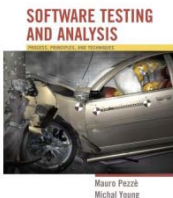
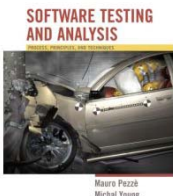


A Framework for Testing and Analysis



Learning objectives

- Introduce dimensions and tradeoff between test and analysis activities
- Distinguish validation from verification activities
- Understand limitations and possibilities of test and analysis



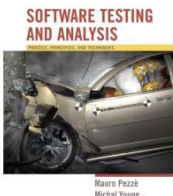
Verification and validation

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

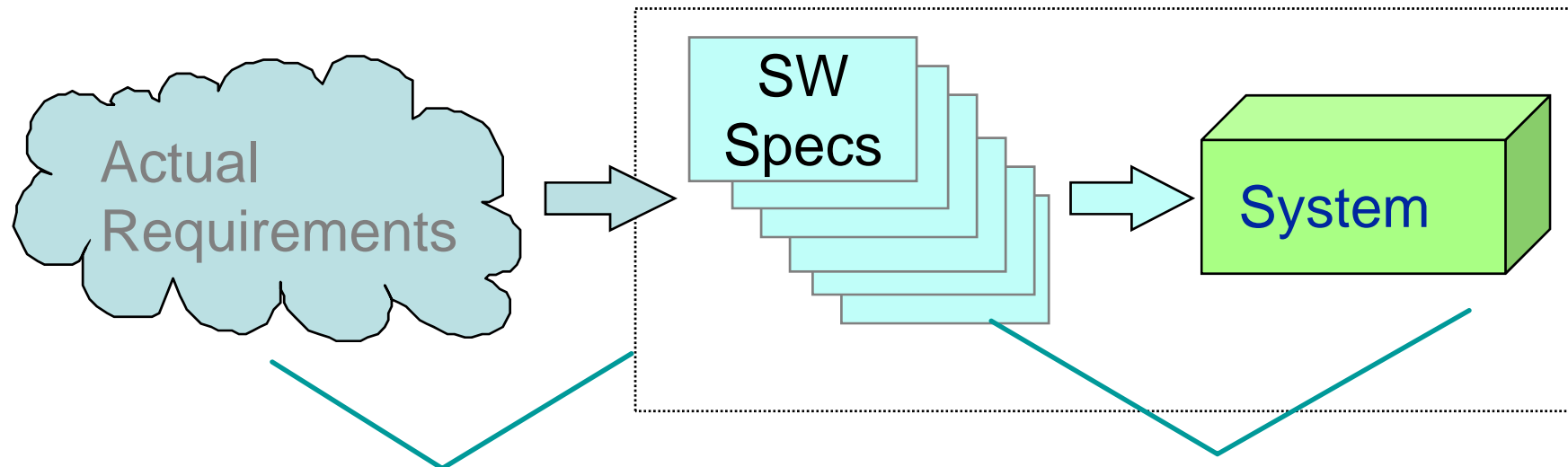
are we building the right software?

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

are we building the software right?



Validation and Verification

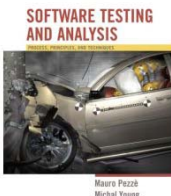


Validation

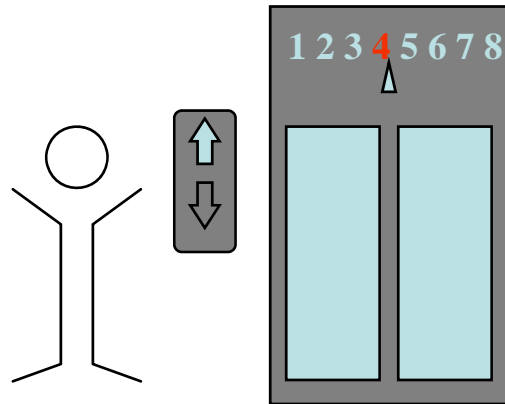
Includes usability testing, user feedback

