Dependability properties

Software Reliability and Testing - Barbara Russo
SERG - Laboratory of Empirical Software Engineering

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

JMSE Summer school, Koblenz, July 12-13 2016

Quality goals

- Product
  - Internal qualities
  - External qualities
    - Usefulness qualities
    - **Dependability qualities**

Dependability qualities

- We have seen:
  - Correctness
  - Reliability
  - Robustness
  - Safety

- Specification Self-Consistency

Correctness

- Software behaviour can be only successful or failing
  - A program cannot be 30% correct
- A program is correct if all its possible behaviours are successes

Correctness

- A program is **correct** if it is consistent with all its specifications
  - Specifications can be very bad defined!
  - Correctness is seldom practical

Reliability

- **Statistical approximation to correctness**: probability that a system deviates form the expected behaviour
  - Likelihood against given specifications
- Unlike correctness it is defined **against an operational profile of a software system**

Measures of Reliability

- **Availability:** the portion of time in which the software operates with no down time
- **Mean Time Between Failures:** The time elapsing between two consecutive failures.

Robustness

- A system **maintains operations** under exceptional circumstances
- It fails “softly” outside its normal operating parameters
- It is “fault tolerant”
  - Despite faults, it operates

Robustness example

- **Unusual circumstance:** unforeseen (not in the specifications) load of users accessing a web site
- **Robust software:**
- Maintain the same throughput speed while stopping last arrived users until the load is decreased
  - It does not decrease performance

Robustness example

- **Action to be taken to increase robustness:** Augment software specifications with appropriate responses to given unusual circumstances (enrich the operational profile with unlikely situations)

Safety

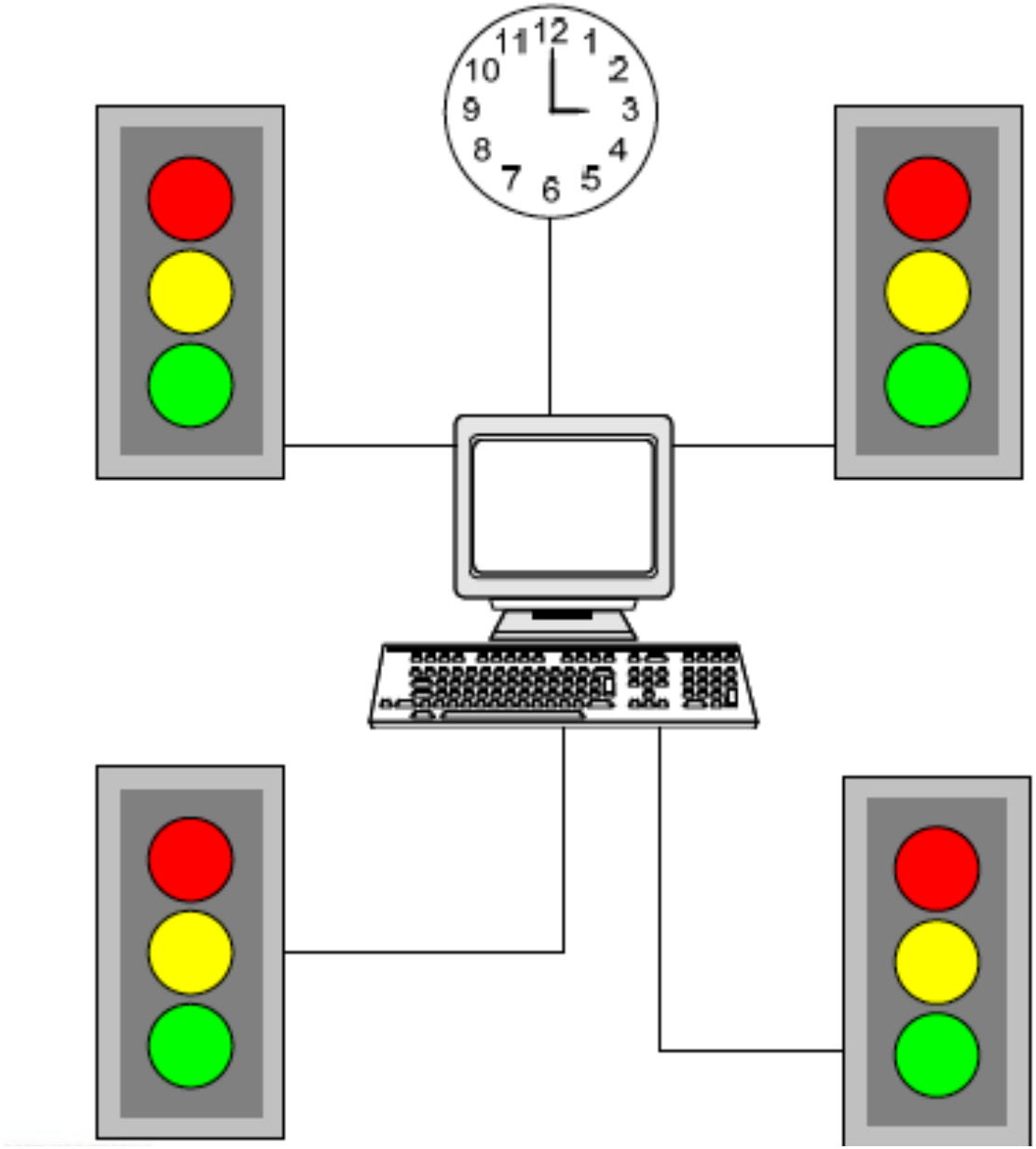
- Robustness in case of hazardous behaviour (hazard)
- Very focused on functionalities that can determine hazards, not concerned with other issues on functionalities
- Often needed in critical systems, but not only:
  - Word crashes -> reliability
  - Word crashes and corrupts existing files -> safety

