



Software engineering revisited

Vu Thi Huong Giang



@2016-2017



Covered topics

- Software engineering
- Software quality



Objectives

- After this lesson, students will be able to:
 - Recall the main concepts about of the software engineering domain.
 - Explain the ways to deal with change and complexity in software production.
 - Demonstrate the quality of a given software and its measurement.



I. SOFTWARE ENGINEERING

1. FAQs
2. Deal with complexity & changes
3. Knowledge Area & Units

1.1. What is software?



Software

- Software = computer programs + associated documentation (e.g. requirements, design models and user manuals)
- Software products may be
 - generic - developed to be sold on open market to any customers
 - customized - developed for a particular customer according to their specification
- New software can be created by
 - developing new programs
 - configuring generic software systems
 - reusing existing softwares



1.2. What is software engineering?



Software

Large / complex
Given budget
Given deadline
Built by teams
Changeable
High quality

Software engineering

Software engineering is concerned with all aspect of software production:

- technical processes of software development activities
- development tools, methods, and theories to support software production





1.3. 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.



1.4. What is a software process model?

- A simplified representation of a software process, presented from a specific perspective
- Examples of process perspectives:
 - Workflow perspective: sequence of activities
 - Data flow perspective: information flow
 - Role/action perspective: who does what



Generic process models: waterfall

Requirement analysis and
specification



Design and specification



Code and module testing

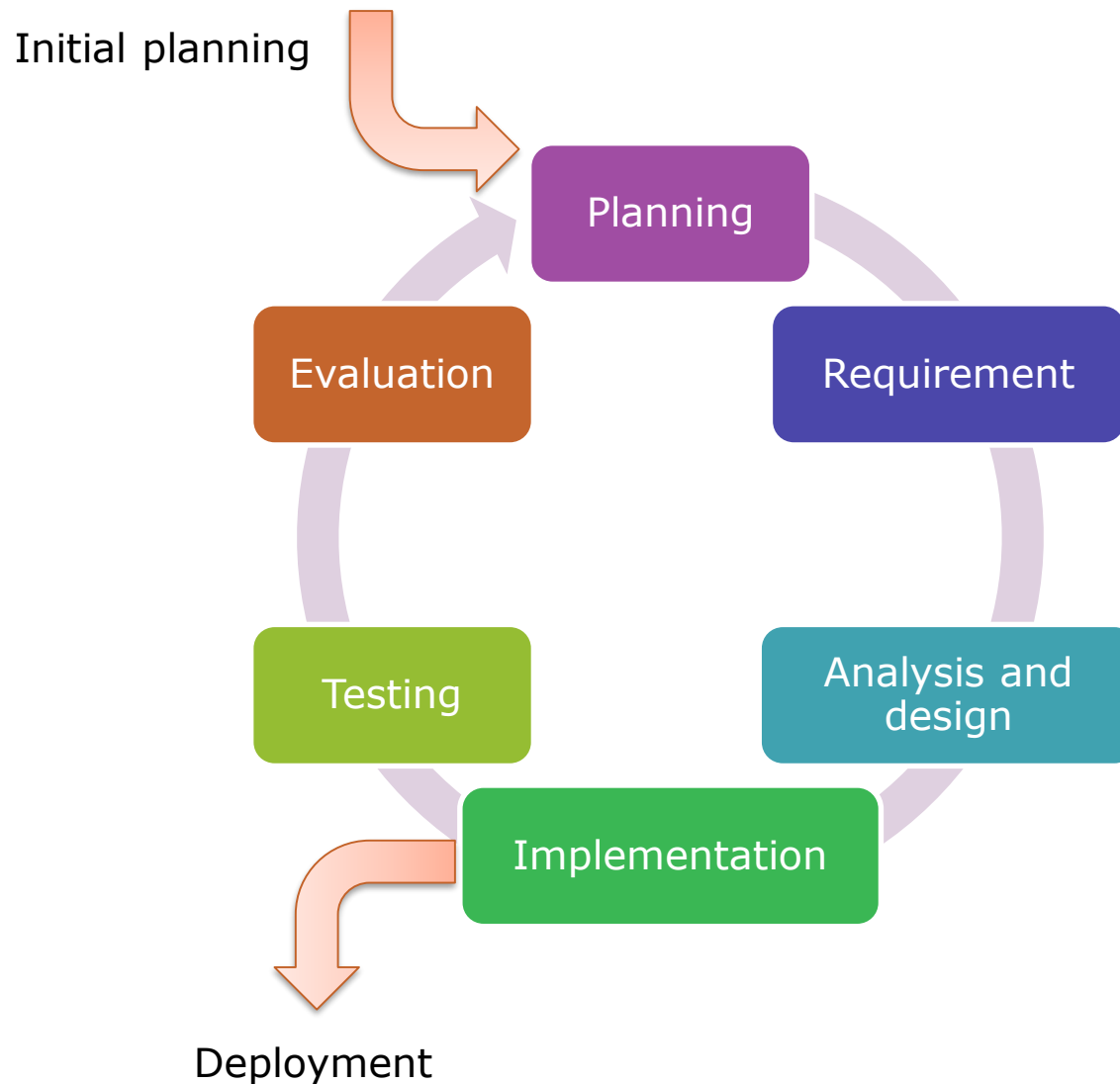


Integration and system testing



Delivery and maintenance

Generic process models: iterative



1.5. What are the attributes of good software?

- The software should deliver the required functionalities 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: it must be understandable, usable and compatible with other systems



1.6. What are software engineering methods?

- Methods are organized ways of producing software, including:
 - Model: graphical descriptions which should be produced
 - Rules: constraints applied to system models
 - Recommendations: advice on good design practice
 - Process guidance: activities to follow
- i.e., structured approaches to software development.



