Verification and Validation

Tools and Techniques for Software Testing - Barbara Russo

SwSE - Software and Systems Engineering group

unibz

Freie Universität Bozen
Libera Università di Bolzano
Università Liedia de Bulsan

Verification and Validation

- Software is not perfect as it is created by human beings

- *Verification and Validation are processes that use techniques and methods to ensure the final product quality*

- Testing is one of these processes

Verification and Validation

- What is Validation?
- What is Verification?

- Are they synonyms? Is there any difference?

- Mentimeters www.menti.com (www.mentimeter.com)

Verification and Validation

- Are they synonyms? **No**
- Is there any difference? **Yes**

- Verification is:

- Validation is:

Verification

- *Check the consistency of an implementation with a specification*

- It checks "How" i.e., the process of building
  - **Are we building the product right?" (B. Boehm)**

- Example: A music player plays (it does play) the music when I press Play

Validation

- *Check the degree at which a software system fulfills user/customer's requirements*

- It checks "What", i.e., the product itself
  - **Are we building the right product ? (B. Boehm)**

- Example: A music player plays a song (it does not show a video) when I press Play

Usefulness vs. dependability

- Requirements are goals of a software system
- Specifications are solutions to achieve such goals

  - **Validation**: Software that matches requirements ⇒ **useful software**
  - **Verification:** Software that matches specifications ⇒ **dependable software**

Example

- **Requirement (goal)**
  - an application must be used in any circumstance
- **Specification (solution)**
  - an application is mobile

Example

- **Requirement (goal)**
  - a music player plays a list of songs of an author
- **Specification (solution)**
  - a music player reproduces an author's playlist from iTunes

Dependability

- *Dependability is the degree at which a software system complies with its specifications*

Examples

- Unit tests cover 75% of code
- Methods have been implemented to cover 95% of the specifications
- Classes cover 60% of the data structures

Make your own example

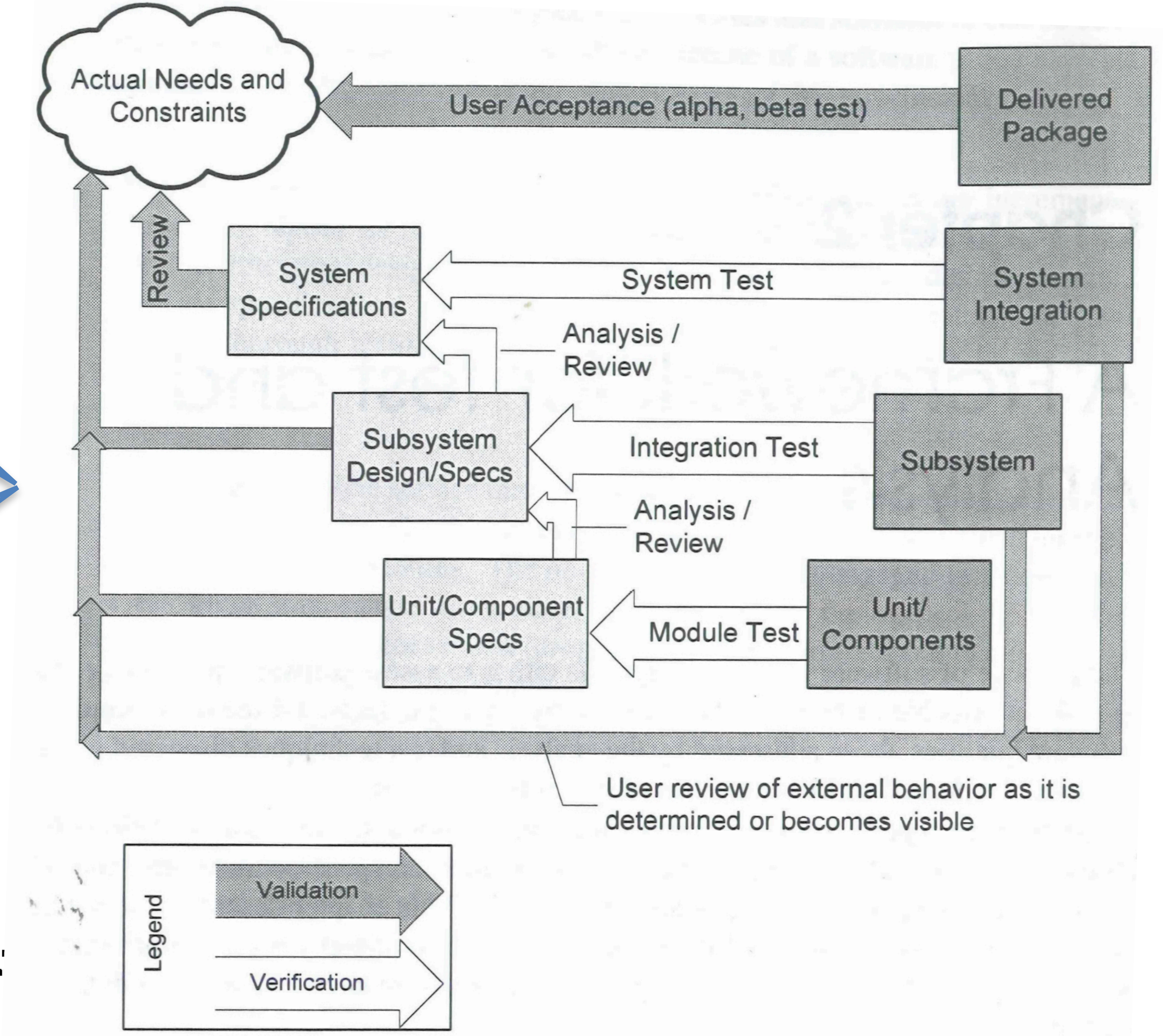
- Go to:

