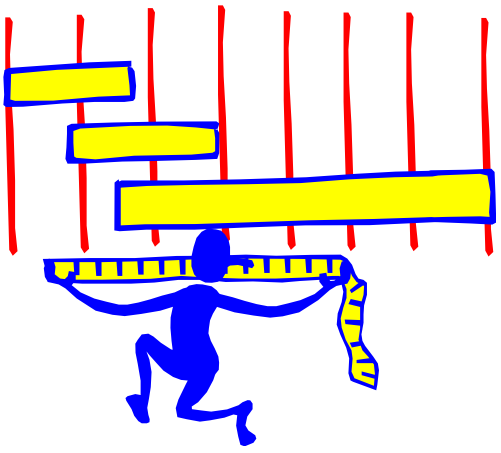
Software Testing and CMMI

Phil Robinson

LonsdaleSystems.com

Comparing CMMI to a Glossary of Software Testing Terms



LonsdaleSystems.com

Comparing CMMI to a Glossary of Software Testing Terms

- CMMI® for Development Version 1.2
  - Carnegie Mellon University
  - Software Engineering Institute (SEI)
- Glossary of Software Testing Terms
  - British Computer Society (BCS)
  - Specialist Group in Software Testing (SIGiST)

LonsdaleSystems.com

Comparing CMMI to a Glossary of Software Testing Terms

- The text of CMMI consists of approximately 136,000 words
- Some glossary terms appear very infrequently!

Term	Count
“System test”	0
“Integration test”	1
“Test plan”	2
“Test case”	4
“Acceptance test”	8
“Unit test”	10

LonsdaleSystems.com

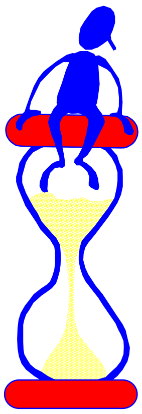
Comparing CMMI to a Glossary of Software Testing Terms

- How can CMMI make so little reference to software testing?
- Is CMMI not relevant to software testing?
- Or maybe software testing is not relevant to process improvement?

- Need understand software testing in the broader context of software quality

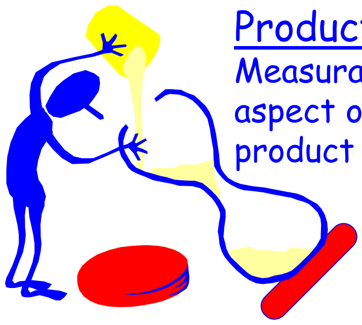
LonsdaleSystems.com

Views of Quality



Excellence
Quality can be recognised but not easily defined

Value view
What the user is prepared to pay for quality



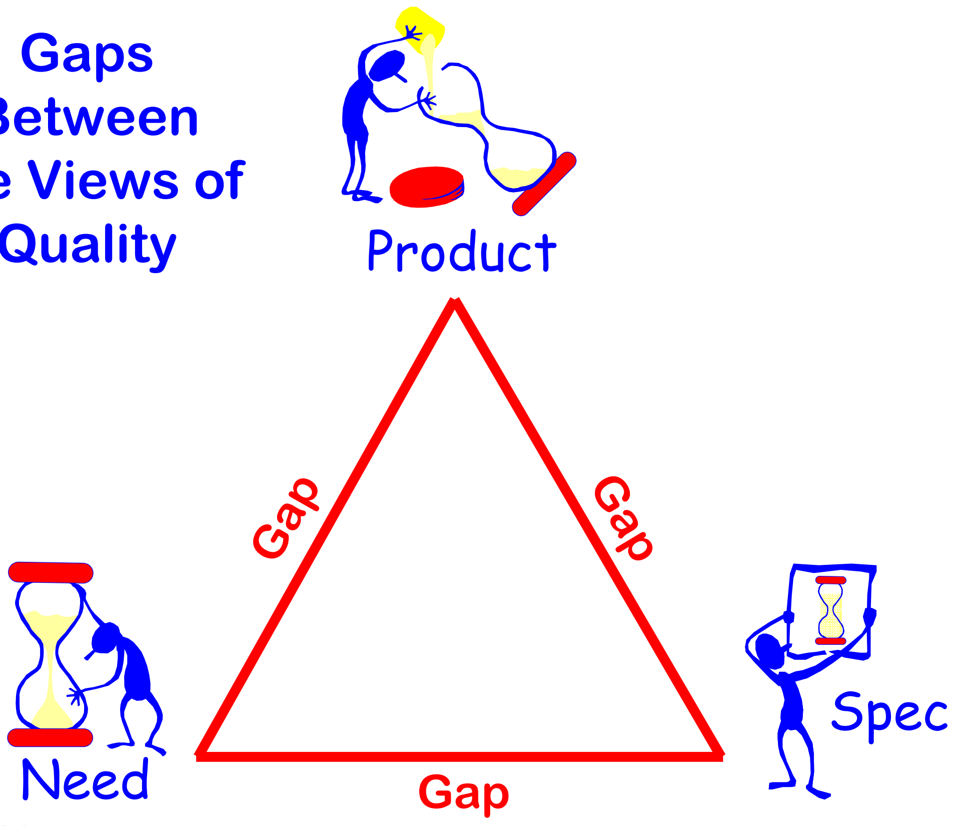
Product view
Measurable aspect of the product