1.7. What are the key challenges facing software engineering?

- 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



I. SOFTWARE ENGINEERING

1. FAQs
2. Deal with complexity & changes
3. Knowledge Area & Units



Approach

- Consider the software engineering as a problem solving activities
 - Analysis: Understand the nature of the problem and break the problem into pieces
 - Synthesis: Put the pieces together into a large structure
- For solving a problem we use:
 - Techniques (methods): Formal procedures for producing results using some well-defined notation
 - Example ?
 - Methodologies: Collection of techniques applied across software development and unified by a philosophical approach
 - Example ?
 - Tools: Instrument or automated systems to accomplish a technique
 - Example ?



2.1. Deal with complexity

- The problem here is the complexity
- Many sources of complexity, but size is the key
- This problem can be solved by a structured design approach:
 - Modeling
 - Decomposition
 - Abstraction
 - Hierarchy
 - Use patterns




2.2. Deal with changes

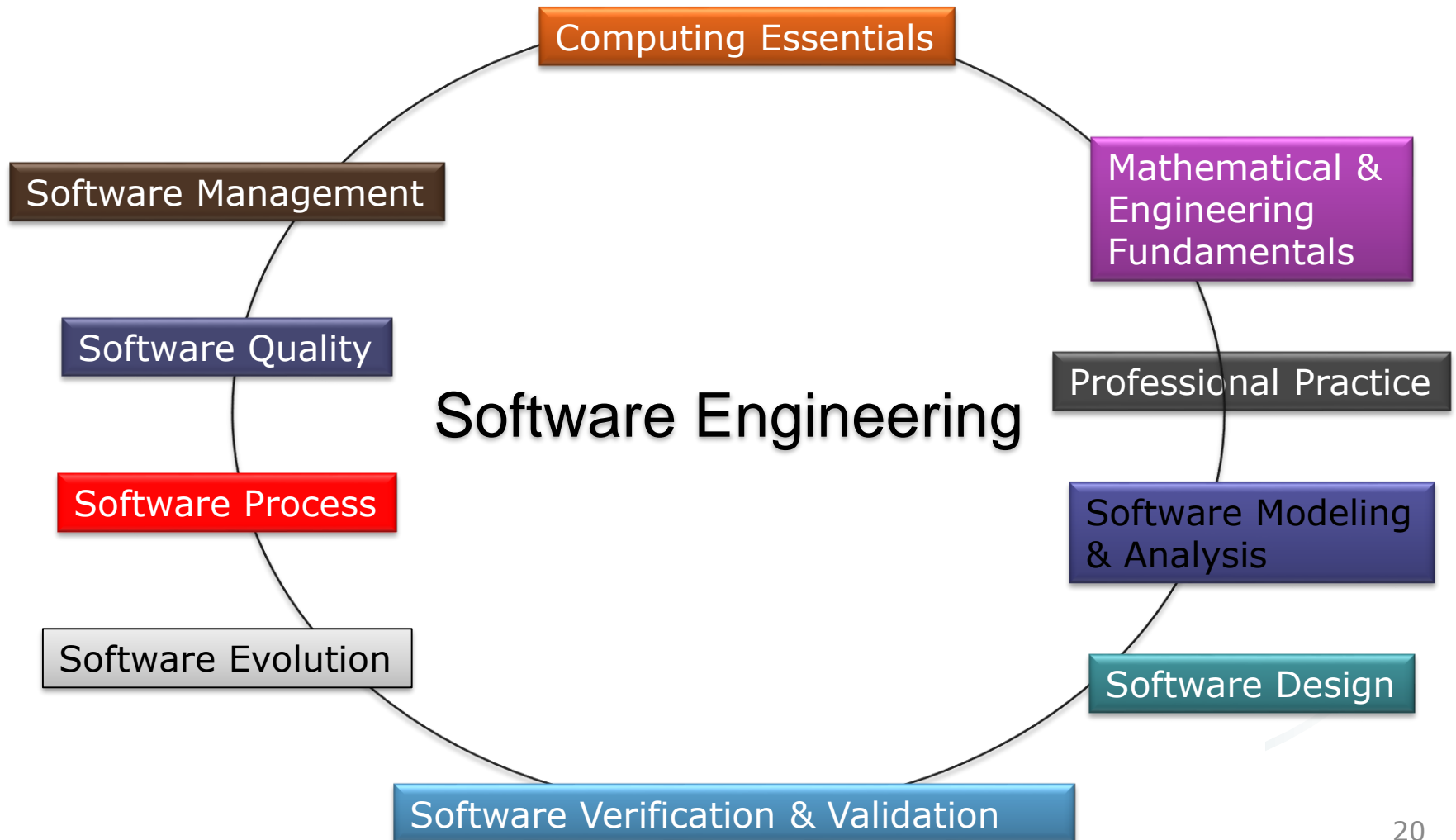
- Changes of project conditions: tailor the software lifecycle
- Changes of requirements or technology: use a nonlinear software lifecycle
- Changes of entities: provide the configuration management



