Software Quality
&
Software Quality Assurance

IT LOOKS LIKE WE'LL RELEASE OUR NEW PRODUCT ON TIME, DESPITE ITS MANY DEFECTS.
WE'VE MINIMIZED THE ECONOMIC IMPACT OF THE DEFECTS VIA AN ADVANCED BUSINESS PROCESS CALLED "HOPING NOBODY NOTICES."
AND WE'VE DOUBLED OUR PROJECTED INCOME BY MODIFYING OUR ASSUMPTIONS!
A LOT OF THIS JOB IS MENTAL.
www.unitedmedia.com
S.Adams
7/25/96 © 1996 United Feature Syndicate, Inc.(NYC)

Software Quality

A definition of quality should emphasize three important points:

1. Software requirements are the foundation from which quality is measured. Lack of conformance to requirement is lack of quality.
2. Specified standards define a set of development criteria that guide the manner in which software is engineered. If the criteria are not followed, lack of quality will almost surely result.
3. There is a set of implicit requirements that often goes unmentioned (e.g. good maintainability). If software conforms to its explicit requirements but fails to meet implicit requirements, software quality is suspect.

Meet the explicit and implicit requirements – the needs

Good product quality correlates with a good engineering process

[DACS]

Software Testing

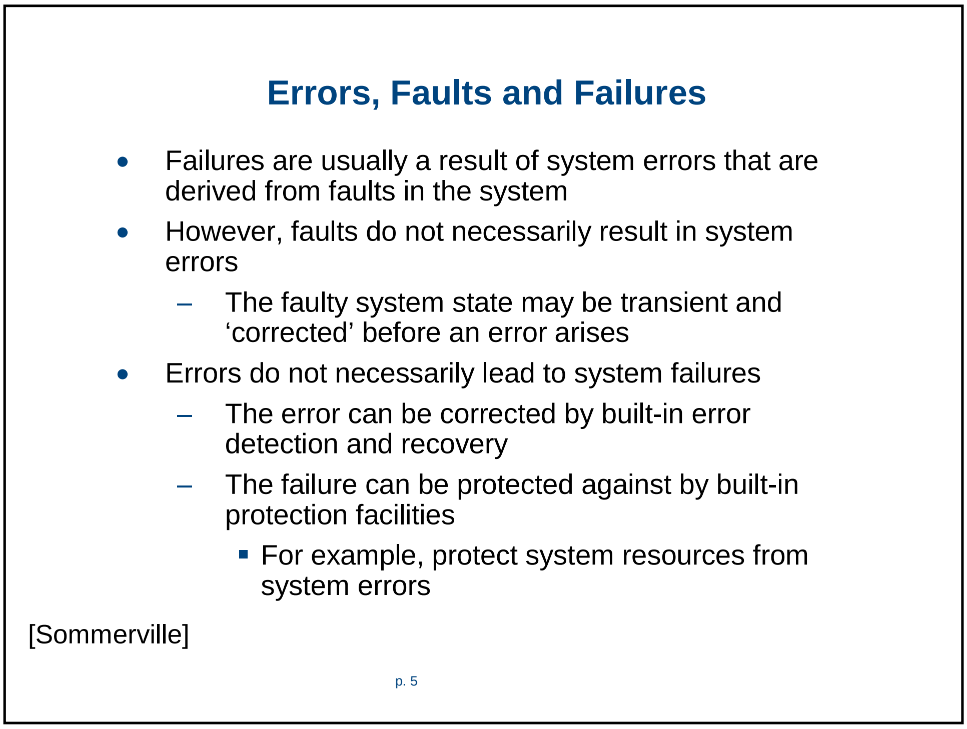
- The purpose of software testing is to assess and evaluate the quality of work performed at each step of the software development process.
- Although it sometimes seems that way, the purpose of testing is NOT to use up all the remaining budget or schedule resources at the end of a development effort.
- The goal of testing is to ensure that the software performs as intended, and to improve software quality, reliability and maintainability.

