Quality Governance
Transforming IT into Business Value

Analyse that....!

(Requirements and how to get them right)

November 2008

Carl Allen
Director of Programme Quality Management

SQS Group Limited

Contents

- Background
- Experiment – Interactive!
- The Problem
- Scope of Service
- The Process
- Example Results
- Case Study

© SQS Software Quality Systems

Background
Why are we here?

- The challenge to any software development project:
  - How to remove defects introduced:
    - Early
    - Quickly
    - Cheaply
  - How to lower the risks they impose on the functioning of the end product.
- This presentation describes the overall process and benefits in subjecting requirements and design documentation to a structured review and testing process very early-on in the project lifecycle.

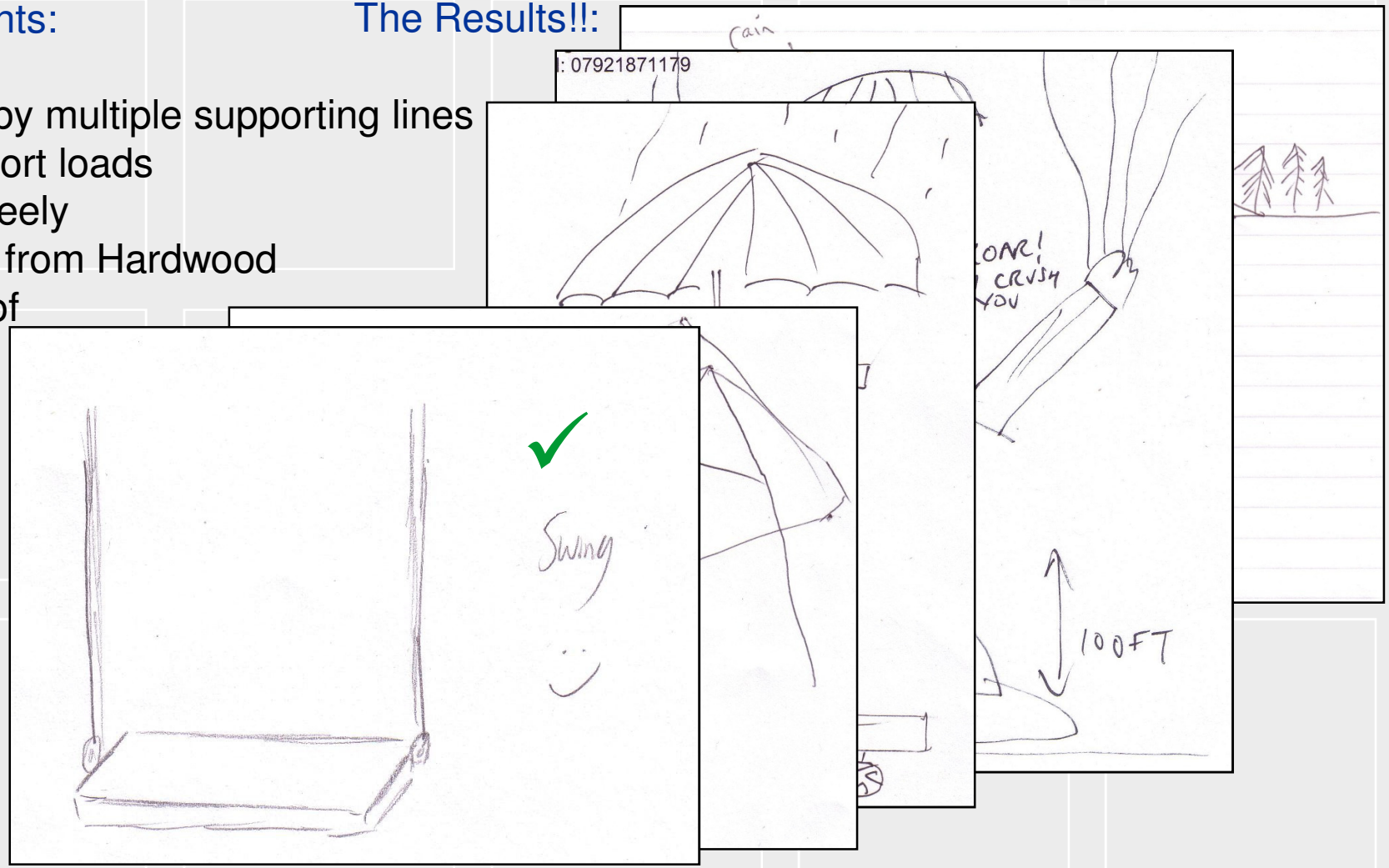
© SQS Software Quality Systems

An experiment
Some creativity required...

The Requirements:

1. Suspended by multiple supporting lines
2. Able to support loads
3. Can move freely
4. Constructed from Hardwood
5. Weatherproof

The Results!!:



© SQS Software Quality Systems

So here's the **problem**....