User's view
Ability to satisfy user's needs

Manufacturer's view
Conforms to the specification

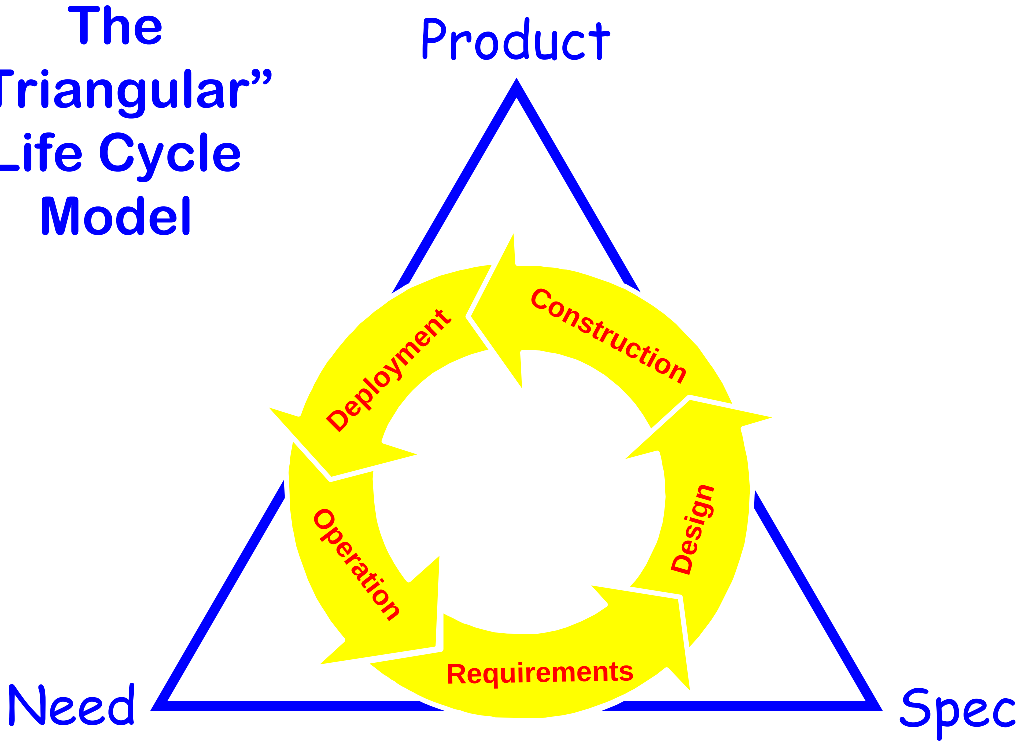
LonsdaleSystems.com

Gaps Between the Views of Quality



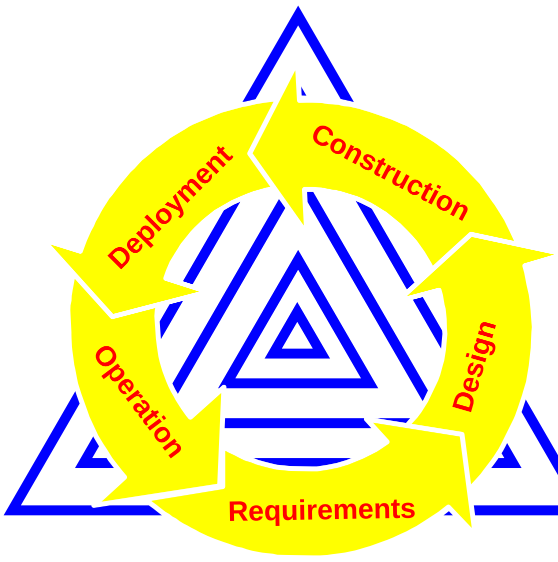
LonsdaleSystems.com

The "Triangular" Life Cycle Model



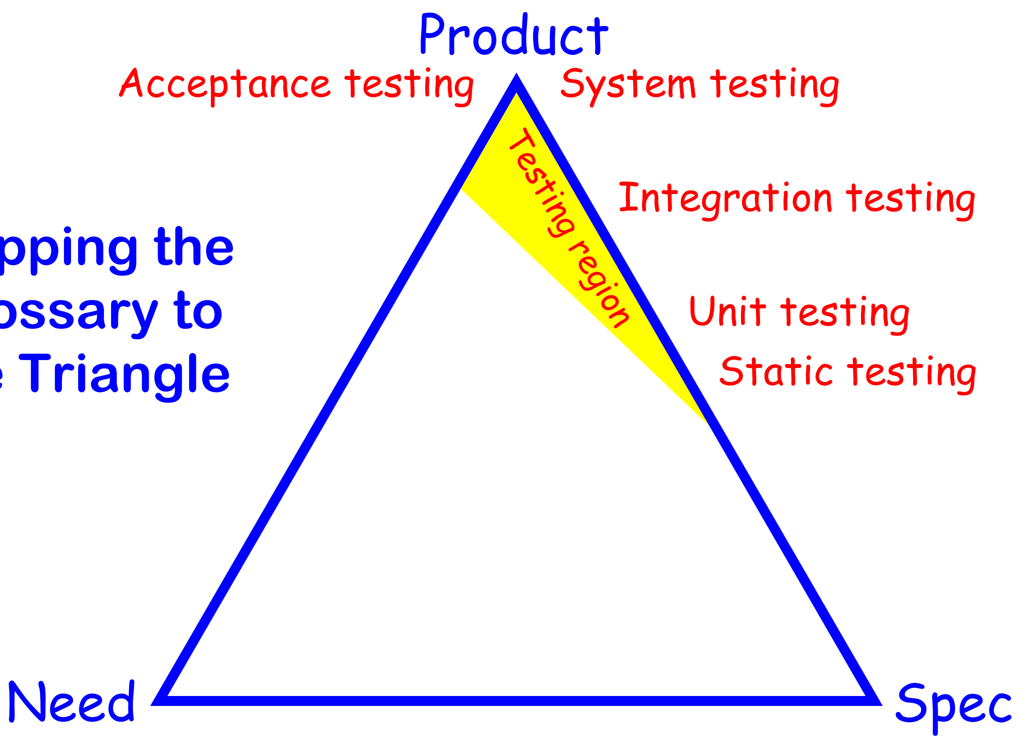
LonsdaleSystems.com

Closing the Gaps



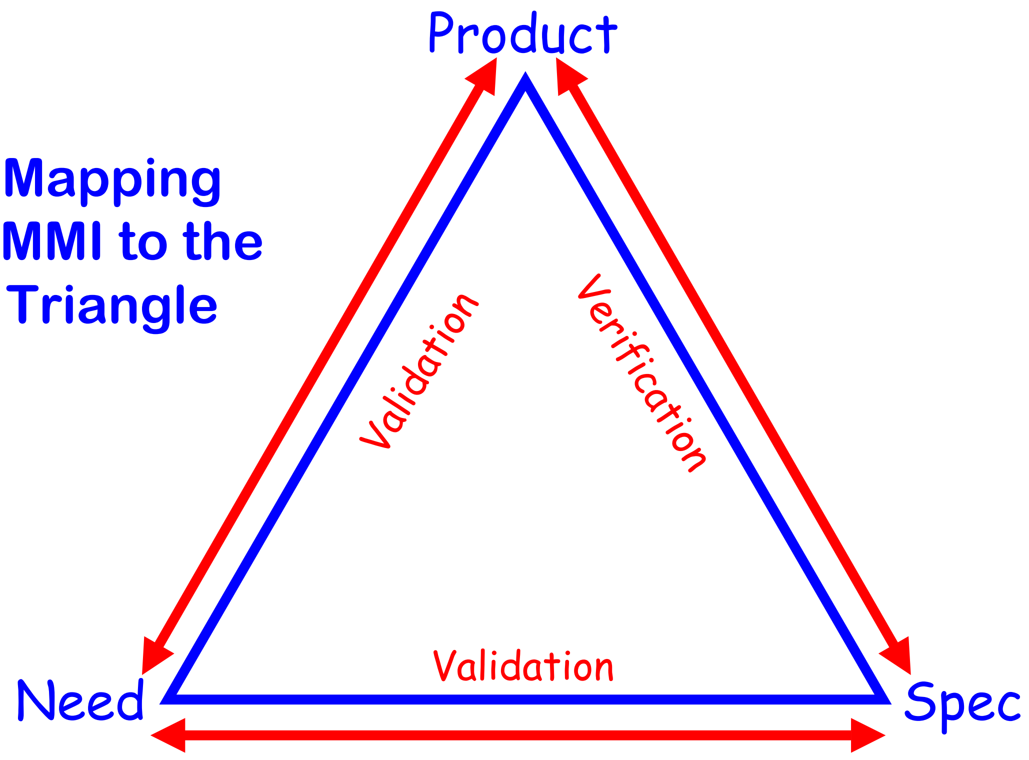
LonsdaleSystems.com

