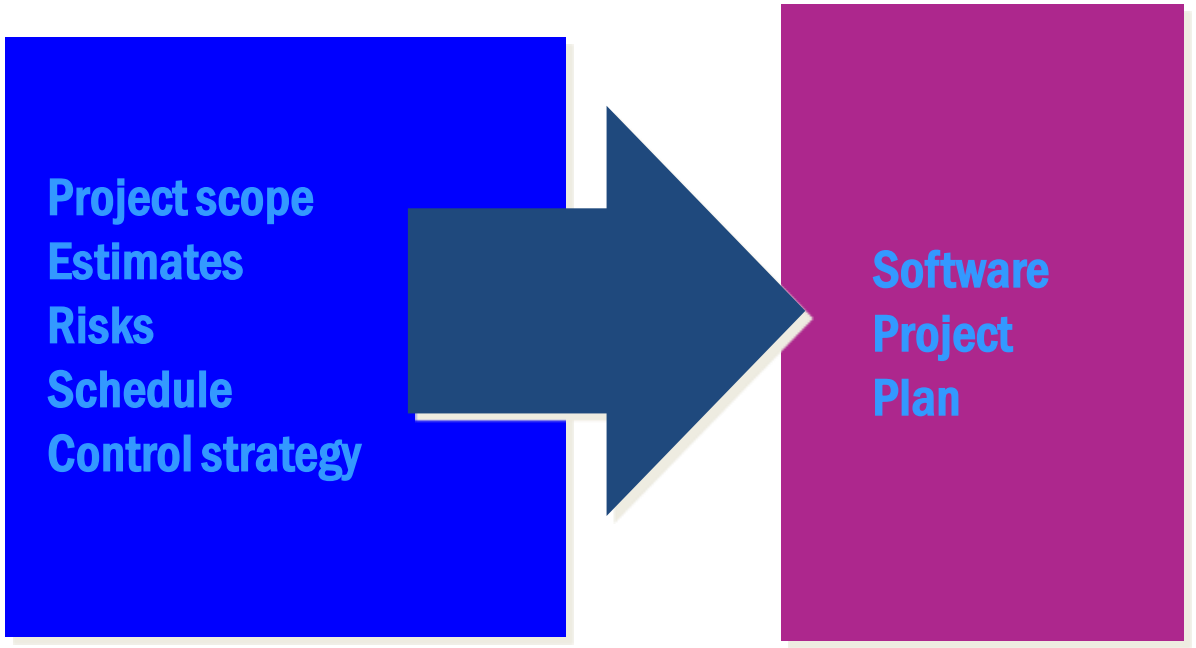
- Estimation of resources, cost, and schedule for a software engineering effort requires
  - experience
  - access to good historical information (metrics)
  - the courage to commit to quantitative predictions when qualitative information is all that exists
- Estimation carries inherent risk and this risk leads to uncertainty

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Software Project Planning

Write it Down!



BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Software Scope

To Understand Scope

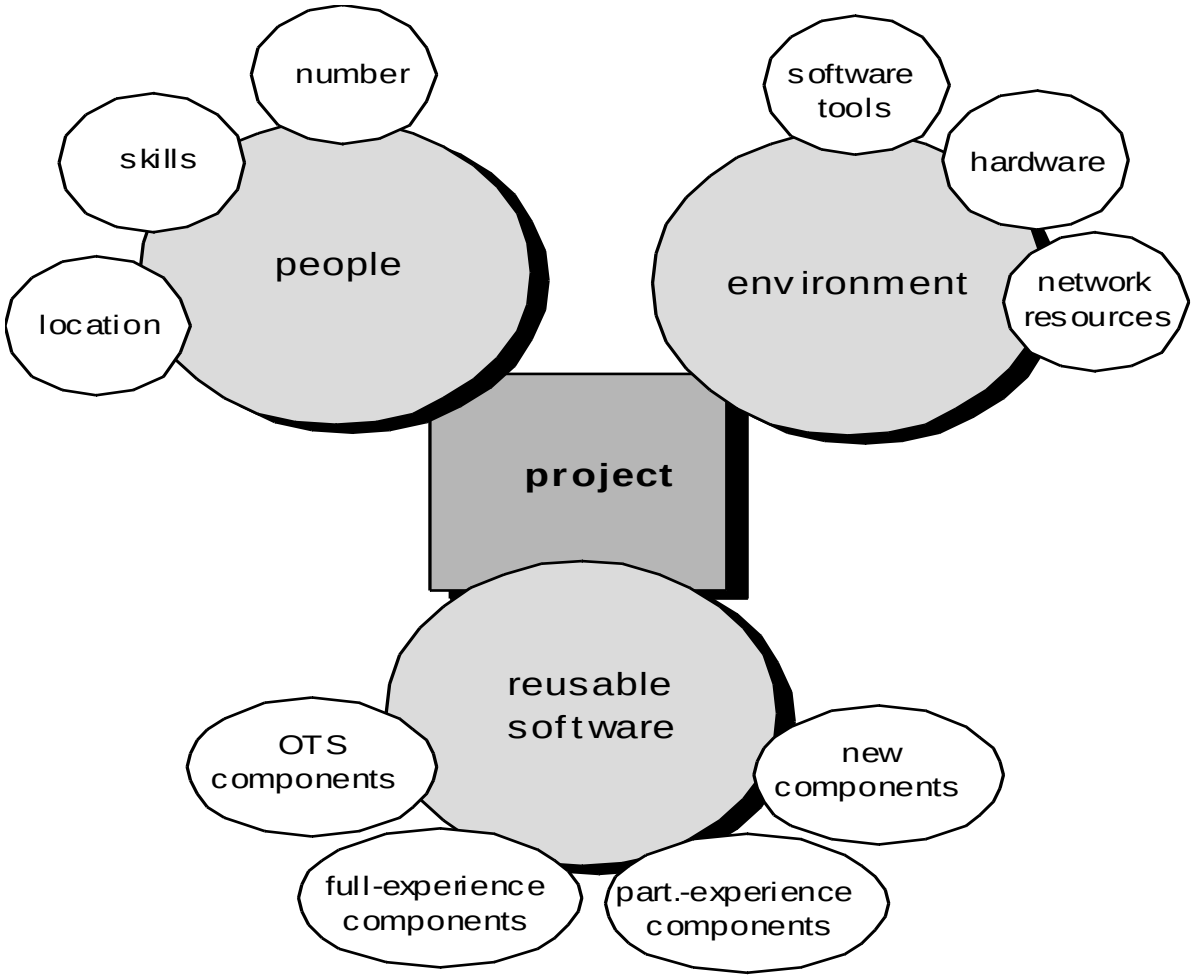
- Understand the customers needs
- understand the business context
- understand the project boundaries
- understand the customer's motivation
- understand the likely paths for change
- understand that ...

Even when you understand, nothing is guaranteed!

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Resources



BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Project Estimation

- Project scope must be understood
- Elaboration (decomposition) is necessary
- Historical metrics are very helpful
- At least two different techniques should be used
- Uncertainty is inherent in the process

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Project Estimation

Estimation Accuracy

- Predicated on ...
  - the degree to which the planner has properly estimated the size of the product to be built
  - the ability to translate the size estimate into human effort, calendar time, and dollars (a function of the availability of reliable software metrics from past projects)
  - the degree to which the project plan reflects the abilities of the software team
  - the stability of product requirements and the environment that supports the software engineering effort.

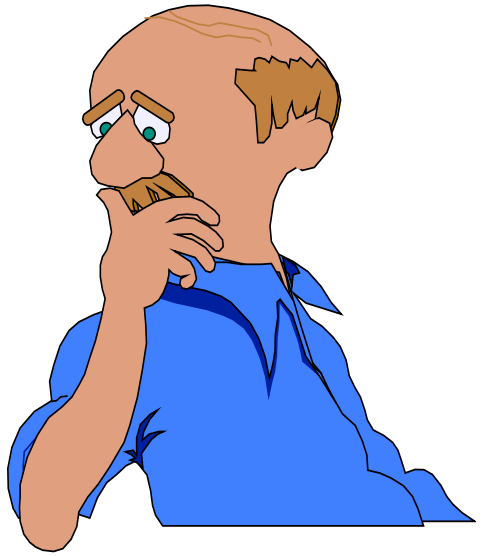
BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Project Estimation

Estimation Techniques

- Past (similar) project experience
- Conventional estimation techniques
  - task breakdown and effort estimates
  - size (e.g., FP) estimates
- Empirical models
- Automated tools

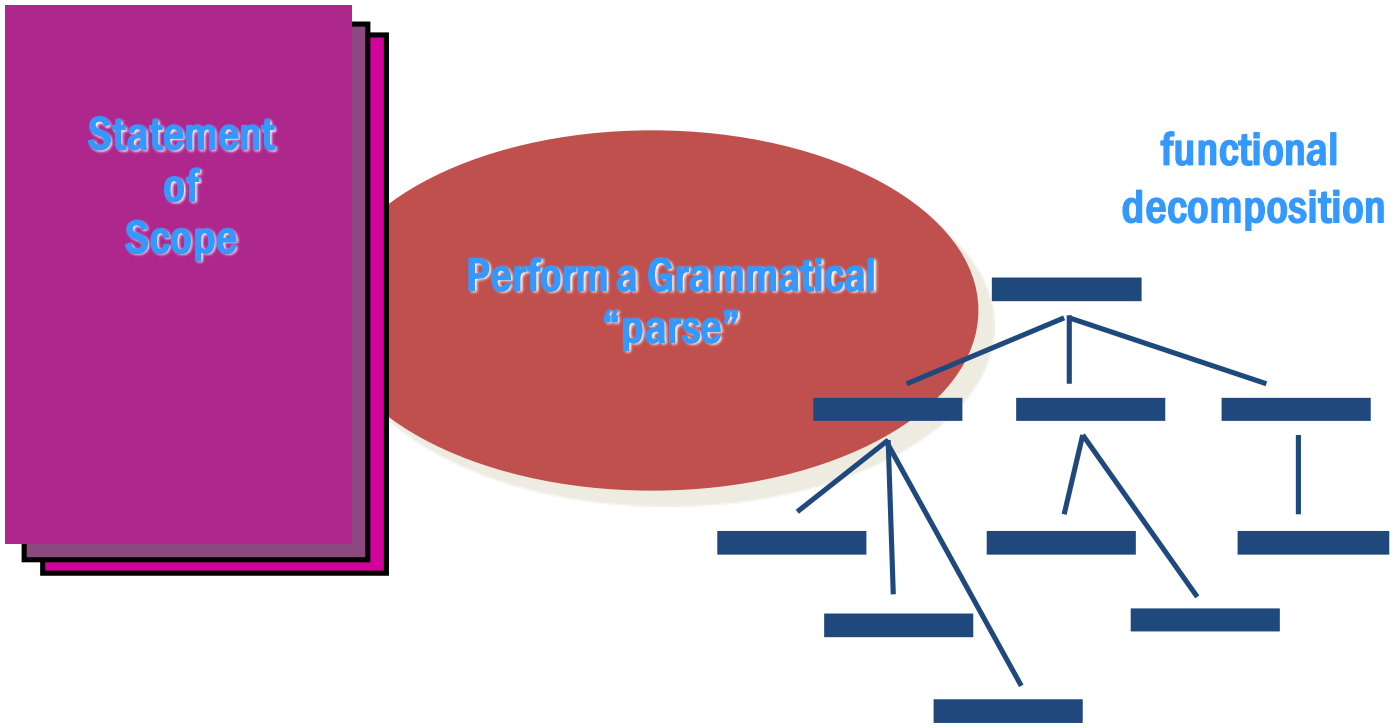


BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Project Estimation

Functional Decomposition



BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Project Estimation

Conventional Methods: LOC/FP Approach

- compute LOC/FP using estimates of information domain values
- use historical data to build estimates for the project

