Intro to Software Eng.

Elements and Qualities

Exercise 01: Review & Preview (in 2 weeks)

Read 1.1-1.2 and preview 1.3-1.4 of TextBook.

Read ch1 of “Beginning Software Engineering”

Read “Essence of Software Engineering”

Preview ch1 of “Design Science Methodology for IS & SE”

Preview ch1 of “A Philosophy of Software Design”

Send an email *in a week to* c.max@yeah.net with a subject like:

SE-id-Adam: Register for SE Course,

simply describe your expectation for the course and report your online learning context if you are learning online.

What is Software Engineering?

SWEs use a disciplined approach to the development of software-driven systems

SWE ! = programmer; SE is a relatively new field of study that applies to all types of systems that are developed as usable products

There are many different jobs that SWEs do

Software Engineering is a challenging career because of the inherent problems of software - as well as the rate of change in computing technologies, and the ever broadening range of applications

Software is Complex

Different from traditional types of products:

- malleable
- Intangible -- difficult to describe and evaluate
- abstract
- solves complex problems
- interacts with other software and hardware
- human intensive -- involves only trivial "manufacturing" process
- consequently, often software is **buggy**

Software engineering

- Software engineering is an engineering discipline that is concerned with all aspects of software production from the early stages of system specification through to maintaining the system after it has gone into use.
- Engineering discipline
  - Using appropriate theories and methods to solve problems bearing in mind organizational and financial constraints.
- All aspects of software production
  - Not just technical process of development. Also project management and the development of tools, methods etc. to support software production.

Importance of software engineering

- More and more, individuals and society rely on advanced software systems. We need to be able to produce reliable and trustworthy systems economically and quickly.
- It is usually cheaper, in the long run, to use software engineering methods and techniques for software systems rather than just write the programs as if it was a personal programming project. For most types of system, the majority of costs are the costs of changing the software after it has gone into use.

4 Technical Aspects of Software

- Software specification (规格描述), where customers and engineers define the software that is to be produced and the constraints on its operation.
- Software development (开发), where the software is designed and programmed.
- Software validation (验证), where the software is checked to ensure that it is what the customer requires. (narrowly also call Software verification)
- Software evolution (演化) , where the software is modified to reflect changing customer and market requirements.
- (+ Project Management – minor topic in this course)

A thinking framework in the form of an actionable kernel.

BY IVAR JACOBSON, PAN-WEI NG, PAUL E. MCMAHON, IAN SPENCE, AND SVANTE LIDMAN

The Essence of Software Engineering: The SEMAT Kernel

Figure 1. Things to work with.

