The Big Picture

Verification and validation

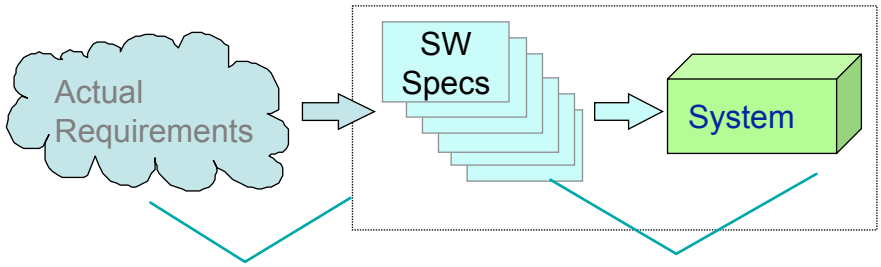
- Validation:
  does the software system meets the user's real needs?

  are we building the right software?

- Verification:
  does the software system meets the requirements specifications?

  are we building the software right?

Validation and Verification



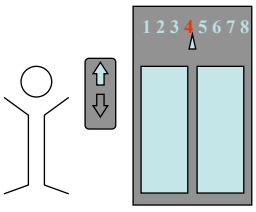
Validation

Includes usability testing, user feedback

Verification

Includes testing, inspections, static analysis

Verification or validation depends on the specification

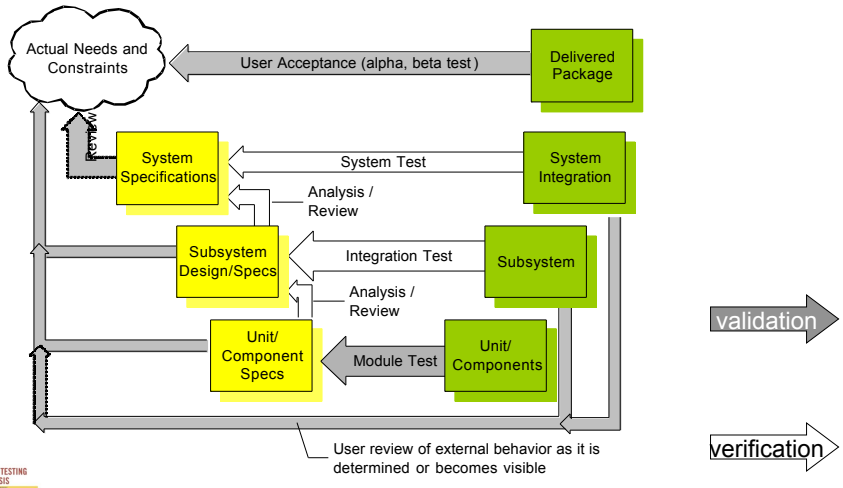


Example: elevator response

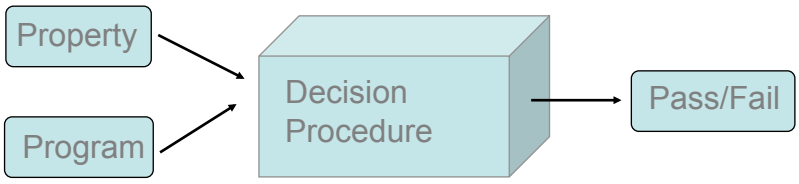
Unverifiable (but validatable) spec: ... if a user presses a request button at floor i, an available elevator must arrive at floor i soon...

Verifiable spec: ... if a user presses a request button at floor i, an available elevator must arrive at floor i within 30 seconds...

Validation and Verification Activities



You can't ~~always~~ ever get what you want

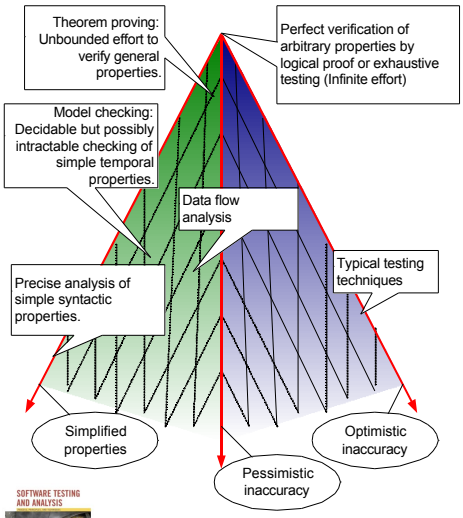


Correctness properties are undecidable

the halting problem can be embedded in almost every property of interest!

In computability theory, the **halting problem** can be stated as follows: "*Given a description of an arbitrary computer program, decide whether the program finishes running or continues to run forever*". This is equivalent to the problem of deciding, given a program and an input, whether the program will eventually halt when run with that input, or will run forever.

Getting what you need ...



- optimistic inaccuracy: we may accept some programs that do not possess the property (i.e., it may not detect all violations).
  - s/w testing
- pessimistic inaccuracy: it is not guaranteed to accept a program even if the program does possess the property being analyzed
  - automated program analysis techniques
- simplified properties: reduce the degree of freedom (eg do not test all possible integer values…)

Some Tricky Terminology

- **Safe**: A safe analysis has no optimistic inaccuracy, i.e., it accepts only correct programs (but may reject some).
- **Sound**: An analysis of a program P with respect to a formula F is sound if the analysis returns true only when the program does satisfy the formula.
  - but it could erroneously return false even if the program does satisfy the formula!
  - If F is an indication of correctness, then same as safe
  - It's tricky to understand what 'sound' means when F is used to indicate a fault.
- **Complete**: An analysis of a program P with respect to a formula F is complete if the analysis always returns true when and only when the program actually does satisfy the formula.