

MACIASZEK, L.A. (2005):
Requirements Analysis and System Design, 2nd ed.
 Addison Wesley, Harlow England, 504p.
 ISBN 0 321 20464 6

Chapter 9
Testing and Change Management

© Pearson Education Limited 2005

Topics

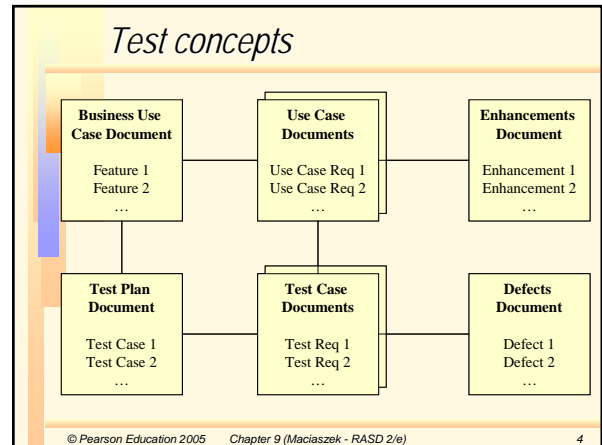
- Test concepts
- Test techniques
- Test driven development
- Managing change
- Traceability

© Pearson Education 2005 Chapter 9 (Maciaszek - RASD 2/e) 2

Main concepts

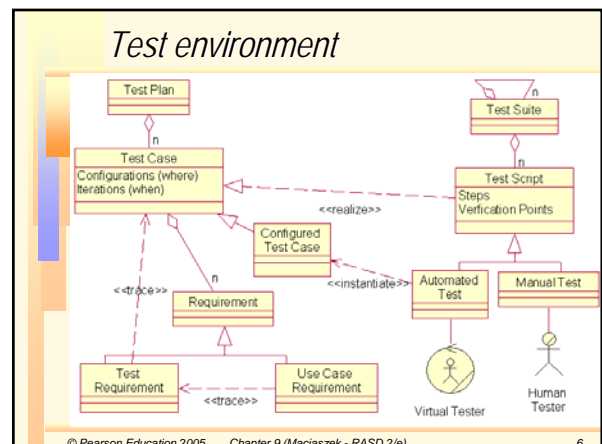
- Testing** is not just the debugging of programs – it is part of quality management:
 - Quality assurance** is about proactive ways of building quality into a software system
 - Quality control** is about (mostly reactive) ways of testing the quality of a software system
- Test driven development** builds quality into a software system by an outright demand that a test code has to be written before the application code and that the application must pass the test to be quality assured
- Change management** is a fundamental aspect of the overall project management
- Traceability** underlies testing and change management

© Pearson Education 2005 Chapter 9 (Maciaszek - RASD 2/e) 3



Test case document

© Pearson Education 2005 Chapter 9 (Maciaszek - RASD 2/e) 5



Testing system services

- Informal testing
- Methodical testing
 - Non-execution-based (formal reviews)
 - Walkthroughs
 - Inspections
 - Execution-based
 - Testing to specs
 - Testing to code

© Pearson Education 2005 Chapter 9 (Maciaszek - RASD 2/e) 7

Walkthrough

- A type of formal brainstorm review that can be conducted in any development phase
- A friendly meeting of developers, carefully planned and with clear objectives, an agenda, duration, and membership
- A few days prior to the meeting, the participants are handed the materials to be reviewed
- During the meeting the problems need to be pinpointed but no solutions attempted
- Acknowledged problems are entered on a walkthrough issues list
- The list is used by the developer to make corrections to the reviewed software product or process
- A follow-up walkthrough may be necessary

© Pearson Education 2005 Chapter 9 (Maciaszek - RASD 2/e) 8

Inspection

- Like the walkthrough, an inspection is a friendly meeting but done under close supervision by the project management
- The purpose is also to identify defects, validate that they are in fact defects, record them, and schedule when and by whom they have to be fixed
- Unlike the walkthroughs, inspections are conducted less frequently, may target only selected and critical issues, and are more formal and more rigorous
- An informational meeting usually takes place one week before the inspection meeting
- During the meeting, the defects are identified, recorded and numbered
- Immediately after the meeting, the moderator prepares the defect log – recorded in a change management tool
- The developer is normally requested to resolve the defects quickly and record the resolution in the change management tool
- The moderator – in consultation with the project manager – submits the development module to the Software Quality Assurance (SQA) group in the organization.