Mapping the Glossary to the Triangle



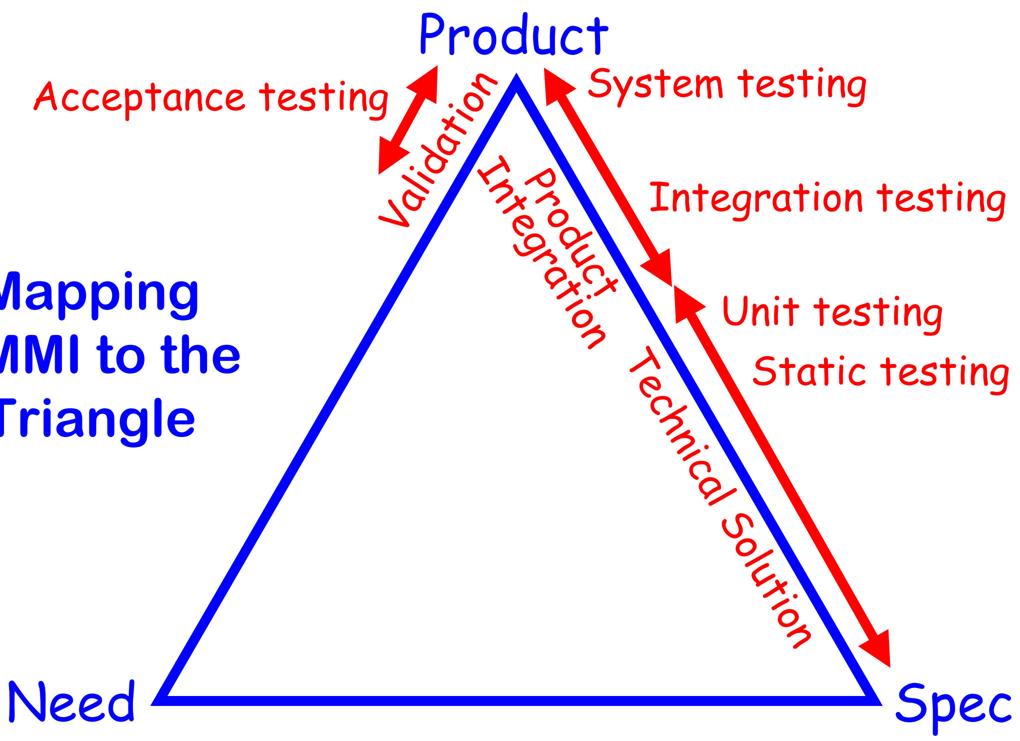
LonsdaleSystems.com

Mapping CMMI to the Triangle



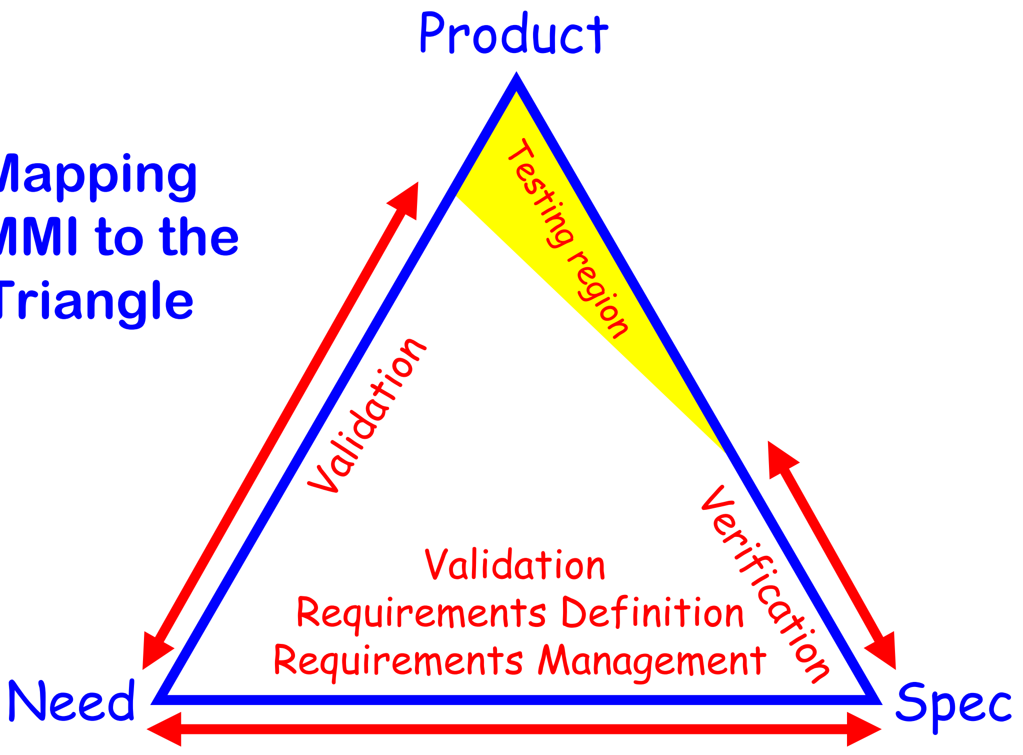
LonsdaleSystems.com

Mapping CMMI to the Triangle



LonsdaleSystems.com

Mapping CMMI to the Triangle



LonsdaleSystems.com

Mapping Testing to CMMI

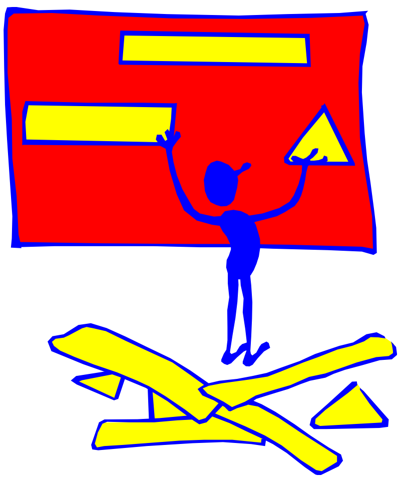
- Testing
  - Covers part of the triangle
  - Single verification and validation technique
  - Least effective verification and validation technique
  - Only closes part of the “Specification-Product” gap
  - Cannot close the “Need-Specification” gap
- CMMI
  - Covers the entire triangle
  - Encourages a variety of verification and validation techniques
  - Closes the “Specification-Product” gap and the “Need-Specification” gap

LonsdaleSystems.com

How can CMMI practices can be applied to software testing?