Software testing is a full-life-cycle assessment of quality

[DACS]

Quality and Testing

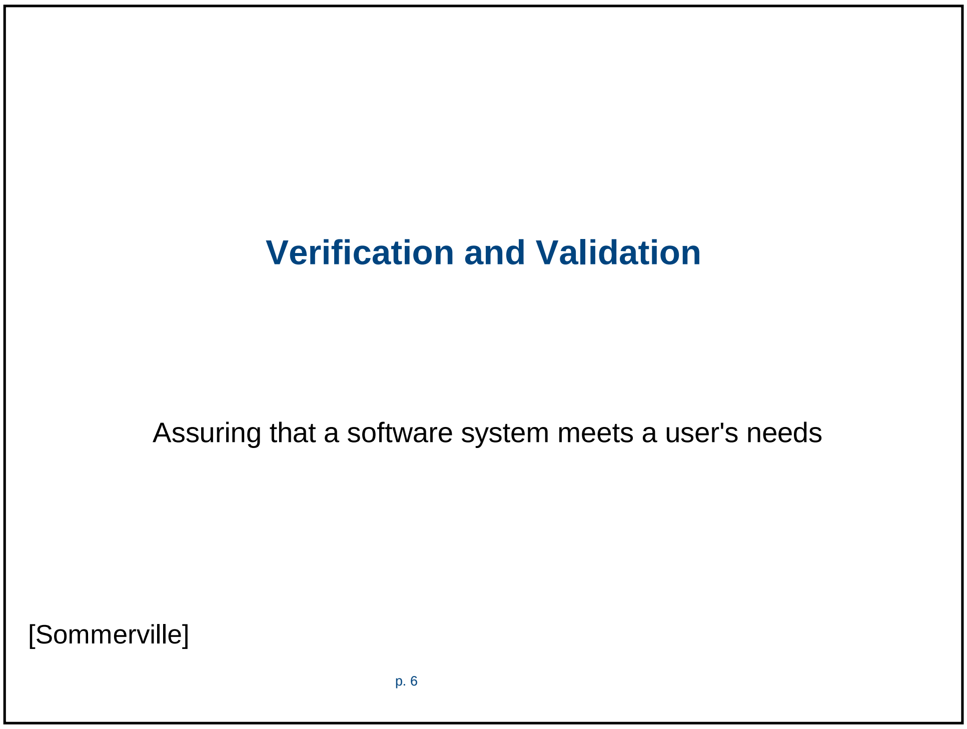
- A good development process, tools, methods, and people go far in providing quality products
- Testing is one aspect of assuring software quality
  - ***It is a measure of quality, it does not deliver quality***
- “Quality cannot be tested into a product”
- ***Software Quality Assurance*** includes
  - Software engineering process improvement
    - Prevent the insertion of defects
  - Fault tolerant software design
    - Tolerate the existence of defects
  - All aspects of software verification and validation
    - Including testing



Errors, Faults and Failures

- Failures are usually a result of system errors that are derived from faults in the system
- However, faults do not necessarily result in system errors
  - The faulty system state may be transient and ‘corrected’ before an error arises
- Errors do not necessarily lead to system failures
  - The error can be corrected by built-in error detection and recovery
  - The failure can be protected against by built-in protection facilities
    - For example, protect system resources from system errors

[Sommerville]



Verification and Validation

Assuring that a software system meets a user's needs

[Sommerville]

Verification vs. Validation

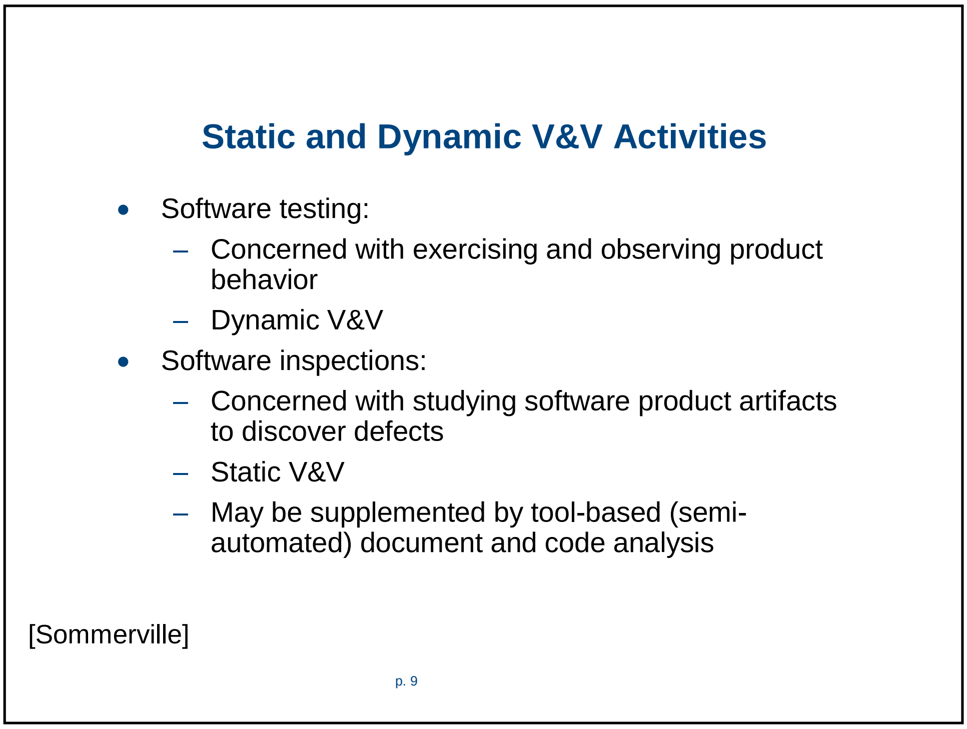
- Verification:
  - “Are we building the product right?”
  - The software should conform to its design
- Validation:
  - “Are we building the right product?”
    - Validate requirements
  - “Did we build the right product?”
    - Validate implementation
  - The software should do what the user really requires
- V&V: Build the right product and build it right!

