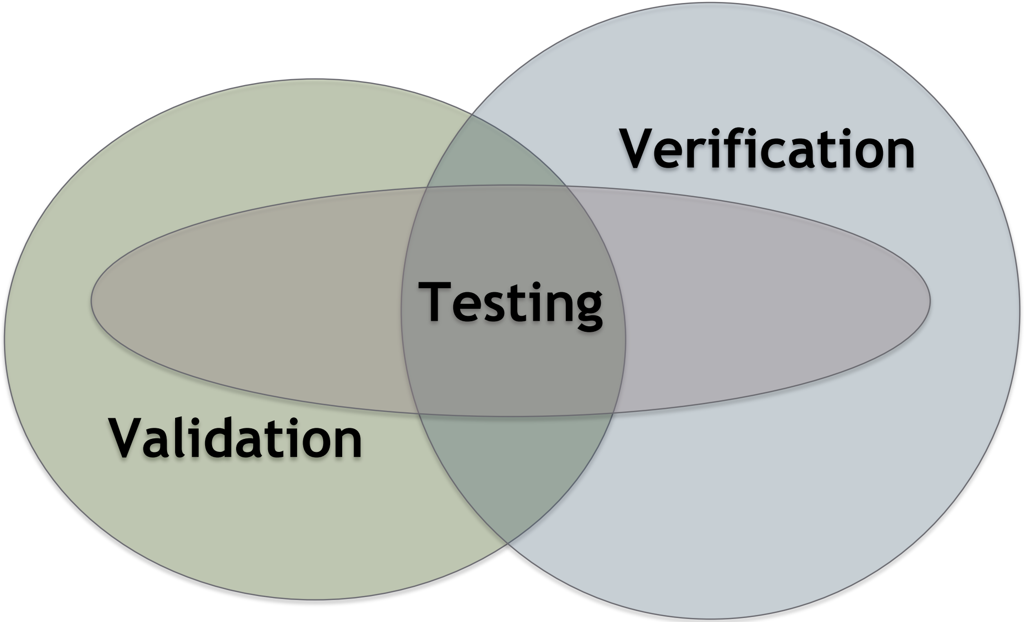
18654

Software Verification and Testing

Terminology



Verification vs. Validation

- **Validation**
  Does the software and its artifacts meet real operational needs?
  Did we build the right software?

- **Verification**
  Does the software and its artifacts meet their respective specifications?
  Are they sound, internally consistent, of high quality?
  Are we building the software right?

Why should we care about SVT?