LonsdaleSystems.com

Test Planning



LonsdaleSystems.com

Glossary of Software Testing Terms

testing: the process of exercising software to

- verify that it satisfies specified requirements and
- to detect errors.

LonsdaleSystems.com

Glossary of Software Testing Terms

test case: a set of inputs, execution preconditions, and expected outcomes developed for a particular objective, such as

- to exercise a particular program path or
- to verify compliance with a specific requirement.

LonsdaleSystems.com

Glossary of Software Testing Terms

coverage: the degree, expressed as a percentage, to which a specified coverage item has been exercised by a test case suite.

coverage item: an entity or property used as a basis for testing

Examples: requirements, paths…

Test Planning and CMMI

Verification

- 100% coverage of requirements is relatively simple
- The Verification process area provides guidance

Error Detection

- 100% coverage of paths is required to detect all errors
- This is usually impossible
- Always a risk that software contains errors
- Error detection is a risk management activity
- The Risk Management process area provides guidance

CMMI Risk Management (RSKM) Process Area

CMMI RSKM Process Area

- SG 1 Prepare for risk management
  - SP 1.1 Determine risk sources and categories
  - SP 1.2 Define risk parameters
  - SP 1.3 Establish a risk management strategy
- SG 2 Identify and analyse risks
  - SP 2.1 Identify risks
  - SP 2.2 Evaluate, categorise and prioritise risks
- SG 3 Mitigate risks
  - SP 3.1 Develop risk mitigation plan
  - SP 3.2 Implement risk mitigation plan