[Sommerville]

The V & V process

- V&V is a whole life-cycle process
  - V & V must be applied at each stage in the software process
- V&V has two principal objectives
  - The discovery of defects in a system
  - The assessment of whether or not the system is usable in an operational situation

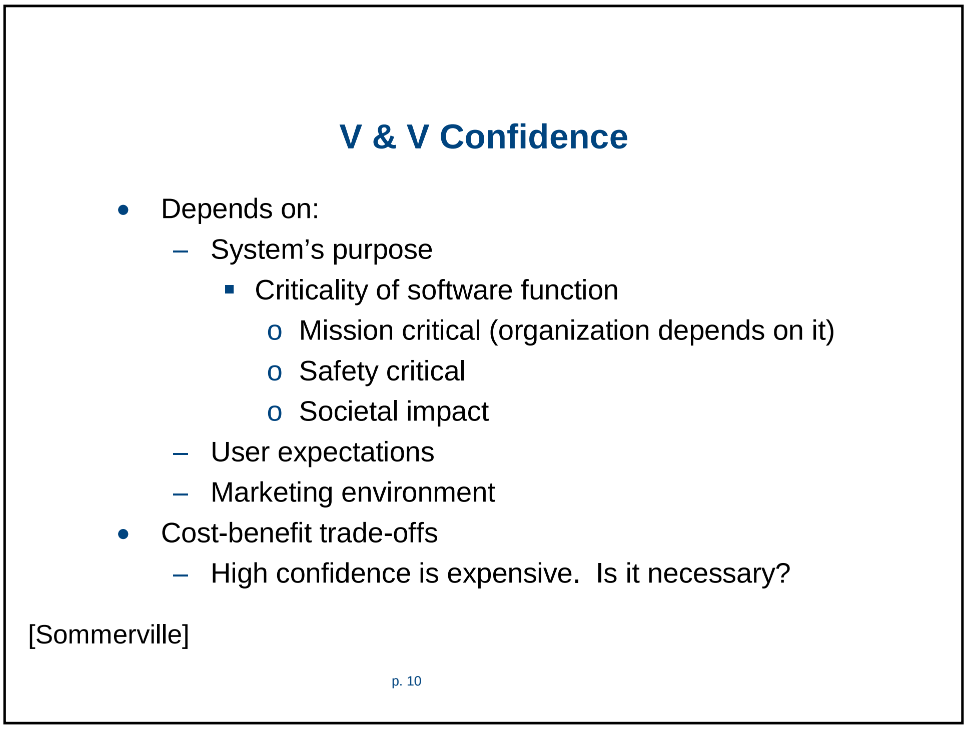
[Sommerville]



Static and Dynamic V&V Activities

- Software testing:
  - Concerned with exercising and observing product behavior
  - Dynamic V&V
- Software inspections:
  - Concerned with studying software product artifacts to discover defects
  - Static V&V
  - May be supplemented by tool-based (semi-automated) document and code analysis

[Sommerville]



V & V Confidence

- Depends on:
  - System's purpose
    - Criticality of software function
      - Mission critical (organization depends on it)
      - Safety critical
      - Societal impact
  - User expectations
  - Marketing environment
- Cost-benefit trade-offs
  - High confidence is expensive. Is it necessary?

