Software, Engineering, Software Engineering

Successful Software Development

• working definition: success

• unsuccessful software development exists

• common reasons for non-success

Course

• Content

• topic areas

• structure of topic areas

• emphasis: formal methods

• relation to other courses

• literature

Organisation

• lectures

• tutorials

• exam

Reference number

ISO/IEC/IEEE 24765:2010(E)

© ISO/IEC 2010

© IEEE 2010

INTERNATIONAL STANDARD

ISO/IEC 24765

First edition

2010-12-15

Systems and software engineering — Vocabulary

Ingénierie des systèmes et du logiciel — Vocabulaire

Software — Computer programs, procedures, and possibly associated documentation and data pertaining to the operation of a computer system. See also:

application software;
support software;
system software.

Contrast with:
hardware.

IEEE 610.12 (1990)

Software — 1. all or part of the programs, procedures, rules, and associated documentation of an information processing system. [...]

2. see 610.123. program or set of programs used to run a computer. [...]

NOTE: includes firmware, documentation, data, and execution control statements.

IEEE 24765 (2010)

Engineering vs. Non-Engineering

workshop (technical product)
studio (artwork)

Mental prerequisite

the existing and available technical know-how
artist’s inspiration, among others

Deadlines

can usually be planned with sufficient precision
cannot be planned due to dependency on artist’s inspiration

Price

oriented on cost, thus calculable
determined by market value, not by cost

Norms and standards

exist, are known, and are usually respected
are rare and, if known, not respected

Evaluation and comparison

can be conducted using objective, quantified criteria
is only possible subjectively, results are disputed

Author

remains anonymous, often lacks emotion to the product
considers the artwork as part of him/herself

Warranty and liability

are clearly regulated, cannot be excluded
are not defined and in practice hardly enforceable

(Ludewig and Lichter, 2013)
6.1.1 suitability

The capability of the software product to provide functions which meet stated and implied needs when the software is used under specified conditions.

6.1 functionality

The extent to which software functions efficiently and effectively on real machines.

6.1.1.1 operability

The ease with which a software system can be used effectively.

6.1.1.2 efficiency

The amount of effort required to use a software system in a given manner.

6.1.1.3 attractiveness

The conditions that determine whether or not a software system is desirable to users.

6.1.1.4 understandability

The ease with which the integral characteristics of software can be identified and grasped by humans.

6.1.1.5 learnability

The extent to which users can be expected to acquire operational skills by themselves.

6.1.1.6 resource utilisation

The extent to which the software system consumes hardware resources defined by the design.

6.1.1.7 time behaviour

The time behaviour with respect to the performance of tasks specified in the requirements.

6.1.1.8 time efficiency

The strategy to perform all activities specified in the requirements in the shortest possible time.

6.1.1.9 time performance

The efficiency with which all activities specified in the requirements are performed.

6.1.1.10 cost

The cost to develop and maintain the software system as specified in the requirements.

6.1.1.11 cost efficiency

The effort incurred to develop the software system in the shortest possible time.

6.1.1.12 cost performance

The ratio of the effort incurred to develop the software system to the time required to perform all activities specified in the requirements.

6.1.1.13 time spent

The time spent for the development of a software system as specified in the requirements.

6.1.1.14 time behaviour

The efficiency with which all activities specified in the requirements are performed.

6.1.1.15 time efficiency

The ratio of the time required to perform all activities specified in the requirements to the time spent for the development of a software system.

6.1.1.16 time performance

The ratio of the time required to perform all activities specified in the requirements to the time spent for the development of a software system.

6.1.1.17 time spent

The time spent for the development of a software system as specified in the requirements.

6.1.1.18 time behaviour

The ratio of the time required to perform all activities specified in the requirements to the time spent for the development of a software system.

6.1.1.19 time efficiency

The ratio of the time required to perform all activities specified in the requirements to the time spent for the development of a software system.

6.1.1.20 time performance

The ratio of the time required to perform all activities specified in the requirements to the time spent for the development of a software system.

6.1.1.21 time spent

The time spent for the development of a software system as specified in the requirements.
A software development project is successful if and only if developer, customer, and user are happy with the result at the end of the project.