To avoid **big** software disasters

1982

Therac-25

Radiation dose calculation

1996

Ariane 5 shuttle

Control software

2003

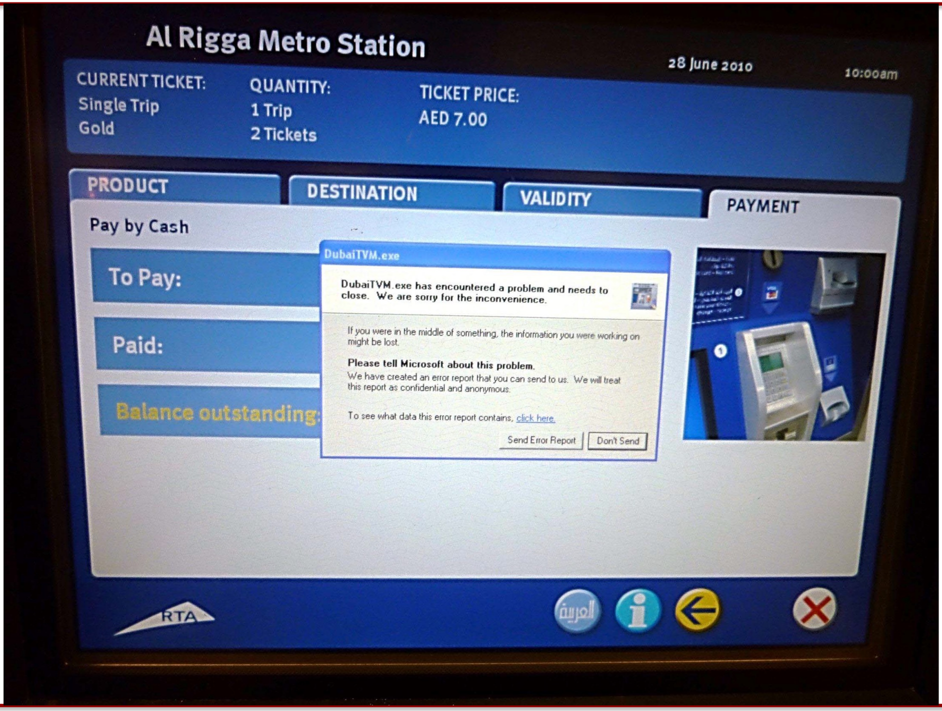
North-east blackout

Alert propagation system

Yes but also…

to avoid small disasters
and for maintaining reputation…

Other examples



Al Rigga Metro Station

28 June 2010

10:00am

CURRENT TICKET: Single Trip Gold

QUANTITY: 1 Trip 2 Tickets

TICKET PRICE: AED 7.00

Pay by Cash

To Pay:

Paid:

Balance outstanding:

DubaiTVM.exe

DubaiTVM.exe has encountered a problem and needs to close. We are sorry for the inconvenience.

If you were in the middle of something, the information you were working on might be lost.

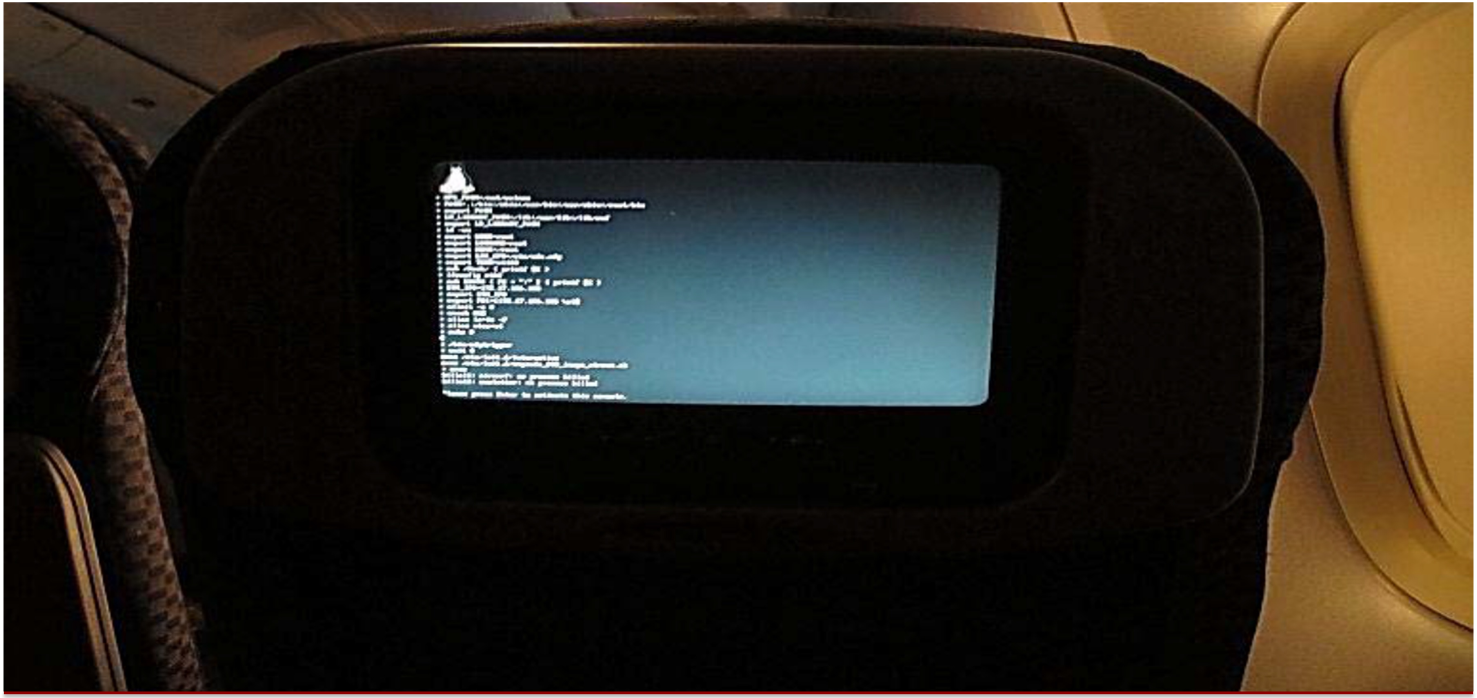
Please tell Microsoft about this problem.

