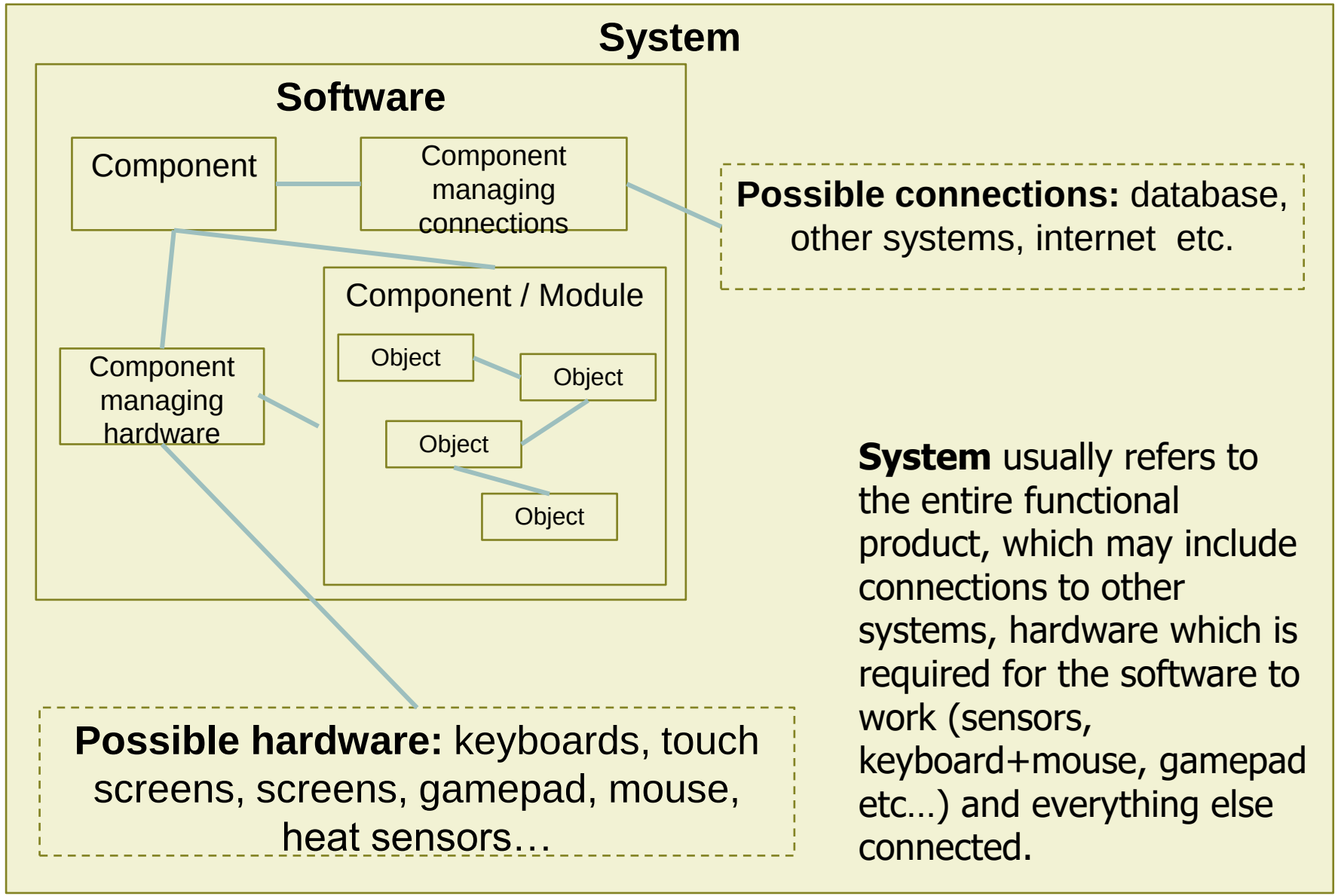
LUT University

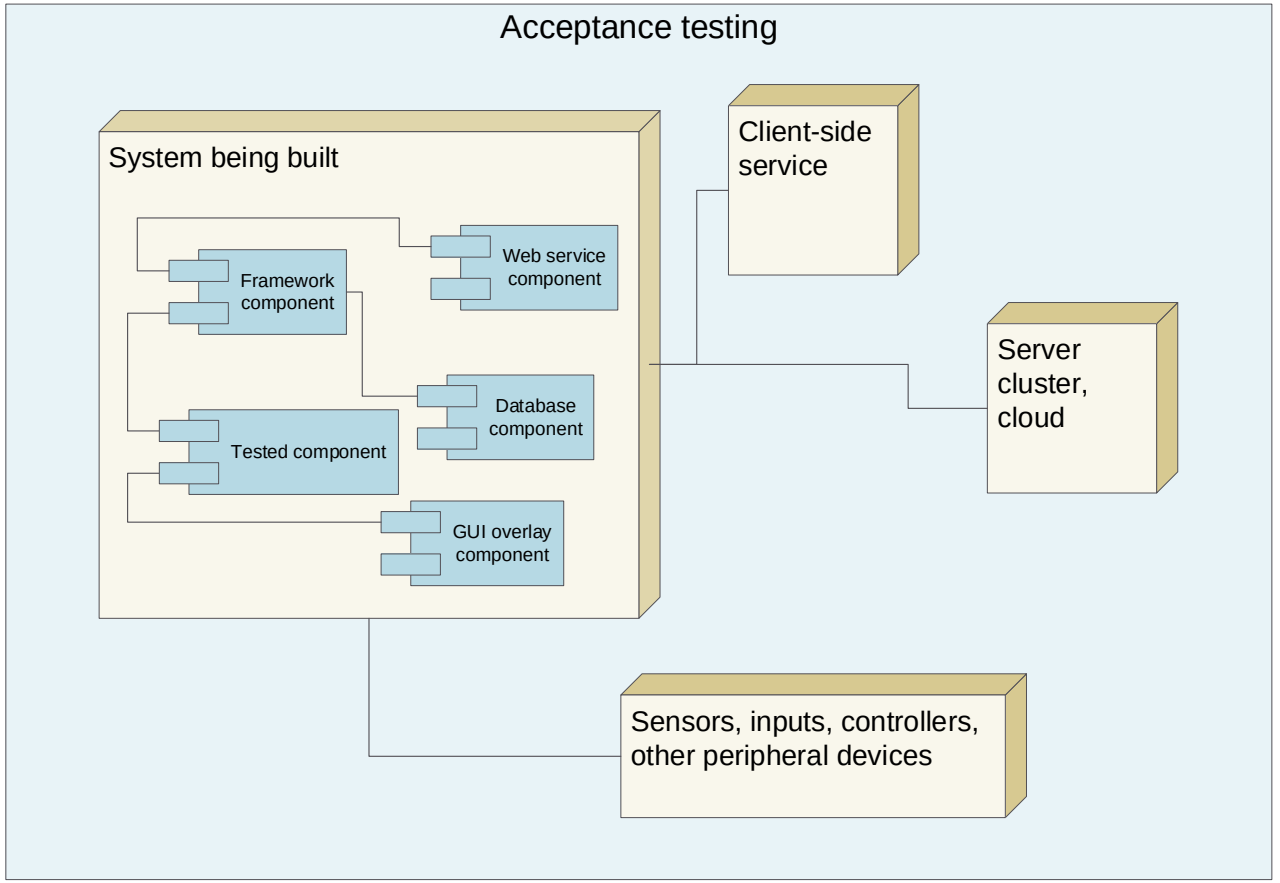
Modeling (and measuring) in the Software Industry