I. Software engineering

1. FAQs
 2. Deal with complexity & changes
 3. Knowledge Area & Units
- 

3. Knowledge Areas & Units





II. SOFTWARE QUALITY

1. Classifications of software qualities
2. Representative qualities
3. Quality measurement



Introduction

- Software products are different from traditional types of products
 - intangible
 - difficult to describe and evaluate
 - malleable
 - human intensive
 - involves only trivial “manufacturing” process
- Good software products require good programming, but ...
programming quality is the means to the end, not the end itself.



1. Classification of software qualities

- Internal vs. external
 - External → visible to users
 - Internal → concern developers
 - Internal qualities affect external qualities
- Product vs. process
 - Our goal is to develop software products
 - The process is how we do it
 - Process quality affects product quality



2. Representative qualities

- Common qualities: 11
- Process-specific qualities: 3
- Application-specific qualities

Correctness

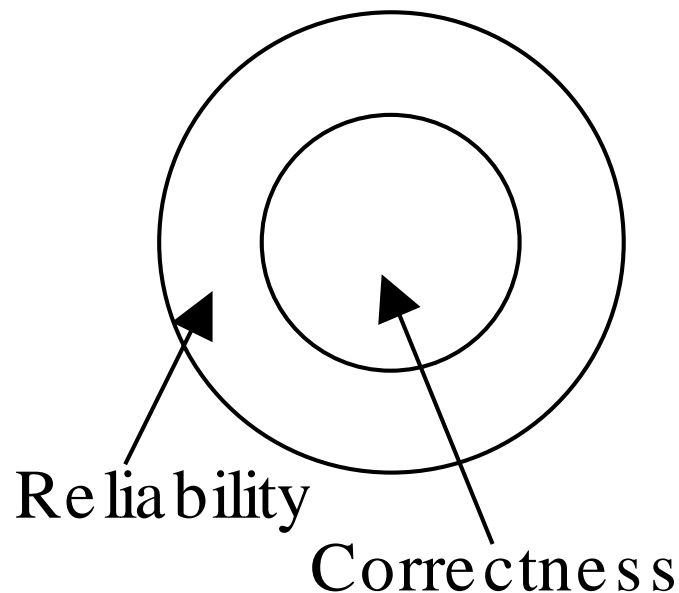
I. Software engineering
II. Software quality
1. Classification
2. Representative qualities
2.1. Common qualities

- Software is correct if it satisfies the functional requirements specifications
 - assuming that specification exists!
- If specifications are formal, since programs are formal objects, correctness can be defined formally
 - It can be proven as a theorem or disproved by counter examples (testing)
- Limits:
 - It is an absolute (yes/no) quality
 - there is no concept of “degree of correctness”
 - there is no concept of severity of deviation
 - What if specifications are wrong? (e.g., they derive from incorrect requirements or errors in domain knowledge)

Reliability

I. Software engineering
II. Software quality
1. Classification
2. Representative qualities
2.1. Common qualities

- Informally, user can rely on it
- Can be defined mathematically as “probability of absence of failures for a certain time period”
- If specifications are correct, all correct software is reliable, but not vice-versa (in practice, however, specs can be incorrect ...)
- Idealized situation: requirements are correct





Robustness

- I. Software engineering
- II. Software quality
 - 1. Classification
 - 2. Representative qualities
 - 2.1. Common qualities**

- Software behaves “reasonably” even in unforeseen circumstances (e.g., incorrect input, hardware failure)



Performance

I. Software engineering

II. Software quality

1.

Classification

2.

Representative qualities

2.1. Common qualities

- Efficient use of resources
 - memory, processing time, communication
- Can be verified
 - complexity analysis
 - performance evaluation (on a model, via simulation)
- Performance can affect scalability
 - a solution that works on a small local network may not work on a large intranet

Usability

I. Software engineering

II. Software quality

1. Classification

2. Representative qualities

2.1. Common qualities

- Expected users find the system easy to use
- Other term: user-friendliness
- Rather subjective, difficult to evaluate
- Affected mostly by user interface
 - e.g., visual vs. textual
- Why is usability important?
 - Users are able to achieve their tasks easily and efficiently, which has public relations benefits for the organization – thereby increasing uptake.
 - Systems having poor usability levels can result in substantial organizational costs
 - People avoid using the application if they find it difficult to use



Verifiability

I. Software engineering

II. Software quality

1. Classification

2. Representative qualities

2.1. Common qualities

- How easy it is to verify properties
 - mostly an internal quality
 - can be external as well (e.g., security critical application)



Maintainability

- Maintainability: ease of maintenance
- Maintenance: changes after release
 - Maintenance costs exceed 60% of total cost of software
 - Three main categories of maintenance
 - corrective: removing residual errors (20%)
 - adaptive: adjusting to environment changes (20%)
 - perfective: quality improvements (>50%)
- Maintainability can be decomposed as
 - Repairability: ability to correct defects in reasonable time
 - Evolvability: ability to adapt software to environment changes and to improve it in reasonable time



Reusability

I. Software engineering

II. Software quality

1. Classification

2. Representative qualities

2.1. Common qualities

- Existing product (or components) used (with minor modifications) to build another product
 - (Similar to evolvability)
- Also applies to process
- Reuse of standard parts measure of maturity of the field



Portability

I. Software engineering

II. Software quality

1.