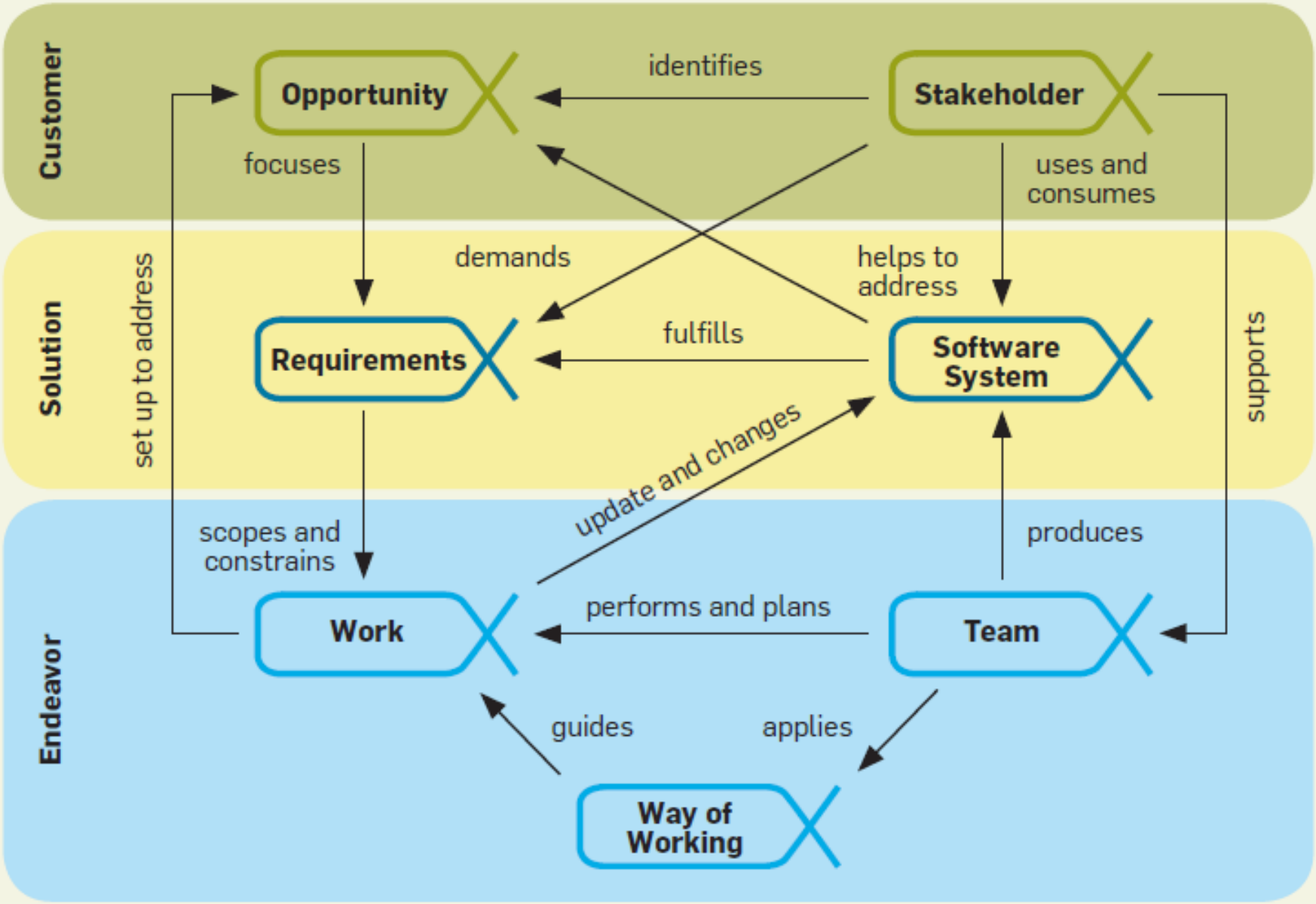


Figure 2. Things to do.

