

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

---

Chapter 9  
*Testing and Change Management*

© Pearson Education Limited 2005

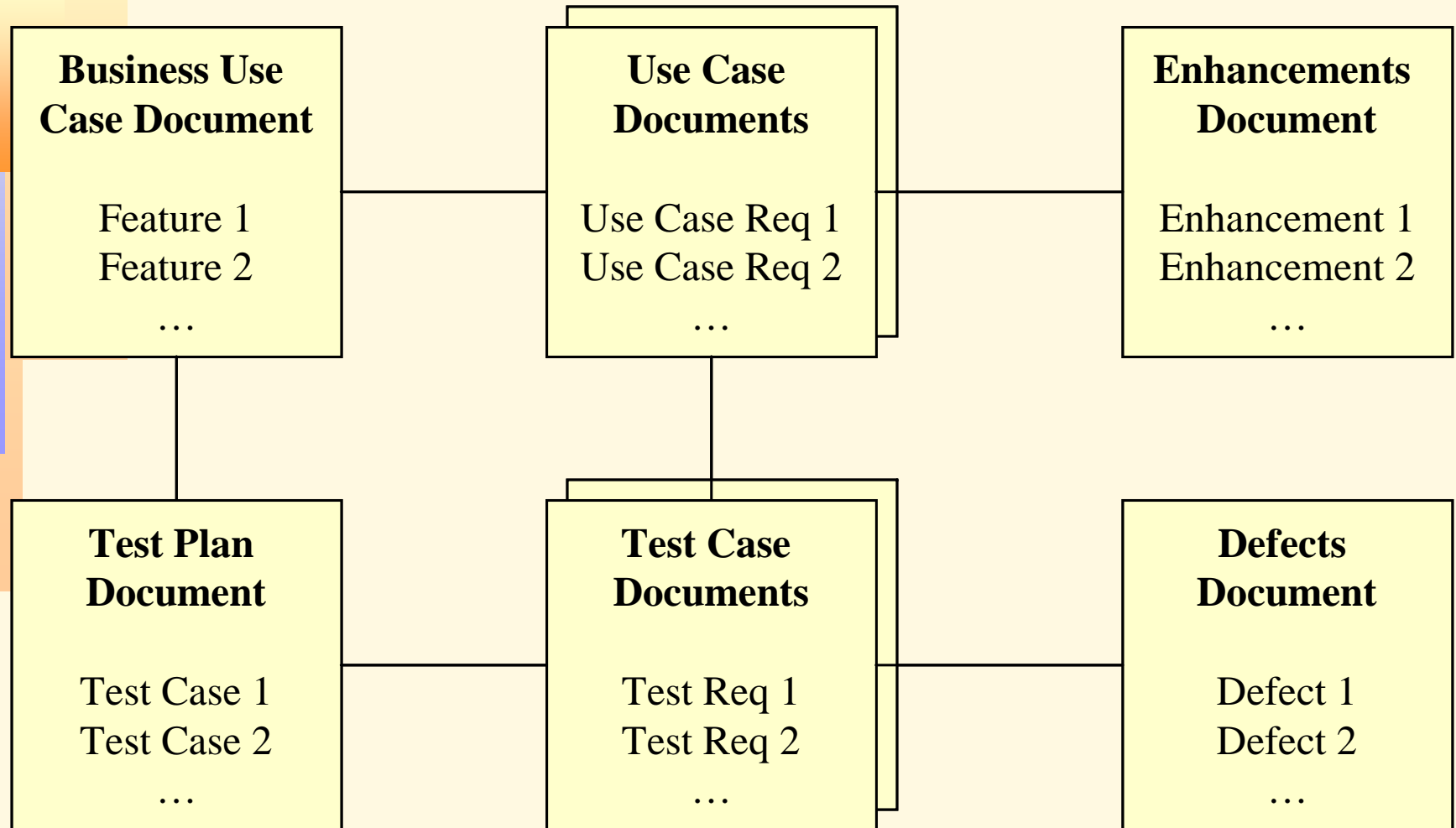
# *Topics*

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

# *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

# Test concepts



# Test case document

The screenshot shows a Microsoft Word window titled 'ConfigTest.TC'. The document contains the following information:

Application Name : OnLine Shopping

Unit Name : [TCR1 Order Configured Computer](#) Version : [ ]

Location : \\Kosciuszko\Projects\Test Cases\Test\_OrdConfCmp

Use Case : OnLineShopping :: Order\_Conf\_Comp

Description : [ ]

Tester : Leszek Maciaszek Test Date : : 08/04/00 Result : Pass  Fail

**Test Results Metrics**

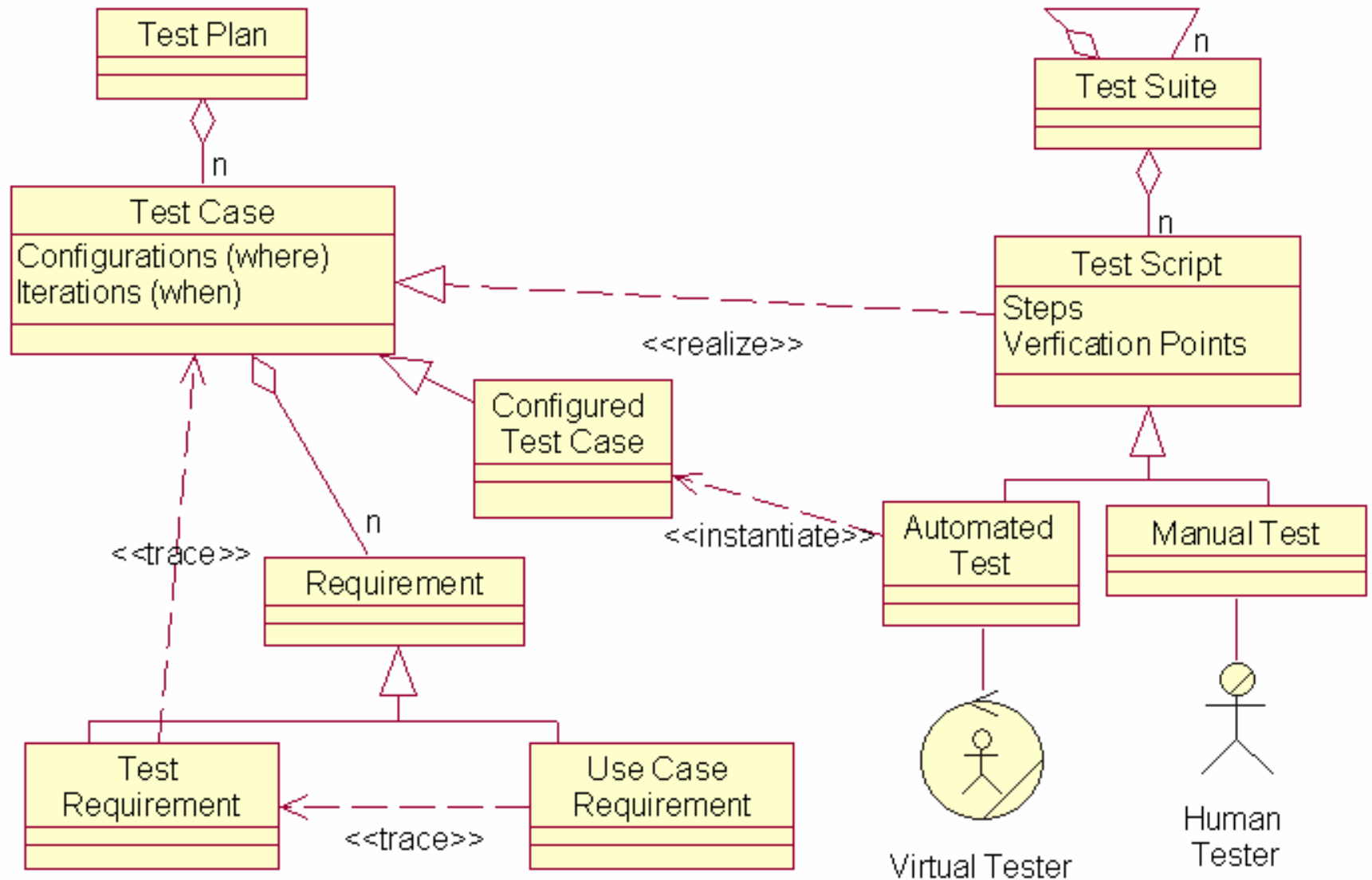
|                   |     |                 |     |                        |     |
|-------------------|-----|-----------------|-----|------------------------|-----|
| No of Classes :   | [ ] | No of Methods : | [ ] | No of Lines :          | [ ] |
| Coding Errors :   | [ ] | Design Errors : | [ ] | Specification Errors : | [ ] |
| Database Errors : | [ ] | GUI Errors :    | [ ] |                        |     |