Define the Organisation’s Approach to Risk-Based Testing

SG 1 Prepare for risk management

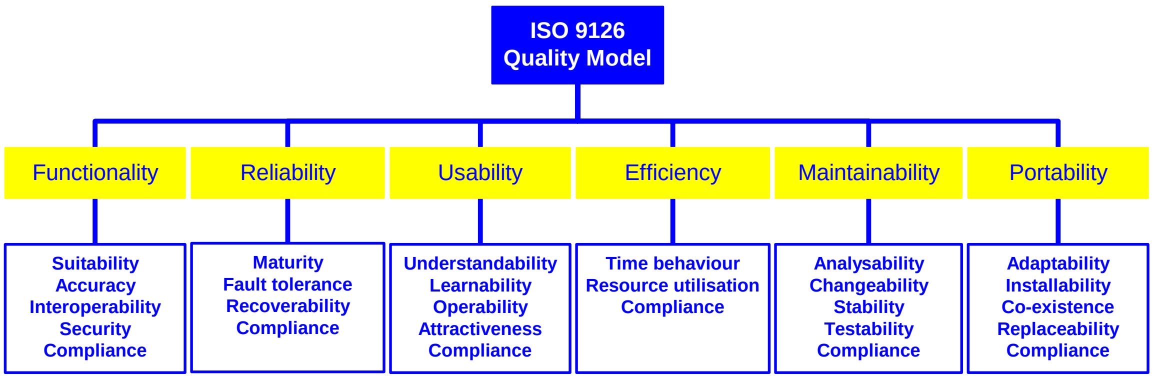
Determine sources of software errors

SP 1.1 Determine risk sources and categories

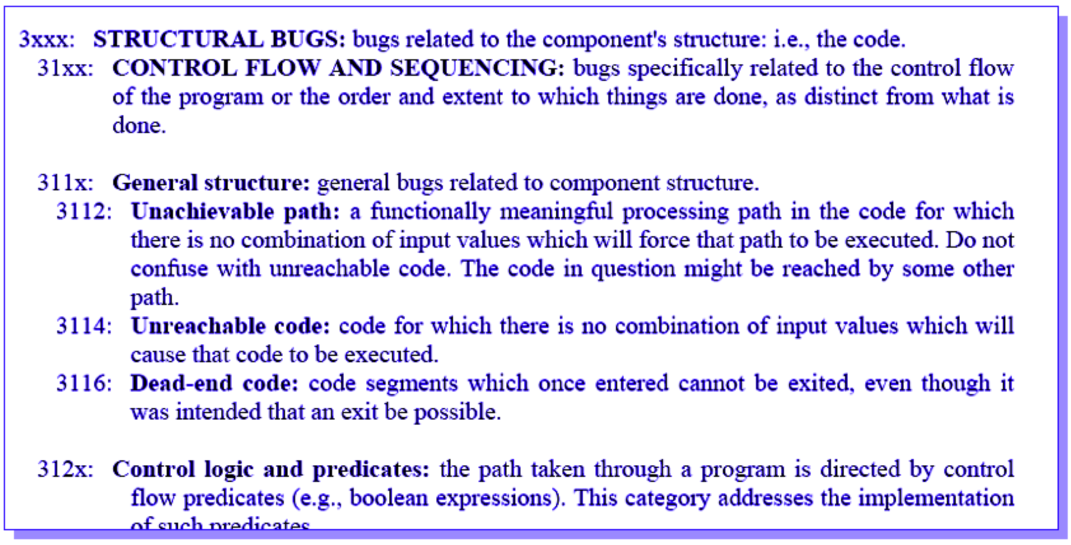
- Product
  - Structure
    - Sub-systems
    - Components
    - Interfaces
  - Characteristics
    - Size
    - Complexity
    - Criticality
  - Quality criteria
    - ISO 9126
- Process
  - Requirements development (RD)
  - Technical solution (TS)
  - Product integration (PI)
- Project
  - Resources
  - Constraints
- Lessons learnt
  - Bug taxonomies
  - SEI taxonomy of risks

Determine sources of software errors

SP 1.1 Determine risk sources and categories


ISO 9126 Quality Model
Functionality
Reliability
Usability
Efficiency
Maintainability
Portability
Suitability Accuracy Interoperability Security Compliance
Maturity Fault tolerance Recoverability Compliance
Understandability Learnability Operability Attractiveness Compliance
Time behaviour Resource utilisation Compliance
Analysability Changeability Stability Testability Compliance
Adaptability Installability Co-existence Replaceability Compliance

Bug Taxonomy



3xxx: **STRUCTURAL BUGS:** bugs related to the component's structure: i.e., the code.

31xx: **CONTROL FLOW AND SEQUENCING:** bugs specifically related to the control flow of the program or the order and extent to which things are done, as distinct from what is done.

311x: **General structure:** general bugs related to component structure.

3112: **Unachievable path:** a functionally meaningful processing path in the code for which there is no combination of input values which will force that path to be executed. Do not confuse with unreachable code. The code in question might be reached by some other path.

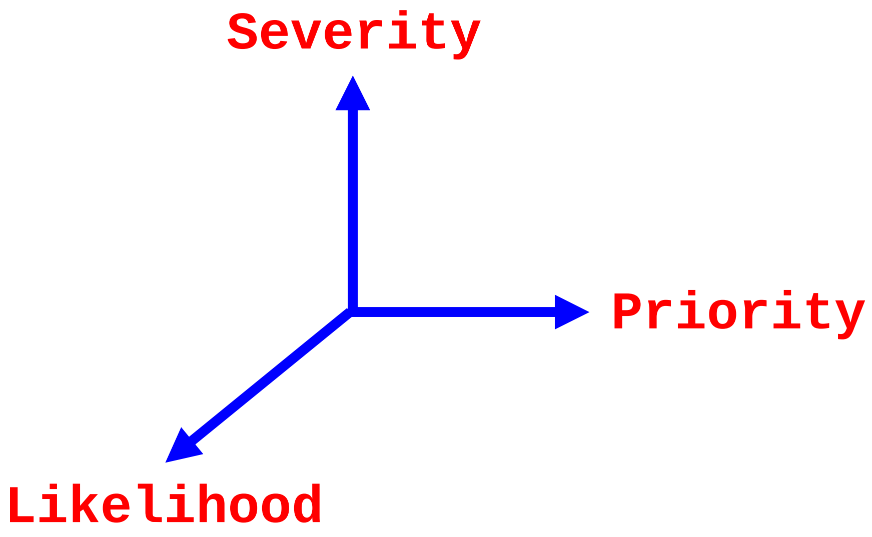
3114: **Unreachable code:** code for which there is no combination of input values which will cause that code to be executed.

3116: **Dead-end code:** code segments which once entered cannot be exited, even though it was intended that an exit be possible.

312x: **Control logic and predicates:** the path taken through a program is directed by control flow predicates (e.g., boolean expressions). This category addresses the implementation of such predicates.

Define parameters of software errors

SP 1.2 Define risk parameters



LonsdaleSystems.com

Define parameters of software errors

SP 1.2 Define risk parameters

Severity

Weight	Description
1	Loss of data
2	Loss of functionality
3	Loss of functionality with workaround
4	Partial loss of functionality
5	Cosmetic error

LonsdaleSystems.com

Define parameters of software errors

SP 1.2 Define risk parameters

Priority

Weight	Description
1	Urgent
2	Essential
3	Valuable
4	Desirable
5	Discretionary

Define parameters of software errors

SP 1.2 Define risk parameters

Likelihood

Weight	Description
1	Very likely
2	Likely
3	Possible
4	Unlikely
5	Very unlikely

Define parameters of software errors

SP 1.2 Define risk parameters

Risk Priority Number

severity × priority × likelihood

125 ⟷ **1**

trivially unimportant

critically dangerous

Establish a quality risk strategy

SP 1.3 Establish a risk management strategy

- Risk Analysis
  - Failure Mode Effect and Analysis (FMEA)
- Risk Mitigation
  - Testing
  - Prototyping
  - Stakeholder discussion
    - Interviews
    - Workshops
  - Reviews
    - Peer
    - Formal
  - Defect prevention