© SQS Software Quality Systems

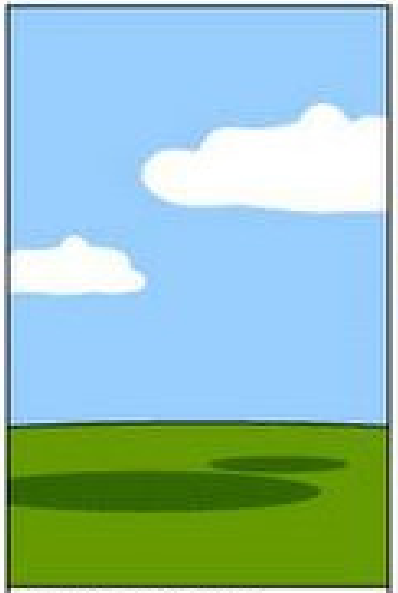
How the customer explained it

How the Project Leader understood it

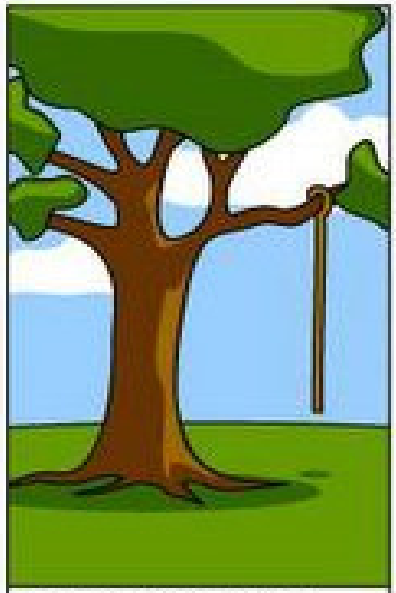
How the Analyst designed it

How the Programmer wrote it

How the Business Consultant described it

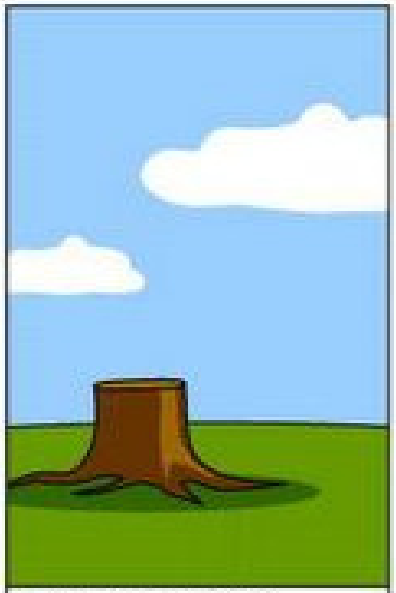


How the project was documented



What operations installed

How the customer was billed



How it was supported

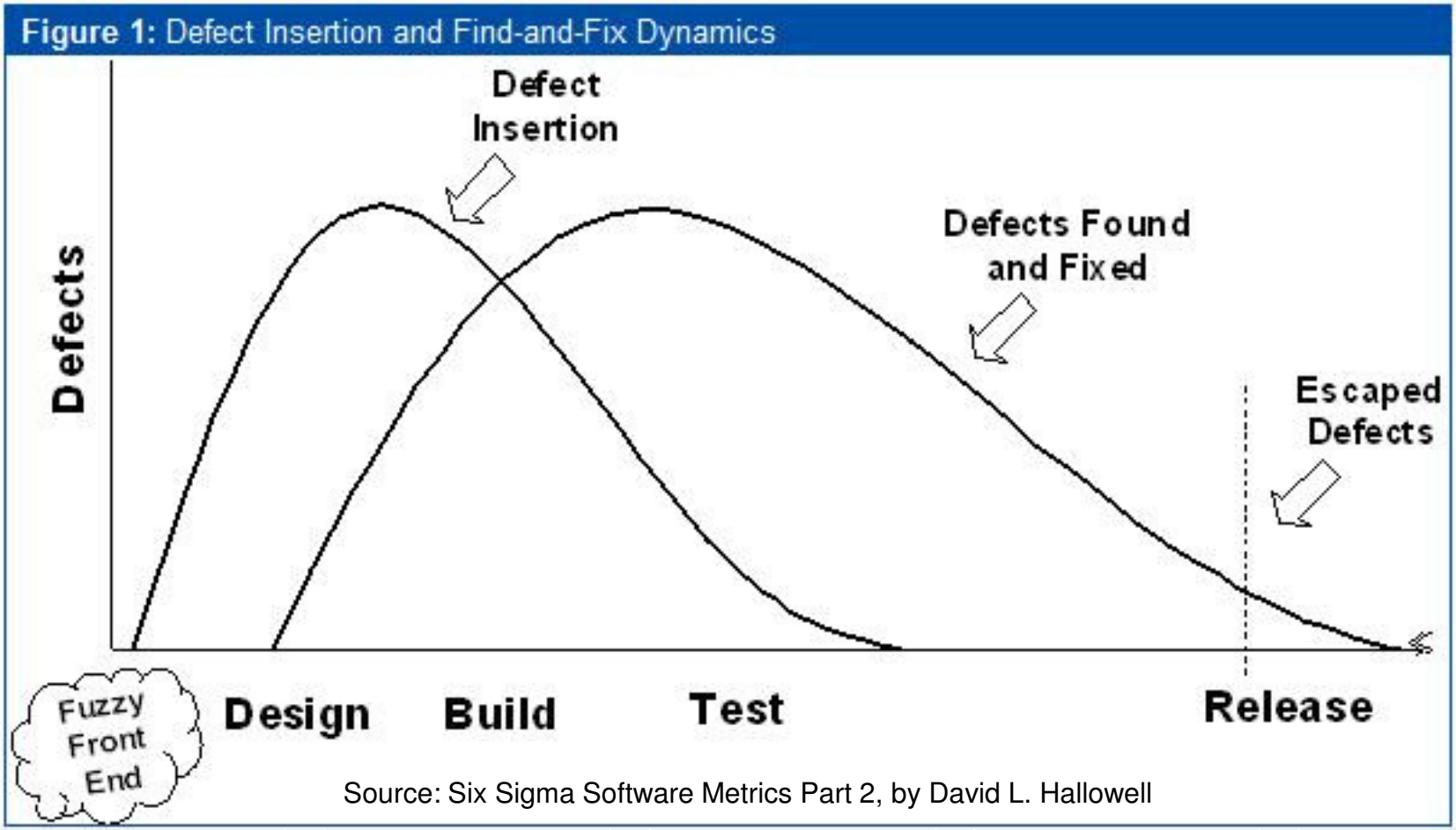
What the customer really needed

Typical Scenario

- Defects introduced at the requirements and design stages
- Requirements signed-off by the business, but do not have enough detail for development and test teams – leading to ambiguity and rework
- Requirements not signed-off by development and test teams
- Offshore or outsourced projects rely on adequate documentation
- Not detected/removed until later stages in the Software Development Lifecycle (SDLC):
  - Build/Unit Testing
  - System/Integration/Non-Functional/End-to-end Testing
  - Operation Acceptance Testing (OAT)
  - Acceptance Testing (UAT)
- Project expends time/effort fixing issues.
- Very rarely are requirements documents or Solution Designs subject to any rigorous process of review or testing.