### Some Empirical Findings (Buschermöhle et al. 2006)

- **Budget**
  - 1-9,999: 33.07%
  - 10,000-99,999: 2.91%
  - 100,000-499,999: 10.05%
  - 500,000-999,999: 22.49%
  - ≥1,000,000: 25.13%
  - Not specified: 25.66%

- **Planned Duration**
  - ≤3: 3.17%
  - 3-6: 30.16%
  - 6-12: 6.88%
  - 12-24: 5.03%
  - >24: 25.66%

- **Criticality**
  - Business critical: 97.35%
  - Mission critical: 2.65%

- **Project Completion**
  - Completed: 72.01%
  - Cancelled: 24.73%

- **Deadline**
  - Kept: 97.35%
  - Late: 2.65%
  - Missed: 0.27%

- **Main Functionality Realised**
  - Completed: 81.52%
  - Below: 11.14%
  - Above: 3.26%

- **Budget**
  - Below: 82.61%
  - Above: 4.89%
  - 20-49%: 5.16%
  - 90-94%: 1.9%
Example: Requirements Engineering

Vocabulary: e.g. consistent, complete, tacit, etc.

Techniques:
- informal
- semi-formal
- formal

In the course:
- e.g. "Whenever a crash..."
- e.g. "Always, if ⟨crash⟩ at t..."
- e.g. "∀t, t′ ∈ Time • ...

Use Cases
- Pattern Language
- Decision Tables
- Live Sequence Charts

Excursion: Informal vs. Formal Techniques

Example: Requirements Engineering, Airbag Controller

DaimlerChrysler AG, CC BY-SA 3.0

Requirement:
Whenever a crash is detected, the airbag has to be fired within 300 ms (±ε).

Developer A: ‘within’ means ‘≤’; so 100 ms is okay, too.

Developer B: ‘within’ means between 300 − ε and 300 + ε.

vs.

• Fix observables:
  - crashdetected: Time → {0, 1}
  - fireairbag: Time → {0, 1}

• Formalise requirement:
  ∀t, t′ ∈ Time • crashdetected(t) ∧ fireairbag(t′) ⇒ t′ ∈ [t + 300 − ε, t + 300 + ε]

→ no more misunderstandings, sometimes tools can objectively decide: requirement satisfied yes/no.
Software Engineering vs. Softwarepraktikum

On popular demand, the chair for software engineering agreed on: strong(er) coupling between both courses.

Introduction

Course: Organisation

Anny Questions So Far?
Course content and organisation

• motivate content of the course

→ humans are at the heart of software engineering.

note:
• state your solution,
• rephrase

• enable objective, tool-based assessment

• Basic rule for high quality submissions:
• Formal (vs. informal) methods
• !

Every exercise task is
• required

• totallyagree
• stronglydisagree

◦ ◦ ◦ ◦ ◦

I have improved my skills in scientific
• gather requirements,
• design,
• implementation,
• quality assurance,
• project management
• (evil rating, lower bound)
• (good-will rating, upper bound)

Problem solving.

We 'll have a
• (i) try to solve yourself
• (ii) discuss with colleagues

Exercises
• (group) forum, contact tutor

“offline”
• permitted exam aids: one A4 paper (max. 21 x 29.7 x 1 mm) of notes, max. two sides inscribed

“online”
• (group) forum

Admission points
• (i) try to solve yourself
• (ii) discuss with colleagues

Grading system:
• (reasonable grading given student' s knowledge
• (most complicated grading system ever

Break:
• (reasonable grading given student' s knowledge

Interaction:
• (usually Thursday, 12:00, local time).
• (usually Wednesday, 12:00, local time),

Organisation: Exam

Grading system:
• "reasonable grading given student' s knowledge
• "most complicated grading system ever

Break:
• "reasonable grading given student' s knowledge

Organisation: Exercises & Tutorials

Admission points
• (i) try to solve yourself
• (ii) discuss with colleagues

Grading system:
• (reasonable grading given student' s knowledge
• (most complicated grading system ever

Break:
• (reasonable grading given student' s knowledge

Organisation: Lectures

Admission points
• (i) try to solve yourself
• (ii) discuss with colleagues

Grading system:
• (reasonable grading given student' s knowledge
• (most complicated grading system ever

Break:
• (reasonable grading given student' s knowledge

Organisation: Tutorials

Admission points
• (i) try to solve yourself
• (ii) discuss with colleagues

Grading system:
• (reasonable grading given student' s knowledge
• (most complicated grading system ever

Break:
• (reasonable grading given student' s knowledge

Organisation: Labs

Admission points
• (i) try to solve yourself
• (ii) discuss with colleagues

Grading system:
• (reasonable grading given student' s knowledge
• (most complicated grading system ever

Break:
• (reasonable grading given student' s knowledge

Organisation: Other

Admission points
• (i) try to solve yourself
• (ii) discuss with colleagues

Grading system:
• (reasonable grading given student' s knowledge
• (most complicated grading system ever

Break:
• (reasonable grading given student' s knowledge

Other