Establish a quality risk strategy

SP 1.3 Establish a risk management strategy

Failure Mode Effect and Analysis (FMEA)

Quality Criteria (ISO 9126)	Failure Mode and Effect	Severity	Priority	Likelihood	Risk Priority Number	Mitigation Strategy

LonsdaleSystems.com

Establish a quality risk strategy

SP 1.3 Establish a risk management strategy

Risk-Based Testing Strategy

- Design tests to expose high-risk errors
- Risk drives decisions on breadth and depth of testing
- Test cases should be traceable to risks
- Need to define a “minimum level” of testing

LonsdaleSystems.com

Identify and Analyse Project Quality Risks

SG 2 Identify and analyse risks

Identify project risks

SP 2.1 Identify risks

- **Product**
  - **ISO 9126**
- Process
- Project
- Lessons Learnt

Identify project risks

SP 2.1 Identify risks

Quality Criteria (ISO 9126)	Failure Mode and Effect
Functionality	Receive mail fails
	Send mail fails
	Folder corruption
	Unable to look up email address
	Address book corruption
	Incorrect formatting of HTML mail
	Attachements not received
	Unable to open attachements
Reliability	
Usability	
Efficency	
Maintainability	
Portability	

Risk assessment

SP 2.2 Evaluate, categorise and prioritise risks

Quality Criteria (ISO 9126)	Failure Mode and Effect	Severity	Priority	Likelihood	Risk Priority Number
Functionality	Receive mail fails	2	1	4	8
	Send mail fails	2	1	4	8
	Folder corruption	1	2	4	8
	Unable to look up email address	3	3	2	18
	Address book corruption	1	3	2	6
	Incorrect formatting of HTML mail	4	4	3	48
	Attachements not received	2	1	3	6
	Unable to open attachements	2	2	3	12
Reliability					
Usability					
Efficency					
Maintainability					
Portability					

Develop and Execute the Project Test Plan

SG 3 Mitigate risks

Develop project risk-based test strategy

SP 3.1 Develop risk mitigation plan

Failure Mode and Effect	Severity	Priority	Likelihood	Risk Priority Number	Mitigation Strategy
Receive mail fails	2	1	4	8	Test with a variety of mai protocolsand firewalls. Test cases should cover all protocol boundary conditions.
Send mail fails	2	1	4	8	Test with a variety of mai protocolsand firewalls. Test cases should cover all protocol boundary conditions.
Folder corruption	1	2	4	8	Attempt to induce folder corruption. Test recovery from folder corruption. Test folder backup and restore.
Unable to look up email address	3	3	2	18	Test with a limited range of typical email addresses. More extensive testing if time permits.
Address book corruption	1	3	2	6	Attempt to induce address book corruption. Test recovery from address book corruption. Test address book export and import.
Incorrect formatting of HTML mail	4	4	3	48	Test with single sample of typical HTML. More extensive testing if time permits.
Attachements not received	2	1	3	6	Test with all supported attachements. Test with attachement boundary conditions (empty, large, etc.)
Unable to open attachements	2	2	3	12	Test with typical attachements.

Test execution

SP 3.2 Implement risk mitigation plan

Glossary of Software Testing Terms

test execution: the processing of a test case suite by the software under test, producing an outcome.

Glossary of Software Testing Terms

outcome: actual outcome or predicted outcome. This is the outcome of a test.

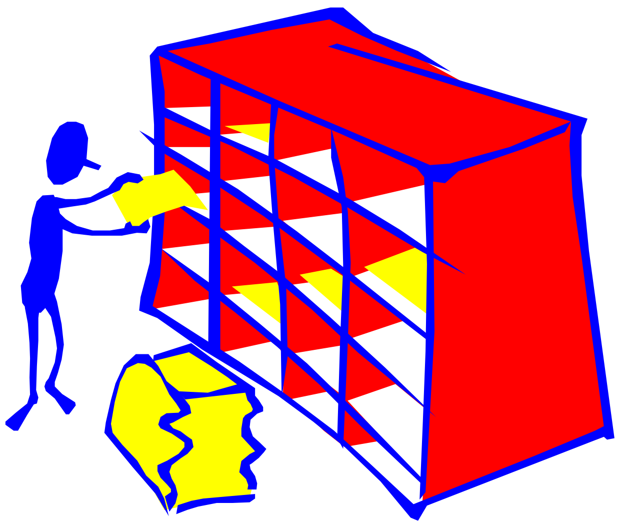
fault: A manifestation of an error in software. A fault, if encountered may cause a failure.

failure: Deviation of the software from its expected delivery or service.