© SQS Software Quality Systems

Relationship between Quality & Time
Defect Insertion and Removal Points

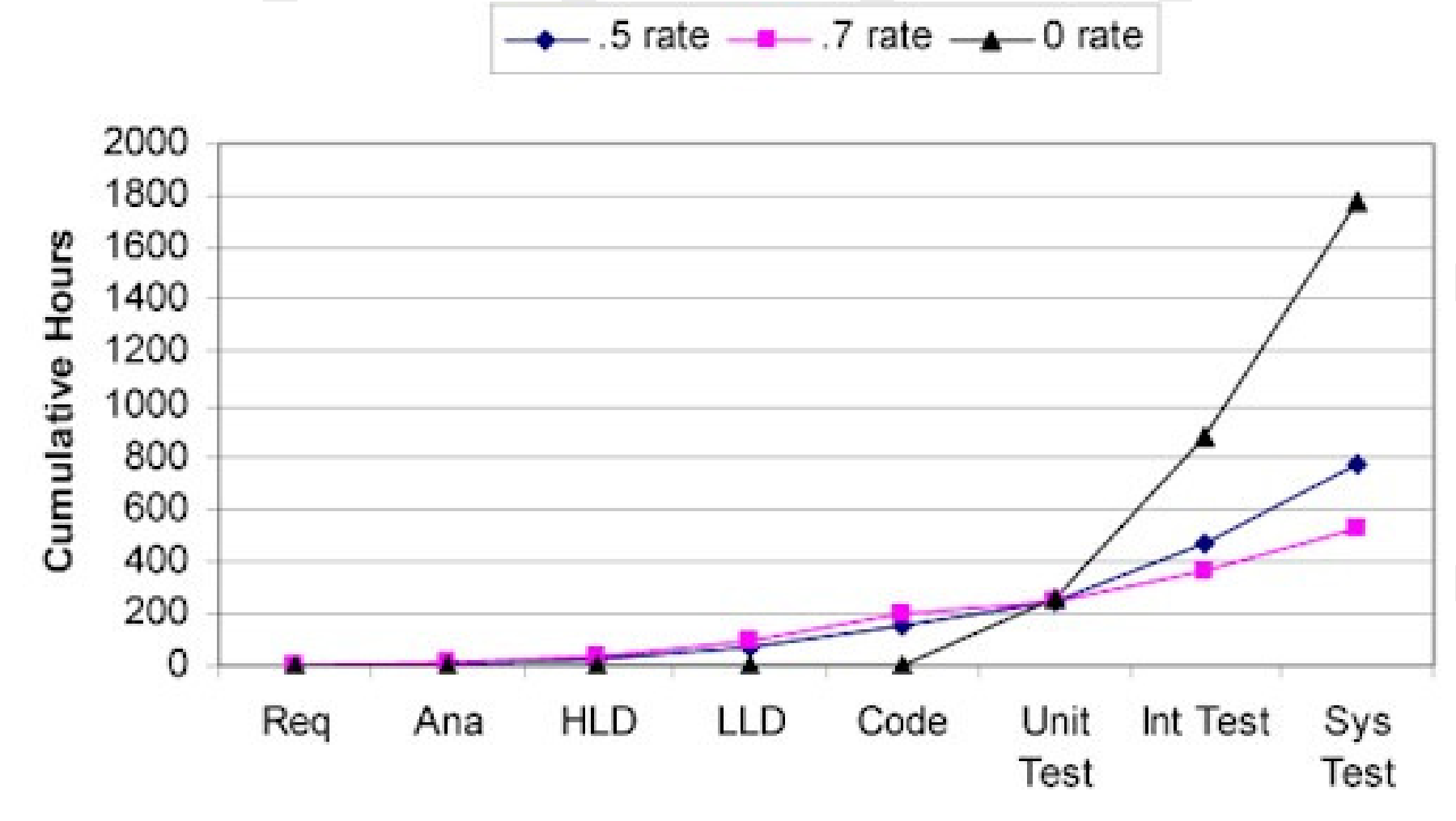


Requirements review & testing, peer reviews and code inspections find defects closer to their insertion point, shifting the find curve to the left, where the fix times and costs are lower

© SQS Software Quality Systems

Cumulative defect removal cost

Increasing costs throughout the SDLC



Source: StickyMinds.com, Calculating the Economics of Inspections by Ed Weller

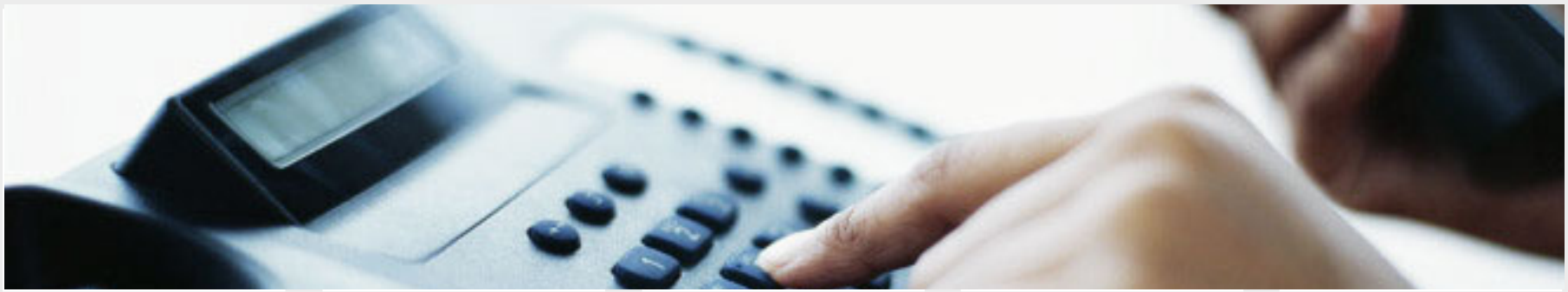
© SQS Software Quality Systems

Some industry figures…

- Research indicates that 40% of IT projects fail to deliver their intended return on investments (Harvard Business School Press)
- 75% of Business managers perceive IT to lack credibility or be failing the organisation (Meta Group)
- 30% of a IT project budget is consumed by rework activities to correct problems in the solution (IBM)

© SQS Software Quality Systems

So doesn’t it make sense to remove the documentation defects early?….



© SQS Software Quality Systems

Quality Governance for Requirements & Solution Design Service

Addressing the problem

Scope:

- Business Analyst & Design Workshop
- High Level Business Requirements Review
- Low Level Business Requirements Review
- Non-functional Requirement (NFR) Acceptance
- Solution Design Document Review
- Retesting following reworked documentation

© SQS Software Quality Systems

First Steps…

- **Define Acceptance Criteria**
  - Requirements and solution design testing should be undertaken against clearly defined acceptance criteria and standards.
  - These criteria should be written in advance of requirements being produced and communicated to the authors so they are aware of the expected quality standards