Prof. Jussi Kasurinen (jussi.kasurinen@lut.fi)
Fall 2020
LUT University

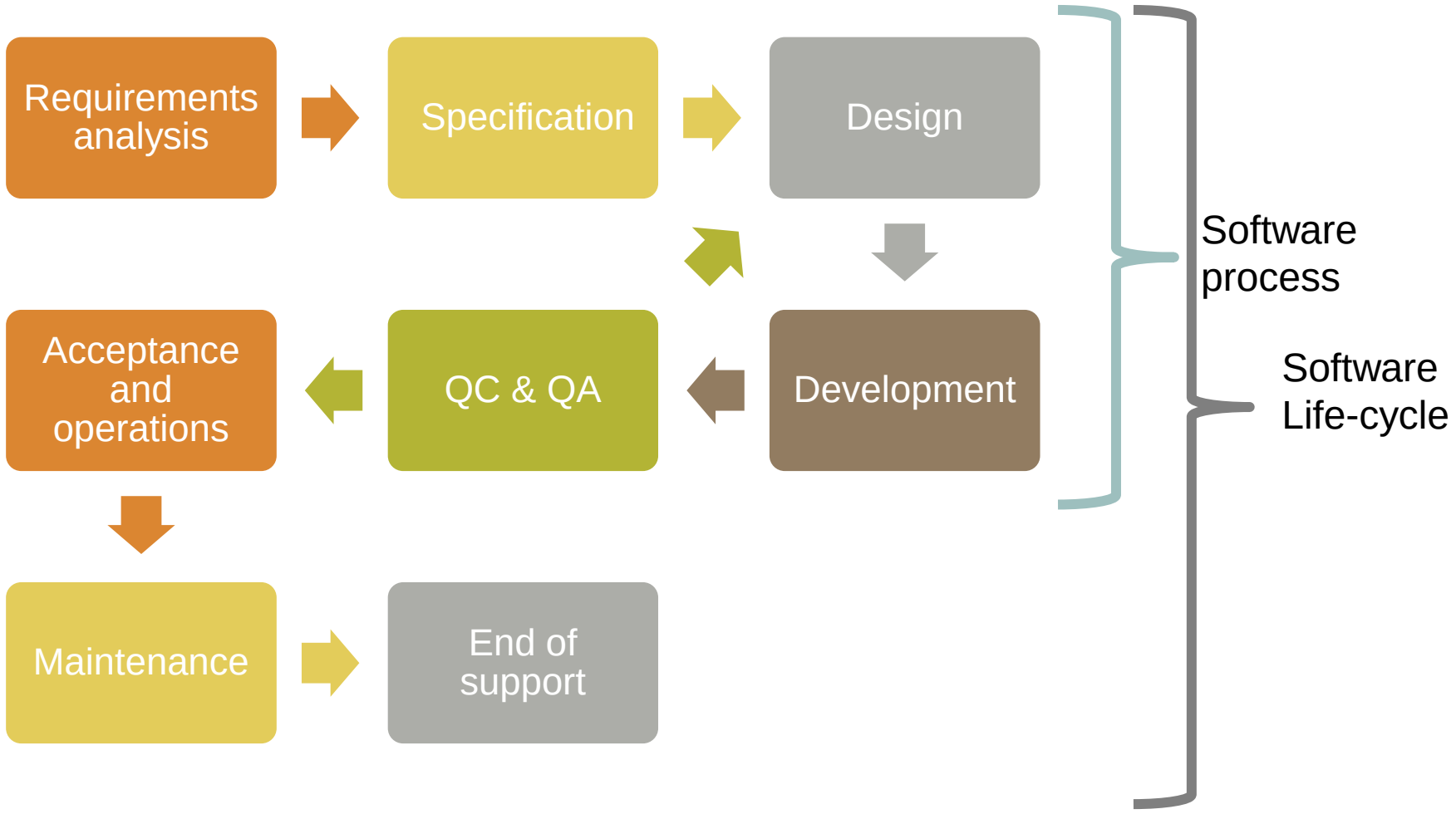
What is system?



Software in acceptance testing



Software lifecycle?



Indicators of system decay

- Frequent failures
- Overly complex structure
- Running in emulation mode
- Very large components
- Excessive resource requirements
- Deficient documentation
- High personnel turnover
- Different technologies in one system

Based on van Vliet, Vrije Universiteit, NL

Quality

Modeling and Measuring in the Software Industry

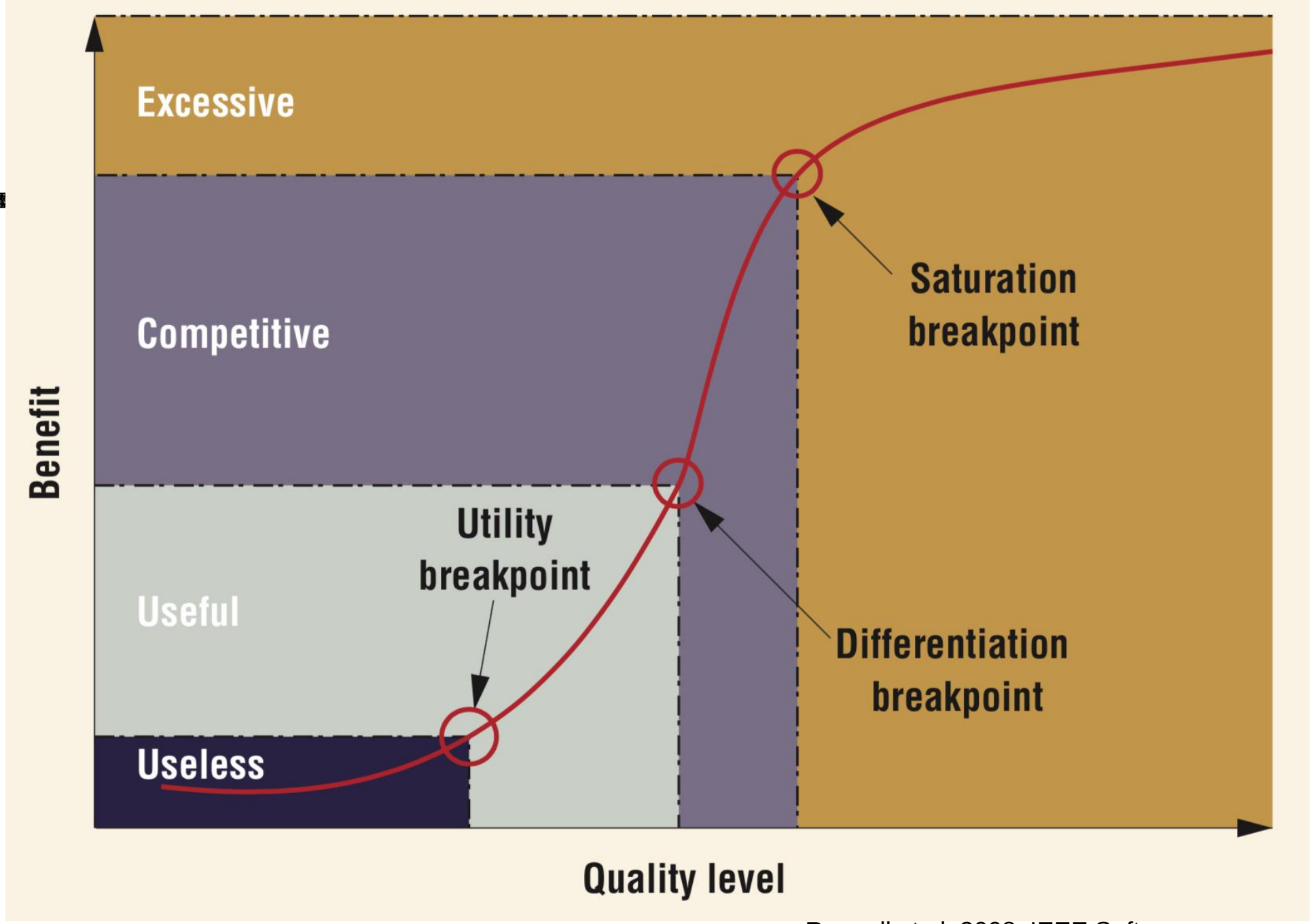
So three things stick out

- **Technical issues:** old software tends to become bloated and full of less-than-ideal fixes to keep the system running.
- **Management issues:** The process leaks brains, and the documentation is not properly maintained along with the system.
- **Quality issues:** The system performance gets weaker, and the overall quality declines over time.

So three things stick out

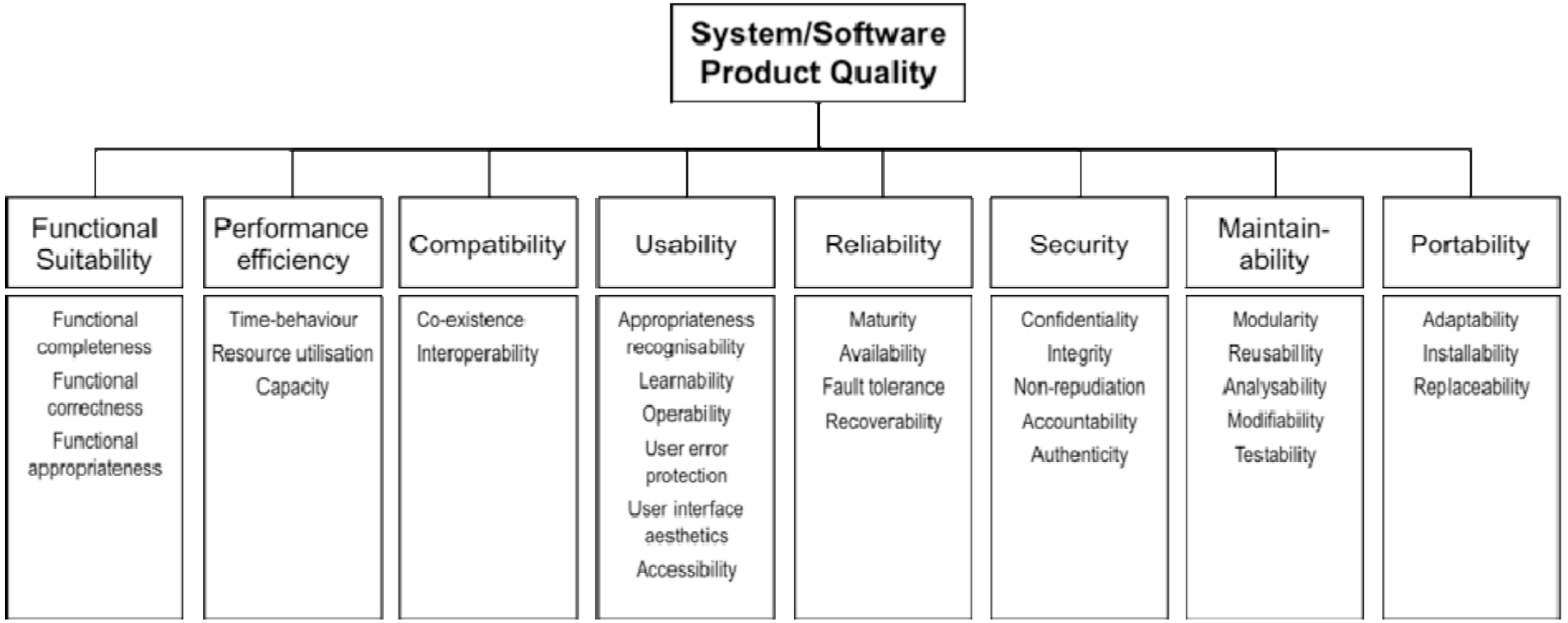
- **All these need to be addressed, but** lets start with the software quality.
- **What is quality?**
  - How can we measure quality? I.e. when can we say that something has declined too far?
  - How can we define quality? Surely different project aim to different types of quality?