We have created an error report that you can send to us. We will treat this report as confidential and anonymous.

To see what data this error report contains, click here.

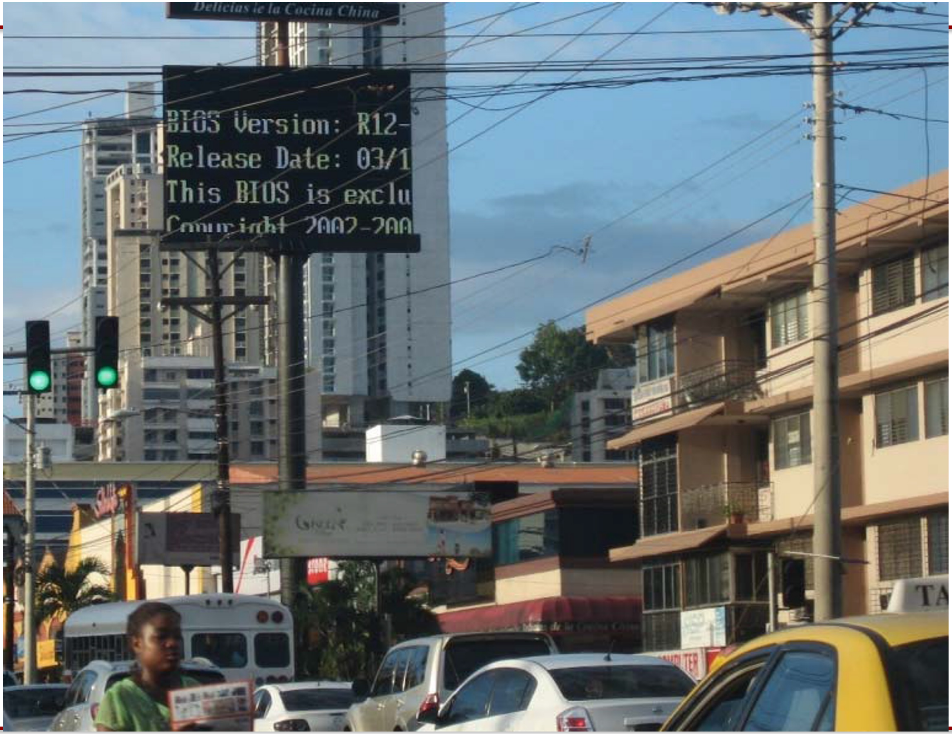
Courtesy of E. Miranda (CMU) – Metro ticket machine crash, Dubai

Other examples



Courtesy of E. Miranda (CMU) – Inflight entertainment system crash

Other examples



Courtesy of E. Miranda (CMU) – Electronic billboard crash, Panama City

Misconceptions about software quality

- Too often quality evaluation is an afterthought
- Performed at the last minute, on a best effort basis with little rigor or planning
- Quality assurance is all about testing
- Quality assurance is performed in isolation, post development
- Quality assurance is too expensive

Courtesy of Jeff Gennari

Why should we care about SVT?