[Sommerville]

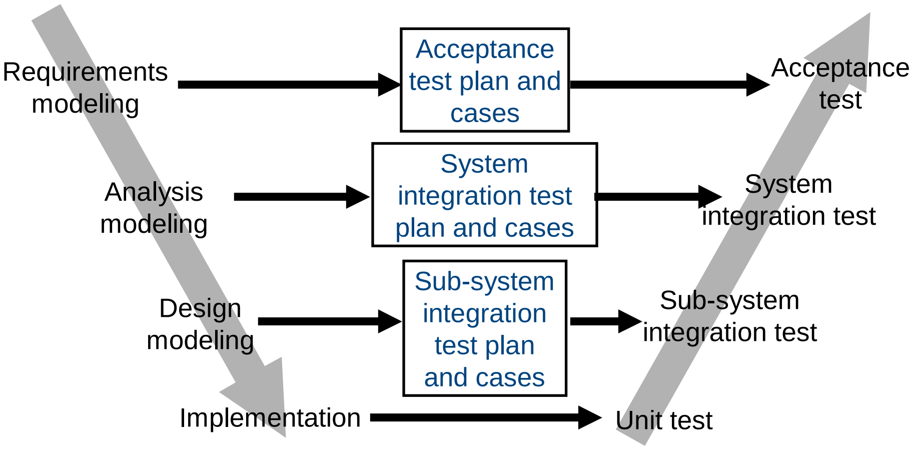
How Do You Plan for V&V?

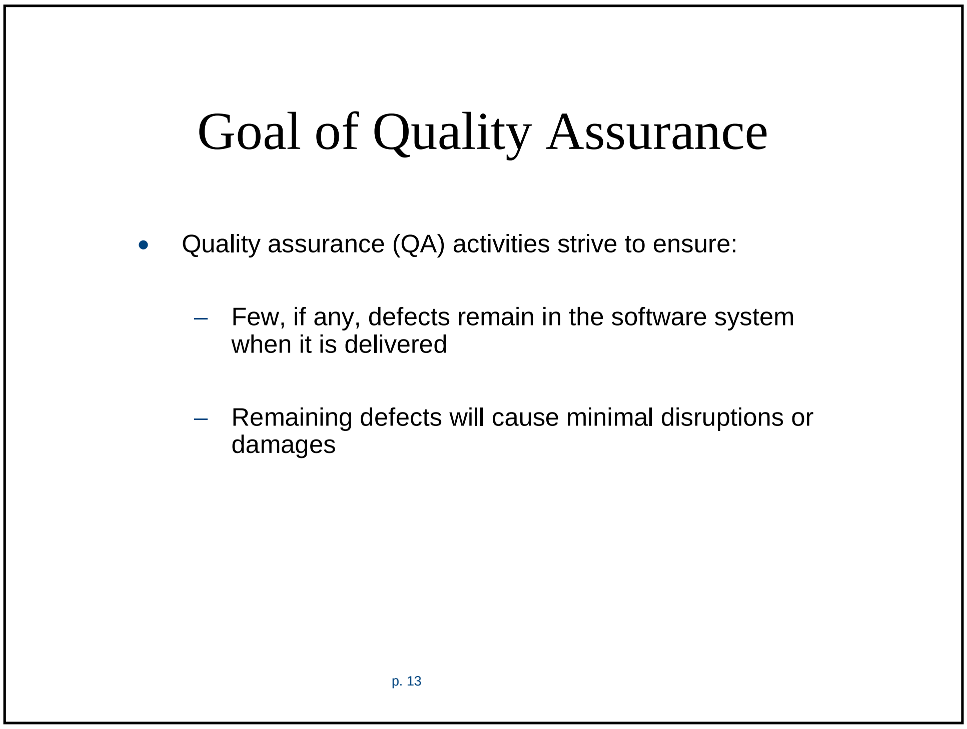
- At each stage of the software development process, there are activities that should be done which will help develop the testing plans and test cases
- Remember: V&V is expensive.
  - Plan to do it right the first time!

