16. Summary and further topics

Aims – review
The BCS Practitioner Certificate – coverage
Other topics in testing
Software testing – industry trends
Research

"Testing is the process of executing a program with the intent of finding errors."

Glen Myers

Syllabus

Aims

- To provide a systematic overview of standards, techniques and tools in software testing.
- To introduce core methodologies for the management and execution of the testing process.
- To introduce practical techniques for testing and apply them to simple examples.

Syllabus

Software testing as a SE discipline

- A systematic approach to software testing
- Framed in context of the software life cycle and as a branch of software engineering
- Builds on the prior knowledge of software engineering
- Extends the topic in both breadth and depth.
- Aims to develop approach to software testing as a professional software engineering discipline.

The Practitioner Certificate

- The second level of qualification in software testing (the first level is the Foundation Certificate)
- Administered by the British Computer Society's Information Systems Examination Board (ISEB) (www.bcs.org.uk/iseb)
- A testing practitioner is anyone involved in software testing
- Includes testers, test analysts, test engineers, test consultants, test managers, user acceptance testers, and software developers

The Practitioner Certificate Syllabus

1. Introduction
- ✓ Testing in the life cycle

2. Test process
- ✓ Generic test process
- ✓ Test planning
- ✓ Test specification
- ✓ Test execution
  - ✓ Preparation for test execution
  - ✓ Executing the tests
- ✓ Test checking and recording
- ✓ Checking for test completion

The Practitioner Certificate Syllabus

3. Test management

- ✓ Test management documentation
  - ✓ Test policy
  - ✓ Test strategy
  - ✓ Project test plan
  - ✓ Phase test plan
- ✓ Test plan documentation
- ✓ Test estimation
- ✓ Scheduling of test planning
- ✓ Test progress monitoring and control

The Practitioner Certificate Syllabus

4. Testing and risk

- ✓ Introduction to testing and risk
- ✓ Risk management
  - ✓ Risk identification
  - ✓ Risk analysis
  - – Risk mitigation

The Practitioner Certificate Syllabus

5. Test techniques

- ✓ Functional/structural testing techniques
- ✓ Non-functional testing techniques
- ✓ Dynamic analysis
- ✓ Static analysis
- – Non-systematic (ad-hoc) testing techniques
- ✓ Choosing test techniques

The Practitioner Certificate Syllabus

6. Reviews

- – The principles of reviews
- – Informal review
- – Walkthrough
- – Technical review
- – Inspection

7. Incident management (√)

8. Test process improvement (√)

The Practitioner Certificate Syllabus

9. Test tools

- ✓ Overview
- ✓ Tool selection
- ✓ Tool implementation

10. People skills

- – Individual skills (√)
- – Test team dynamics
- – Fitting testing within an organisation
- – Motivation

Other topics in testing

- Object-oriented testing
- GUI testing
- Network testing
- Testing of internet applications
- Testing concurrent systems
- Testing of safety-critical applications
- . . .

Software testing – into the future

- As software becomes more complex, it is harder to develop reliable systems, on time and on schedule
- Statistics (an USA technology consultancy)
  - 30% of all software projects are cancelled
  - 60% are considered failures by organisations that initiated them
  - Over 50% come over budget
  - 90% come in late

Software testing – into the future

- Analysis of the failures
  - The main source of weakness is the lack and poor quality of tools available to software developers
- Can this be fixed?

  "to enable average software developer to write above-average programs"

Software testing – into the future

- Idea One – change the software development model
  - Changes in the specification are the greatest problem, but cannot be avoided
  - Waterfall model outdated, but commonly used
  - Slow in responding to changes in the specification
  - Proposed solution: Iterative model, which continually cycles through the five phases (requirements, design, coding, testing, deployment)
  - Supported by major players in SE industry (e.g. Borland, IBM, Microsoft)

Software testing – into the future

- Idea Two – use tools which support the iterative software development model
  - Model development tools ☆
  - Tools for converting the model into code (although programmer input still necessary) ☆
  - Requirements automatically updated from the code, available for immediate inspection by the business developers
  - A portion of the code is tested as soon as it is implemented
  - If requirements are changed, testing will reveal discrepancies, leading to altering the code

Software testing – into the future

- Examples of tools which support the iterative process
- Borland
  - Software Delivery Optimisation approach
  - Includes management software Core SDP (Software Delivery Platform)
- IBM
  - Rational
- Microsoft
  - Visual Studio

Software testing – into the future

- Idea Three – use automated software testing tools
  - Software testing tools used infrequently of inappropriately
  - Rapid growth of the testing tools industry, especially for testing web applications (high impact)
    - Mercury (Quality Center)
    - Microsoft (.NET)
    - Borland
    - IBM WebSphere
    - Infragistics
    - HP Unified Functional Testing

Software testing – into the future

- Idea Four – open source solutions
  - "Open-source ethos allows programmers to work together more efficiently. Freedom to improve tools without restriction is the best route to efficiency"
    (Behlendorf: CollabNet, Apache web server)
  - Selling services not tools. A business client (e.g. a software house) "rents" the tools and gets the benefit of the company's expertise.

Software testing – into the future

Agile programming

- A movement which brings together the life-cycle management, software testing tools and open source software development
- Key principles
  - Developers must talk to each other often
  - They must talk to the clients often
  - They should share tools
  - Software development cycle must be shortened to weeks
- This stops projects going on for years and producing nothing

Software testing – into the future

- Research – examples
  - Semantic models of computer programs
  - Languages for modelling systems and their properties (e.g. UML)
  - Software verification (including probabilistic verification)
  - Verification and validation of properties of systems
  - Model checking (translation of computer program into a finite-state model, which can be analysed mathematically / symbolically)
  - Component – based software development

Software testing – into the future

- Research in the School of Computer Science
  - Dr Rami Bahsoon
  - Dr Dan Ghica
  - Prof. Uday Reddy
  - Dr Eike Ritter
  - Dr Mark Ryan
  - Dr Hayo Thielecke
  - Prof. Xin Yao