- **Acceptance Criteria Workshop**
  - A workshop should held with the authors of the requirements and solution design documents (the Business Analysts and Developers) to express the details of the acceptance criteria on which they will be measured.
- **Speed-up Document Delivery**
  - A by-product of having measurable acceptance criteria, or test checks, against the requirements is that is it very clear to the requirement authors what they have to produce and to what quality.
  - This helps to speed up delivery of the requirements and reduces the amount of re-work required post review.

© SQS Software Quality Systems

Defining Acceptance Criteria
Know what you are measuring against

- Certify that the requirements document is an acceptable description of the system to be implemented
- Check a requirements document for
  - Completeness and consistency
  - Conformance to IEEE standards
  - Requirements conflicts
  - Technical errors
  - Ambiguous requirements
- Requirements tested using the IEEE 1998 standard for Software Validation and Verification with the following criteria:
  - Correctness (Adequacy)
  - Non-Ambiguity
  - Completeness
  - Consistency
  - Verifiability
  - Modifiability
  - Traceability

© SQS Software Quality Systems

Testing against acceptance criteria – the details…

- Correctness (Adequacy)
- Non-Ambiguity
- Completeness
- Consistency
- Verifiability
- Modifiability
- Traceability

1. Correctness (Adequacy)

- Requirements should accurately reflect the business drivers that induce them.
- Example:

  Wrong

  - The system shall store user information in a database.

  Correct

  - The system shall store user information in a retrievable and updateable manner.