Hazard

- Safety is meaningless without a specification of hazards
- Identify and classify hazards for the given software

Hazard

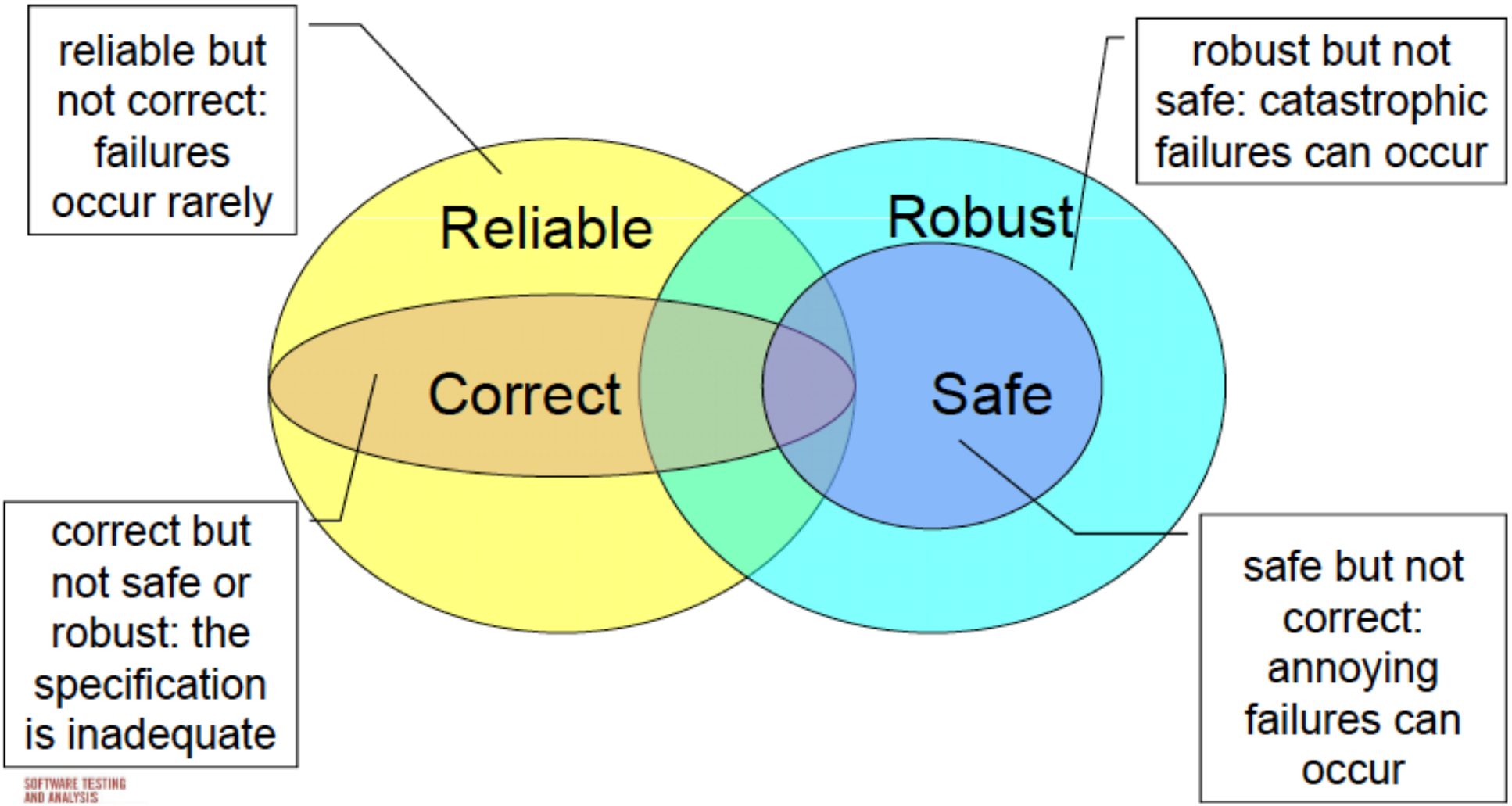
- Do it in the embedded environment (often hazards are related to specific environmental circumstances)
  - Separate the analysis of hazards from any other verification activity



- **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

Relations


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

Exercise

- Analyse an issue report and classify it in terms of the properties
- Find examples of
  - correct but not safe,
  - reliable but not correct,
  - robust but not safe,
  - safe but not correct

Self-Consistency

- Consistency (Specification vs specification, no conflicts)
  - No ambiguity (open to interpretations)
  - Adherence to standards

Which type of issue is this?

Problem Statement:
For many downloads (especially large files) the publisher often releases MD5 checksums in order to assure that the download arrives as intended. The problem is that this checksum often requires a manual check by the recipient of the download. The recipient is often unwilling or unable to do this verification and continues on assuming the download is good, without actually checking.

Relates to Spec section:
http://dev.w3.org/html5/spec/links.html#links-created-by-a-and-area-elements
http://dev.w3.org/html5/spec/text-level-semantics.html#the-a-element

Possible Solution:
A possible solution would be to include the checksum as a machine readable attribute or tag that specifies the checksum and algorithm for the user agent to verify the download after the download finishes, giving instant feedback to the user.