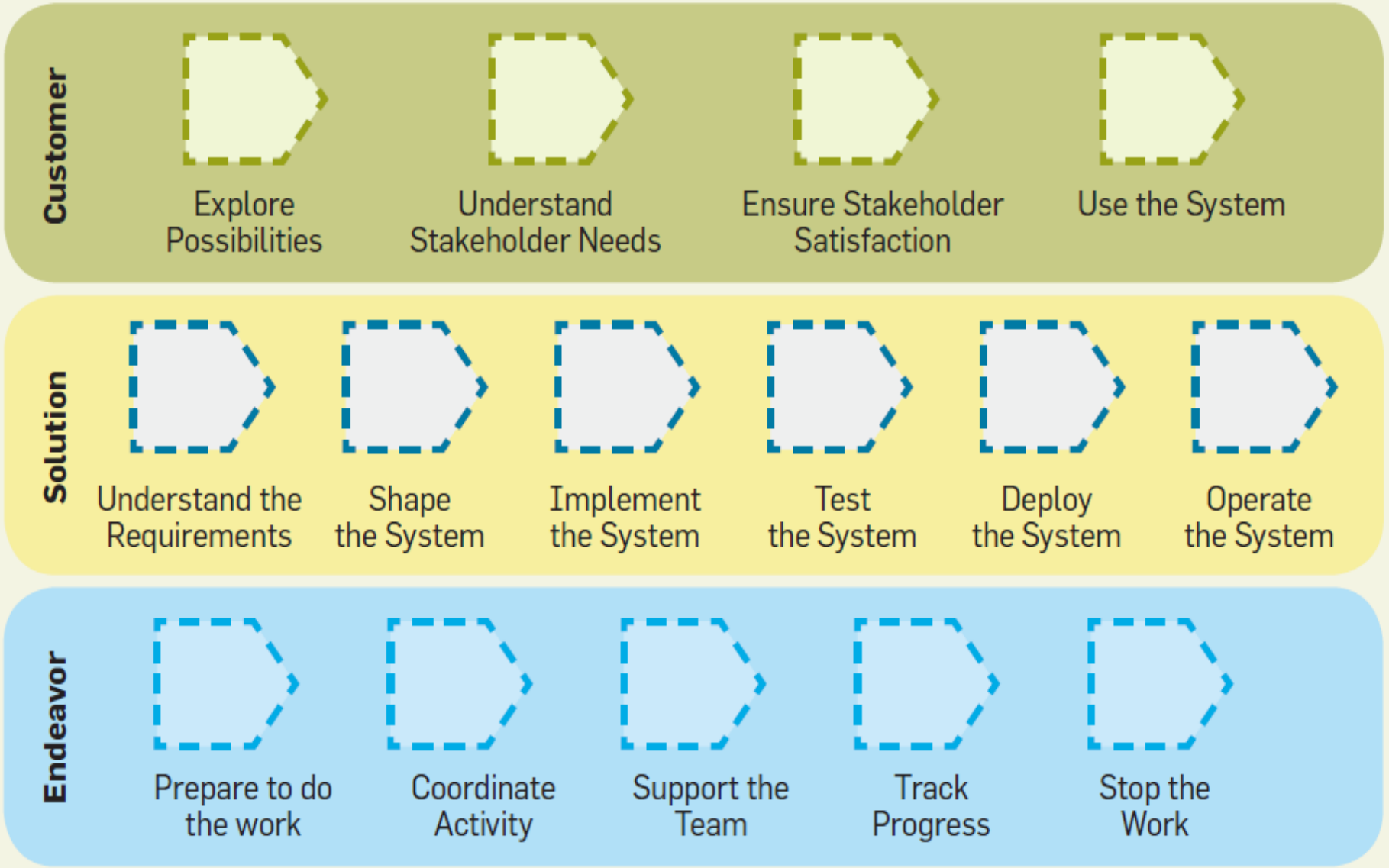
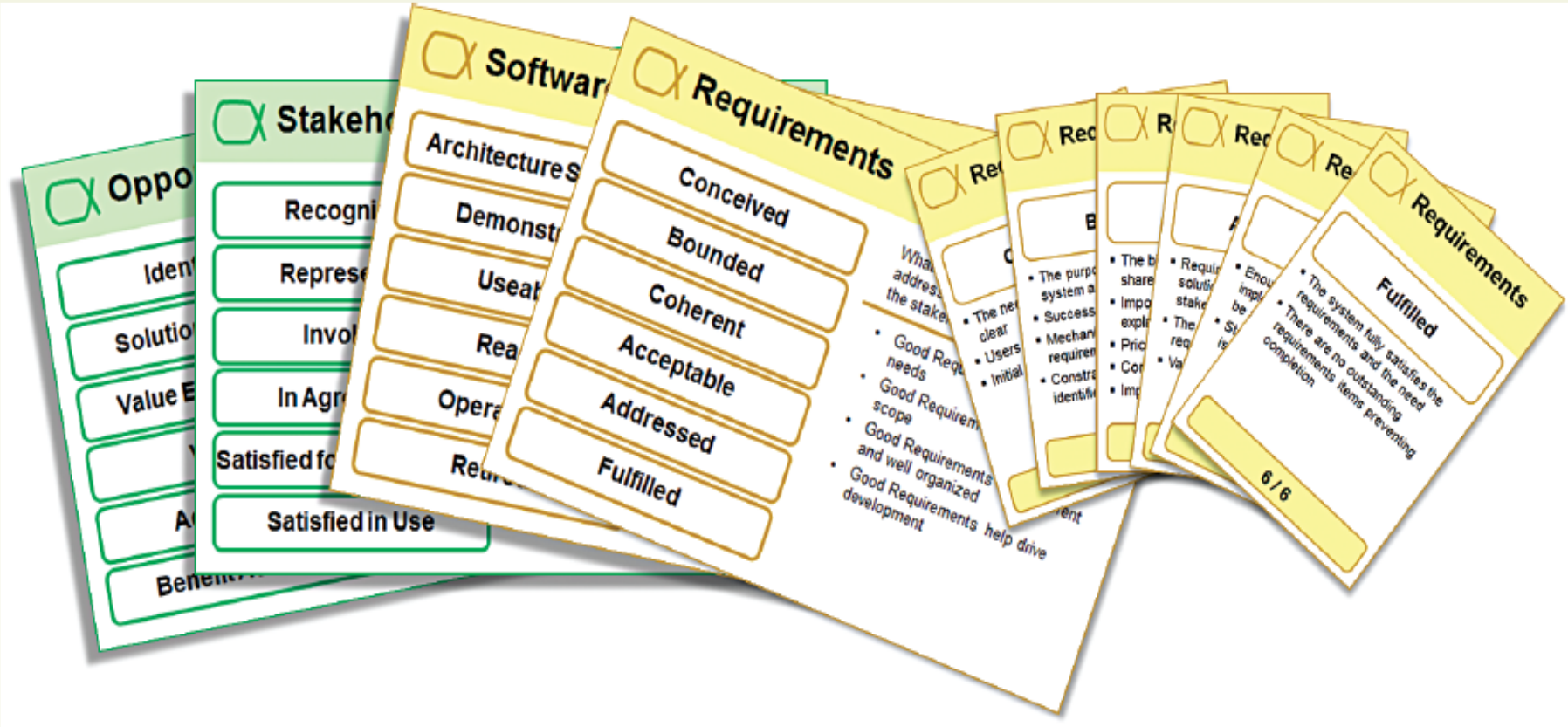