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Project Estimation

Example: LOC Approach

Function	Estimated LOC
user interface and control facilities (UICF)	2,300
two-dimensional geometric analysis (2DGA)	5,300
three-dimensional geometric analysis (3DGA)	6,800
database management (DBM)	3,350
computer graphics display facilities (CGDF)	4,950
peripheral control (PC)	2,100
design analysis modules (DAM)	8,400
estimated lines of code	33,200

Average productivity for systems of this type = 620 LOC/pm.

Burdened labor rate =$8000 per month, the cost per line of code is approximately $13.

Based on the LOC estimate and the historical productivity data, the total estimated project cost is $431,000 and the estimated effort is 54 person-months.

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Project Estimation

Example: FP Approach

Information Domain Value	opt.	likely	pess.	est. count	weight	FP-count
number of inputs	20	24	30	24	4	97
number of outputs	12	15	22	16	5	78
number of inquiries	16	22	28	22	5	88
number of files	4	4	5	4	10	42
number of external interfaces	2	2	3	2	7	15
count-total						321

The estimated number of FP is derived:

$$FP_{estimated} = count\text{-}total \times [0.65 + 0.01 \times \Sigma (F_i)]$$

$$FP_{estimated} = 375$$

organizational average productivity = 6.5 FP/pm.

burdened labor rate = $8000 per month, approximately $1230/FP.

Based on the FP estimate and the historical productivity data, total estimated project cost is $461,000 and estimated effort is 58 person-months.

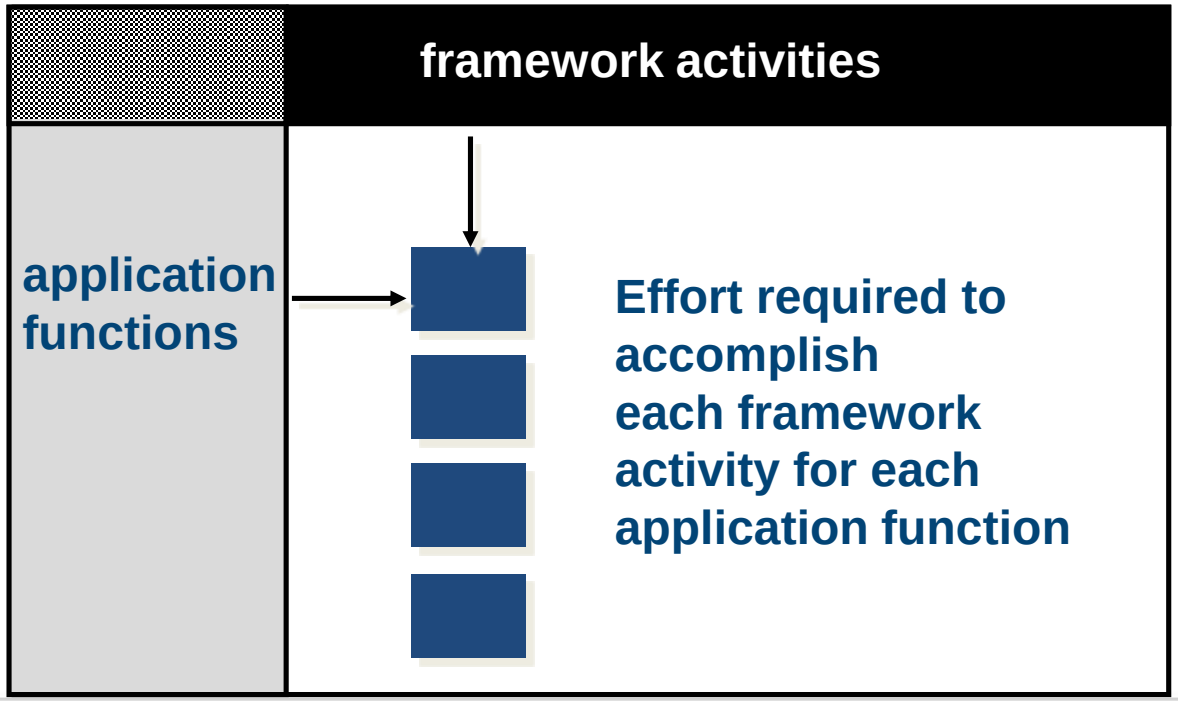
BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Project Estimation

Process-Based Estimation

Obtained from "process framework"



BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Project Estimation

Process-Based Estimation Example

Activity →	CC	Planning	Risk Analysis	Engineering		Construction Release		CE	Totals
Task →				analysis	design	code	test		
Function									
▼									
UICF				0.50	2.50	0.40	5.00	n/a	8.40
2DGA				0.75	4.00	0.60	2.00	n/a	7.35
3DGA				0.50	4.00	1.00	3.00	n/a	8.50
CGDF				0.50	3.00	1.00	1.50	n/a	6.00
DSM				0.50	3.00	0.75	1.50	n/a	5.75
PCF				0.25	2.00	0.50	1.50	n/a	4.25
DAM				0.50	2.00	0.50	2.00	n/a	5.00
Totals	0.25	0.25	0.25	3.50	20.50	4.50	16.50		46.00
% effort	1%	1%	1%	8%	45%	10%	36%		

CC = customer communication CE = customer evaluation

Based on an average burdened labor rate of $8,000 per month, the total estimated project cost is $368,000 and the estimated effort is 46 person-months.

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

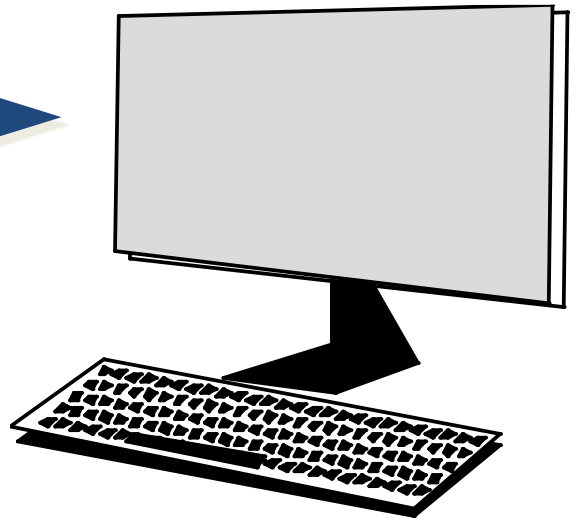
Project Estimation

Tool-Based Estimation

project characteristics

calibration factors

LOC/FP data

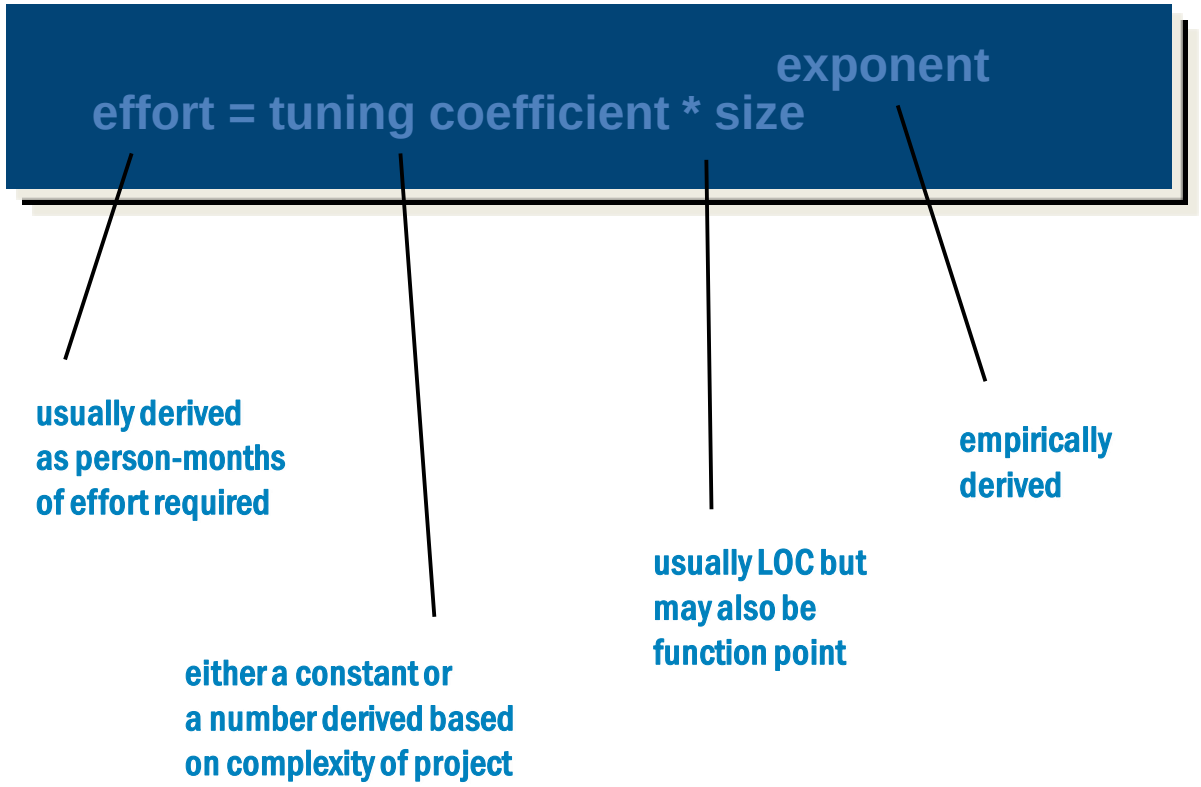


BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Empirical Estimation Models

General form:



BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Empirical Estimation Models

COCOMO-II

- COCOMO II is actually a hierarchy of estimation models that address the following areas:
  - *Application composition model.* Used during the early stages of software engineering, when prototyping of user interfaces, consideration of software and system interaction, assessment of performance, and evaluation of technology maturity are paramount.
  - *Early design stage model.* Used once requirements have been stabilized and basic software architecture has been established.
  - *Post-architecture-stage model.* Used during the construction of the software.