- Correctness checks:
  - Is the requirement within the scope of the release?
  - Is the requirement feasible from a technology point of view?
  - Does the requirement refrain from being unnecessarily over-stringent?
  - Is it a requirement (what) and not a description of implementation (how)?

- Correctness (Adequacy)
- Non-Ambiguity
- Completeness
- Consistency
- Verifiability
- Modifiability
- Traceability

© SQS Software Quality Systems

2. Non-Ambiguity

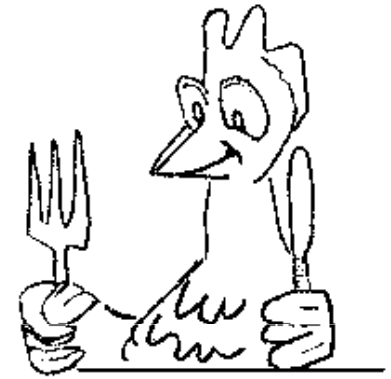
- It should **not** be possible for a requirement to be interpreted in more than one way.
- Where this is the case and all interpretations are valid, break the requirement into two or more individual components.
- Examples:

Wrong

- The system shall be user-friendly.

Correct

- The system's front end shall be developed in accordance with the Microsoft Windows Logo GUI guidelines.



- Does the requirement refrain from using words and phrases open to misinterpretation?
  - Minimise, maximise, user-friendly, rapid, quick, easy, sufficient
- Does the requirement refrain from using non-specific phrases referring to time?
  - Instantaneous, earlier, timely

- Correctness (Adequacy)
- Non-Ambiguity
- Completeness
- Consistency
- Verifiability
- Modifiability
- Traceability

© SQS Software Quality Systems

3. Completeness

- A requirement should contain enough information within itself to be of value.
- If dependencies exist between requirements on the same level, they should be merged if possible.
- Example:

  Wrong
  - R1.1 The system shall monitor widgets.

  Correct
  - R1.1 The system shall monitor 200 widgets for conditions X,Y,Z.
  - R1.2 The system shall update the status of widgets (refer to R1.1) every 1 second and display it in the administrator window.
- Does the requirement contain enough information within itself to be of value without a dependency on another requirement at the same level?
- Does the requirement consider all possible paths and scenarios and contains information on how to deal with them?
- Does the requirement avoid the use of incomplete terms?
  - etc, TBD, unknown at this point, but not limited to, and/or

- Correctness (Adequacy)
- Non-Ambiguity
- Completeness
- Consistency
- Verifiability
- Modifiability
- Traceability

© SQS Software Quality Systems

3. Completeness...
Avoiding ‘hidden’ problems not yet explored...

© SQS Software Quality Systems

4. Consistency

- The formatting and wording of requirements must follow a common set of conventions.
- These conventions should be documented and publicised to the project team.
- Do the descriptions of different requirements include contradictions?
- Are there contradictions between individual requirements and overall system requirements?
- Is the requirement consistent within the scope of the document?
- Example:

  Wrong
  - R1.1 The system shall perform action X.
  - R1.2 In addition to R 1.1 it should allow the user to perform action Y.
  - The system shall be expected to perform action Z (1.3).

  Correct
  - R1.1 The system shall perform action X.
  - R1.2 The system shall allow the user to perform action Y.
  - R1.3 The system shall perform action Z.

- Correctness (Adequacy)
- Non-Ambiguity
- Completeness
- Consistency
- Verifiability
- Modifiability
- Traceability

© SQS Software Quality Systems

5. Verifiability

- Is it possible to prove that a requirement has been achieved?
- Each requirement should be testable i.e. it should be possible to define tests to check whether or not that requirement has been met.
- Inventing requirements tests is an effective validation technique as missing or ambiguous information in the requirements description may make it difficult to formulate tests
- Also refers to the degree of ease with which the test infrastructure can be put in place to prove a requirement has been met.
- The requirement should be written such that test case authoring, test data, and test execution follow naturally from the requirement.
- Example:

  Wrong
  - R2.3 The system shall minimise user input to complete procedure X.

  Correct
  - R2.3 The system shall require 2 user inputs to complete procedure X.

- Correctness (Adequacy)
- Non-Ambiguity
- Completeness
- Consistency
- Verifiability
- Modifiability
- Traceability

© SQS Software Quality Systems

6. Modifiability/Maintainability

- Relates to a requirement's ability to respond to change.
- A requirement should be written in such a way that the structure and the style of the requirement should be preserved even if the content of the requirement changes.
- Example:

  Correct
  - R2.3 The system shall require 2 user inputs to complete procedure X.
  - R2.3 The system shall require 3 user inputs to complete procedure Y.
- Can the requirement be rewritten to change the content without changing the style and structure?

- Correctness (Adequacy)
- Non-Ambiguity
- Completeness
- Consistency
- Verifiability
- Modifiability
- Traceability

© SQS Software Quality Systems

7. Traceability

- All requirements must have unique identifiers that can be used to map tests and software features to the requirements from which they were spawned.
- All lower level requirements must be traceable to higher-level requirements.
- Infrastructure must be in place to support traceability, e.g. a Requirements Traceability Matrix (RTM).
- Related requirements should be grouped together and should refer to each other.
- Example:

  Wrong
  - R2.3 The system shall allow the user to change today's date as specified previously.