L

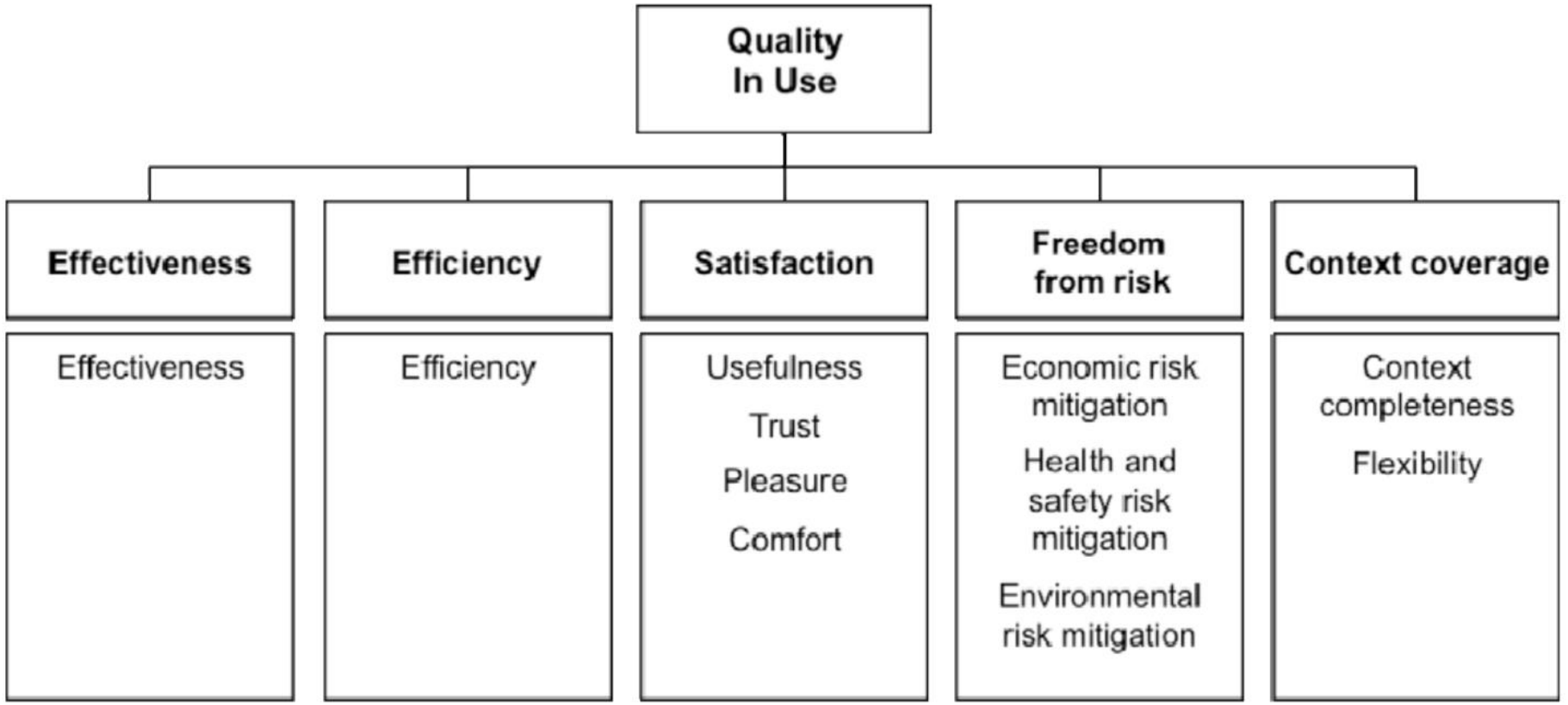


Regnell et al. 2008, IEEE Software

Example: ISO/IEC 25010 quality characteristics

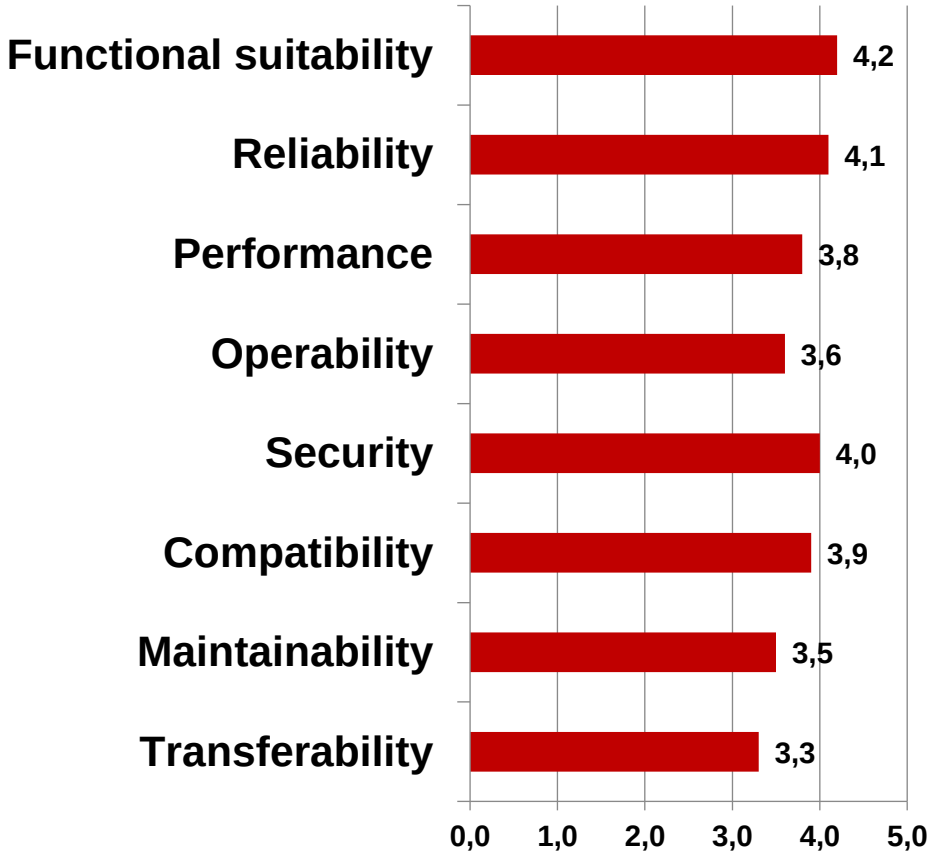


Example: ISO/IEC 25010 quality-in-use characteristics



Importance of Different Types of Quality

- In all interviewed organizations, every attribute is at least somewhat important.
- Only in 9 out of 248 (3,6%) assessments “not important”
- Perceived quality, not absolute quality!
- Common, universally acknowledged “good quality” is a myth.
- Industry differences; usability for games
  - In-use completely separate model!