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Empirical Estimation Models

The Software Equation

A dynamic multivariable model

$$E = [LOC \times B^{0.333}/P]^3 \times (1/t^4)$$

where

E = effort in person-months or person-years
t = project duration in months or years
B = "special skills factor"
P = "productivity parameter"

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Estimation for OO Projects

1. Develop estimates using effort decomposition, FP analysis, and any other method that is applicable for conventional applications.
2. Using object-oriented requirements modeling, develop use-cases and determine a count.
3. From the analysis model, determine the number of key classes.
4. Categorize the type of interface for the application and develop a multiplier for support classes:

Interface type	Multiplier
No GUI	2.0
Text-based user interface	2.25
GUI	2.5
Complex GUI	3.0

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Estimation for OO Projects

5. Multiply the number of key classes (step 3) by the multiplier to obtain an estimate for the number of support classes.
6. Multiply the total number of classes (key + support) by the average number of work-units per class. Lorenz and Kidd suggest 15 to 20 person-days per class.
7. Cross check the class-based estimate by multiplying the average number of work-units per use-case

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

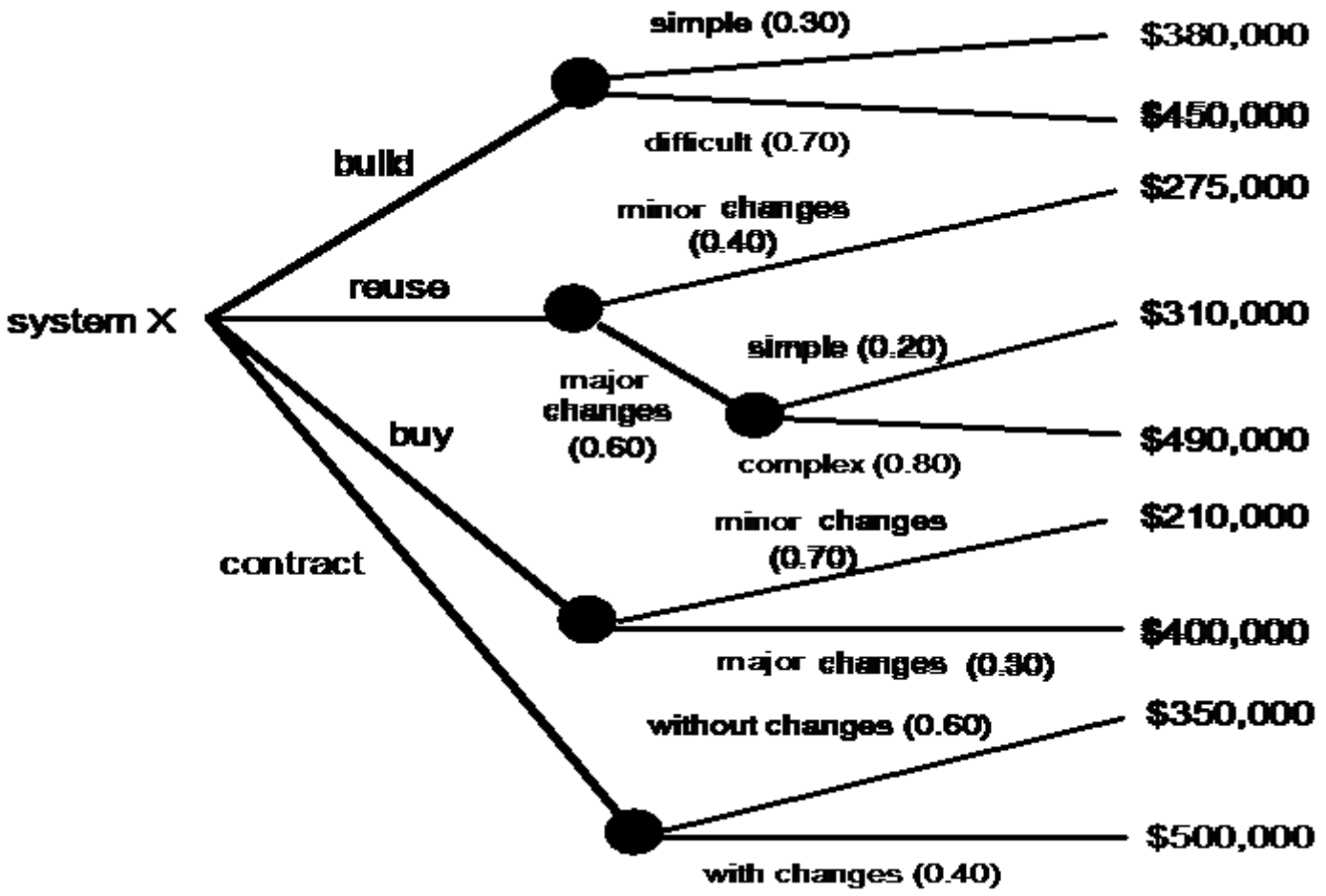
Estimation for Agile Projects

- Each user scenario (a mini-use-case) is considered separately for estimation purposes.
- The scenario is decomposed into the set of software engineering tasks that will be required to develop it.
- Each task is estimated separately. Note: estimation can be based on historical data, an empirical model, or "experience."
  - Alternatively, the 'volume' of the scenario can be estimated in LOC, FP or some other volume-oriented measure (e.g., use-case count).
- Estimates for each task are summed to create an estimate for the scenario.
  - Alternatively, the volume estimate for the scenario is translated into effort using historical data.
- The effort estimates for all scenarios that are to be implemented for a given software increment are summed to develop the effort estimate for the increment.

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

The Make-Buy Decision



BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

The Make-Buy Decision

Computing Expected Cost

$$\text{expected cost} = \sum (\text{path probability})_i \times (\text{estimated path cost})_i$$

For example, the expected cost to build is:

$$\text{expected cost}_{\text{build}} = 0.30\ (\$380K) + 0.70\ (\$450K)$$

$$= \$429\ K$$

similarly,

$$\text{expected cost}_{\text{reuse}} = \$382K$$

$$\text{expected cost}_{\text{buy}} = \$267K$$

$$\text{expected cost} = \$410K$$

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

The Make-Buy Decision

Computing Expected Cost

$$\text{expected cost} = \sum (\text{path probability})_i \times (\text{estimated path cost})_i$$

For example, the expected cost to build is:

$$\text{expected cost}_{\text{build}} = 0.30\ (\$380K) + 0.70\ (\$450K)$$

$$= \$429\ K$$

similarly,

$$\text{expected cost}_{\text{reuse}} = \$382K$$

$$\text{expected cost}_{\text{buy}} = \$267K$$

$$\text{expected cost} = \$410K$$

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Project Scheduling

Scheduling Principles

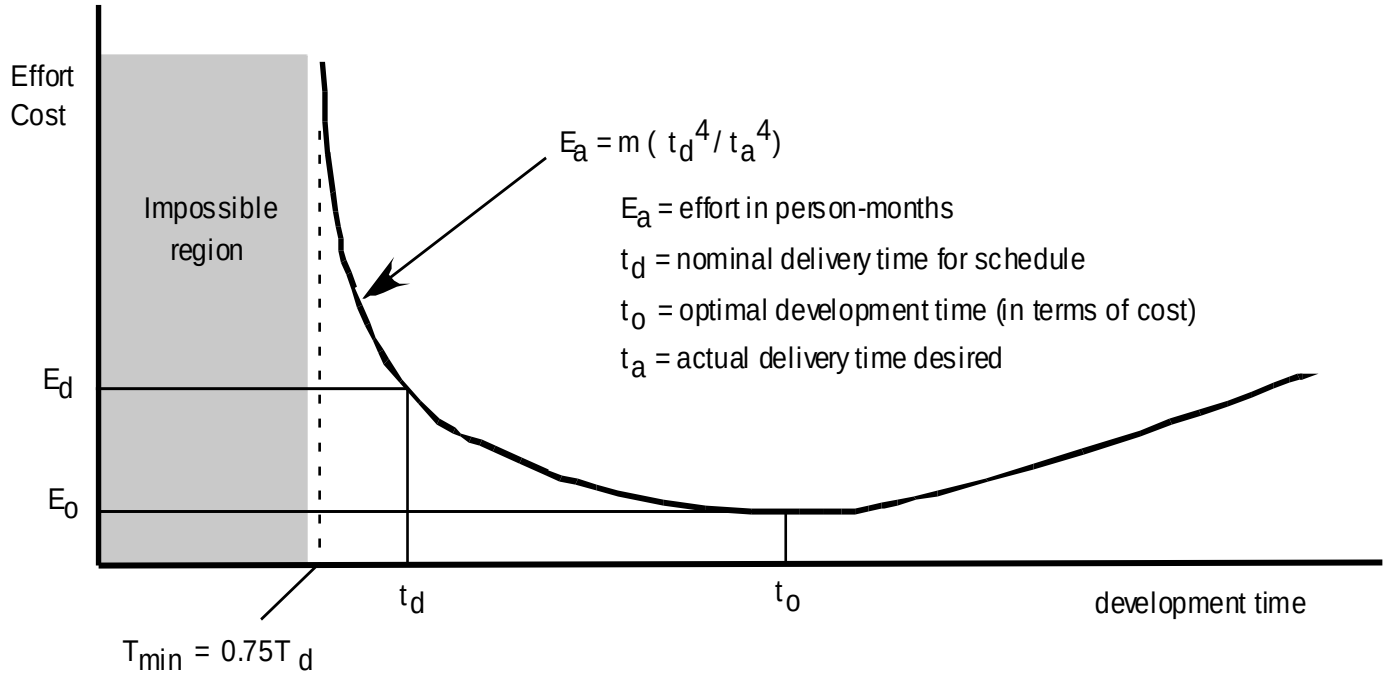
- Compartmentalization –define distinct tasks
- Interdependency –indicate task interrelationship
- Effort validation –be sure resources are available
- Defined responsibilities –people must be assigned
- Defined outcomes –each task must have an output
- Defined milestones –review for quality