Figure 4. Cards make the kernel tangible.



Key Points on Software

Software engineering (SE) is an intellectual activity and thus human-intensive

Software is built to meet a certain functional goal and satisfy certain qualities

Software processes also must meet certain qualities

Software qualities are sometimes referred to as “ilities”

Classification of sw qualities : "ilities"

Internal vs. external

- External→ visible to users
- Internal→ concern developers

Product vs. process

- Our goal is to develop software products
- The process is how we do it

Internal qualities affect external qualities

Process quality affects product quality

Correctness (正确性)

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 counterexamples (testing)

The limits of correctness

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)

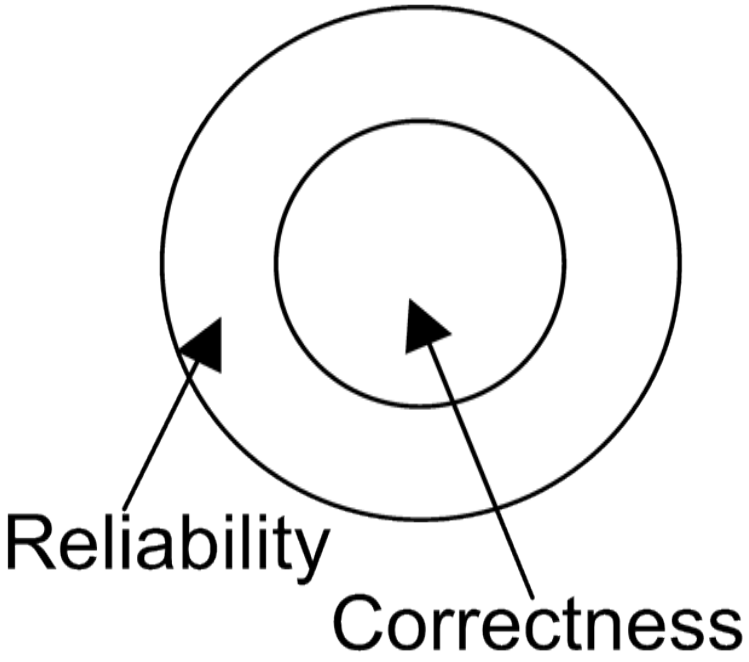
Reliability (可靠性)

Reliability

- informally, user can rely on it
- can be defined mathematically as “probability of absence of failures for a certain time period”
- if specs are correct, all correct software is reliable, but not vice-versa (in practice, however, specs can be incorrect …)

Idealized situation

Requirements are correct



Robustness (稳健性，鲁棒性)

Robustness

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

Performance (性能，效率)

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 (可用性)

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

Better expression: Usage centric

Verifiability (可验证性)

How easy it is to verify properties

- mostly an internal quality
- can be external as well (e.g., security critical application)

Maintainability (可维护性)

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 sw to environment changes and to improve it in reasonable time

Reusability (重用性，复用性)

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 (可移植性)

Software can run on different hw platforms or sw environments

Remains relevant as new platforms and environments are introduced (e.g. digital assistants)

Relevant when downloading software in a heterogeneous network environment

Understandability (可理解性)

Ease of understanding software

Program modification requires program understanding

Interoperability (互操作性)

Ability of a system to coexist and cooperate with other systems

- e.g., word processor and spreadsheet

Criteria for Good Programs (Software)