Verification

Includes testing, inspections, static analysis



Verification or validation depends on the specification

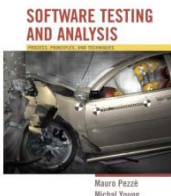
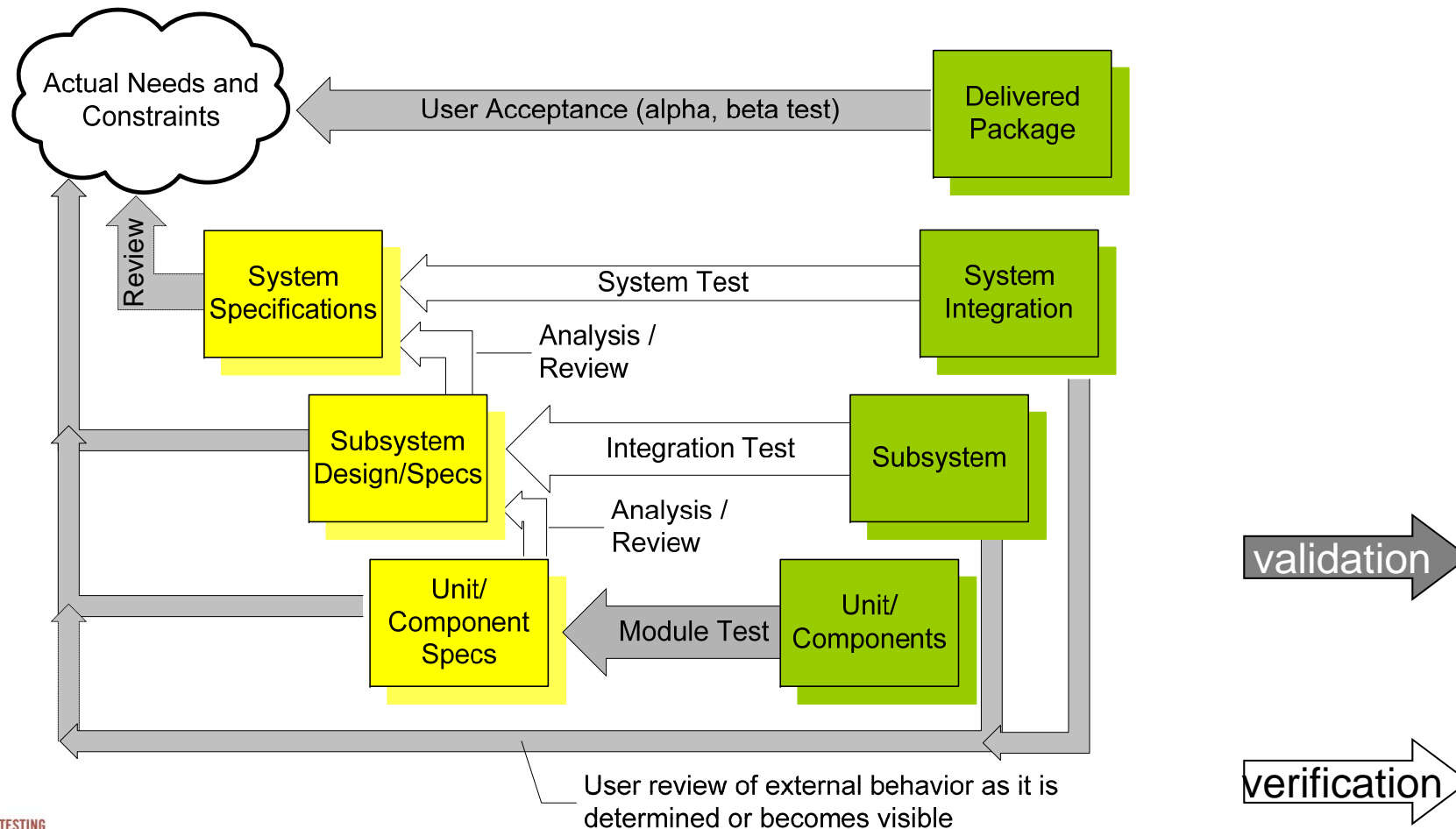


Example: elevator response

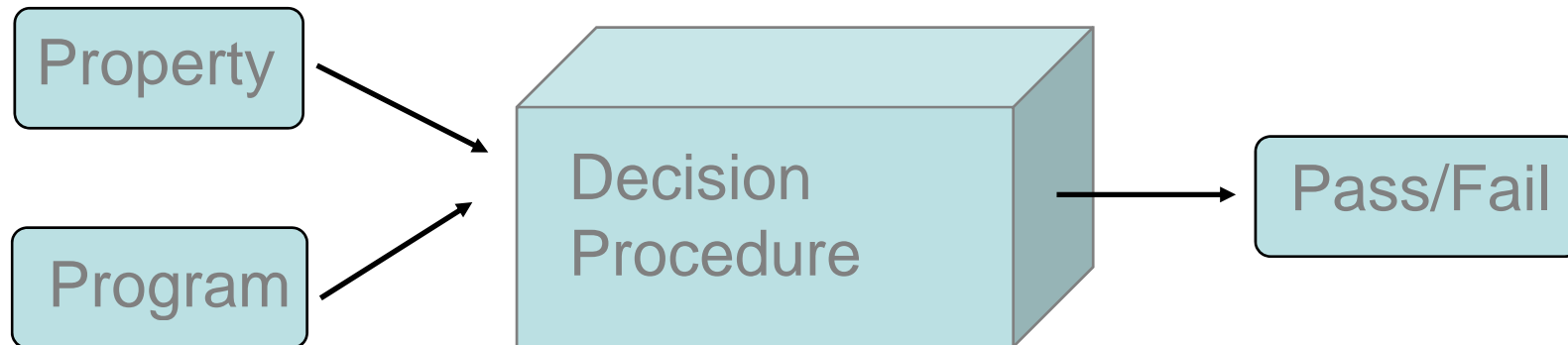
Unverifiable (but validatable) spec: ... if a user presses a request button at floor i , an available elevator must arrive at floor i soon...