Key developer practices

- Two key developer practices that support quality outcomes are:
  - version control
  - testing

Version Control Best Practices

1. Complete **one change at a time** and commit it
   - If you committing several changes together you cannot undo/redo them individually
   - If you don't commit and your hard disk crashes…
2. **Don't break the build**
   - Test your changes before committing
3. Commit **only the source** files
4. **Use the log** by writing a summary for each commit
   - What has been changed and why
5. **Communicate** with the other developers
   - See who else is working on a part before changing it
   - Discuss and agree on a design
   - Follow the project guidelines & specifications

Some best practices of an agile developer

- Integrate early integrate often
- Keep your project releasable at all times
- Automate acceptance testing
- Use automated unit tests
- Use it before you build it – TDD
- Write code in short edit/build/ test cycles
- Emphasize collective ownership of code
- Keep a solutions log
- Keep it simple
- Don't fall for the quick hack
- Write code to be clear not clever
- Warnings are really errors
- Provide useful error messages
- Be a mentor
- Share code only when it's ready
- Review code
- Keep others informed of the status of your work

V. Subramaniam & A. Hunt, "*Practices of an Agile Developer*", The Pragmatic Bookshelf, 2006.

Comparison to ISTQ-B principles

- 1. Testing shows the presence of bugs
- 2. Exhaustive testing is impossible
- 3. Test early as possible
- 4. Defects cluster: 20% contain the 80%
- 5. The pesticide paradox
- 6. Testing is context dependent
- 7. Absence of errors fallacy

MEASUREMENT

Some slides based on “Software measurement”, Mark Micallef, U of Malta

What is measurement?

Measurement is the process by which numbers or symbols are assigned to attributes of entities in the world *according to clearly defined rules*.

Uses of Measurement

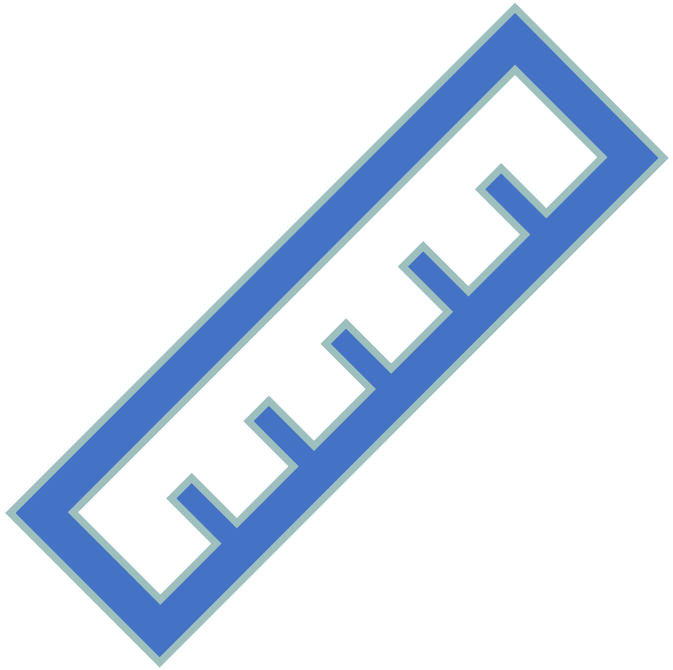
- Measurement helps us to understand
  - Makes the current activity visible
  - Measures establish guidelines
- Measurement allows us to control
  - Predict outcomes and change processes
- Measurement encourages us to improve
  - When we hold our product up to a measuring stick, we can establish quality targets and aim to improve

On measuring stuff (Katara, ISO/IEC 29119)

- Large projects or organizations need measurement planning:
  - **Why are things measured**
  - **What is measured**
  - **Who will measure**
  - **What parts of the system**
  - **When**
  - **How will the data be collected and analyzed**

On measurements (Katara, ISO/IEC 29119)

- Define how the measurements affect the business; basis on organizing the work towards the management!
- Ease of use to ensure reliable results.
- Understandability; also explanation for fluctuation between the measurements
- Selectivity
  - Results can indicate what component or event caused the result.



Requirements for any measurement ever.

- **Objectivity;** The measurement is not dependent on the people or environment of the measurement process.
- **Reliability;** Accuracy, repeatability and usability are robust enough to provide precise and readable result.
- **Economicality;** The measurement provides information which is useful, and saves more than the measurement costs.
- **Reusability;** The measurement is repeatable, and repeated measurements provide metrics which are comparable to earlier measurements.

Basic measurements

- Basic meaning “simple enough to count from any type of system or project”
  - Remaining budget vs. remaining time
  - Amount of features vs. remaining time.
  - Amount of features
  - Lines of code
    - Lines of code per feature
  - Amount bugs
    - Amount of critical bugs
  - Amount of fixed buges
  - Hours used, hours used per feature
  - Mean time between failures (MTBF)
  - Mean time between unique (new) failures
  - etc

Other measurements for industry

- Function Point analysis
  - General system characteristics-extension
- McCabe's Cyclomatic complexity metric
- Halstead's software science formulae

→ Assessment of ***amount of work*** needed or predicted, or probability to become a problem!

Modeling and planning

Modeling and Measuring in the Software Industry

UML 2 Chart types

Structure diagrams

- **Class diagram**
- **Component diagram**
- **Composite structure diagram** (added in UML 2.x)
- **Deployment diagram**
- **Object diagram**
- **Package diagram**

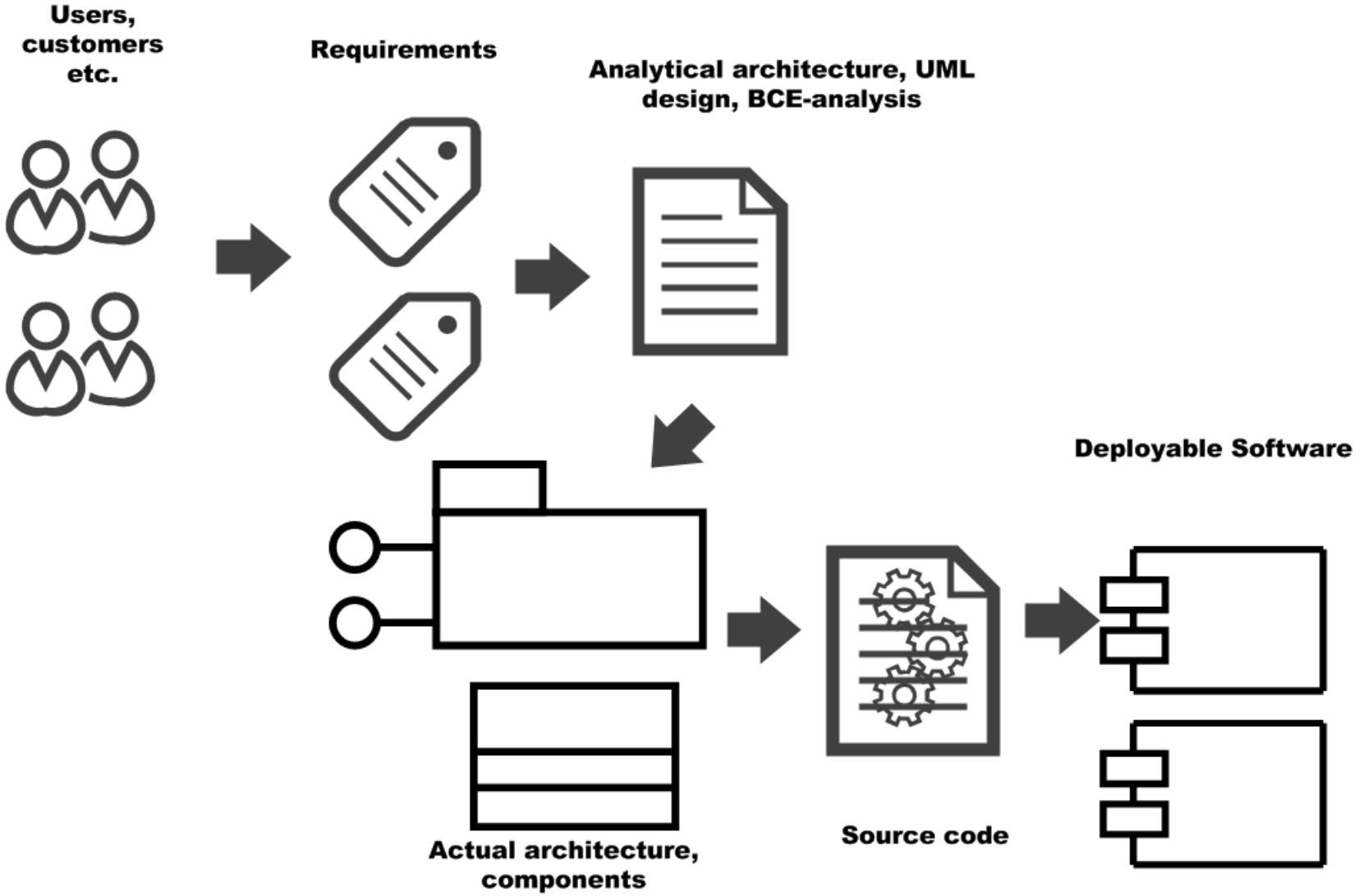
Behavior diagrams

- **Activity diagram**
- **State Machine diagram** (*was Statechart diag. in UML1*)
- **Use case diagram**