Classification

2.

Representative qualities

2.1. Common qualities

- Software can run on different hardware platforms or software environments
- Remains relevant as new platforms and environments are introduced
 - e.g. digital assistants
- Relevant when downloading software in a heterogeneous network environment



Understandability

I. Software engineering

II. Software quality

1.

Classification

2.

Representative qualities

2.1. Common qualities

- Ease of understanding software
- Program modification requires program understanding



Interoperability

I. Software engineering

II. Software quality

1. Classification

2. Representative qualities

2.1. Common qualities

- Ability of a system to coexist and cooperate with other systems
 - Capable of exchange information with other systems
 - Capable of use the exchanged information



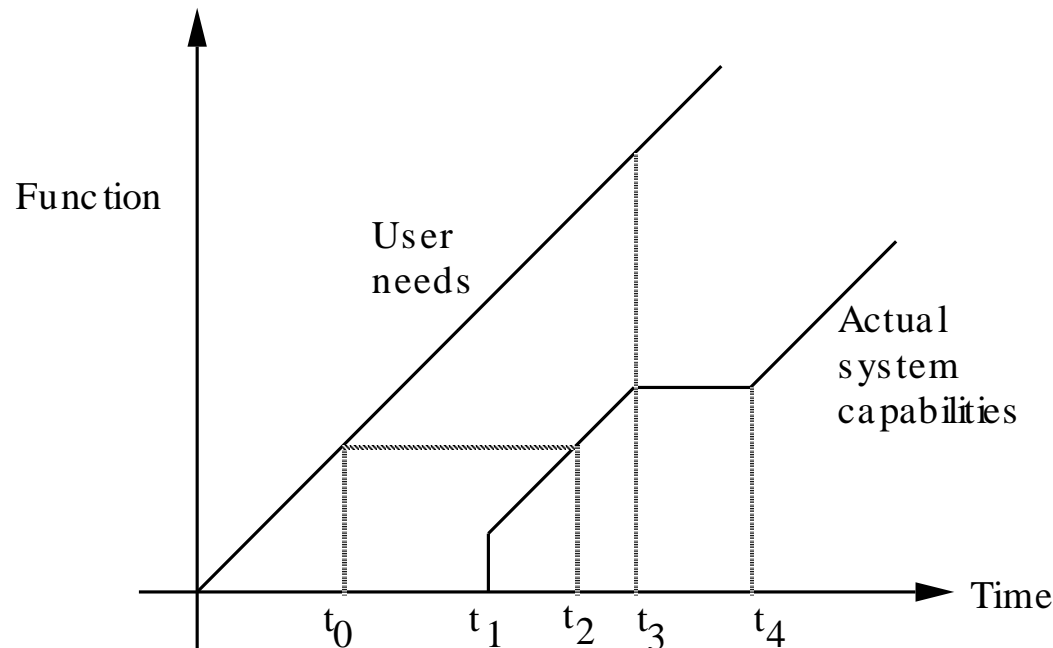
2.2. Typical process qualities

- Productivity
 - denotes its efficiency and performance
- Timeliness
 - ability to deliver a product on time
- Visibility
 - all of its steps and current status are documented clearly



Timeliness

- Often the development process does not follow the evolution of user requirements
- A mismatch occurs between user requirements and status of the product