In general

to achieve "acceptable" quality!

All engineering disciplines pair design and construction with activities that check intermediate and final products

What to apply SVT to

- Final products
  - Working code
  - System in operation
- Intermediate artifacts
  - Designs
  - Models/Abstractions
  - Documents
  - Intermediate code

SVT Landscape

	Static	Dynamic
Validation Verification	artifact under investigation is verified/validated based on its static descriptions, without “executing” them	artifact under investigation is verified/validated by “running” it with supplied or derived inputs in an execution environment

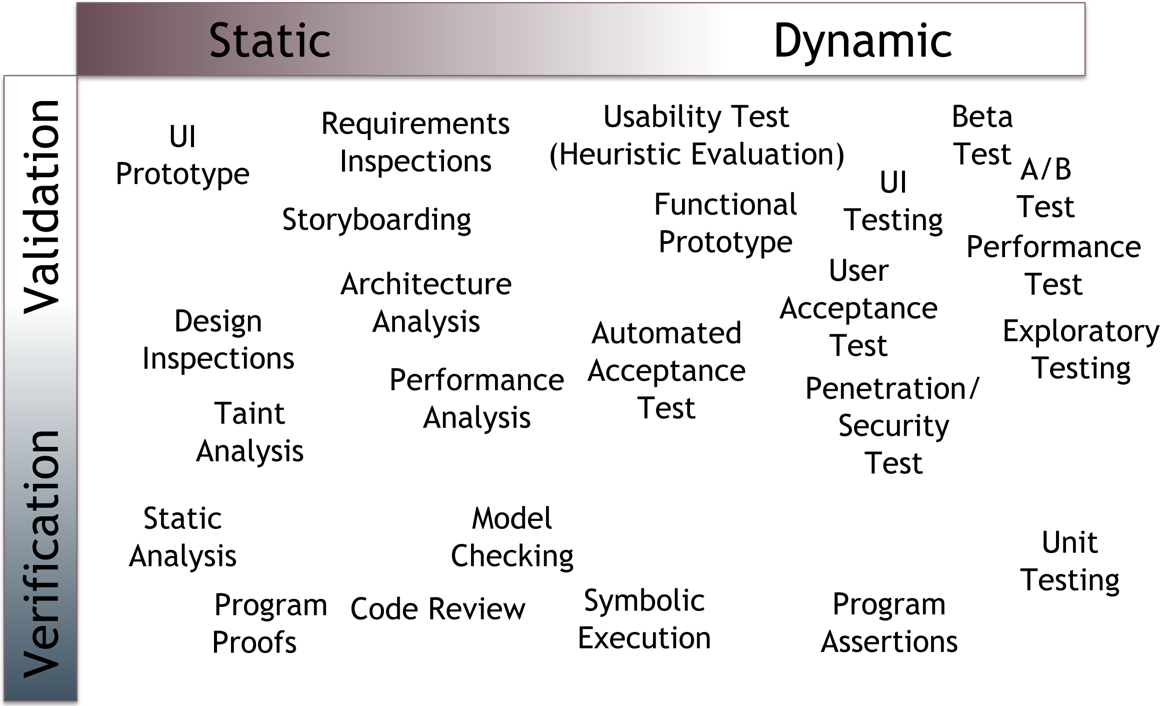
Which dynamic quality practices do you know of?

Which quality practices are dynamic?