  Correct
  - R2.3 The system shall allow the user to change today's date as specified in R2.1.
- Does the requirement have a unique identifier?
- If it is a low level requirement can be it be traced to a high level requirement(s) from which it spawned?
- Does the requirement have a priority assigned?

- Correctness (Adequacy)
- Non-Ambiguity
- Completeness
- Consistency
- Verifiability
- Modifiability
- Traceability

© SQS Software Quality Systems

Example Results Template

Document:	
Reviewed By:	

Area			Unambiguous				
Check No	1.4	1.5	2.1	2.2	2.3	2.4	2.5
Check	Is it is requirement (what) and not a description of implementation (how)?	Is it a requirement and not a statement or a goal?	Does the requirement refrain from using words and phrases open to misinterpretation?	Does the requirement refrain from using non-specific phrases referring to time?	Is the requirement simple and clear to understand?	Does the requirement refrain from making assumptions about the reader's knowledge of external systems or business processes? If no, are these assumptions clearly stated elsewhere?	Does the requirement refrain from using indefinite pronouns?
Example(s):		- requirements use shall or must - statements of fact use will - goals use should	minimise, maximise, user-friendly, rapid, fast, quick, easy, sufficient, adequate	until, after, before, until, when, earlier, instantaneous, simultaneous, timely	no unnecessary adjectives, adverbs, comparisons		anything; both; each; either; every; everybody; everyone; everything; few; many; most; much; neither; no one; nobody; none; one; several; some; somebody; someone; something; such

Req Desc	Req No							
Create Base Bank Users	2.3.1.1	y	n	n	n	y	y	y
	2.3.1.2	y	y	y	y	y	y	y
	2.3.1.3	n	n	n	y	n	y	y
Load Bank Users	2.3.2.1	y	y	n		n	y	y
	2.3.2.2	y	y	n		y	y	y
	2.3.2.3	y	y	n		y	y	n
	2.3.2.4	y	y	y		y	y	y
	2.3.2.5	y	n	y		y	y	y
	2.3.2.6	y	n	n	y	y	n	y
Audit & Control	2.3.3.1	y	y	y	y	y	y	y
	2.3.3.2	n	y	y	y	n	y	y
	2.3.3.3	y	y	y	y	n	y	y
	2.3.3.4	y	n	y	y	y	y	y
Required Functionality	2.3.4.1	y	y	y	y	y	y	y
	2.3.4.2	y	y	y	y	n	y	y
	2.3.4.3	y	y	y	y	y	n	y
	2.3.4.4	n	y	n	y	y	n	y
	2.3.4.5	y	y	y	y	y	n	y

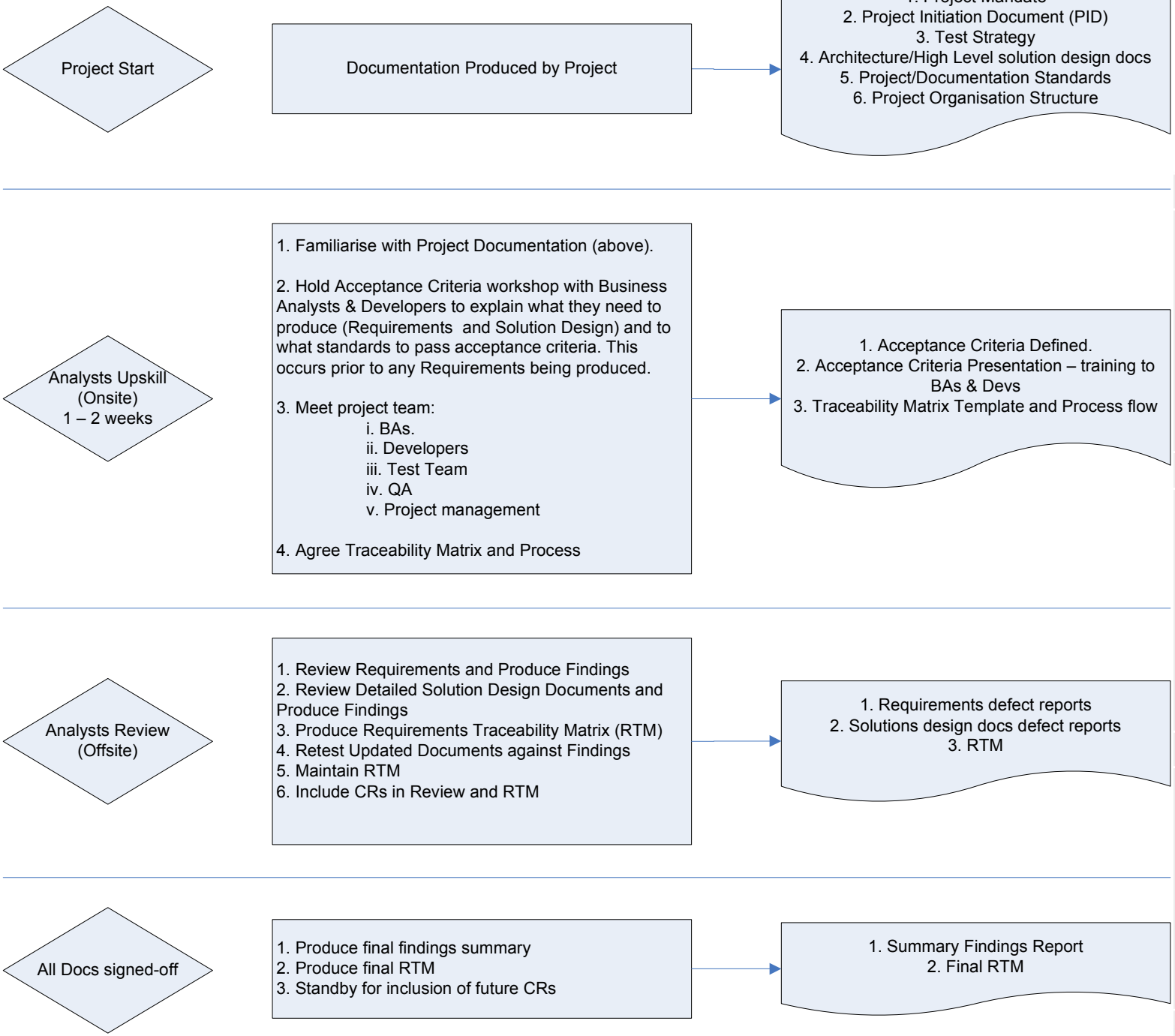
Defect:
Ambugious: "Easily"

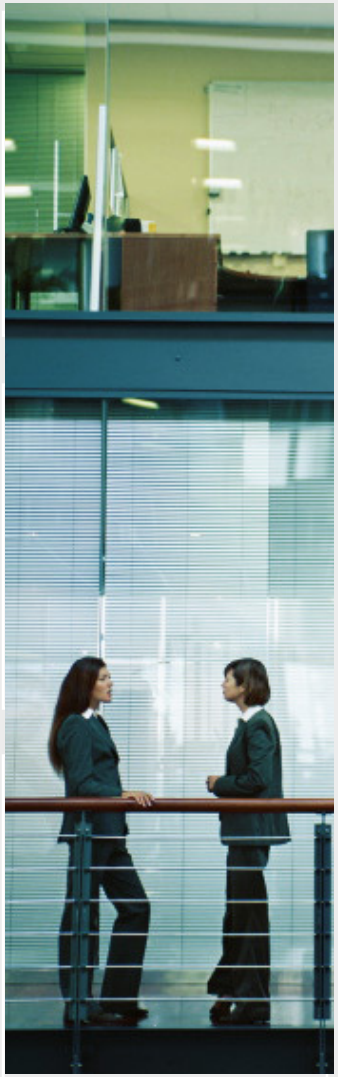
Example of Non-Functional Acceptance Criteria

ID	Title	Detail	Acceptance Criteria	Example	Notes
PT001	TPS at peak/Average TPS	Transactions per second. A transaction is defined as an action that is undertaken by a customer that would have a corresponding output on the SI log.	The requirement must show the peak TPS as an integer for both the peak and average levels. This must be reproduced for all releases.	R1 Peak TPS=32, Average TPS=12 R2 Peak TPS=38, Average TPS=15	The TPS during test phases will be monitored via Cresta SI tools.
PT002	Transaction Response Times/SLAs	How long it takes the server to respond to a client machine.	Minimum and maximum acceptable transaction response times to be defined for each area of functionality per release.	Retrieve 10 UKBA records = 3 seconds min + 5 seconds max	
PT003	Burstiness.	The percentage amount by which a sustained peak can suddenly increase.	The Burstiness ratio is to be defined as a percentage and accurately represents the worst case scenario of a surge within the system.	Burstiness = 25% within 5 seconds.	Not to be confused with contingency. This differs in that there has been no h/w failure etc.
PT004	Transaction profile at peak volumes.	The mix of different transaction types passing through the system at any given point during the peak hour.	Peak hours defined in terms of the mix of transactions at any given point. Each transaction type defined as an integer (rounding up if necessary) This should be at as granular level as possible (every 15 mins at the least) Cresta will take these projections down to the "per second" level. All volumetrics must be easily graphable.	15:00 PAY3=12 transactions in last 15 minutes; IAT=10 transaction in last 15 minutes.	The presence of Ajax will greatly change the profile, as traffic will continually be passing from the client to the server and vice versa.

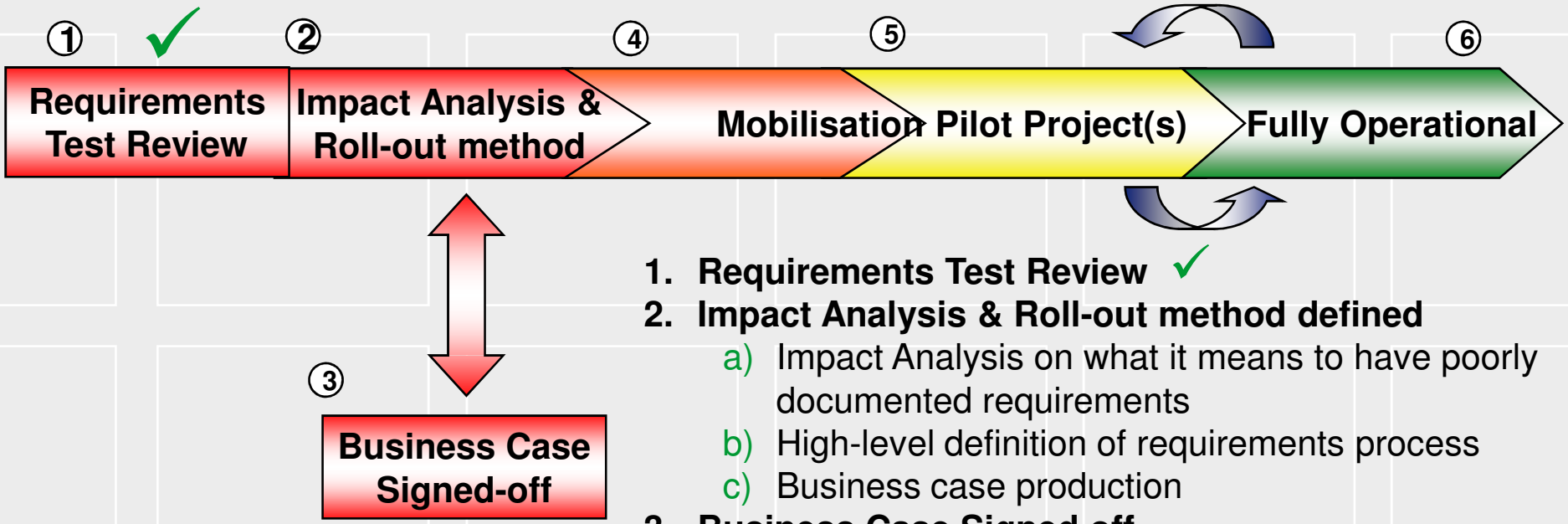
© SQS Software Quality Systems

Summary of the process


Project Start
Documentation Produced by Project
1. Project Mandate
2. Project Initiation Document (PID)
3. Test Strategy
4. Architecture/High Level solution design docs
5. Project/Documentation Standards
6. Project Organisation Structure
Analysts Upskill (Onsite) 1 – 2 weeks
1. Familiarise with Project Documentation (above).
2. Hold Acceptance Criteria workshop with Business Analysts & Developers to explain what they need to produce (Requirements and Solution Design) and to what standards to pass acceptance criteria. This occurs prior to any Requirements being produced.
3. Meet project team:
i. BAs.
ii. Developers
iii. Test Team
iv. QA
v. Project management
4. Agree Traceability Matrix and Process
1. Acceptance Criteria Defined.
2. Acceptance Criteria Presentation – training to BAs & Devs
3. Traceability Matrix Template and Process flow
Analysts Review (Offsite)
1. Review Requirements and Produce Findings
2. Review Detailed Solution Design Documents and Produce Findings
3. Produce Requirements Traceability Matrix (RTM)
4. Retest Updated Documents against Findings
5. Maintain RTM
6. Include CRs in Review and RTM
1. Requirements defect reports
2. Solutions design docs defect reports
3. RTM
All Docs signed-off
1. Produce final findings summary
2. Produce final RTM
3. Standby for inclusion of future CRs
1. Summary Findings Report
2. Final RTM



The long-term – Enterprise Approach



1. **Requirements Test Review** ✓
2. **Impact Analysis & Roll-out method defined**
   a) Impact Analysis on what it means to have poorly documented requirements
   b) High-level definition of requirements process
   c) Business case production
3. **Business Case Signed off**
4. **Mobilisation**
   - Ramping up of team and engagement
5. **Pilot Projects**
   - Engagement on pilot projects, ideally covering key project
6. **Fully Operational**
   - Review and improvement
   - Operating across several lines of business

© SQS Software Quality Systems

Case Study: Retail Bank

Case Study – Retail Bank

- Every project asset was tested against defined acceptance criteria
- This yielded not only better quality documentation, but also faster production of documentation, reduced re-work and the reduction of the risk of defects moving forward into development and testing.
- The defect detection rate here on average was 17.5 defects per hour, which is a very high and efficient rate of detection.
- On average, there were 3.5 defects per requirement, which were prevented in moving forward into development and testing.
- If these defects were not found at this stage, they would be carried forward into development and testing and based on the Return on Investment model, they would cost a factor of **30 times** more to find and fix.

© SQS Software Quality Systems

Case Study – Retail Bank

References:

- "As discussed, the interactive review with your team was very useful and we were also very grateful for your flexibility in helping us meet our deadlines this week. The feedback from the stakeholders was that the quality of the R2 document was better than R1 and I am sure that this is in part due to the work we have been doing together to remove ambiguous language. Thanks"

  Head of Business Design

  Retail Bank

© SQS Software Quality Systems

Any Questions?

Carl Allen

Director of Programme Quality Management

SQS Group

Quality Governance
Transforming IT into Business Value

SQS Group Limited

7 Moorgate
London, EC2M 6SS
United Kingdom

Phone: +44 (0) 20 7448 4620
Fax: +44 (0) 20 7448 4651

email: carl.allen@sqs-uk.com

Internet: www.sqs-uk.com