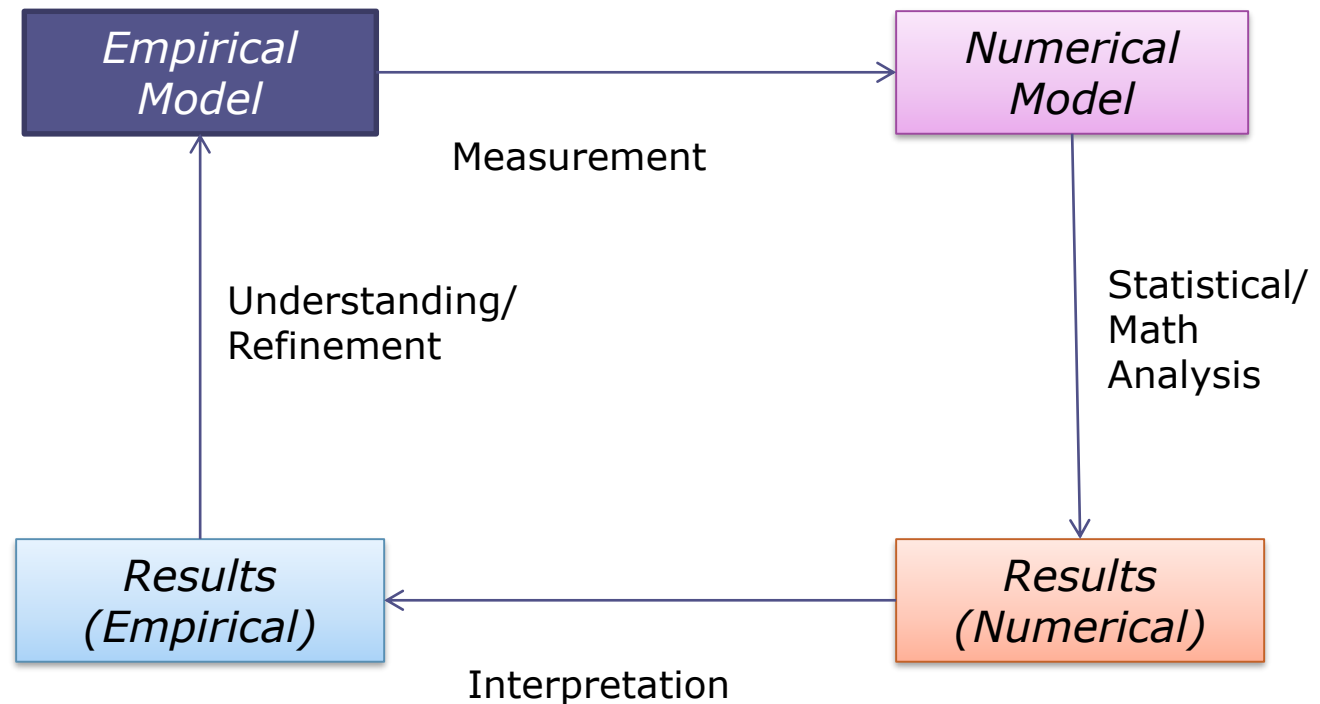


2.3. Application-specific qualities

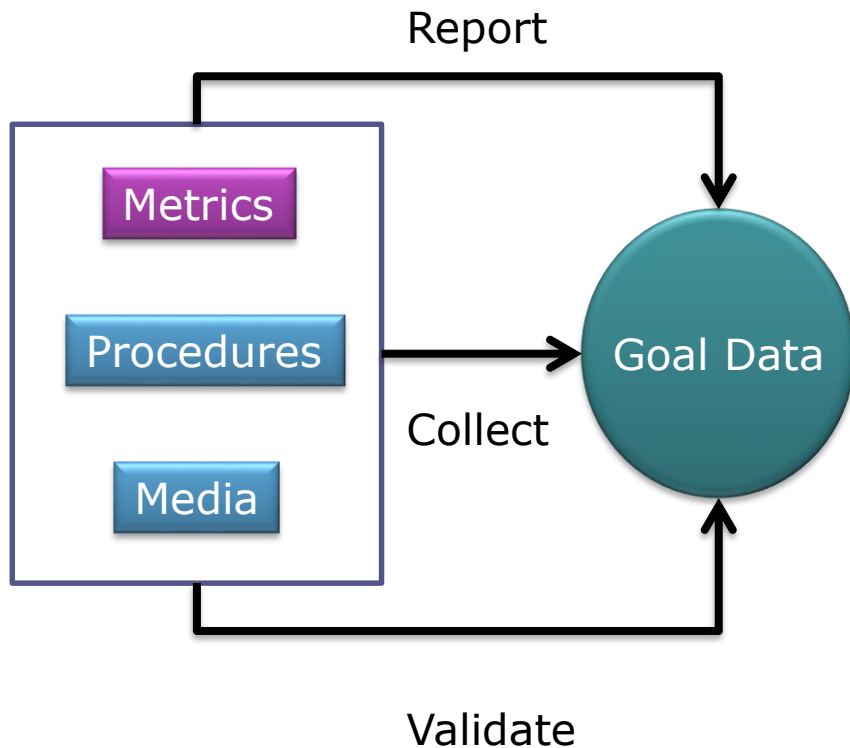
- Information systems
 - Security
 - Data availability
 - ..
- Real-time systems
 - Scheduling priority
 - ..
- Distributed systems
 - Code mobility

3. Quality measurement

- To measure the software quality indicators, we use:
 - Metrics
 - Methods



Measurement Plan



- name + description for each unique metric
- classification for each metric
- association point in product development that identifies when and how data is to be collected
- definitions of the data collection forms
- data reporting, collection, and validation procedures



Quiz and Exercises

- Now let's go over what you have learned through this lesson by taking a quiz.
- When you're ready, press Start button to take the quiz



1. Which of the following statements are true ? Why ?

- A. Software engineering is an engineering discipline that is concerned with all aspects of software production.
- B. The essential product attributes of GOOD SOFTWARE is interoperability, timeliness, productivity.
- C. Software engineering is a method of software production.
- D. Software products consist of developed programs and associated documentation.
- E. Producing software includes suggestions for the process to be followed, the notations to be used, rules governing the system descriptions which are produced and design guidelines.