Interaction diagrams

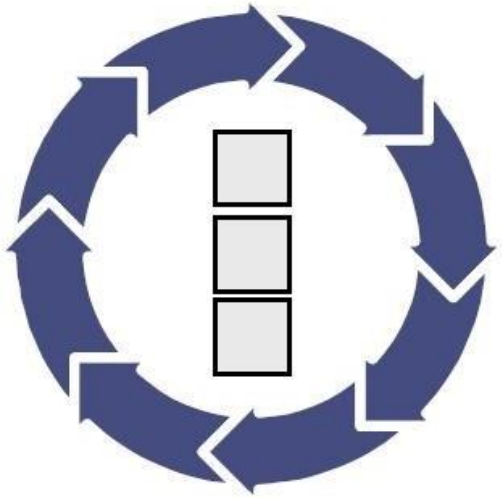
- **Communication diagram** (was *Collaboration diag. in UML1)*
- **Interaction overview diagram** (added in UML 2.x)
- **Sequence diagram**
- **Timing diagram** (added in UML 2.x)

From Requirements to implementation, basic concept

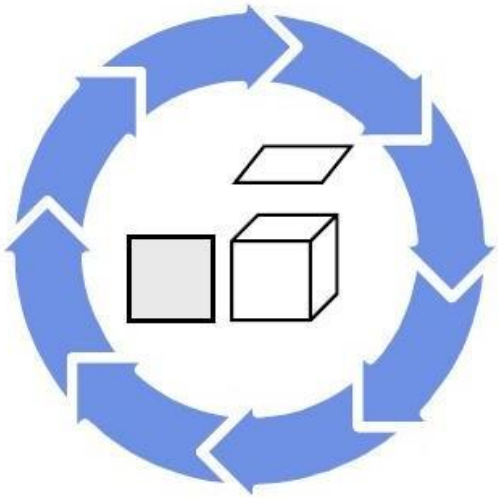


Additional basic concepts (for large companies)