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Project Scheduling

Effort and Delivery Time

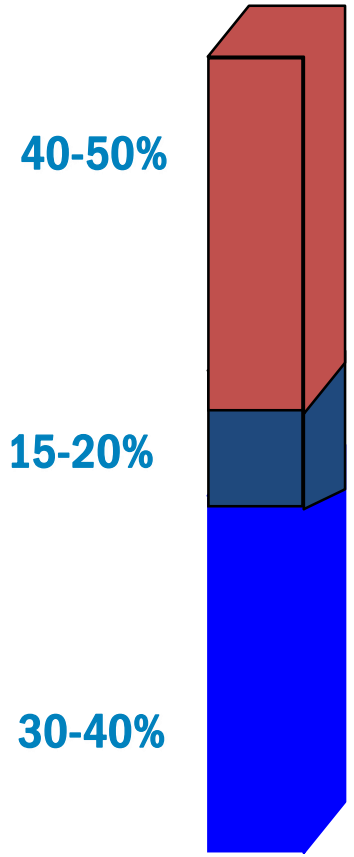


BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Project Scheduling

Effort Allocation



- "front end" activities
  - customer communication
  - analysis
  - design
  - review and modification
- construction activities
  - coding or code generation
- testing and installation
  - unit, integration
  - white-box, black box
  - regression

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Project Scheduling

Defining Task Sets

- determine type of project
- assess the degree of rigor required
- identify adaptation criteria
- select appropriate software engineering tasks

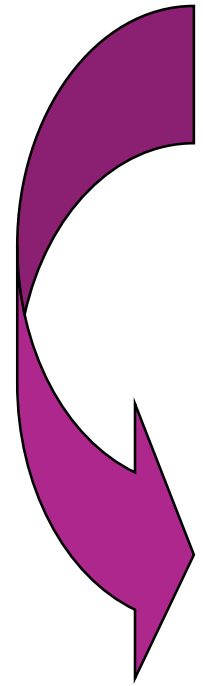
BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Project Scheduling

Task Set Refinement

1.1 Concept scoping determines the overall scope of the project.



is refined to

Task definition: Task 1.1 Concept Scoping
- 1.1.1 Identify need, benefits and potential customers;
- 1.1.2 Define desired output/control and input events that drive the application;
  - Begin Task 1.1.2
    - 1.1.2.1 FTR: Review written description of need
      FTR indicates that a formal technical review (Chapter 26) is to be conducted.
    - 1.1.2.2 Derive a list of customer visible outputs/inputs
    - 1.1.2.3 FTR: Review outputs/inputs with customer and revise as required;
  - endtask Task 1.1.2
- 1.1.3 Define the functionality/behavior for each major function;
  - Begin Task 1.1.3
    - 1.1.3.1 FTR: Review output and input data objects derived in task 1.1.2;
    - 1.1.3.2 Derive a model of functions/behaviors;
    - 1.1.3.3 FTR: Review functions/behaviors with customer and revise as required;
  - endtask Task 1.1.3
- 1.1.4 Isolate those elements of the technology to be implemented in software;
- 1.1.5 Research availability of existing software;
- 1.1.6 Define technical feasibility;
- 1.1.7 Make quick estimate of size;
- 1.1.8 Create a Scope Definition;