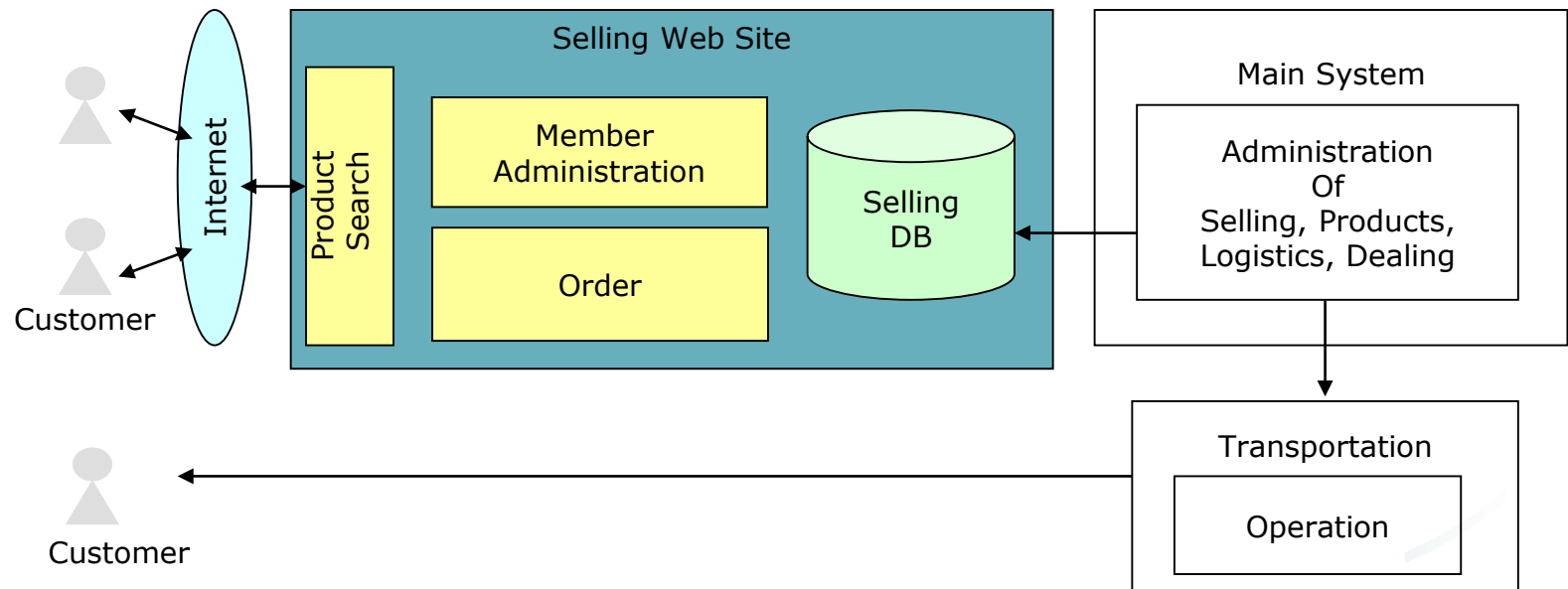


2. Which of the following statements are false ? Why ?

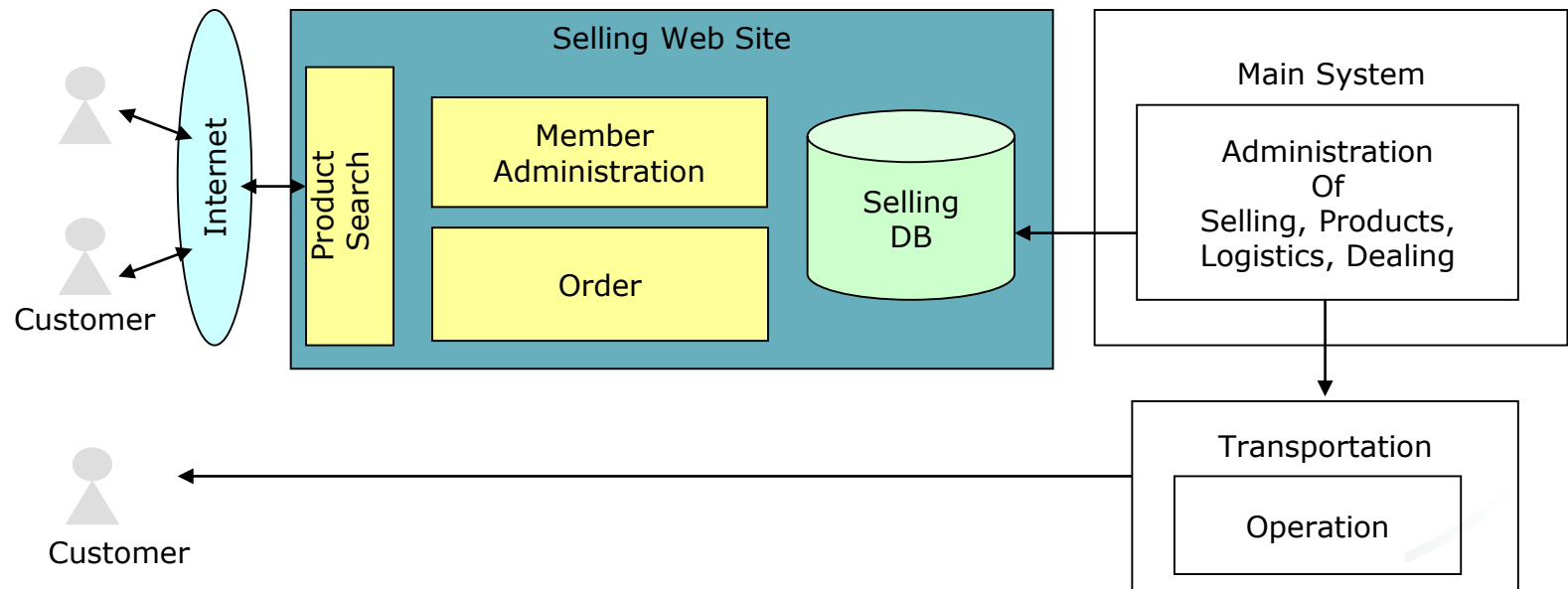
- A. Reusability is an application-specific quality indicator of software.
- B. Reusability is a common quality indicator of software.
- C. Reusability is a typical process quality indicator of software.
- D. Reusability is a measurement plan.
- E. Reusability is a software metrics.