© Pearson Education 2005 Chapter 9 (Maciaszek - RASD 2/e) 9

Testing to specs

- **Execution-based** test type
 - applies to executable software products, not to documents or models
 - also known as
 - black-box testing
 - functional testing
 - input/output driven testing
- Test module treated as a **black box** that takes some input and produces some output (no need to understand the program logic or computational algorithms)
- Requires that the test requirements are derived from the use case requirements, and then identified and documented in a separate **test plan** and **test case documents**
- Test scenarios can be recorded in a **capture-playback** tool and used repeatedly for **regression testing**
- Likely to discover defects normally difficult to catch by other means, such as missing functionality

© Pearson Education 2005 Chapter 9 (Maciaszek - RASD 2/e) 10

Testing to code

- **Execution-based** testing
- Also known as
 - white-box testing
 - glass-box testing
 - logic-driven testing
 - path-oriented testing
- Starts with the careful analysis of the program's algorithms
- Test cases are derived to **exercise** the code – i.e. to guarantee that all possible execution paths in the program are verified
 - the **test data** are specially contrived to exercise the code
- Can be supported by the **capture-playback** tools and used then for **regression testing**
 - playback scripts need to be written by the programmer rather than generated by the tool
- Like all other forms of execution-based testing, testing to code cannot be exhaustive

© Pearson Education 2005 Chapter 9 (Maciaszek - RASD 2/e) 11

Testing system constraints

- Predominantly **execution-based**
- Includes such issues as:
 - user interface testing
 - database testing
 - authorization testing
 - performance testing
 - stress testing
 - failover testing
 - configuration testing
 - installation testing

© Pearson Education 2005 Chapter 9 (Maciaszek - RASD 2/e) 12

User interface testing

- Is the window modal or modeless? Which should it be?
- Is a visual distinction made between the required and optional fields?
- Are any fields missing?
- Are there any spelling mistakes in titles, labels, prompt names, etc.?
- Are command buttons (OK, Cancel, Save, Clear, etc.) used consistently across all dialog boxes?
- Is it possible always to abort the current operation (including the delete operation)?
- Are all static fields protected from editing by users? If the application can change the static text, is this being done correctly?
- Do the sizes of edit boxes correspond to ranges of values that they take?
- Are the values entered into edit boxes validated by the client program?
- Are the values in drop-down lists populated correctly from the database?
- Are edit masks used in entry fields as specified?
- ...

© Pearson Education 2005 Chapter 9 (Maciaszek - RASD 2/e) 13

Database testing

- Verify that the transaction executes as expected with correct input. Is the system's feedback to the UI correct? Is the database content correct after the transaction?
- Verify that the transaction executes as expected with incorrect input. Is the system's feedback to the UI correct? Is the database content correct after the transaction?
- Abort the transaction before it finishes. Is the system's feedback to the UI correct? Is the database content correct after the transaction?
- Run the same transaction concurrently in many processes. Deliberately make one transaction hold a lock on a data resource needed by other transactions. Are the users getting understandable explanations from the system? Is the database content correct after the transactions have terminated?
- Extract every client SQL statement from the client program and execute it interactively on the database. Are the results as expected and the same as when the SQL is executed from the program?
- ...

© Pearson Education 2005 Chapter 9 (Maciaszek - RASD 2/e) 14

Authorization testing

- The **user interface** of the program should be able to configure itself dynamically to correspond to the **authorization level** of the current user (**authenticated** by the user id and password)
- Server permissions (privileges):**
 - to **access individual server objects** (tables, views, columns, stored procedures, etc.)
 - to **execute SQL** statements (select, update, insert, delete, etc.)
- Permissions for a user may be assigned at a **user level** or at a **group level**
- Most DBMSs support also the **role level**
- An **authorization database** may need to be set up alongside the application database to store and manipulate the client and server permissions