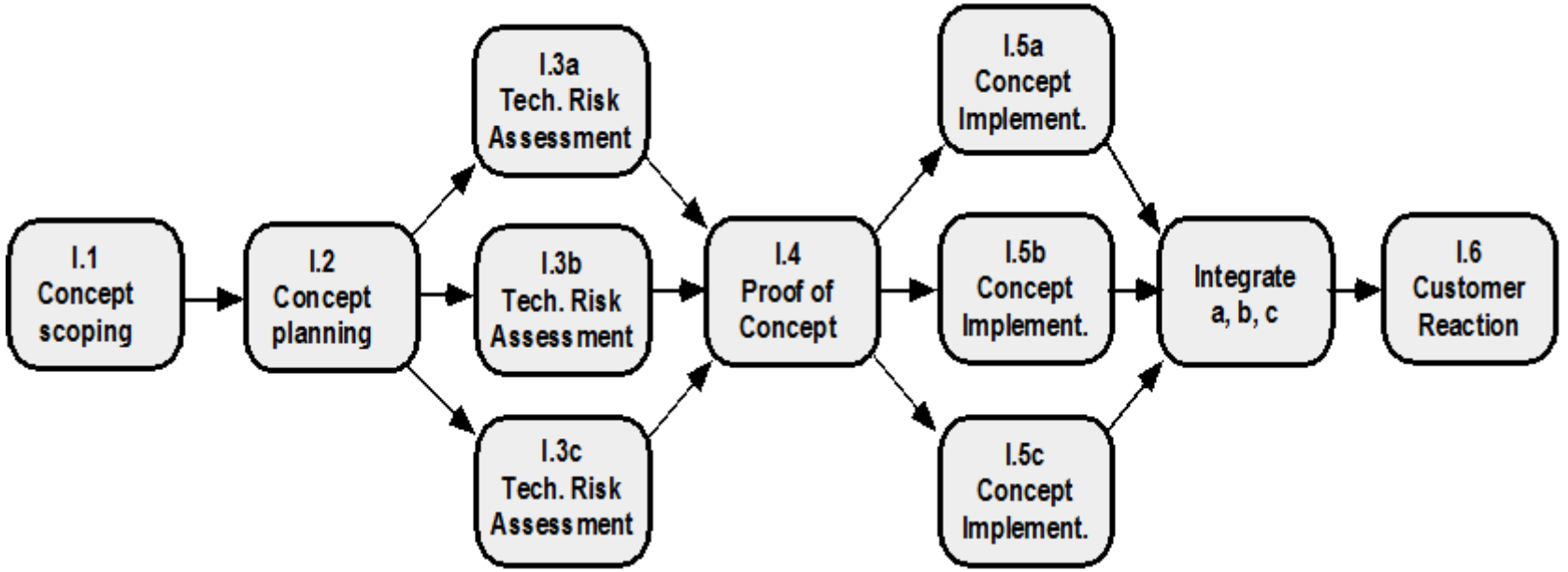
endTask definition: Task 1.1

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Project Scheduling

Define a Task Network

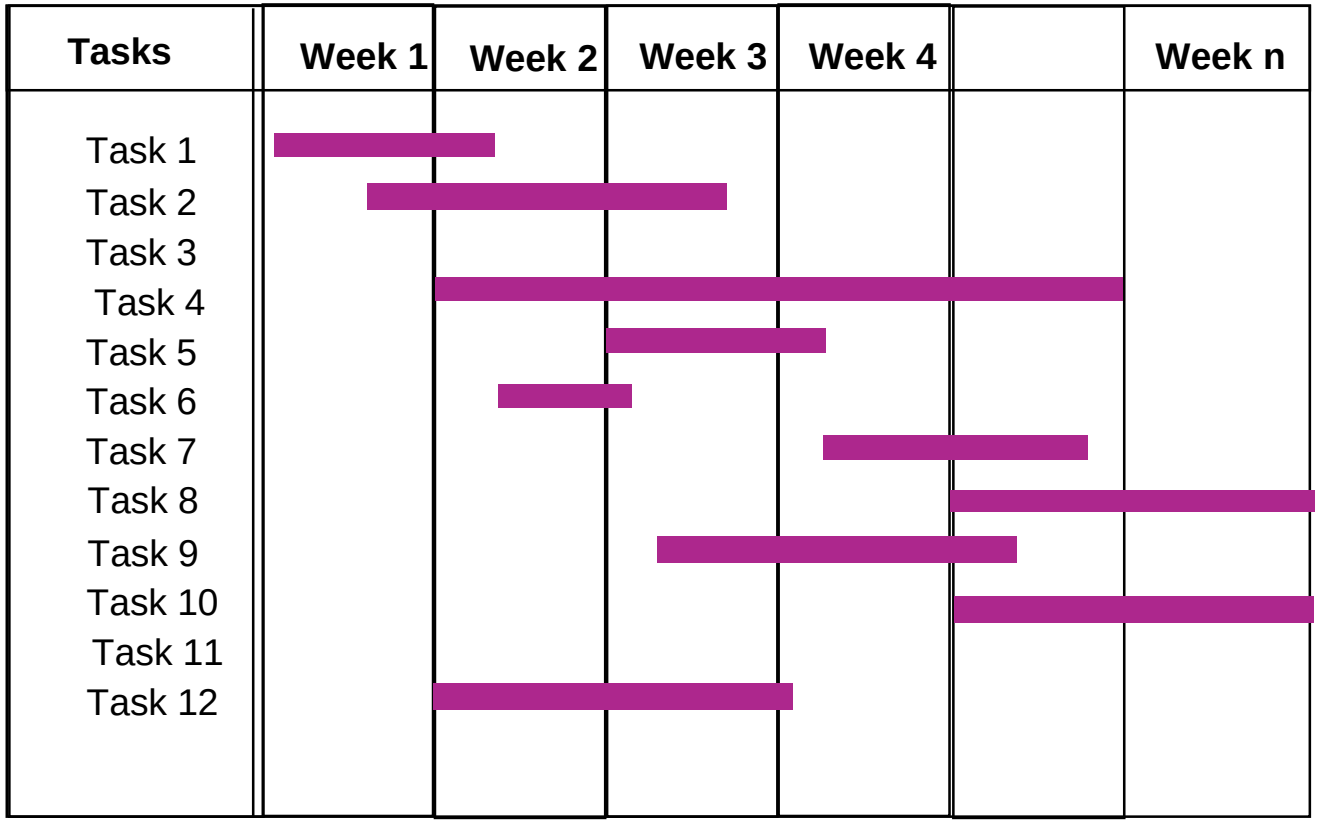


BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Project Scheduling

Timeline Charts

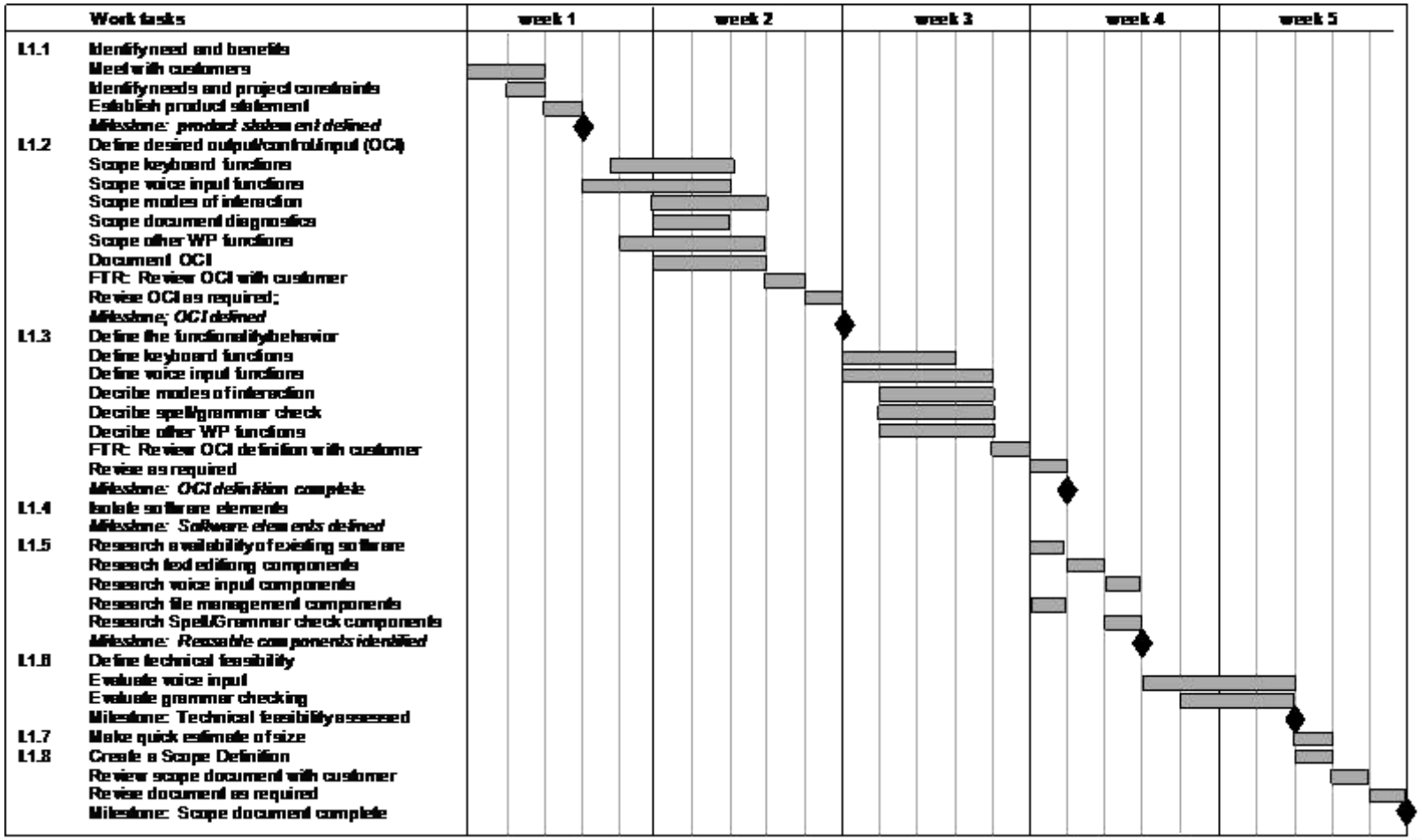


BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Project Scheduling

Use Automated Tools to Derive a Timeline Chart



BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Project Scheduling

Schedule Tracking

- conduct periodic project status meetings in which each team member reports progress and problems.
- evaluate the results of all reviews conducted throughout the software engineering process.
- determine whether formal project milestones have been accomplished by the scheduled date.
- compare actual start-date to planned start-date for each project task listed in the resource table.
- meet informally with practitioners to obtain their subjective assessment of progress to date and problems on the horizon.
- use earned value analysis to assess progress quantitatively.

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Project Scheduling

Progress on an OO Project-I

- *Technical milestone: OO analysis completed*
  - All classes and the class hierarchy have been defined and reviewed.
  - Class attributes and operations associated with a class have been defined and reviewed.
  - Class relationships have been established and reviewed.
  - A behavioral model has been created and reviewed.
  - Reusable classes have been noted.
- *Technical milestone: OO design completed*
  - The set of subsystems has been defined and reviewed.
  - Classes are allocated to subsystems and reviewed.
  - Task allocation has been established and reviewed.
  - Responsibilities and collaborations have been identified.
  - Attributes and operations have been designed and reviewed.
  - The communication model has been created and reviewed.

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Scheduling for WebApp and Mobile Projects

Seven increments for Web and MobileApps

1. Basic company and product information
2. Detailed product information and downloads
3. Product quotes and processing product orders
4. Space layout and security system design
5. Information and ordering of monitoring srvices
6. Online control of monitoring equipment
7. Accessing control information

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Scheduling for WebApp and Mobile Projects

Design the interface for the fourth increment

- Develop a sketch of the page layout for the space design page
- Review layout with stakeholders
- Design space layout navigation mechanisms
- Design “drawing board” layout
- Develop procedural details for the graphical wall layout function
- Develop procedural details for the wall length computation and display function
- Develop procedural details for the graphical window layout function
- Develop procedural details for the graphical door layout function
- Design mechanisms for selecting security system components

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Project Scheduling

Progress on an OO Project-II

- *Technical milestone: OO programming completed*
  - Each new class has been implemented in code from the design model.
  - Extracted classes (from a reuse library) have been implemented.
  - Prototype or increment has been built.
- *Technical milestone: OO testing*
  - The correctness and completeness of OO analysis and design models has been reviewed.
  - A class-responsibility-collaboration network has been developed and reviewed.
  - Test cases are designed and class-level tests have been conducted for each class.
  - Test cases are designed and cluster testing is completed and the classes are integrated.
  - System level tests have been completed.

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Earned Value Analysis (EVA)

- Earned value
  - is a measure of progress
  - enables us to assess the "percent of completeness" of a project using quantitative analysis rather than rely on a gut feeling
  - "provides accurate and reliable readings of performance from as early as 15 percent into the project." [Fle98]

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Earned Value Analysis (EVA)

Computing Earned Value-I

- The *budgeted cost of work scheduled* (BCWS) is determined for each work task represented in the schedule.
  - $BCWS_i$ is the effort planned for work task *i*.
  - To determine progress at a given point along the project schedule, the value of BCWS is the sum of the $BCWS_i$ values for all work tasks that should have been completed by that point in time on the project schedule.
- The BCWS values for all work tasks are summed to derive the *budget at completion,* BAC. Hence,

$$BAC = \sum (BCWS_k) \text{ for all tasks } k$$

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Earned Value Analysis (EVA)

Computing Earned Value-II

- Next, the value for *budgeted cost of work performed* (BCWP) is computed.
  - The value for BCWP is the sum of the BCWS values for all work tasks that have actually been completed by a point in time on the project schedule.
- "the distinction between the BCWS and the BCWP is that the former represents the budget of the activities that were planned to be completed and the latter represents the budget of the activities that actually were completed." [Wil99]
- Given values for BCWS, BAC, and BCWP, important progress indicators can be computed:
  - Schedule performance index, SPI = BCWP/BCWS
  - Schedule variance, SV = BCWP – BCWS
  - SPI is an indication of the efficiency with which the project is utilizing scheduled resources.

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Earned Value Analysis (EVA)

Computing Earned Value-III

- Percent scheduled for completion = BCWS/BAC
  - provides an indication of the percentage of work that should have been completed by time *t.*
- Percent complete = BCWP/BAC
  - provides a quantitative indication of the percent of completeness of the project at a given point in time, *t.*
- *Actual cost of work performed,* ACWP, is the sum of the effort actually expended on work tasks that have been completed by a point in time on the project schedule. It is then possible to compute
  - Cost performance index, CPI = BCWP/ACWP
  - Cost variance, CV = BCWP – ACWP

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Case Study

The use of project management tools

- In practical project, the use of project management tools are very effective to manage the project process including:
  - Scheduling
  - Resources
  - Estimation
  - Costing
  - etc

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

References

- Pressman, R.S. (2015). ***Software Engineering : A Practioner's Approach. 8th ed***. McGraw-Hill Companies.Inc, Americas, New York. ISBN : 978 1 259 253157.
- IT and sw estimation

  http://www.charismatek.com/_public4/html/services/service_estimation.htm
- Overview of COCOMO

  http://www.softstarsystems.com/overview.htm
- Function Point Language

  http://www.qsm.com/resources/function-point-languages-table/index.html
- PERT

  http://en.wikipedia.org/wiki/Program_Evaluation_and_Review_Technique

BINUS UNIVERSITY ONLINE LEARNING

People
Innovation
Excellence

Q & A

BINUS
UNIVERSITY
ONLINE
LEARNING

People
Innovation
Excellence

Thank You

BINUS UNIVERSITY ONLINE LEARNING

People
Innovation
Excellence

Software Engineering

Topic 10

Software Risk Management & Reengineering

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Acknowledgement

These slides have been adapted from Pressman, R.S. (2015). *Software Engineering : A Practioner's Approach. 8th ed*. McGraw-Hill Companies.Inc, Americas, New York. ISBN : 978 1 259 253157. Chapter 35 and 36

BINUS UNIVERSITY ONLINE LEARNING

People
Innovation
Excellence

Learning Objectives

LO 4 : Analyze the software project management and the proposed potential business project

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Contents

- Reactive versus Proactive Risk Strategies
- Software Risk
- Software Maintenance
- Business Process Reengineering
- Software Reengineering
- Reverse Engineering
- Forward Engineering

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Reactive versus Proactive Risk Strategies

Reactive Risk Management

- project team reacts to risks when they occur
- mitigation—plan for additional resources in anticipation of fire fighting
- fix on failure—resource are found and applied when the risk strikes
- crisis management—failure does not respond to applied resources and project is in jeopardy

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Reactive versus Proactive Risk Strategies

Proactive Risk Management

- formal risk analysis is performed
- organization corrects the root causes of risk
  - TQM concepts and statistical SQA
  - examining risk sources that lie beyond the bounds of the software
  - developing the skill to manage change