[Sommerville]

V-Model

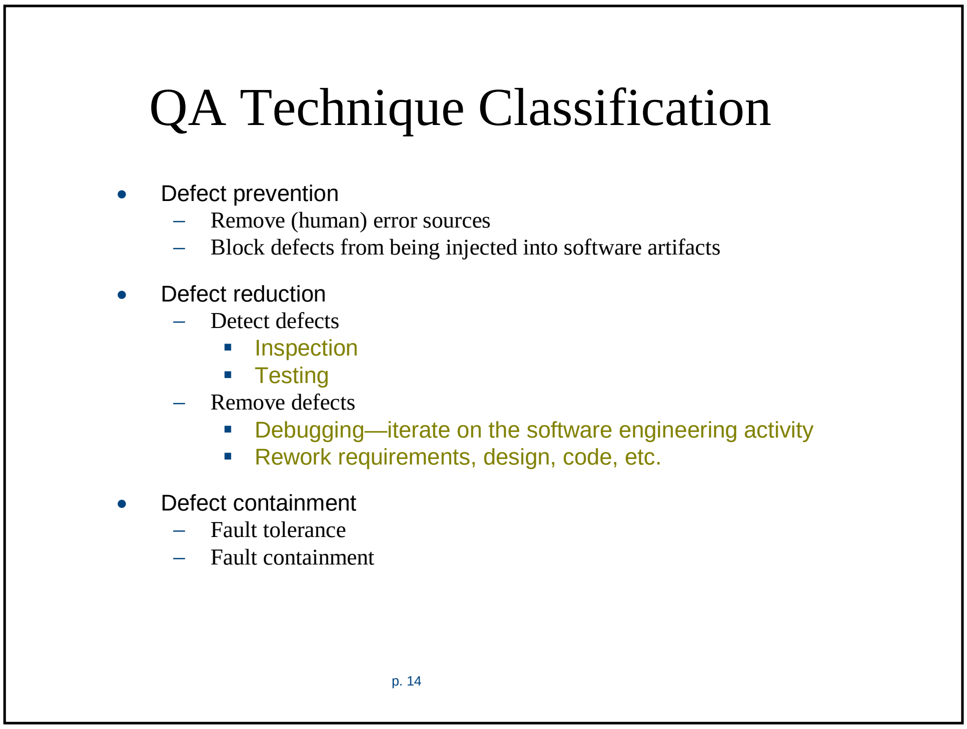
- Plan and develop tests throughout the life cycle
- Implement tests when there is an implementation ready to test
- Iterative and incremental: Repeat “V” at each iteration


Requirements modeling
Acceptance test plan and cases
Acceptance test
Analysis modeling
System integration test plan and cases
System integration test
Design modeling
Sub-system integration test plan and cases
Sub-system integration test
Implementation
Unit test



Goal of Quality Assurance

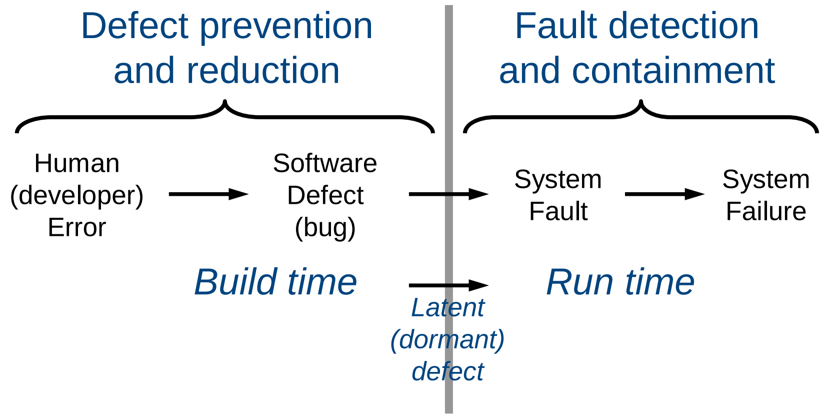
- Quality assurance (QA) activities strive to ensure:
  - Few, if any, defects remain in the software system when it is delivered
  - Remaining defects will cause minimal disruptions or damages



QA Technique Classification

- Defect prevention
  - Remove (human) error sources
  - Block defects from being injected into software artifacts
- Defect reduction
  - Detect defects
    - Inspection
    - Testing
  - Remove defects
    - Debugging—iterate on the software engineering activity
    - Rework requirements, design, code, etc.
- Defect containment
  - Fault tolerance
  - Fault containment

Dealing with Pre-Release and Post-Release Defects



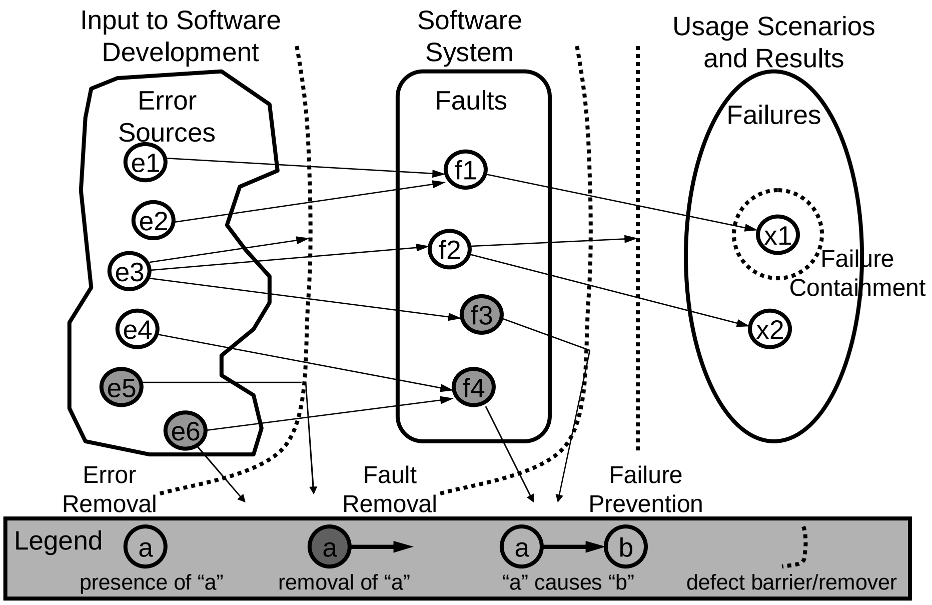
Defect prevention and reduction

Fault detection and containment

Human (developer) Error → Software Defect (bug) → System Fault → System Failure

Build time → *Run time*

Latent (dormant) defect



Input to Software Development

Software System

Usage Scenarios and Results

Error Sources

e1

e2

e3

e4

e5

e6

Faults

f1

f2

f3

f4

Failures

x1

x2

Failure Containment

Error Removal

Fault Removal

Failure Prevention

Legend

a presence of “a”

a removal of “a”

a b “a” causes “b”

defect barrier/remover

Defect Prevention

Remove the *root causes* of errors

- Education and training address human misconceptions that cause errors
  - Domain and product knowledge
  - Software engineering process
  - Technology knowledge
- *Formal methods* can help identify and correct imprecise specifications, designs and implementations
- *Standards conformance,* use of best practices and patterns can help prevent fault injection

Defect Reduction

- Discover and remove defects
- Inspection: direct fault detection
  - requirements, design, code, manuals, test cases
- Testing: *failure observation and fault isolation*
  - Execute the software and observe failures
  - Use execution history/records to analyze and locate fault(s) and defect(s) causing the failure

Issues with Testing

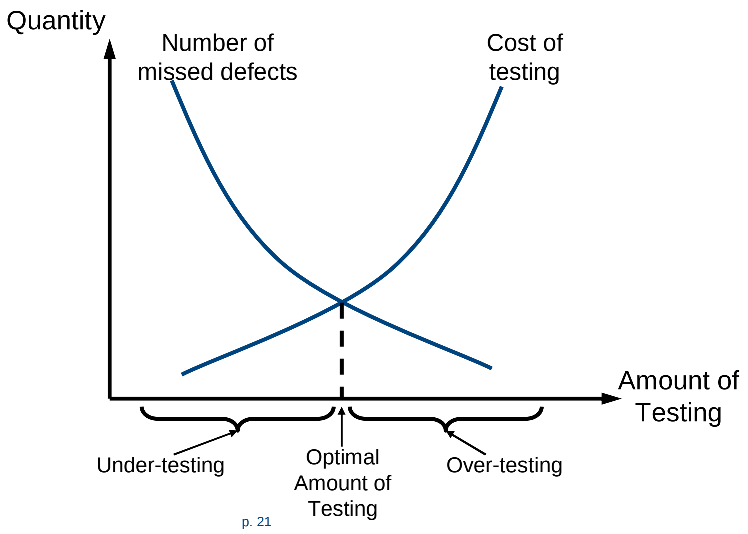
- Need implemented software to execute
- Need software instrumentation, execution history to:
  - isolate faults
  - trace to defects
- Impossible to test everything
  - Expensive to test most things
- Risk of too much and not enough testing
  - Use project risks to guide investment

Risk

Denotes a potential negative impact that may arise from some present process or from some future event.