- Style-checking code (a type of static analysis)
- Static analysis
- Compilation
- Unit testing ✓
- Paper prototyping
- Acceptance testing ✓
- Code review
- Performance testing ✓

SVT Landscape

Testing are towards the middle or the right of this spectrum



Static

Dynamic

Validation

Verification

UI Prototype

Requirements Inspections

Usability Test (Heuristic Evaluation)

Beta Test

A/B Test

UI Testing

Storyboarding

Functional Prototype

Performance Test

User Acceptance Test

Architecture Analysis

Design Inspections

Automated Acceptance Test

Exploratory Testing

Performance Analysis

Penetration/ Security Test

Taint Analysis

Static Analysis

Model Checking

Unit Testing

Program Proofs

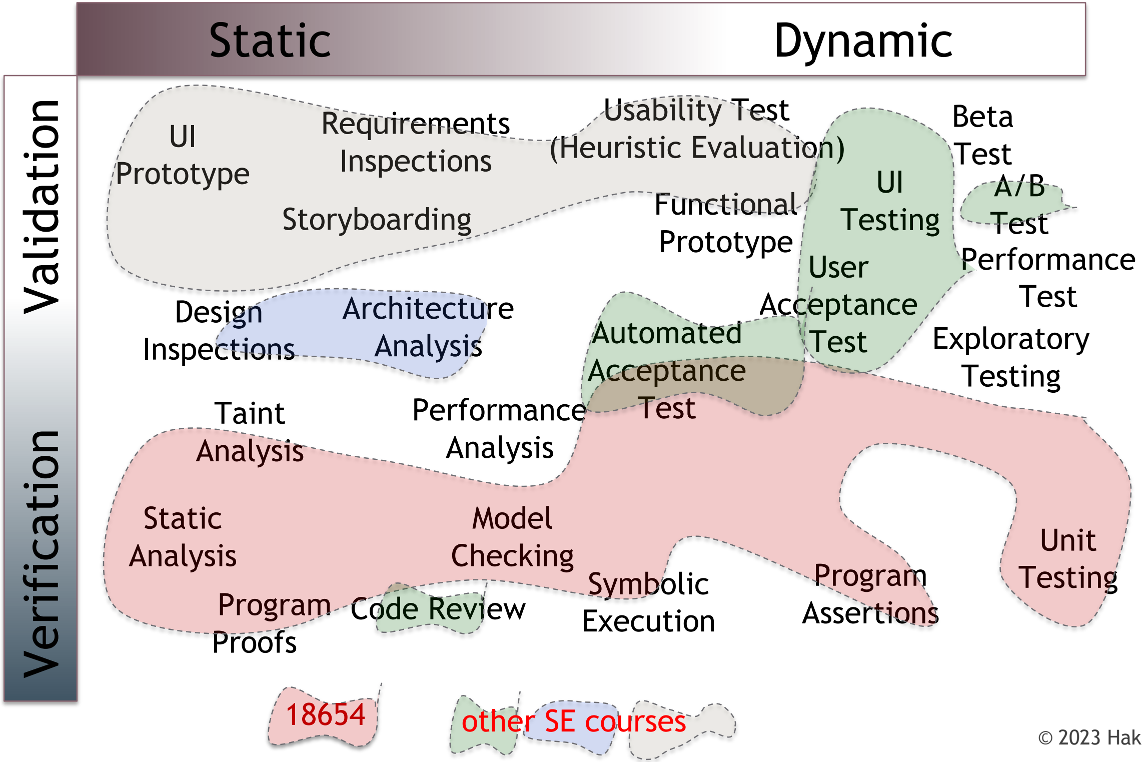
Code Review

Symbolic Execution

Program Assertions

SVT Landscape - Scope of 18654

in this course vs. other courses



We will revisit these main principles recurring throughout the course: think about them when we discuss a new technique

Redundancy

Partitioning

Approximation

Visibility

Feedback

Repeatability

Main Principles of SVT

- **Redundancy**: triangulate results by using different SVT techniques and applying them differently or in an overlapping manner
- **Partitioning**: divide and conquer -- apply STV to subparts, then aggregate results
- **Approximation**: make the SVT problem easier by simplifying the system or SVT task
- **Visibility**: make information about the system accessible to SVT tools; make the health of the system visible to the team
- **Feedback**: SVT should provide actionable results
- **Repeatability**: SVT should give the same result each time (*better to fail every time than only sometimes*)

Flaky test and some other non-deterministic dependencies