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

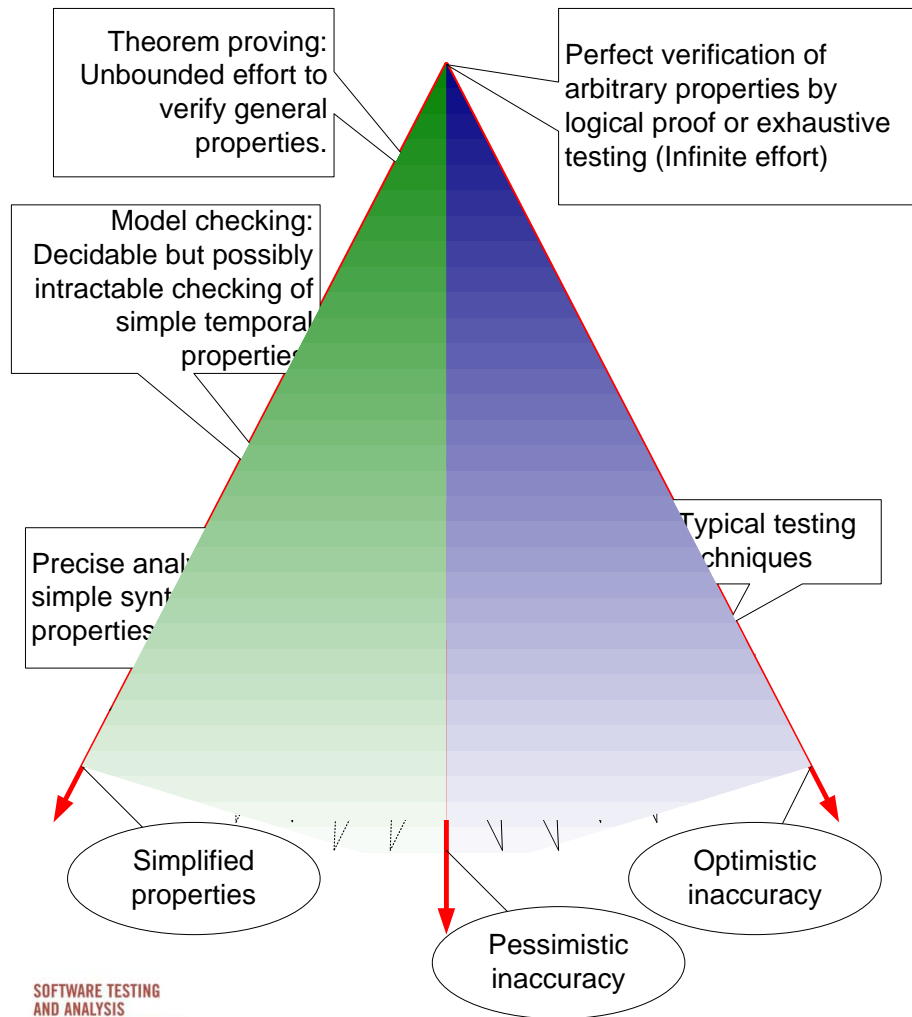


You can't ~~always~~ ^{ever} get what you want

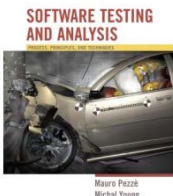


Correctness properties are undecidable
the halting problem can be embedded in almost
every property of interest

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).
 - 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 for simplifying the property to check



Example of simplified property: Unmatched Semaphore Operations

original problem

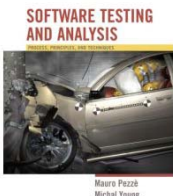
```
if ( .... ) {  
    ...  
    lock(S);  
}  
...  
if ( ... ) {  
    ...  
    unlock(S);  
}
```

Static
checking for
match is
necessarily
inaccurate ...

simplified property

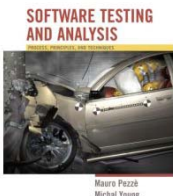
Java prescribes a
more restrictive, but
statically checkable
construct.

```
synchronized(S) {  
    ...  
    ...  
}
```



Some Terminology

- **Safe:** A safe analysis has no optimistic inaccuracy, i.e., it accepts only correct programs.
- **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.
- **Complete:** An analysis of a program P with respect to a formula F is complete if the analysis always returns true when the program actually does satisfy the formula.



Summary

- Most interesting properties are undecidable, thus in general we cannot count on tools that work without human intervention
- Assessing program qualities comprises two complementary sets of activities: validation (does the software do what it is supposed to do?) and verification (does the system behave as specified?)
- There is no single technique for all purposes: test designers need to select a suitable combination of techniques