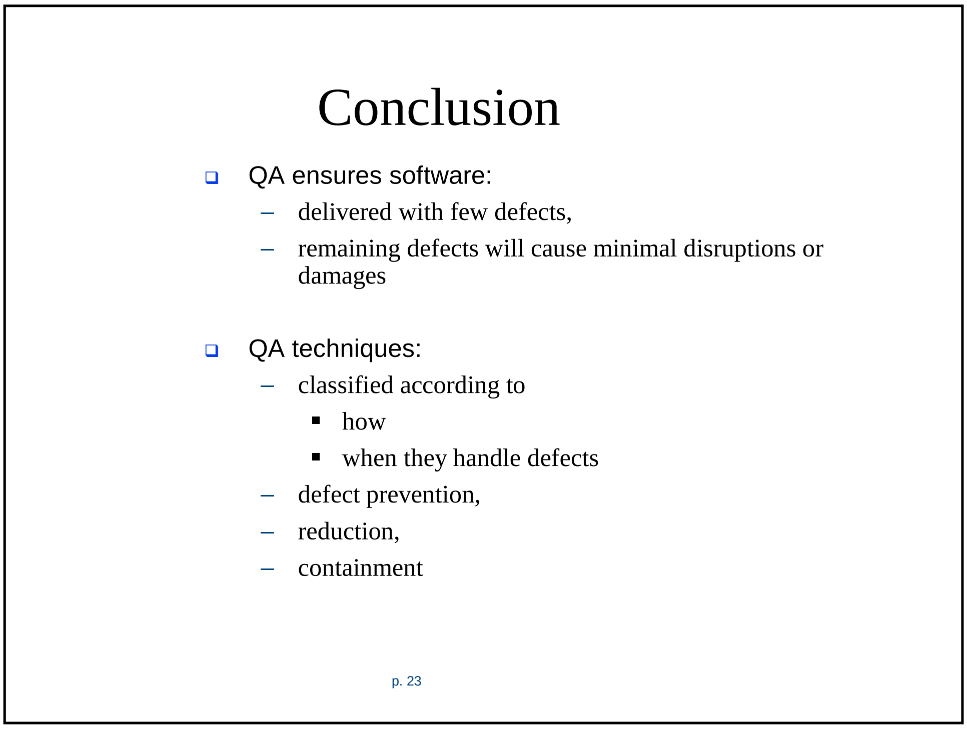
- What is your risk exposure to a defect that is hidden?
  - Likelihood of defect existence
  - Likelihood of failure occurrence
  - Impact if failure occurs
- **Risk exposure determines** ...
  - Testing priority
  - Testing depth
  - What ***to*** test and ***not*** to test

Testing Sweet Spot



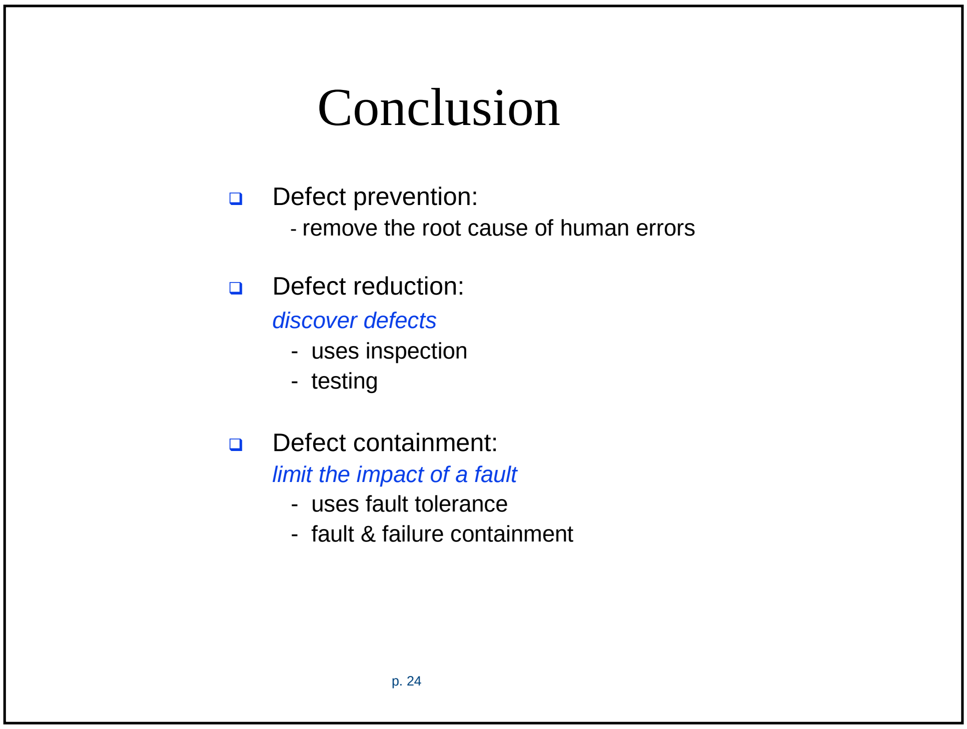
Defect Containment

- Software fault tolerance
  - Safety-critical or mission-critical software often must be fault tolerant
    - The system can continue in operation in spite of a fault occurrence
  - Techniques: exception handling, recovery blocks
- Software failure containment
  - Fault detection and isolation
  - Techniques:
    - safety interlocks,
    - physical containment (barriers),
    - disaster planning, etc.



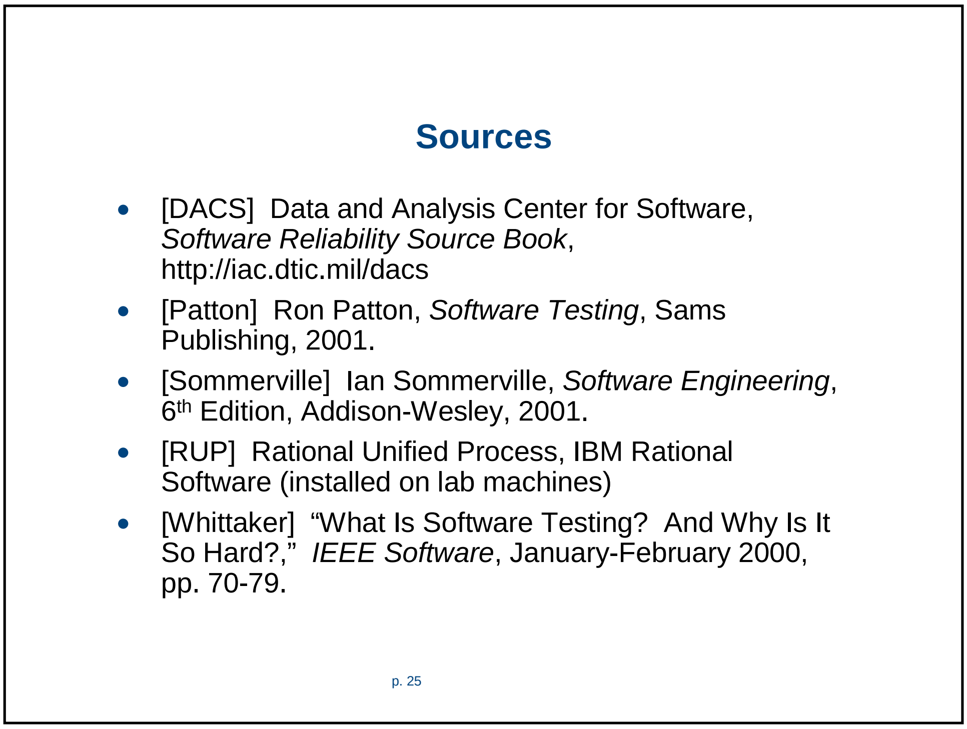
Conclusion

- QA ensures software:
  - delivered with few defects,
  - remaining defects will cause minimal disruptions or damages
- QA techniques:
  - classified according to
    - how
    - when they handle defects
  - defect prevention,
  - reduction,
  - containment



Conclusion

- Defect prevention:
  - remove the root cause of human errors
- Defect reduction:
  discover defects
  - uses inspection
  - testing
- Defect containment:
  limit the impact of a fault
  - uses fault tolerance
  - fault & failure containment



Sources

- [DACS] Data and Analysis Center for Software, *Software Reliability Source Book*, http://iac.dtic.mil/dacs
- [Patton] Ron Patton, *Software Testing*, Sams Publishing, 2001.
- [Sommerville] Ian Sommerville, *Software Engineering*, 6th Edition, Addison-Wesley, 2001.
- [RUP] Rational Unified Process, IBM Rational Software (installed on lab machines)
- [Whittaker] “What Is Software Testing? And Why Is It So Hard?,” *IEEE Software*, January-February 2000, pp. 70-79.