© Pearson Education 2005 Chapter 9 (Maciaszek - RASD 2/e) 15

Testing of other constraints

- performance testing**
 - transaction speed and throughput
 - peak loads
- stress testing**
 - to break the system when abnormal demands are placed on it
 - frequently coupled with performance testing
- failover testing**
 - system's response to a variety of hardware, network or software malfunctions
 - closely related to the DBMS recovery procedures
- configuration testing**
 - how the system operates on various software and hardware configurations.
- installation testing**
 - extends the configuration testing
 - verifies that the system operates properly on every platform installed

© Pearson Education 2005 Chapter 9 (Maciaszek - RASD 2/e) 16

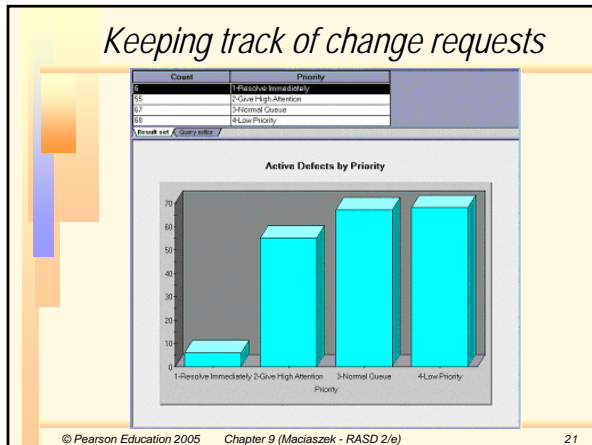
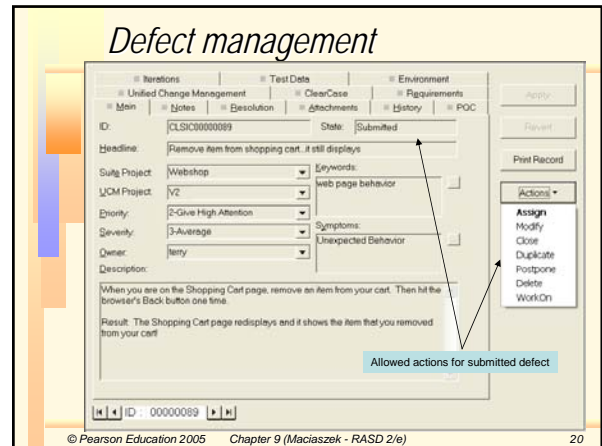
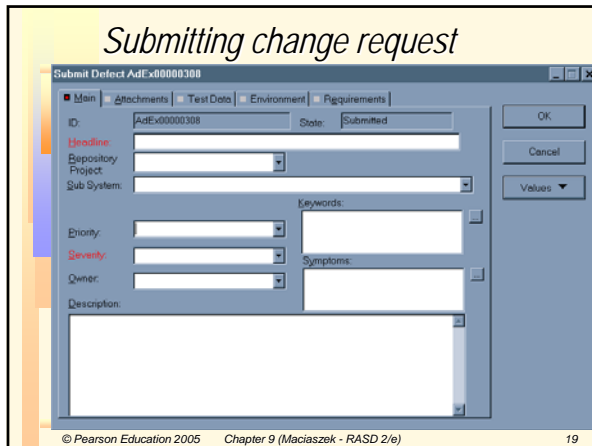
Test driven development

- The idea is to write **test cases** and **scripts** as well as **test programs** before the application code (**the unit under test**) is developed (designed and programmed)
 - application code** is written as a response to a **test code** and the test code can be used to test the application code as soon as it is available
- Advantages**
 - allows to clarify user requirements (and the use case specifications) before the programmer writes the first line of the application code
 - drives in fact the software development, not just the software verification
 - supported by testing frameworks, such as JUnit

© Pearson Education 2005 Chapter 9 (Maciaszek - RASD 2/e) 17

Managing change

© Pearson Education 2005 Chapter 9 (Maciaszek - RASD 2/e) 18



- ### Traceability
- There is a significant **cost** to the project associated with the traceability, testing and change management
 - the **cost-benefit analysis** should determine the scope and depth of project traceability
 - as a minimum, the traceability should be maintained between the use case requirements and defects
 - What can be traced?
 - Features are linked to test cases and to use case requirements in the use case documents
 - Test requirements in the test case documents can be traced back to test cases and use case requirements
 - Test requirements are linked to defects and enhancements are traced to use case requirements
- © Pearson Education 2005 Chapter 9 (Maciaszek - RASD 2/e) 22