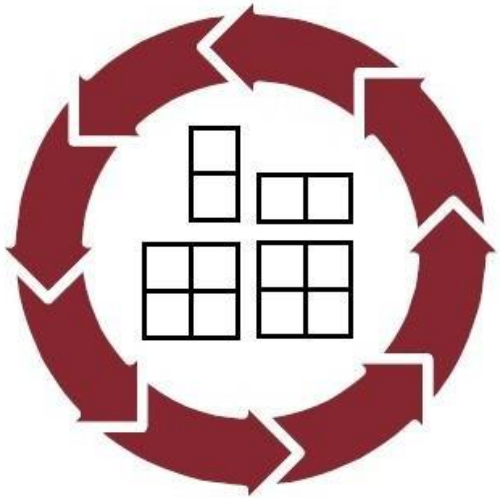
Use of a core asset base



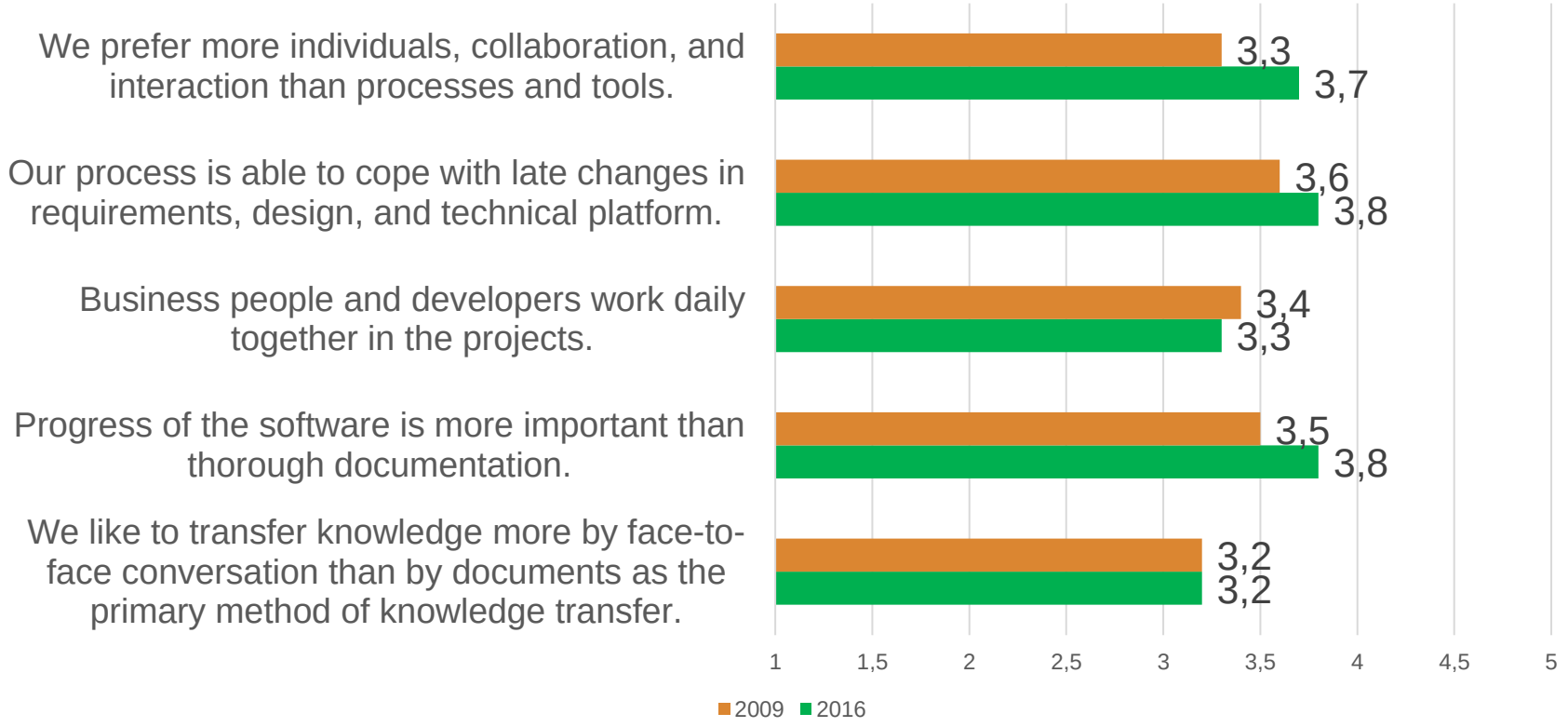
in production



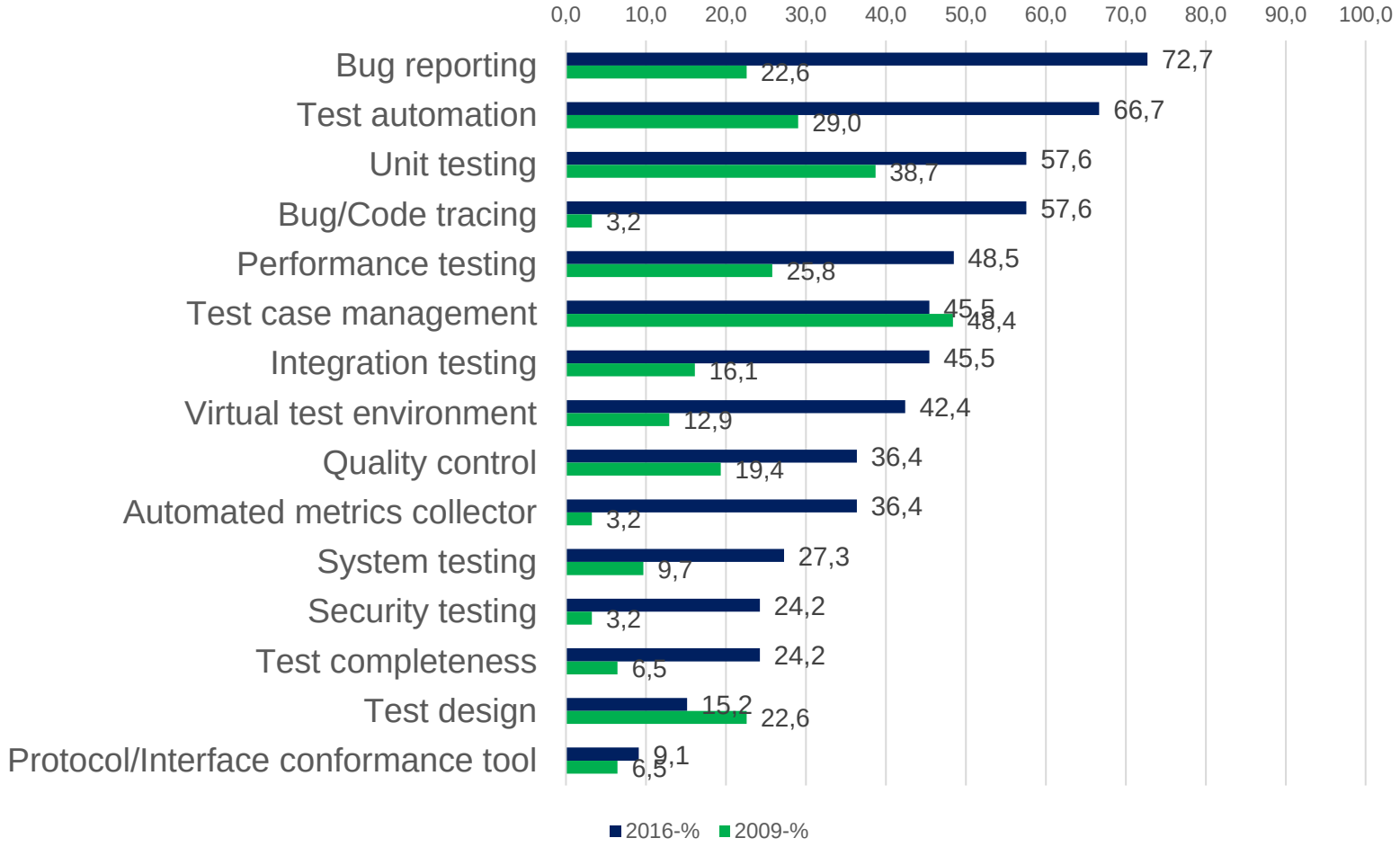
of a related set of products



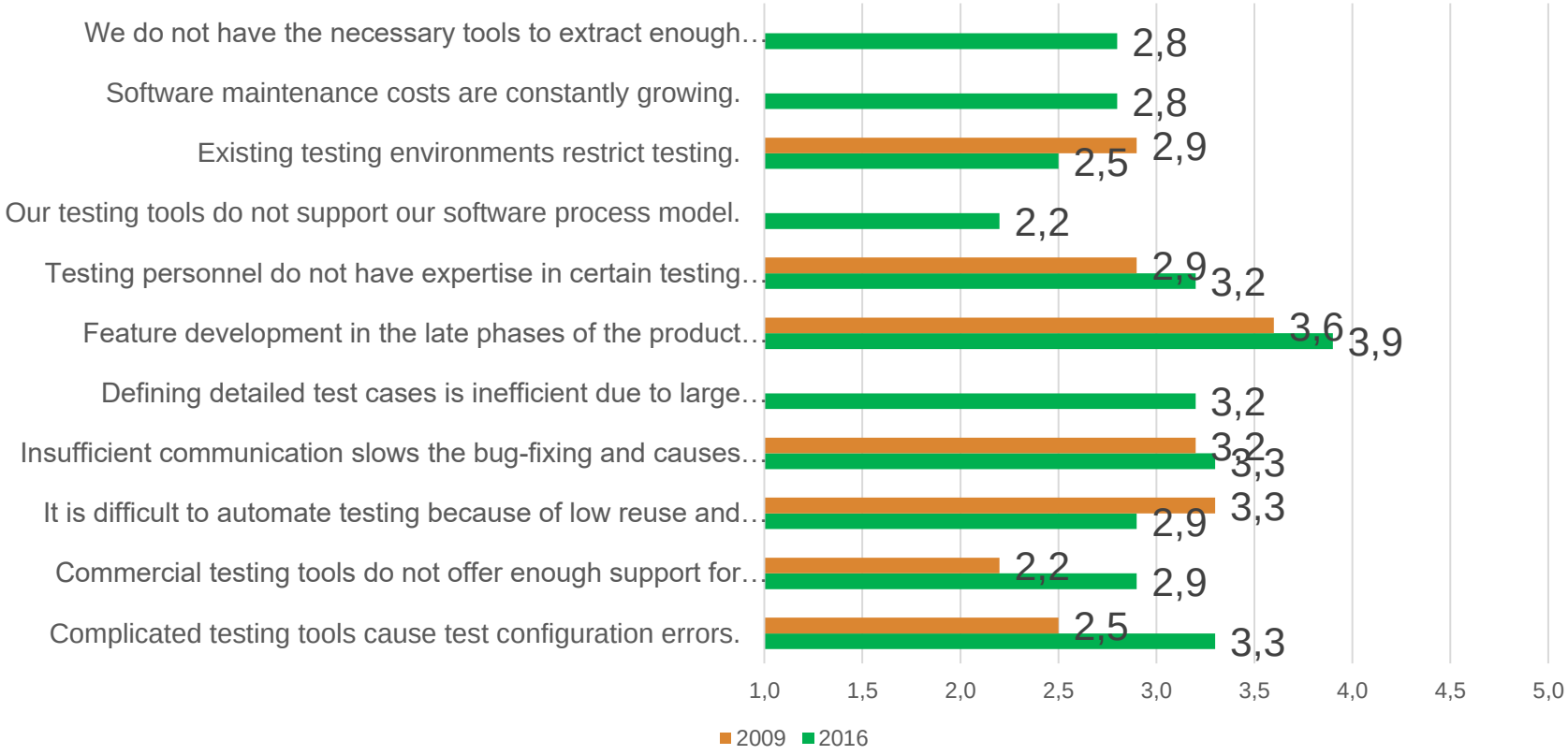
How Do We Priorize Things?



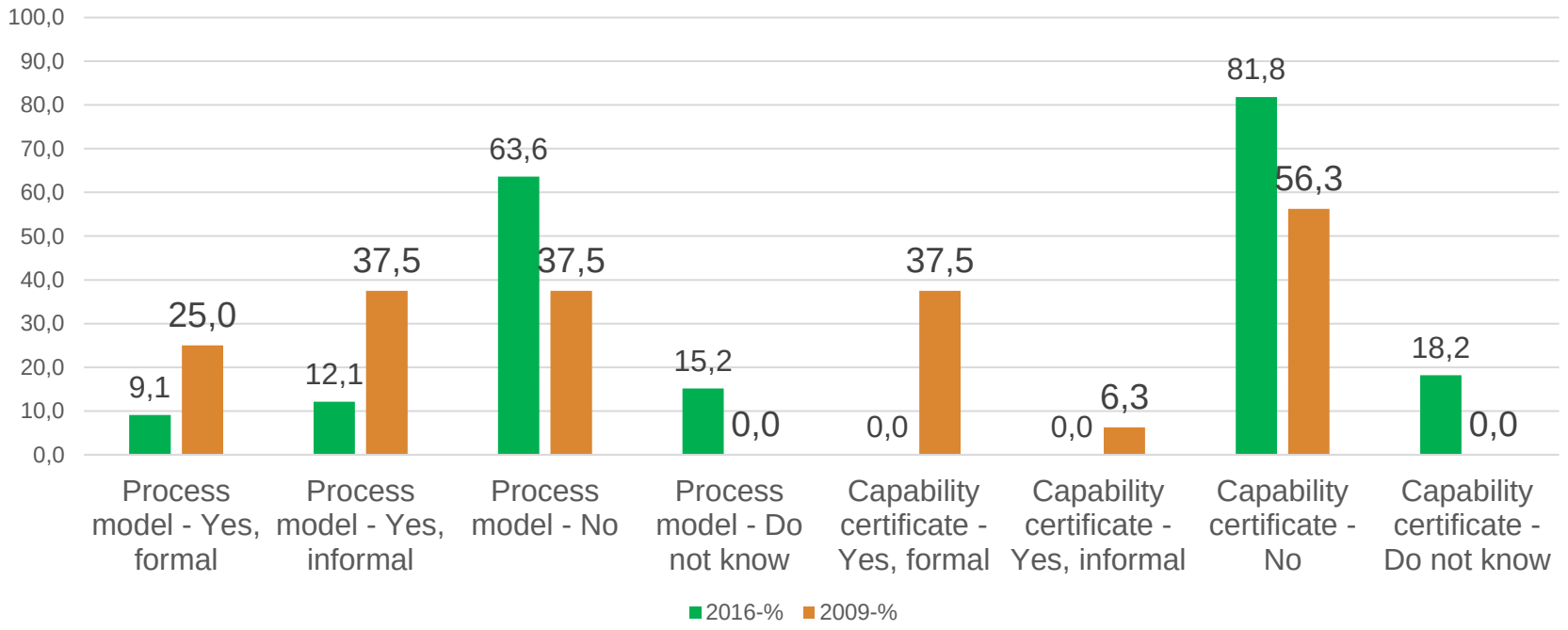
Applied testing tools (from 2009 and 2016 surveys)



Things That Bother



Formal Application level of Standards and Certifications

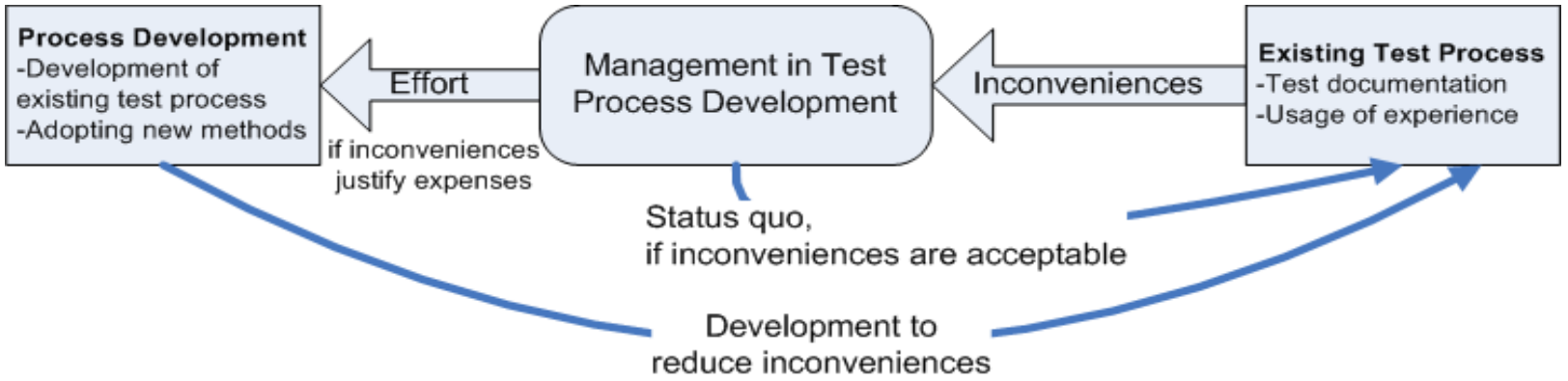


How do the organizations develop their (test) processes?

Organizations do not tend to try out new ideas.

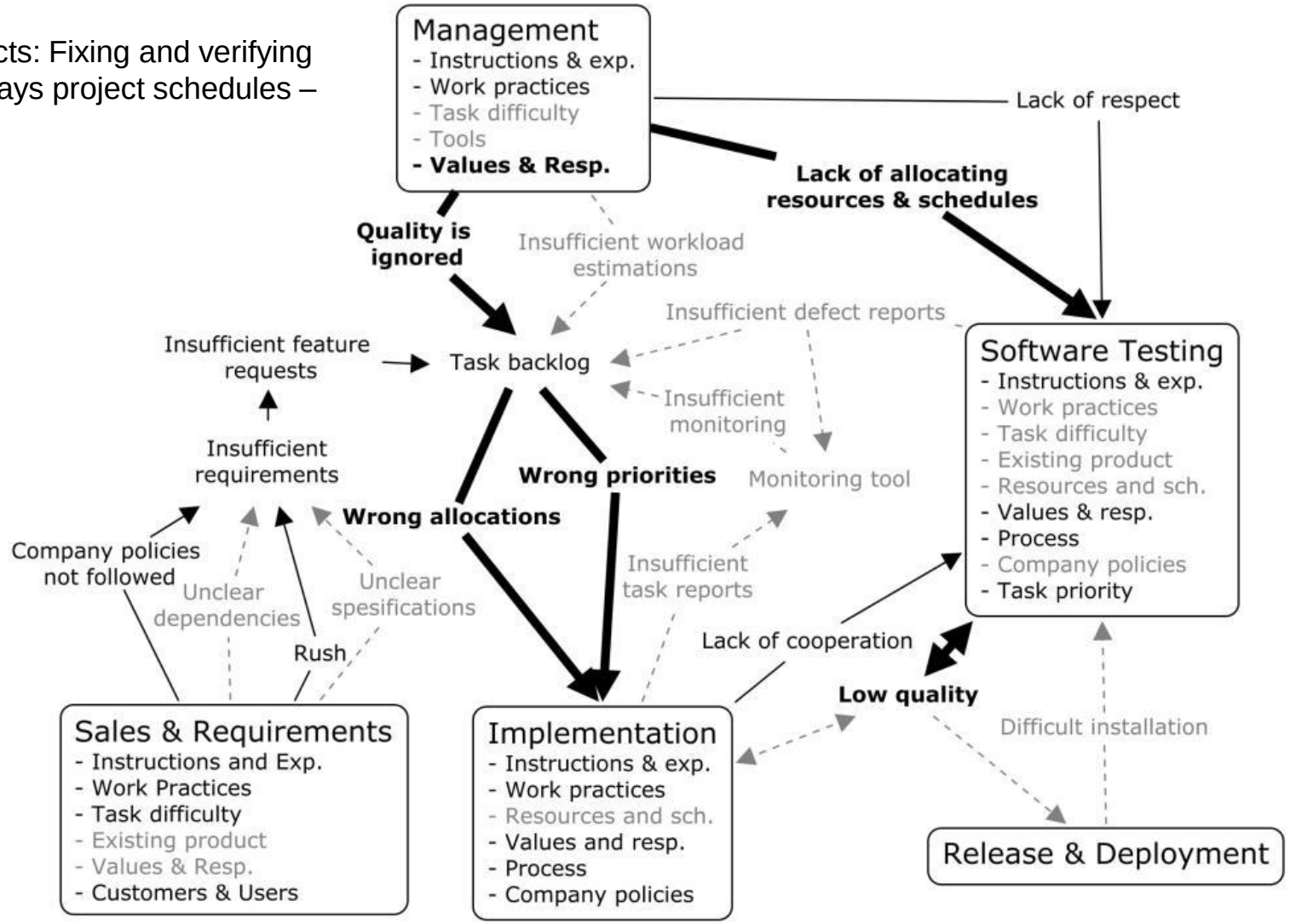
Sporadic development is done when the inconveniences overcome acceptable losses.

Even if the test process feedback is collected, it is often neglected if the process is “good enough”.



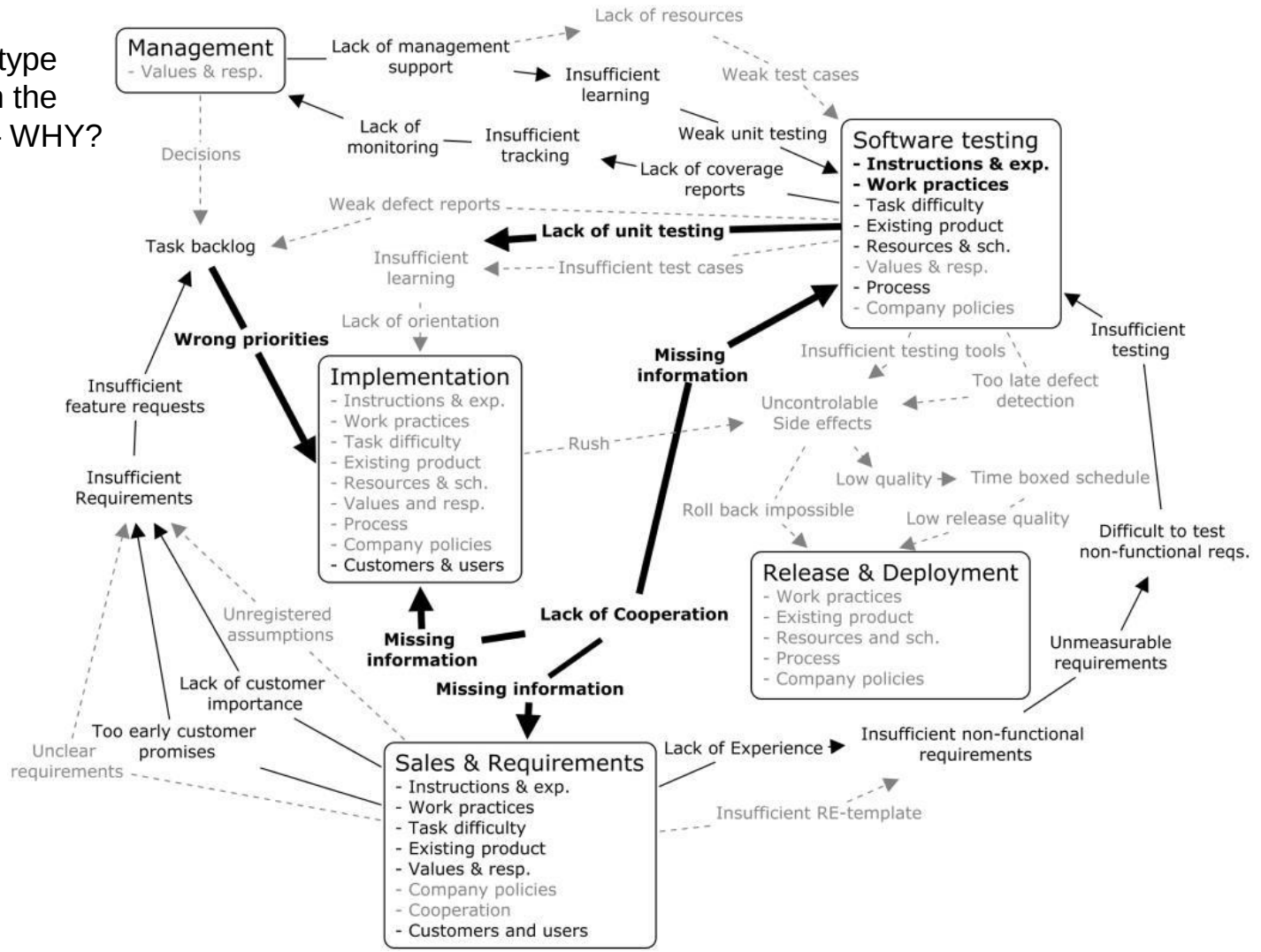
Case Defects: Fixing and verifying defects delays project schedules – WHY?