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Software Risk

Seven Principles

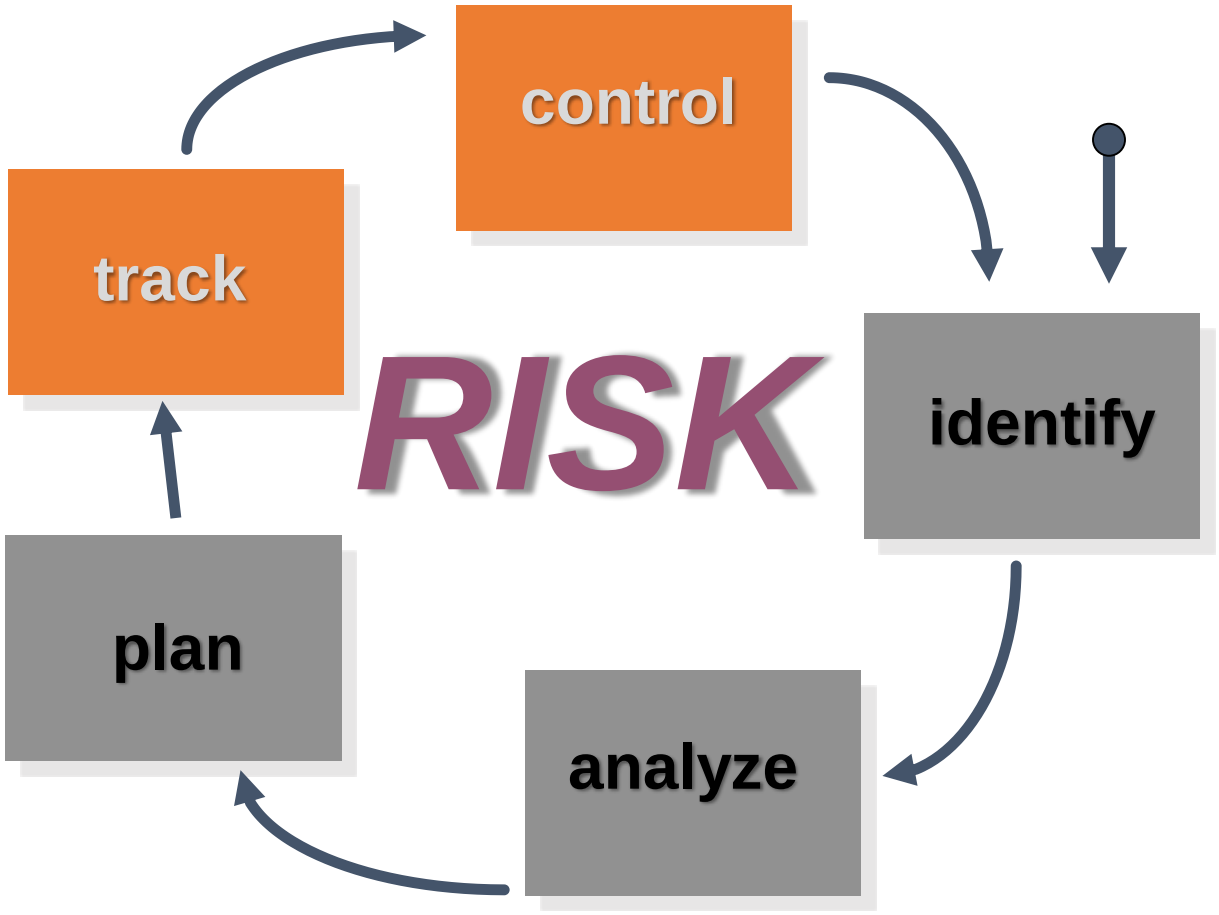
- **Maintain a global perspective**—view software risks within the context of system and the business problem
- **Take a forward-looking view**—think about the risks that may arise in the future; establish contingency plans
- **Encourage open communication**—if someone states a potential risk, don't discount it.
- **Integrate**—a consideration of risk must be integrated into the software process
- **Emphasize a continuous process**—the team must be vigilant throughout the software process, modifying identified risks as more information is known and adding new ones as better insight is achieved.
- **Develop a shared product vision**—if all stakeholders share the same vision of the software, it likely that better risk identification and assessment will occur.
- **Encourage teamwork**—the talents, skills and knowledge of all stakeholder should be pooled

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Software Risk

Risk Management Paradigm



BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Software Risk

Risk Identification

- ***Product size*** —risks associated with the overall size of the software to be built or modified.
- ***Business impact*** —risks associated with constraints imposed by management or the marketplace.
- ***Customer characteristics*** —risks associated with the sophistication of the customer and the developer's ability to communicate with the customer in a timely manner.
- ***Process definition*** —risks associated with the degree to which the software process has been defined and is followed by the development organization.
- ***Development environment*** —risks associated with the availability and quality of the tools to be used to build the product.
- ***Technology to be built*** —risks associated with the complexity of the system to be built and the "newness" of the technology that is packaged by the system.
- ***Staff size and experience*** —risks associated with the overall technical and project experience of the software engineers who will do the work.

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Software Risk

Assessing Project Risk-I

- Have top software and customer managers formally committed to support the project?
- Are end-users enthusiastically committed to the project and the system/product to be built?
- Are requirements fully understood by the software engineering team and their customers?
- Have customers been involved fully in the definition of requirements?
- Do end-users have realistic expectations?

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Software Risk

Assessing Project Risk-II

- Is project scope stable?
- Does the software engineering team have the right mix of skills?
- Are project requirements stable?
- Does the project team have experience with the technology to be implemented?
- Is the number of people on the project team adequate to do the job?
- Do all customer/user constituencies agree on the importance of the project and on the requirements for the system/product to be built?

Software Risk

- ***performance risk***—the degree of uncertainty that the product will meet its requirements and be fit for its intended use.
- ***cost risk***—the degree of uncertainty that the project budget will be maintained.
- ***support risk***—the degree of uncertainty that the resultant software will be easy to correct, adapt, and enhance.
- ***schedule risk***—the degree of uncertainty that the project schedule will be maintained and that the product will be delivered on time.

Risk Projection

- *Risk projection*, also called *risk estimation,* attempts to rate each risk in two ways
  - the likelihood or probability that the risk is real
  - the consequences of the problems associated with the risk, should it occur.
- The are four risk projection steps:
  - establish a scale that reflects the perceived likelihood of a risk
  - delineate the consequences of the risk
  - estimate the impact of the risk on the project and the product,
  - note the overall accuracy of the risk projection so that there will be no misunderstandings.

BINUS UNIVERSITY ONLINE LEARNING

People
Innovation
Excellence

Software Risk

Impact assessment

Category \ Components		Performance	Support	Cost	Schedule
Catastrophic	1	Failure to meet the requirement would result in mission failure		Failure results in increased costs and schedule delays with expected values in excess of $500K	
	2	Significant degradation to nonachievement of technical performance	Nonresponsive or unsupportable software	Significant financial shortages, budget overrun likely	Unachievable IOC
Critical	1	Failure to meet the requirement would degrade system performance to a point where mission success is questionable		Failure results in operational delays and/or increased costs with expected value of $100K to $500K	
	2	Some reduction in technical performance	Minor delays in software modifications	Some shortage of financial resources, possible overruns	Possible slippage in IOC
Marginal	1	Failure to meet the requirement would result in degradation of secondary mission		Costs, impacts, and/or recoverable schedule slips with expected value of $1K to $100K	
	2	Minimal to small reduction in technical performance	Responsive software support	Sufficient financial resources	Realistic, achievable schedule
Negligible	1	Failure to meet the requirement would create inconvenience or nonoperational impact		Error results in minor cost and/or schedule impact with expected value of less than $1K	
	2	No reduction in technical performance	Easily supportable software	Possible budget underrun	Early achievable IOC

Note: [1] The potential consequence of undetected software errors or faults.
[2] The potential consequence if the desired outcome is not achieved.

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Building the Risk Table

- Estimate the probability of occurrence
- Estimate the impact on the project on a scale of 1 to 4, where
  - 1 = catastrophic
  - 2 = critical
  - 3 = marginal
  - 4 = negligible
- sort the table by probability and impact

Software Risk

Sample risk table prior to sorting

Risks	Category	Probability	Impact	RMMM
Size estimate may be significantly low	PS	60%	2	
Larger number of users than planned	PS	30%	3	
Less reuse than planned	PS	70%	2	
End users resist system	BU	40%	3	
Delivery deadline will be tightened	BU	50%	2	
Funding will be lost	CU	40%	1	
Customer will change requirements	PS	80%	2	
Technology will not meet expectations	TE	30%	1	
Lack of training on tools	DE	80%	3	
Staff inexperienced	ST	30%	2	
Staff turnover will be high	ST	60%	2	
...				

Impact values:
1—catastrophic
2—critical
3—marginal
4—negligible

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Risk Exposure (Impact)

The overall *risk exposure*, RE, is determined using the following relationship [Hal98]:

$$RE = P \times C$$

where

P is the probability of occurrence for a risk, and

C is the cost to the project should the risk occur.

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Risk Exposure Example

- **Risk identification.** Only 70 percent of the software components scheduled for reuse will, in fact, be integrated into the application. The remaining functionality will have to be custom developed.
- **Risk probability.** 80% (likely).
- **Risk impact.** 60 reusable software components were planned. If only 70 percent can be used, 18 components would have to be developed from scratch (in addition to other custom software that has been scheduled for development). Since the average component is 100 LOC and local data indicate that the software engineering cost for each LOC is $14.00, the overall cost (impact) to develop the components would be 18 x 100 x 14 = $25,200.
- **Risk exposure.** RE = 0.80 x 25,200 ~ $20,200.

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Software Risk

Risk Mitigation, Monitoring, and Management

- Mitigation — how can we avoid the risk?
- Monitoring — what factors can we track that will enable us to determine if the risk is becoming more or less likely?
- Management — what contingency plans do we have if the risk becomes a reality?

BINUS UNIVERSITY ONLINE LEARNING

People
Innovation
Excellence

Software Risk

Risk information sheet

Risk information sheet			
Risk ID: P02-4-32	Date: 5/9/09	Prob: 80%	Impact: high
Description: Only 70 percent of the software components scheduled for reuse will, in fact, be integrated into the application. The remaining functionality will have to be custom developed.			
Refinement/context: Subcondition 1: Certain reusable components were developed by a third party with no knowledge of internal design standards. Subcondition 2: The design standard for component interfaces has not been solidified and may not conform to certain existing reusable components. Subcondition 3: Certain reusable components have been implemented in a language that is not supported on the target environment.			
Mitigation/monitoring: 1. Contact third party to determine conformance with design standards. 2. Press for interface standards completion; consider component structure when deciding on interface protocol. 3. Check to determine number of components in subcondition 3 category; check to determine if language support can be acquired.			
Management/contingency plan/trigger: *RE* computed to be $20,200. Allocate this amount within project contingency cost. Develop revised schedule assuming that 18 additional components will have to be custom built; allocate staff accordingly. Trigger: Mitigation steps unproductive as of 7/1/09			
Current status: 5/12/09: Mitigation steps initiated.			
Originator: D. Gagne		Assigned: B. Laster	

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Software Maintenance

- Software is released to end-users, and
  - within days, bug reports filter back to the software engineering organization.
  - within weeks, one class of users indicates that the software must be changed so that it can accommodate the special needs of their environment.
  - within months, another corporate group who wanted nothing to do with the software when it was released, now recognizes that it may provide them with unexpected benefit. They'll need a few enhancements to make it work in their world.
- All of this work is *software maintenance*

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Maintainable Software

- Maintainable software exhibits effective modularity
- It makes use of design patterns that allow ease of understanding.
- It has been constructed using well-defined coding standards and conventions, leading to source code that is self-documenting and understandable.
- It has undergone a variety of quality assurance techniques that have uncovered potential maintenance problems before the software is released.
- It has been created by software engineers who recognize that they may not be around when changes must be made.
    - *Therefore, the design and implementation of the software must "assist" the person who is making the change*

BINUS UNIVERSITY ONLINE LEARNING

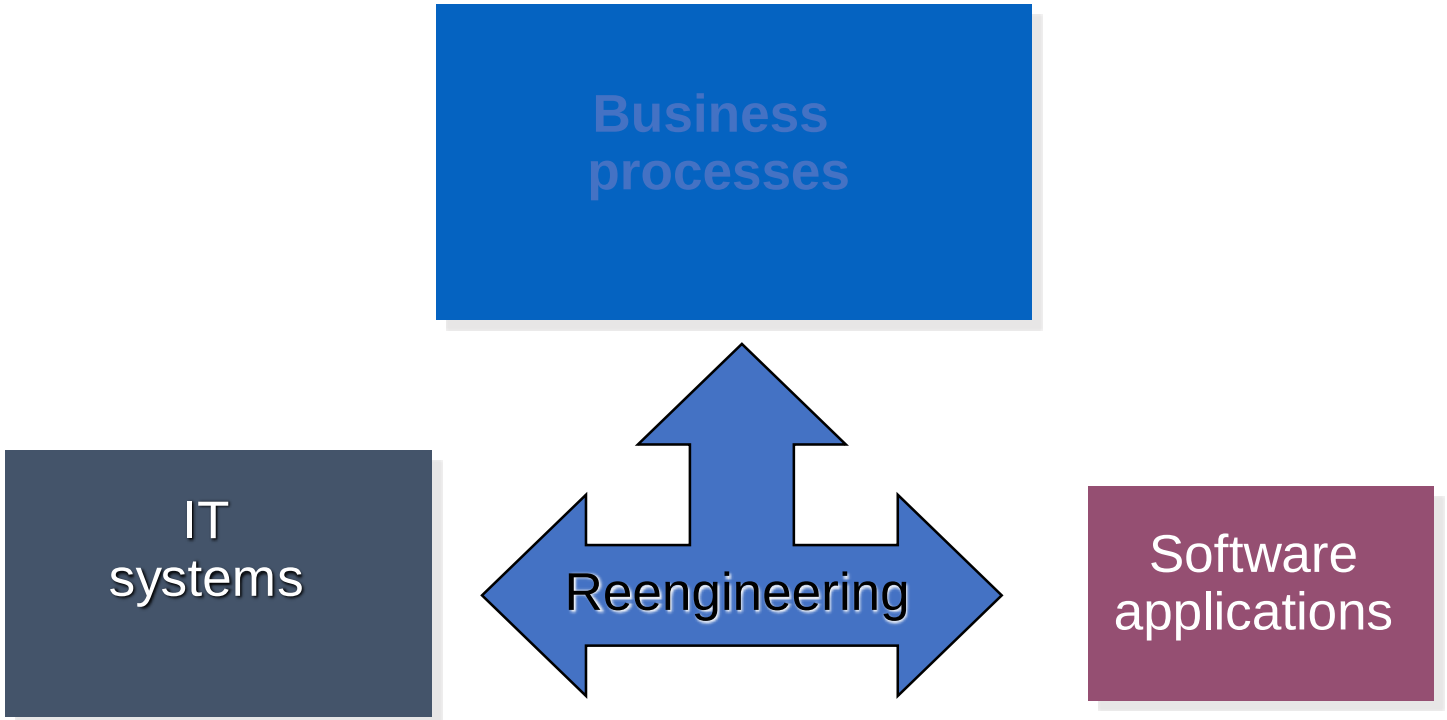
People Innovation Excellence

Software Supportability

- "the capability of supporting a software system over its whole product life.
  - This implies satisfying any necessary needs or requirements, but also the provision of equipment, support infrastructure, additional software, facilities, manpower, or any other resource required to maintain the software operational and capable of satisfying its function." [SSO08]
- The software should contain facilities to assist support personnel when a defect is encountered in the operational environment (and make no mistake, defects *will* be encountered).
- Support personnel should have access to a database that contains records of all defects that have already been encountered—their characteristics, cause, and cure.

Business Process Reengineering

Reengineering



BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

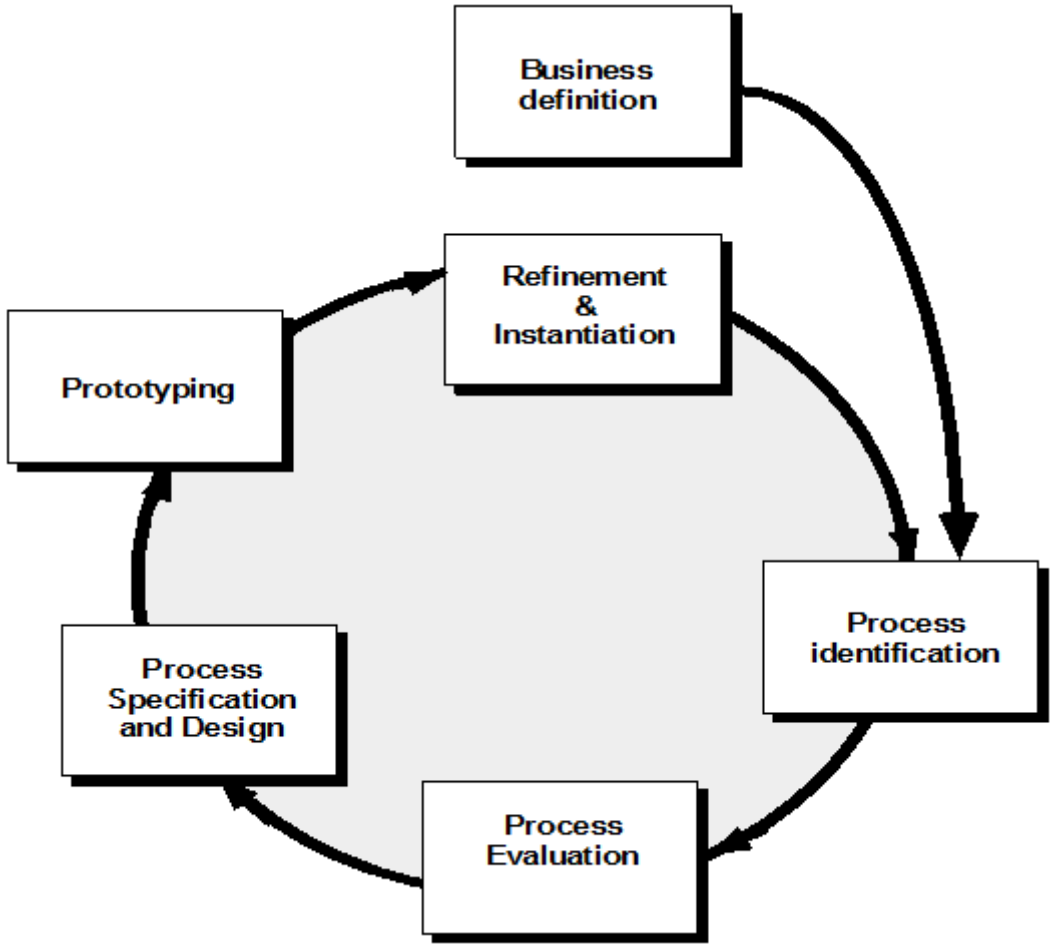
Business Process Reengineering

- **Business definition.** Business goals are identified within the context of four key drivers: cost reduction, time reduction, quality improvement, and personnel development and empowerment.
- **Process identification.** Processes that are critical to achieving the goals defined in the business definition are identified.
- **Process evaluation.** The existing process is thoroughly analyzed and measured.
- **Process specification and design.** Based on information obtained during the first three BPR activities, use-cases are prepared for each process that is to be redesigned.
- **Prototyping.** A redesigned business process must be prototyped before it is fully integrated into the business.
- **Refinement and instantiation.** Based on feedback from the prototype, the business process is refined and then instantiated within a business system.

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Business Process Reengineering



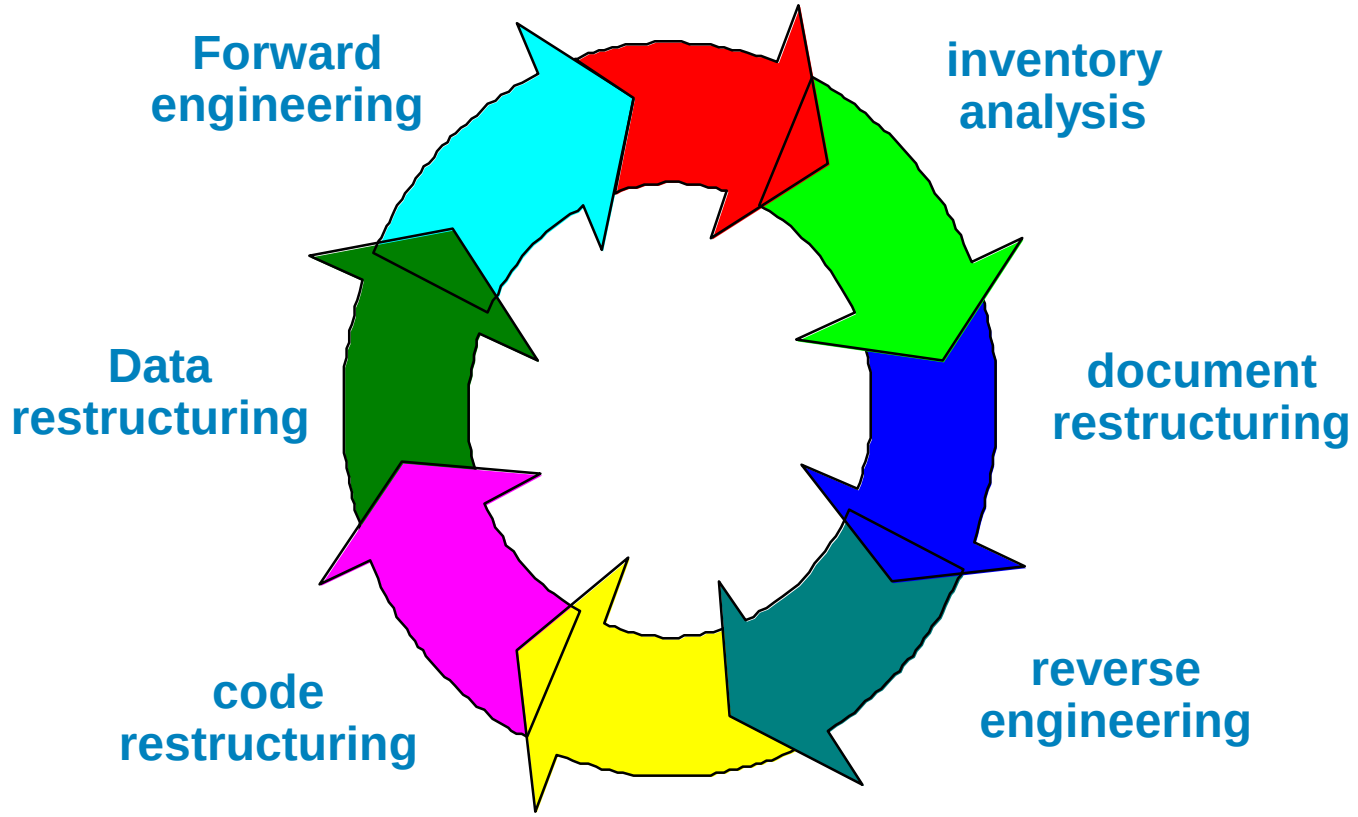
BPR Principles

- Organize around outcomes, not tasks.
- Have those who use the output of the process perform the process.
- Incorporate information processing work into the real work that produces the raw information.
- Treat geographically dispersed resources as though they were centralized.
- Link parallel activities instead of integrated their results. When different
- Put the decision point where the work is performed, and build control into the process.
- Capture data once, at its source.

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Software Reengineering



BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Inventory Analysis

- build a table that contains all applications
- establish a list of criteria, e.g.,
  - name of the application
  - year it was originally created
  - number of substantive changes made to it
  - total effort applied to make these changes
  - date of last substantive change
  - effort applied to make the last change
  - system(s) in which it resides
  - applications to which it interfaces, ...
- analyze and prioritize to select candidates for reengineering

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

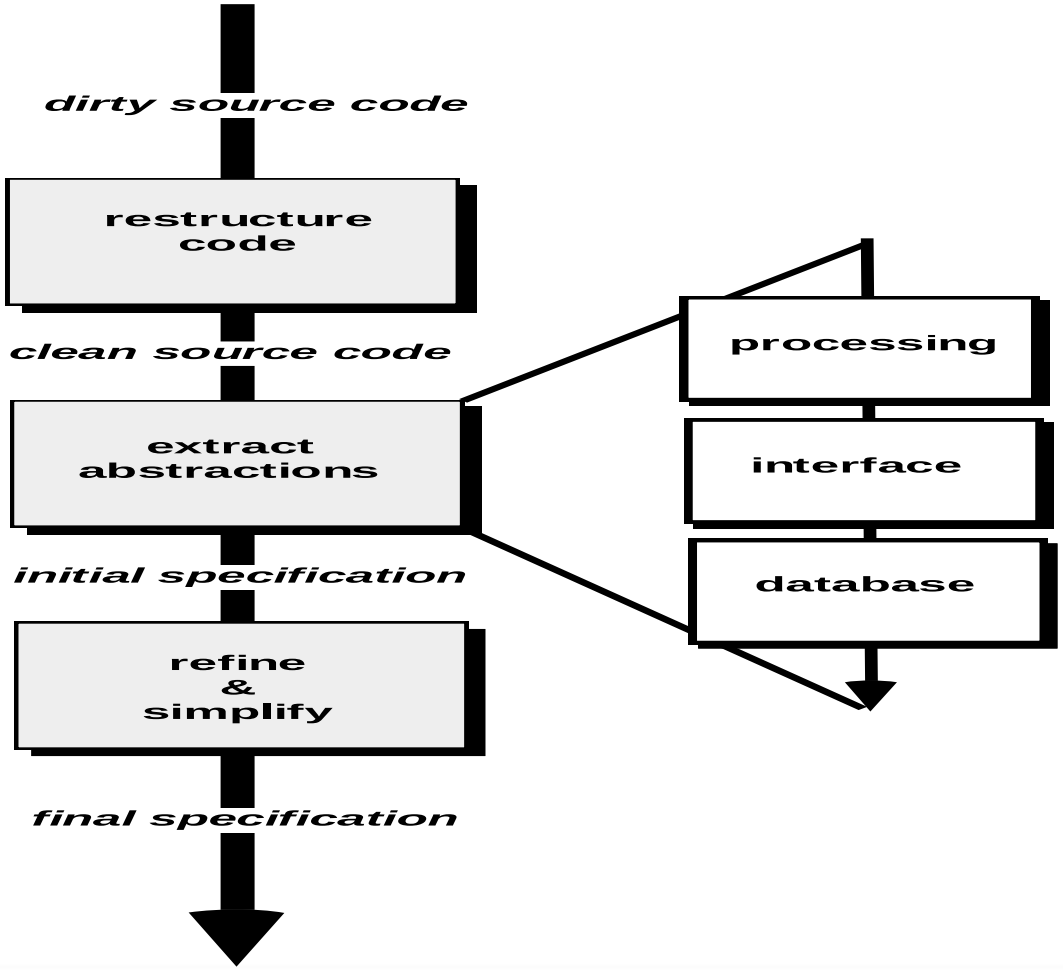
Document Restructuring

- Weak documentation is the trademark of many legacy systems.
- But what do we do about it? What are our options?
- Options …
  - *Creating documentation is far too time consuming.* If the system works, we'll live with what we have. In some cases, this is the correct approach.
  - *Documentation must be updated, but we have limited resources.* We'll use a "document when touched" approach. It may not be necessary to fully redocument an application.
  - *The system is business critical and must be fully redocumented.* Even in this case, an intelligent approach is to pare documentation to an essential minimum.

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Reverse Engineering



BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Code Restructuring

- Source code is analyzed using a restructuring tool.
- Poorly design code segments are redesigned
- Violations of structured programming constructs are noted and code is then restructured (this can be done automatically)
- The resultant restructured code is reviewed and tested to ensure that no anomalies have been introduced
- Internal code documentation is updated.

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Data Restructuring

- Unlike code restructuring, which occurs at a relatively low level of abstraction, data structuring is a full-scale reengineering activity
- In most cases, data restructuring begins with a reverse engineering activity.
  - Current data architecture is dissected and necessary data models are defined.
  - Data objects and attributes are identified, and existing data structures are reviewed for quality.
  - When data structure is weak (e.g., flat files are currently implemented, when a relational approach would greatly simplify processing), the data are reengineered.
- Because data architecture has a strong influence on program architecture and the algorithms that populate it, changes to the data will invariably result in either architectural or code-level changes.

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Forward Engineering

1. The cost to maintain one line of source code may be 20 to 40 times the cost of initial development of that line.
2. Redesign of the software architecture (program and/or data structure), using modern design concepts, can greatly facilitate future maintenance.
3. Because a prototype of the software already exists, development productivity should be much higher than average.
4. The user now has experience with the software. Therefore, new requirements and the direction of change can be ascertained with greater ease.
5. CASE tools for reengineering will automate some parts of the job.
6. A complete software configuration (documents, programs and data) will exist upon completion of preventive maintenance.

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Economics of Reengineering-I

- A cost/benefit analysis model for reengineering has been proposed by Sneed [Sne95]. Nine parameters are defined:
  - P_1 = current annual maintenance cost for an application.
  - P_2 = current annual operation cost for an application.
  - P_3 = current annual business value of an application.
  - P_4 = predicted annual maintenance cost after reengineering.
  - P_5 = predicted annual operations cost after reengineering.
  - P_6 = predicted annual business value after reengineering.
  - P_7 = estimated reengineering costs.
  - P_8 = estimated reengineering calendar time.
  - P_9 = reengineering risk factor (P_9 = 1.0 is nominal).
  - L = expected life of the system.

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Economics of Reengineering-II

- The cost associated with continuing maintenance of a candidate application (i.e., reengineering is not performed) can be defined as

$$C_{maint} = [P_3 - (P_1 + P_2)] \times L$$

- The costs associated with reengineering are defined using the following relationship:

$$C_{reeng} = [P_6 - (P_4 + P_5) \times (L - P_8) - (P_7 \times P_9)]$$

- Using the costs presented in equations above, the overall benefit of reengineering can be computed as

$$\text{cost benefit} = C_{reeng} - C_{maint}$$

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Case Study (1)

Risk and Issues

- Some people have a confusion between risk and issues.
  - Issue happen in the current condition
    - Example: the application have some bugs during the report calculation
  - Risks will happen in the future
    - Example: Since there is no backup server, there is a possibility to have the system unavailability if the current production server is down

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Risk and Issues

Case Study (2)

Issue

Risk

→ Issue Management (Problem solving, Corrective action)

→ Risk Management (Preventive action)

Case Study (3)

- Some people have a confusion between risk and issues.
  - Issue happen in the current condition
    - Example: the application have some bugs during the report calculation
  - Risks will happen in the future
    - Example: Since there is no backup server, there is a possibility to have the system unavailability if the current production server is down

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

References

- Pressman, R.S. (2015). ***Software Engineering : A Practioner's Approach. 8th ed***. McGraw-Hill Companies.Inc, Americas, New York. ISBN : 978 1 259 253157.
- Risk Management & operational Risk

  http://www.grafp.com/products/risk-manage.html
- Sw maintenance and Reengineering

  http://www.csse.monash.edu.au/~jonmc/CSE2305/Topics/13.25.SWEng4/html/text.html

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

References

- Pressman, R.S. (2015). ***Software Engineering : A Practioner's Approach. 8th ed***. McGraw-Hill Companies.Inc, Americas, New York. ISBN : 978 1 259 253157.
- Risk Management & operational Risk

  http://www.grafp.com/products/risk-manage.html
- Sw maintenance and Reengineering

  http://www.csse.monash.edu.au/~jonmc/CSE2305/Topics/13.25.SWEng4/html/text.html

BINUS UNIVERSITY ONLINE LEARNING

People
Innovation
Excellence

Q & A

BINUS
UNIVERSITY
ONLINE
LEARNING

People
Innovation
Excellence

Thank You