



5

- When: where the system has a significant user interface and it is important to avoid user error (e.g. this could be a critical application e.g. cockpit design in an aircraft or a consumer product that we want to be an enjoyable system to use or we might be considering efficiency (e.g. call-centre software)).
- (e.g. call-centre software), What/How: we could construct a simulator (e.g. cockpit) in the case of embedded systems or we could just have many users try the system in a controlled environment. We need to structure the test with clear objectives (e.g. to reduce decision time, to support concurrent use of certain functions...) and have good means of collecting & analysing data. Why: there may be safety issues, we may want to produce something more useable than competitors' products...
- Strengths: in well-defined contexts this can provide very good feedback often underpinned by some theory e.g. estimates of cognitive load.
- Weaknesses: some usability requirements are hard to express and hard to test, it is possible to test extensively and then not know what to do with the data.

3 March 2009

Software Testing: Lecture 13

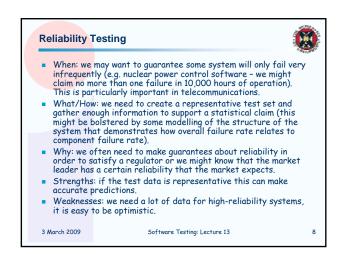
Security Testing When: most systems that are open to the outside world and have a function that should not be disrupted require some kind of security test. Usually we are concerned to thwart malicious users. What/How: there are a range of approaches. One is to use league tables of bugs/errors to check and review the code (e.g. SAMS top twenty security problems). We might also form a team that attempts to break/break into the system. Why: some systems are essential and need to keep running, e.g. the telephone system, some systems need to be secure to mainta reputation. Strengths: this is the best approach we have - most of the effort should go into design and the use of known secure components. Weaknesses: we only cover known ways in using checklists and we do not take account of novelty – using a team to try to break does introduce this. 3 March 2009 Software Testing: Lecture 13

Performance Testing

- When: many systems are required to meet performance targets laid down in a service level agreement (e.g. does your ISP give you 2Mb/s download?).
- What/How: there are two approaches modelling/simulation, and direct test in a simulated environment (or in the real environment).
- Why: often a company charges for a particular level of service this may be disputed if the company fails to deliver. E.g. the VISA payments system guarantees 5s authorisation time delivers faster and has low variance. Customers would be unhappy with less.
- Strengths: can provide good evidence of the performance of the system, modelling can identify bottlenecks and problems.
- Weaknesses: issues with how representative tests are.

3 March 2009

Software Testing: Lecture 13





Documentation Testing

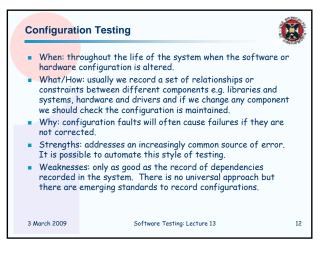


11

- When: most systems that have documentation should have it tested - and should be tested against the real system. Some systems embed test cases in the documentation and using the doc tests is an essential part of a new release.
- What/How: test set is maintained that verifies the doc set matches the system behaviour. Could also just get someone to do the tutorial and point out the errors.
- Why: the user gets really confused if the system does not conform to the documentation.
- Strengths: ensures consistency.
- Weaknesses: not particularly good on checking consistency of narrative rather than examples.

3 March 2009

Software Testing: Lecture 13



Summary



13

- There are a very wide range of potential tests that should be applied to a system.
 Not all systems require all tests.
 Managing the test sets and when they should be applied is a very complex task.
 The quality of test sets is critical to the quality of a running implementation.

3 March 2009

Software Testing: Lecture 13