LonsdaleSystems.com

43

Bug Taxonomies and Defect Classification



LonsdaleSystems.com

44

CMM CAR Process Area

- SG 1 Determine causes of defects
  - SP 1.1 Select data for analysis
  - SP 1.2 Analyse causes
- SG 2 Address causes of defects
  - SP 2.1 Implement the action proposals
  - SP 2.2 Evaluate the effect of changes
  - SP 2.3 Record data

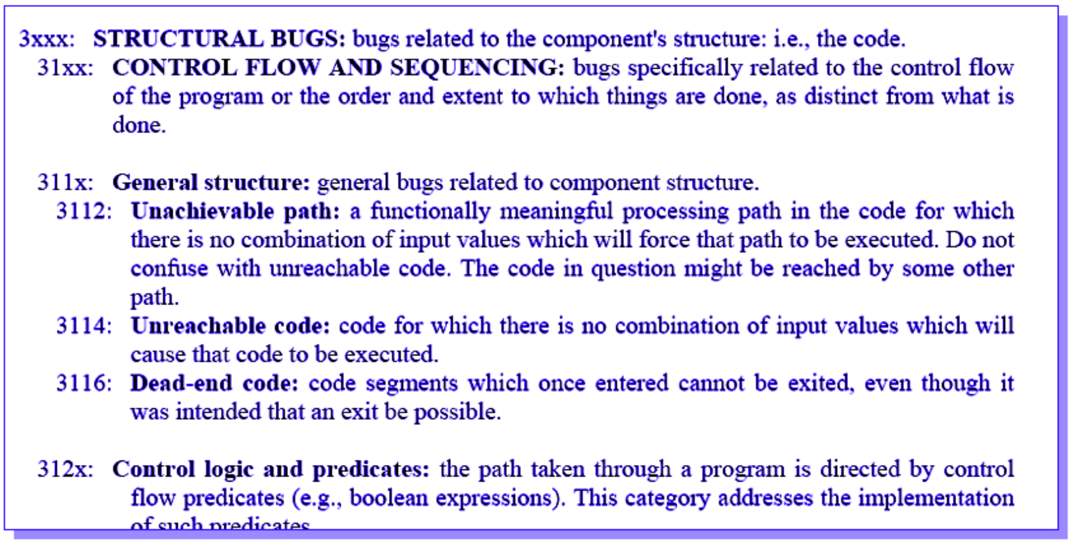
Bug Taxonomy

SP 1.1 Select data for analysis
SP 1.2 Analyse causes

- Massive amount of data
- Large variety of causes
- Need to classify

Bug Taxonomy

SP 1.1 Select data for analysis
SP 1.2 Analyse causes



3xxx: **STRUCTURAL BUGS:** bugs related to the component's structure: i.e., the code.

31xx: **CONTROL FLOW AND SEQUENCING:** bugs specifically related to the control flow of the program or the order and extent to which things are done, as distinct from what is done.

311x: **General structure:** general bugs related to component structure.

3112: **Unachievable path:** a functionally meaningful processing path in the code for which there is no combination of input values which will force that path to be executed. Do not confuse with unreachable code. The code in question might be reached by some other path.

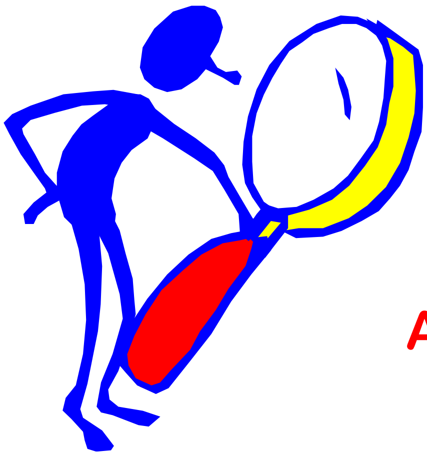
3114: **Unreachable code:** code for which there is no combination of input values which will cause that code to be executed.

3116: **Dead-end code:** code segments which once entered cannot be exited, even though it was intended that an exit be possible.

312x: **Control logic and predicates:** the path taken through a program is directed by control flow predicates (e.g., boolean expressions). This category addresses the implementation of such predicates.

Defect Prevention

SG 2 Address causes of defects



Another time…

Software Testing and CMMI

Phil Robinson

LonsdaleSystems.com

References

- CMMI
  - `http://www.sei.cmu.edu/cmmi/`
- Glossary of Software Testing Terms
  - `http://www.testingstandards.co.uk/glossary.htm`
- ISO 9126
  - `http://www.iso.org`
  - `http://en.wikipedia.org/wiki/ISO_9126`
- SEI Risk Taxonomy
  - `http://www.sei.cmu.edu/publications/documents/93.reports/93.tr.006.html`

LonsdaleSystems.com

References

- **Bug Taxonomy**
  - http://inet.uni2.dk/~vinter/bugtaxst.doc
- **FMEA**
  - Rex Black, *Critical Testing Process: Plan, Prepare, Perform Perfect*, Addison Wesley, 2004
  - http://en.wikipedia.org/wiki/FMEA

Notes: