Software Reliability and Testing

Summer semester 2013/2014
Fundamentals

prof. Barbara Russo

February 27, 2014

Table of Contents

Verification and Validation

- Software products is imperfect. It is created by human beings
- Verification and validation practices are a set of methods to ensure the final product quality
- Among them software analysis and testing is a technique to analyze a software item and detect the differences between existing and required conditions and to evaluate the features of the software item

Definition of Verification

- Check of consistency of an implementation with a specification
- It is about "How" – 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

Definition of Verification

- Check of consistency between two descriptions (roles) of the system at different stages of the development process, e.g.,
  - UML class diagram and its implementation in code
  - Specification document and UML class diagram
- Chain of Two Roles: Specification $\rightarrow$ Implementation (Specification) $\rightarrow$ Implementation

Definition of Validation

- Degrees to which a software system fulfils the user requirements
- It is about "What" – the product itself
- "Are we building the right product ? (B. Boehm)
- A music player plays a music (it does not show a video) when I press Play

Usefulness vs. dependability

- Requirements are goals of a software system
- Specifications are solution to achieve the goals / user needs (requirements)
  - Software that matches requirements $\rightarrow$ **useful** software
  - Software that matches specifications $\rightarrow$ **dependable** software

Definition of dependability

- Degrees to which a software system complies with its specifications (focus on verification)
  - Specifications are solutions to a problem described in the requirement analysis
  - They are prone to defects as they have been written by human beings

Validation vs. Verifications activities

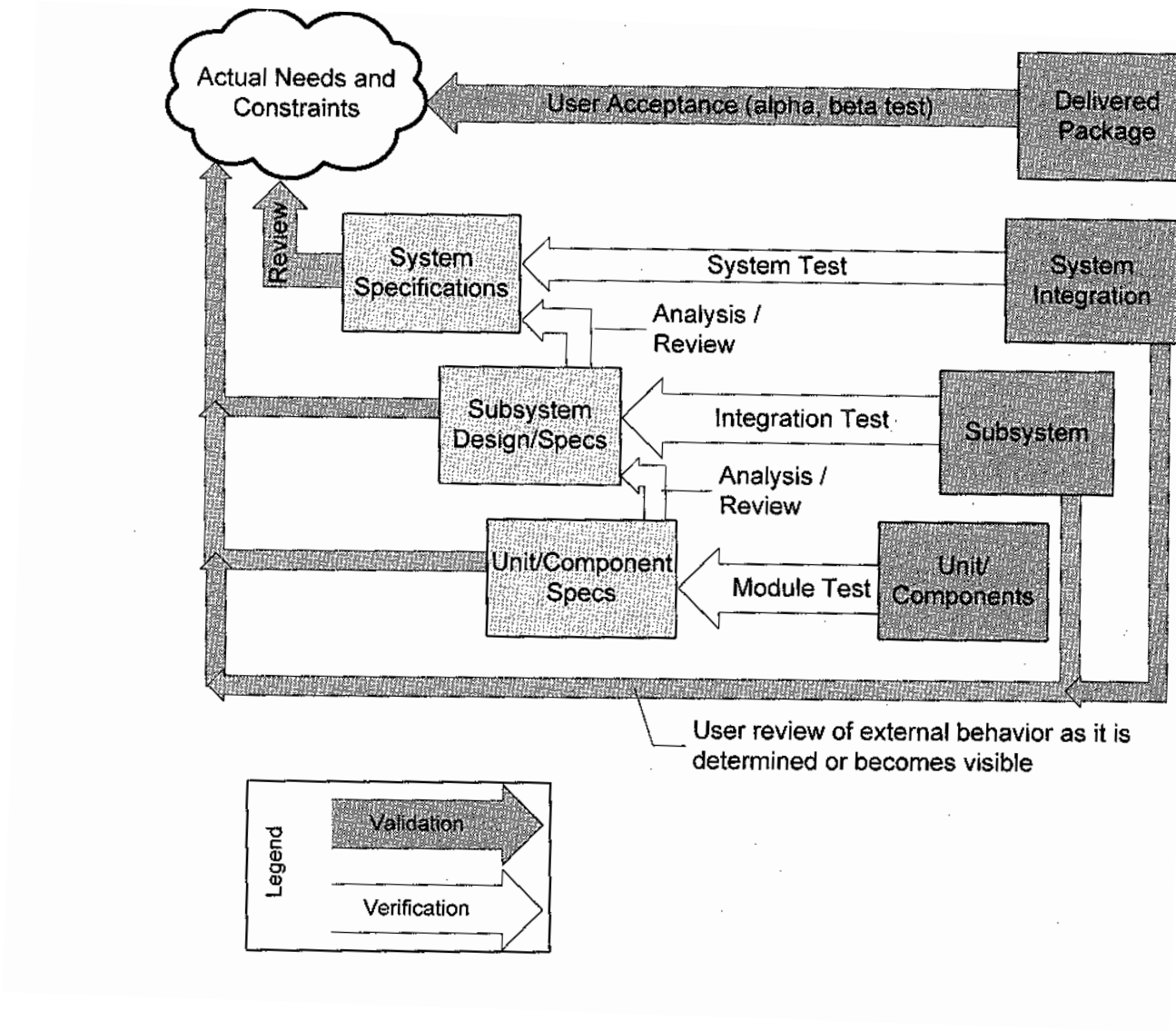


Figure : Verification vs. validation. Source: Pezze and Young, Software Testing and Analysis, Wiley, 2008

Validation vs. Verification

Validation involves stakeholders' judgment

Exercise: Discuss a validation technique

Validation vs. Verification

Verification manly focuses on dependability and concerns five software properties:

- Correctness: consistency with specification
- Reliability: statistical approximation to correctness: probability that a system deviates form the expected behavior
- Robustness: evaluate properties that can maintain system operations also under exception circumstances of not full-functionality
- Safety: robustness in case of hazardous behavior (attacks)
- Self-Consistency
  - Consistency (Specification vs specification, no conflict)
  - Ambiguity (open to interpretations)
  - Adherence to standards

Undecidability of problems

Given a set of specifications and a program we want to find some logical procedure to say that the program satisfies the specifications.

- Alan Turing: some problems cannot be solved by any computer program
- Logical paradox: If there exists a program P that can determine whether for some arbitrary input I and program Q, Q eventually halts, then there should be a program P' that can determine for some arbitrary input I' whether P halts.

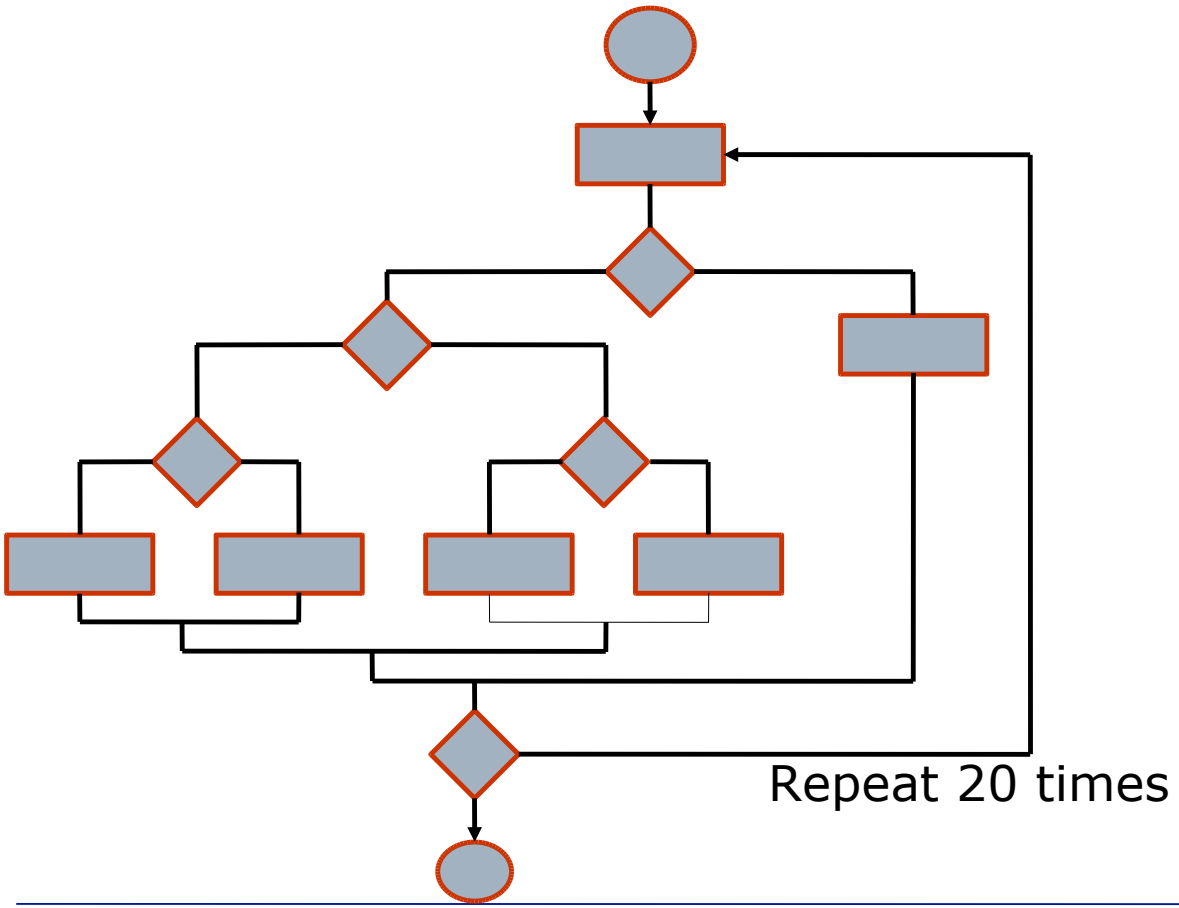
Halting problem

- 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 or it runs forever
- There is no infallible algorithm that can check a property of interest

Checking a property with algorithms

... and even when checking is feasible it might be very expensive

Example



Assume that there are 32 possible different execution flows repeated 20 times. If we execute one test per millisecond, it would take more than two days to test this program!!

Why such number?

- 2 execution flows per decision point
- 5 decision points
- You have to determine all possible ways to combine 32 tokens 20 times…
- Combination with repetition of 32 elements in 20 groups
- See Pressman's book at pg. 448 for further discussion

Analysis of a property

	Analysis output	
	Pos. output	Neg. output
Pos.	TP	FN
Neg.	FP	TN

Exercise: property = correctness. Suppose we know how many classes are correct/incorrect in our program. Use test coverage technique to determine whether a class is correct.

Optimistic and Pessimistic accuracy of techniques

- Undecidability implies that for each verification technique there exists a program that cannot be verified in finite time.
- Techniques for verification are inaccurate:
  - Optimistic inaccuracy: technique that verifies a property S can return TRUE on programs that does not have the property (FALSE POSITIVE)
    - Testing is an optimist technique. It returns that a program is correct even if no finite number of tests can guarantee correctness
  - Pessimistic inaccuracy: technique that verifies a property S can return FALSE on programs that have the property (FALSE NEGATIVE). Conservative technique.
    - Automated programs might be pessimistic.

Analysis of programs for correctness

Assume that we want to verify the property "Correctness:"

- Safe analysis accepts only correct programs
- Optimistic analysis might also return TRUE for non correct programs
- Pessimistic analysis might also return FALSE also for correct programs

Sound analysis for correctness

P= program, A=analysis

- A of P is TRUE $\Longrightarrow$ P is correct

You can substitute any other property.

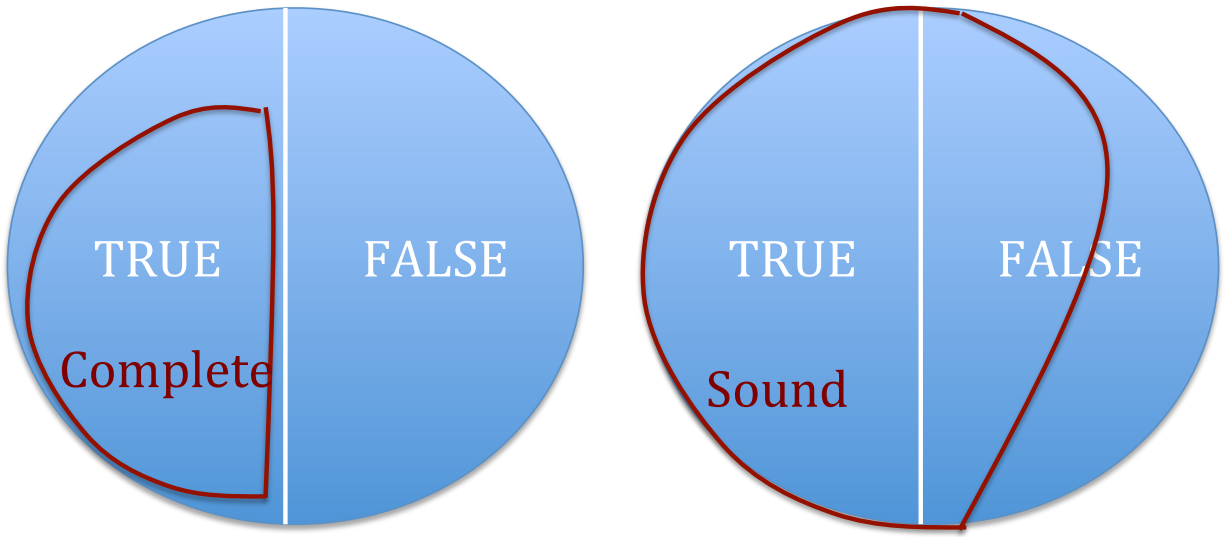
Complete analysis for correctness

P=program A=analysis

- P is correct $\Longrightarrow$ A of P is TRUE

You can substitute any other property.

Sound and complete analyses



Substituting principle

In complex system, verifying properties can be infeasible. Often this happens when properties are related to specific human judgements:

- Substituting a property with one that can be easier verified or contain the class of programs to verify
- Example: "Race condition": interference between writing data in one process and reading or writing related data in another process (an array accessed by different threads). Testing the integrity of shared data is difficult as it is checked at run time. Typical solution is to adhere to a protocol of serialization

Serialization

Definition of serialization:

- When group of objects or states can be transmitted as one entity and then at arrival reconstructed into a the original distinct objects.
- Like the Transporter of Star Trek!!

Serialization

- The 2 phase locking (2PL) is typically used with transactions
  - Expanding phase: locks are acquired and no locks are released (the number of locks can only increase)
  - Shrinking phase: locks are only released (the number of locks can only decrease)

Serialization

Solution for serialization in transactions

- 2PL is a sufficient property for serialization:
  - 2PL $\implies$ serialization,
- but it is not necessary as there are other techniques.
  - serialization $\not\implies$ 2PL ,

Serialization

Another solution is provided by some programming languages

Java object serialization:

- An object can be represented as a sequence of bytes that includes the object's data as well as information about the object's type and its types of data.
- After a serialized object has been written into a file, it can be read from the file and deserialized: the type information and bytes that represent the object and its data can be used to recreate the object in memory.

Java object serialization:

- The ObjectOutputStream class contains:
  `public final void writeObject(Object x) throws IOException`
- The method serializes an Object and sends it to the output stream.
- Similarly, the ObjectInputStream class contains the method for deserializing an object:
  `public final Object readObject() throws IOException, ClassNotFoundException`
- This method retrieves the next Object out of the stream and deserializes it.