- menti.com

Verification and validation activities

Do you know any of them?

What do they have in common?



Exercise

- **What is what (Ver or Val)?**
  - **Acceptance test (with customer):** negotiated with the customer. It defines the input and the output of each software feature
  - **alpha test (acceptance test with user)**: performed by users in a controlled environment. Evaluate the operational profile as defined by the organisation
  - **beta test (acceptance test with user)**: performed by users in a their own environment. Capture real operational profiles

Testing as a verification process

Readings

- Pezzè & Young, Software Testing and Analysis: Process, Principles and Techniques,Wiley, 2007. University Shelf ST 233 P522, Chap.1-4, 5-6 8-12 17, access from unibz library 15- Textbook Collection ST 233
- Chapter 1

Types of Verification process

- **Software analysis and review** are verification processes to examine a software artifact and to approve it
- **Software testing** is a verification process that detects differences between existing and required conditions and to evaluate the features of the software item

IEEE definition

What is the relation between testing and dependability?

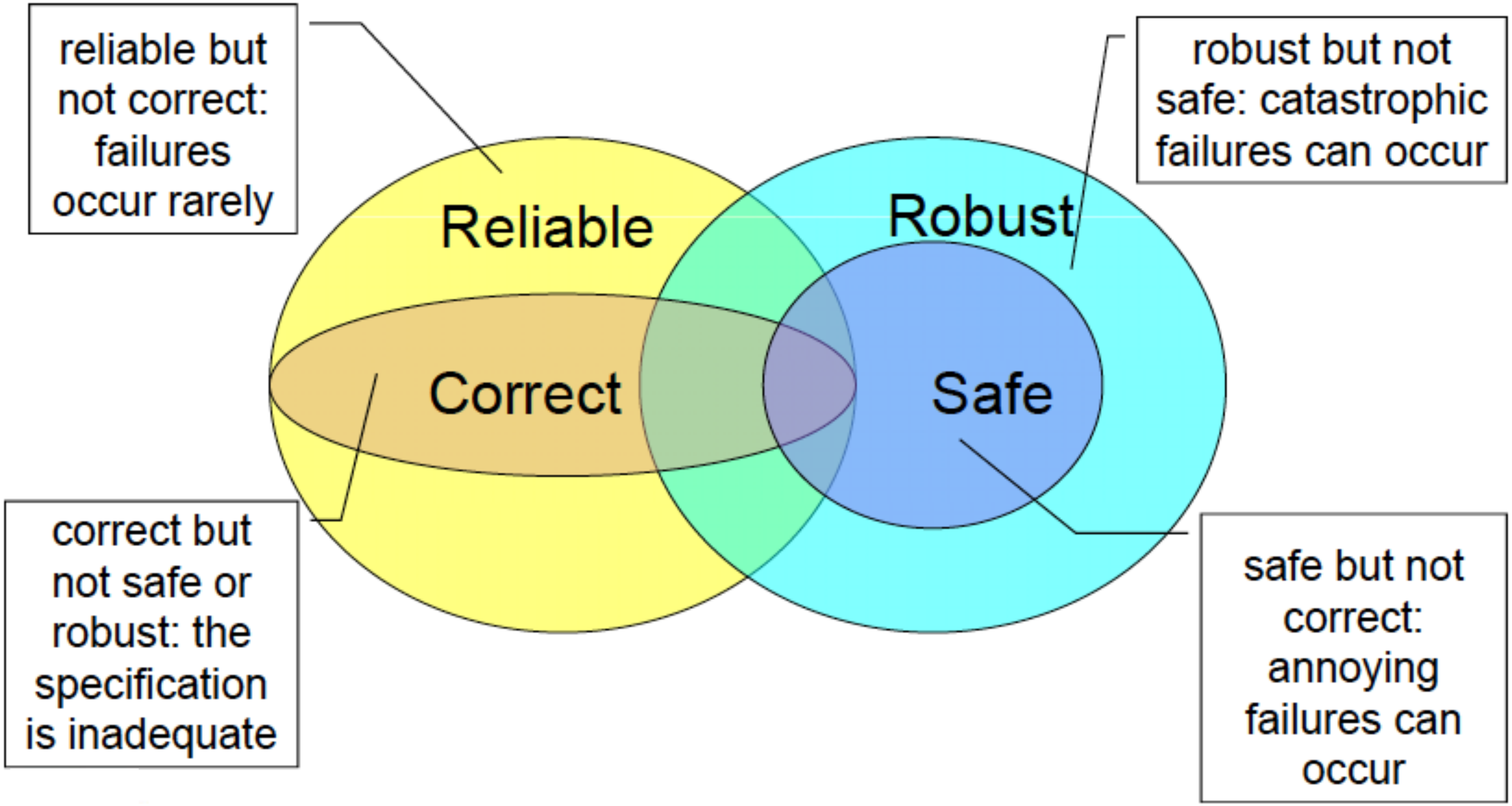
Goal of testing

- Testing aims at verifying four software **dependability properties**:
  - **Correctness**: consistency with specification
  - **Reliability**: statistical approximation to correctness; probability that a system deviates from the expected behavior

Goal of testing

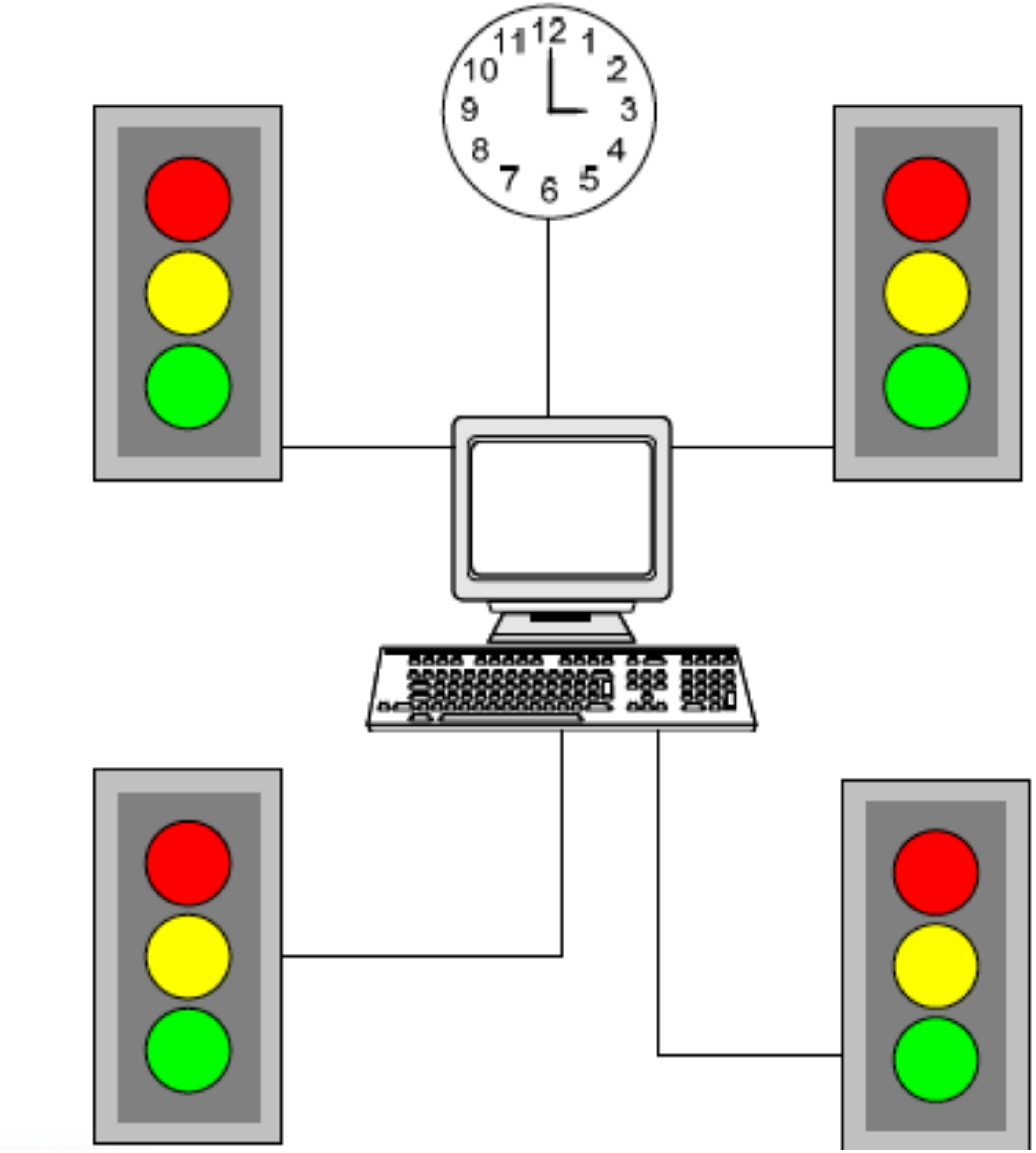
- **Robustness:** being able to maintain operations under exceptional circumstances of not full-functionality
- **Safety:** robustness in case of hazardous behavior (e.g., attacks)

Relations


reliable but not correct: failures occur rarely
robust but not safe: catastrophic failures can occur
Reliable
Robust
Correct
Safe
correct but not safe or robust: the specification is inadequate
safe but not correct: annoying failures can occur

unibz Freie Universität Bozen Libera Università di Bolzano Università Liedia de Bulsan

Source: Mauro Pezze' and Michal Young



- **Reliability:** built according to central scheduling and practice
- **Robustness, safety:** degraded function when possible; never signal conflicting greens
  - Blinking red / blinking yellow is better than no lights;
  - No lights is better than conflicting greens

Source: Mauro Pezze' and Michal Young

Testing techniques

- Testing is a process
- Different testing techniques can be used all along the process

Specification Self-consistency

- Pay attention testing does not question specifications!!! Thus, it can be affected by specifications that do not have:
  - **Consistency**: Specification vs specification, no conflicts
  - **No ambiguity**: open to interpretations, uncertainty
  - **Adherence to standards**: consistency with benchmarks

Application vs. testing specs

- Application specification:
  - Show list of ongoing auctions by vocal command
- Testing specifications:
  - At the vocal command “Show auctions,” a list of auctions $X_1, \ldots, X_n$ that are ongoing is displayed on the screen
  - At the vocal command “Show,” the question “what?” is replayed

What is the requirement?

What is different?

Checking dependability

- *How can we check whether our software satisfies any of the dependability properties?*

- Can we use a “proof”?
- For example, correctness: given a set of specifications and a program we want to find some logical procedure **(e.g., a proof)** to say that the program satisfies the specifications

Undecidability of problems

Some problems cannot be solved by any computer program (Alan Turing)

The halting problem

Given a program P and an input I, it is not decidable whether P will eventually halt when it runs with that input I or it runs forever

Verifying a program

- Undecidability implies that given a program P and a set of verification techniques, *we do not know whether the techniques can verify the program in finite time*
- ... and even when it is feasible it might be very expensive

Inaccuracy of verification

- Thus, verification is inaccurate and can be expensive
- => E.g., modern testing uses automation

Inaccuracy of verification

- Thus, techniques for verification are inaccurate when checking **dependability properties:**

- A verification technique has **optimistic or pessimistic inaccuracy**

- Verification starting point: specify the technique and the dependability property

Optimistic Inaccuracy

- A technique that verifies a dependability property **can return TRUE on programs that do not have the property (FALSE POSITIVE)**

Example

- Testing is optimistic as it returns that a program is correct even if no finite number of tests can guarantee correctness
- Positive: *a program is correct*

Pessimistic Inaccuracy

- Pessimistic inaccuracy: technique that verifies a property **S can return FALSE on programs that have the property (FALSE NEGATIVE)**

Example

- Old test cases can have pessimistic inaccuracy for *robustness/safety* as they may return FALSE on newer versions of the system although they are robust/safe (e.g., the newer versions have implemented new specifications that include hazard)

Accuracy: confusion matrix

Predicted by the technique

Truth

	Pred. TRUE	***Pred. FALSE***
TRUE	*TP*	*FN*
FALSE	*FP*	*TN*

Examples - false positive

- As the exception expectation is placed around the whole test method, this might not actually test what is intended to be tested

```
@Test(expected = FooException.class)
public void testWithExceptions() {
   foo.prepareToDoStuff();
   foo.doStuff();
}
```