What caused the failures?

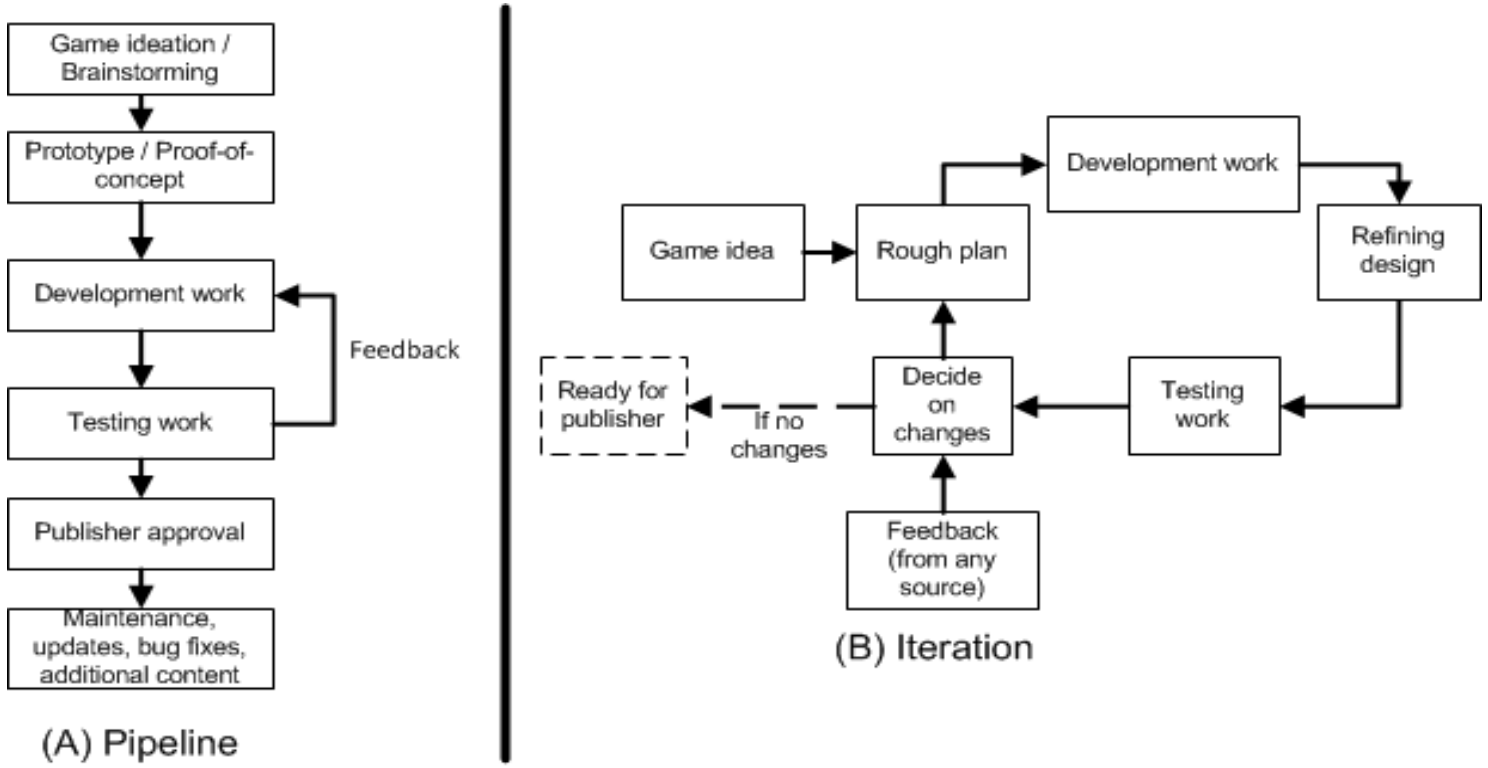


Case Quality: Blocker type defects are detected in the product after release – WHY?

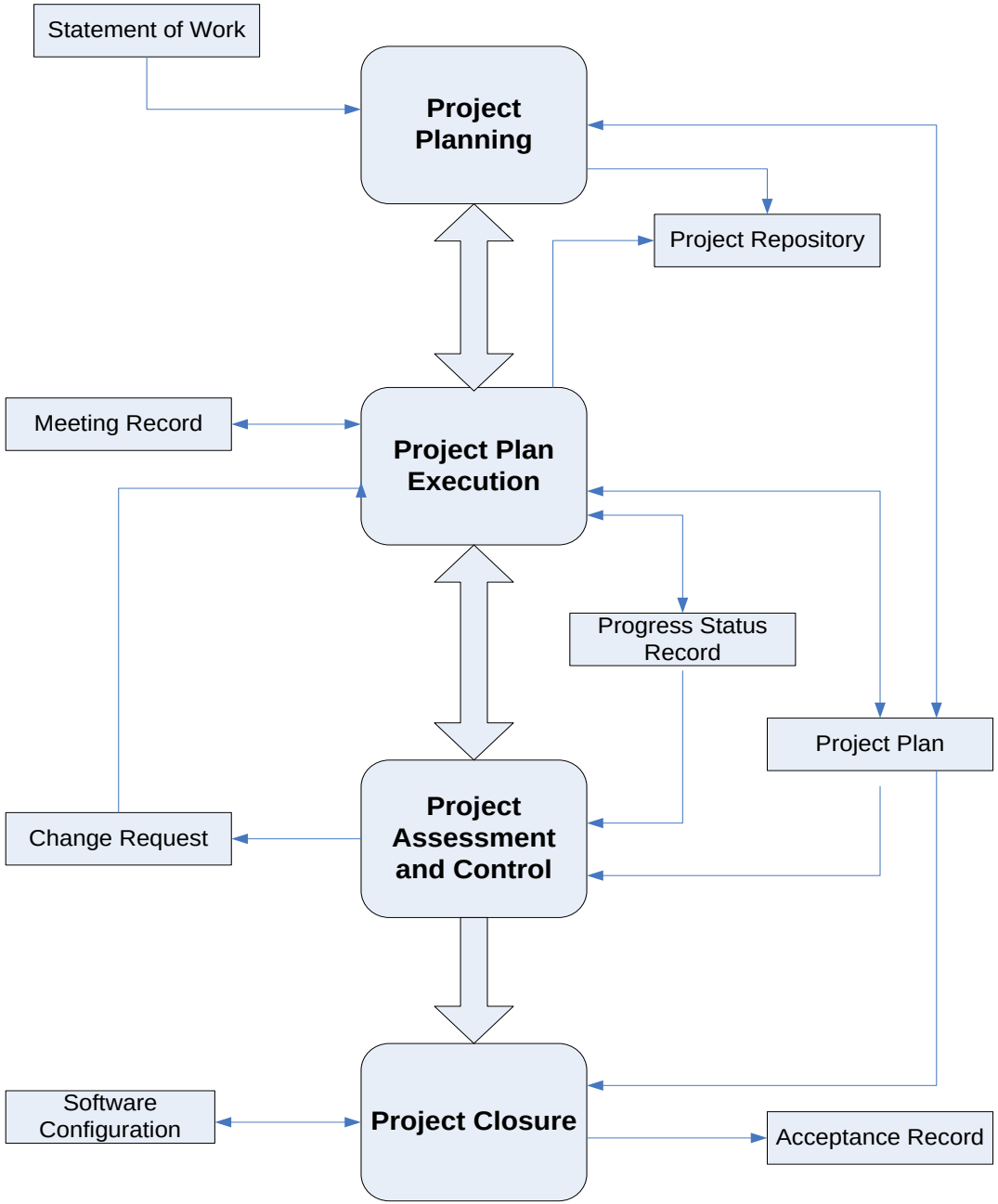
What caused the failures?



Workflows: example, development methods in games



ISO/IEC 29110 For small organizations



Conclusions

- The industry applies design and measurement methods they feel justified. No design for design's sake.
- Measurements to predict costs, problems and resource needs, assess reasonable amount of work to get things done.
- Prototypes to do the proof-of-concept models, actual modeling work to set up product lines or reusability.
- The key aspect is **timely availability and correctness of information.**
  - Models, prototypes, measurements, requirements gathering, customer meetings, testing work all tools to generate relevant and correct information.

LUT
University