3. Exercises

- The ABC Wear Corporation plans to develop (from system analysis to operation testing) the web site for selling wears through Internet and to form the new whole system as follow:

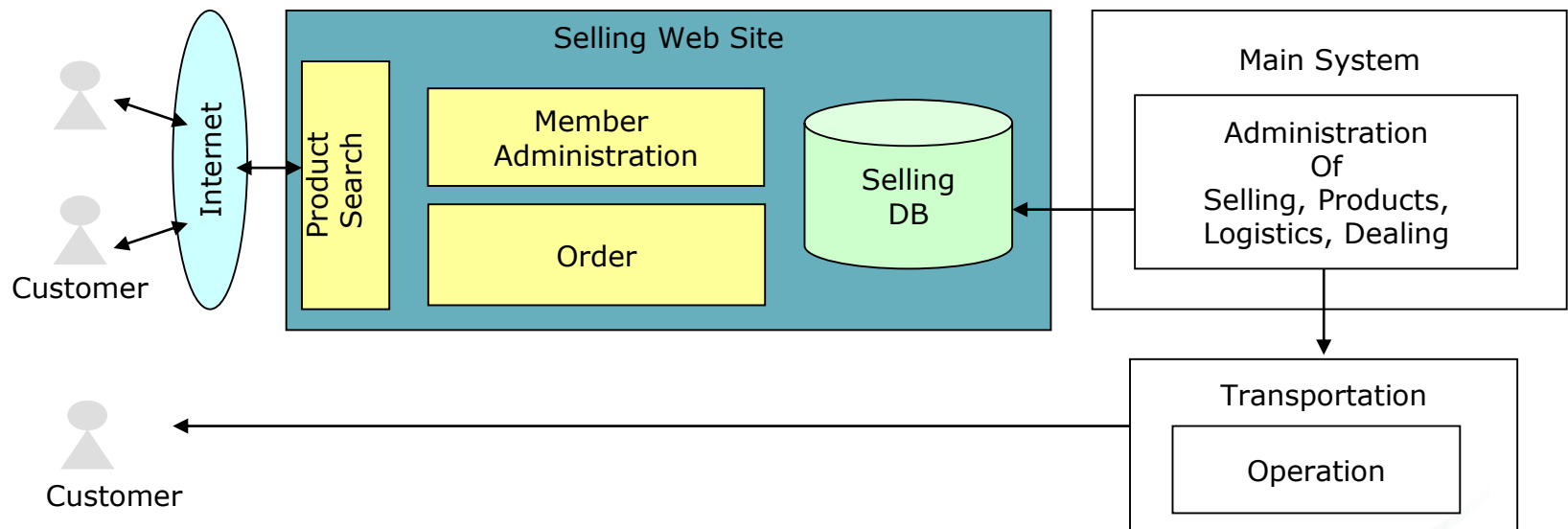


a. Identify the must-have functionalities of the website



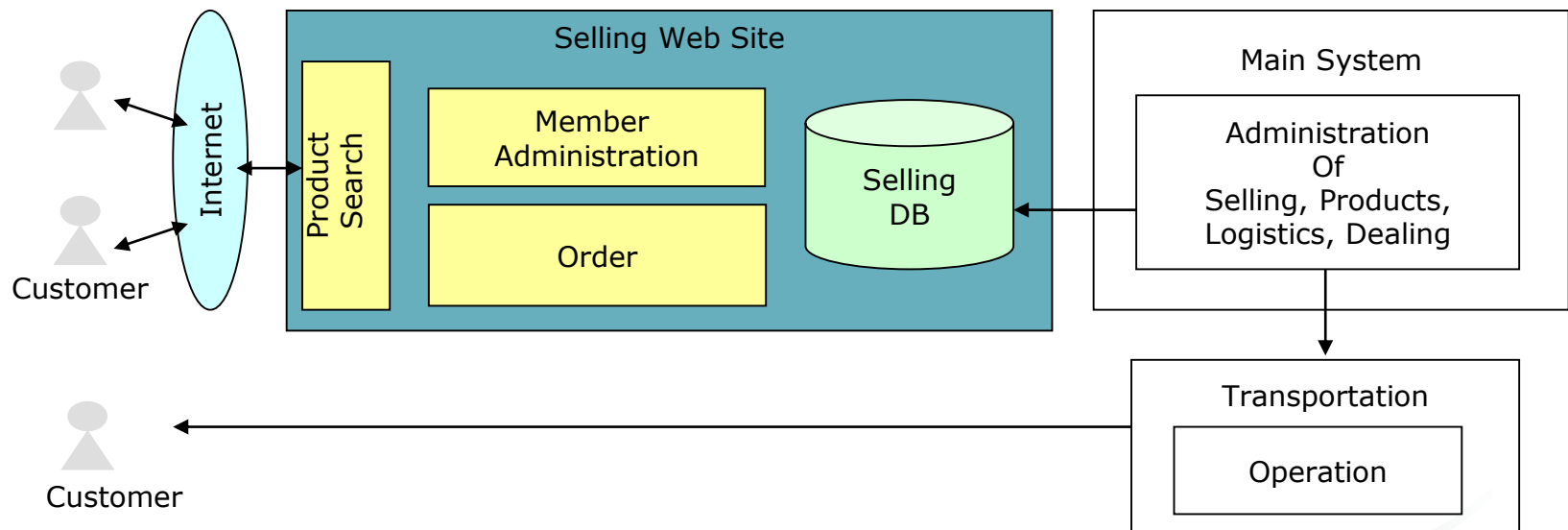
b. How to deal with complexity in the following cases:

- Developing new selling website.
- Activating the website in the whole system configuration.
- Reusing a search engine and a member administration module.



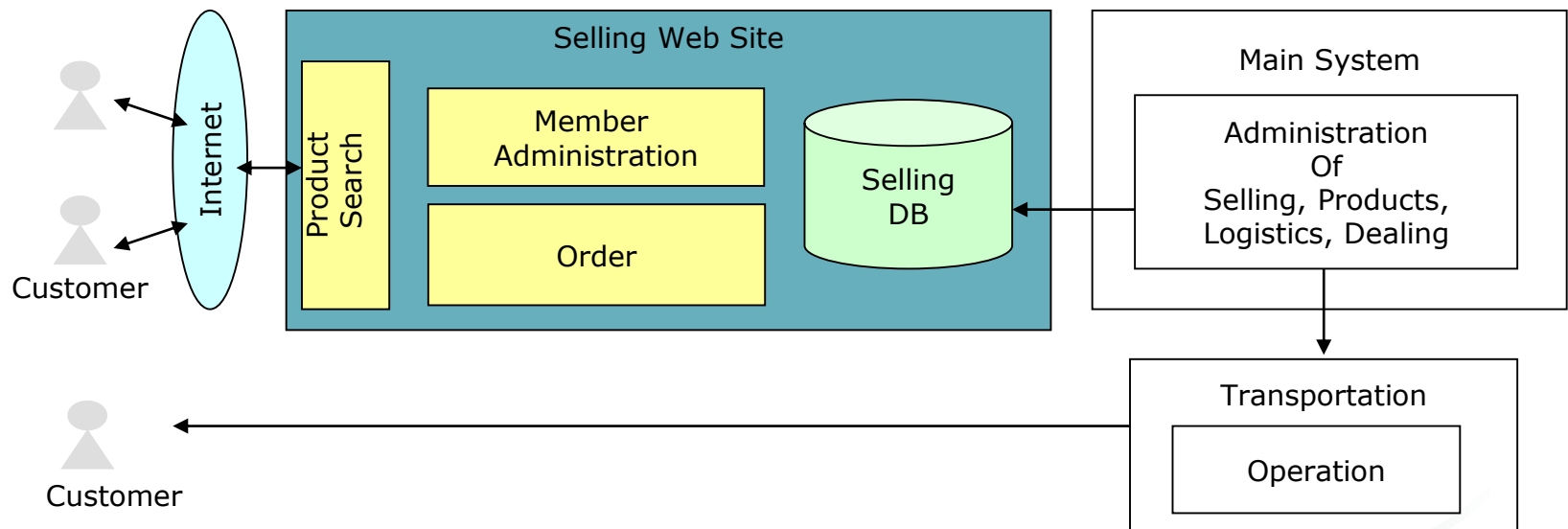
c. How to deal with changes in the following cases:

- Developing new selling website.
- Activating the website in the whole system configuration.
- Reusing a search engine and a member administration module.



d. Identify the quality indicators of the website in the following cases:

- Developing new selling website.
- Activating the website in the whole system configuration.
- Reusing a search engine and a member administration module.



e. Describe a measurement plan for the whole system that are suitable for the following cases:

- Developing new selling website.
- Activating the website in the whole system configuration.
- Reusing a search engine and a member administration module.