1) Run Effectively (Correctly) & Efficiently (Objectives 目的)
2) Easy to be Extended and Modified (Approaches 手段)
3) Easy to be Understood (Pre-conditions 前提)

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

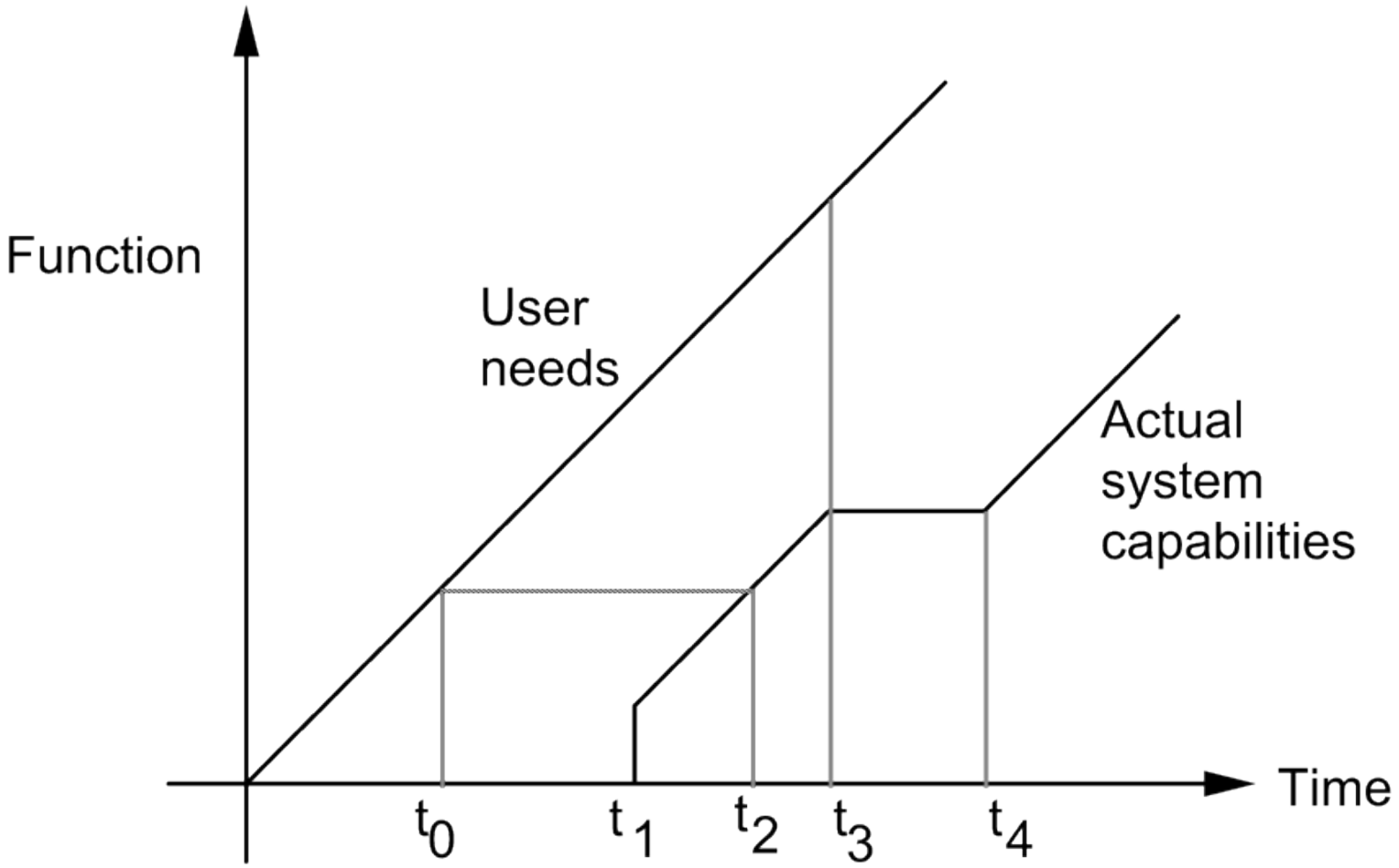
Timeliness: issues

Often the development process does not follow the evolution of user requirements

A mismatch occurs between user requirements and status of the product

Timeliness:

a visual description of the mismatch



Application-specific qualities

E.g., information systems

- Data integrity
- Security
- Data availability
- Transaction performance.

Quality measurement

Many qualities are subjective

No standard metrics defined for most qualities