**Conformance to Use Case Specs**

|  | Fail                     | Pass conditionally | Pass                     |
|--|--------------------------|--------------------|--------------------------|
| <a href="#">TCR1.1 Order entry form displays in the Web browser after pressing the "I Want to Buy It" button</a> | <input type="checkbox"/> |                    | <input type="checkbox"/> |
| <a href="#">TCR1.1.1 The title of the form is "Order Your Computer"</a>  | <input type="checkbox"/> |                    | <input type="checkbox"/> |

Page 1 Sec 1 1/7 At 2.3" Ln 5 Col 33 REC TRK EXT OVR WPH

# Test environment



# *Testing system services*

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

# 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

# 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.

# 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*

# 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

# *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*

# *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?*
- *...*

# 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?*
- ...

# 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*

# *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*

# *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*

# Managing change

Rational ClearQuest - [CLSIC : Rational Demo (All Defects (Defect))]

File Edit View Actions Query Window

Workspace browser with predefined queries, charts and reports

| id            | Headline  | Severity   | Owner | State    |
|---------------|---|------------|-------|----------|
| CLSIC00000037 | spelling error in login screen                    | 3-Average  | alex  | Resolved |
| CLSIC00000038 | sales tax incorrect if item deleted from purchase | 1-Critical | dale  | Resolved |
| CLSIC00000039 | cancel sale doesn't correctly repaint screen      | 3-Average  | sandy | Resolved |
| CLSIC00000040 | columns out of alignment                          | 3-Average  | devon | Resolved |
| CLSIC00000041 | delete item not working correctly                 | 2-Major    | sandy | Opened   |
| CLSIC00000042 | override price does not work                      | 2-Major    | chris | Resolved |
| CLSIC00000043 | alt-C does not invoke cancel operation            | 3-Average  | chris | Resolved |
| CLSIC00000044 | clerk allowed to charge too much on credit card   | 4-Minor    | jan   | Resolved |

Result set | Query editor | Display editor /

List of defects

Iterations | Test Data | Environment

Unified Change Management | ClearCase | Requirements

Main | Notes | Resolution | Attachments | History | PQC

ID: CLSIC00000039 State: Resolved

Headline: cancel sale doesn't correctly repaint screen

Suite Project: ClassicsPOS Keywords:

UCM Project:

Priority: 3-Normal Queue

Severity: 3-Average

Owner: sandy

Description: cancel sale function does not redraw the form - refresh clears up the problem

Symptoms: Cosmetic Flaw

Apply

Revert

Print Record

Actions

- Validate
- Modify
- Reject
- Duplicate
- Delete
- WorkOn

Details of selected defect

Possible actions (state changes)

Record: 3 Count: 55

# Submitting change request

Submit Defect AdEx00000308

Main  Attachments  Test Data  Environment  Requirements

ID: AdEx00000308 State: Submitted

**H**eadline:

**R**epository:

**P**roject:

**S**ub System:

**P**riority:

**S**everity:

**O**wner:

**D**escription:

Keywords:

Symptoms:

OK

Cancel

Values ▾

# Defect management

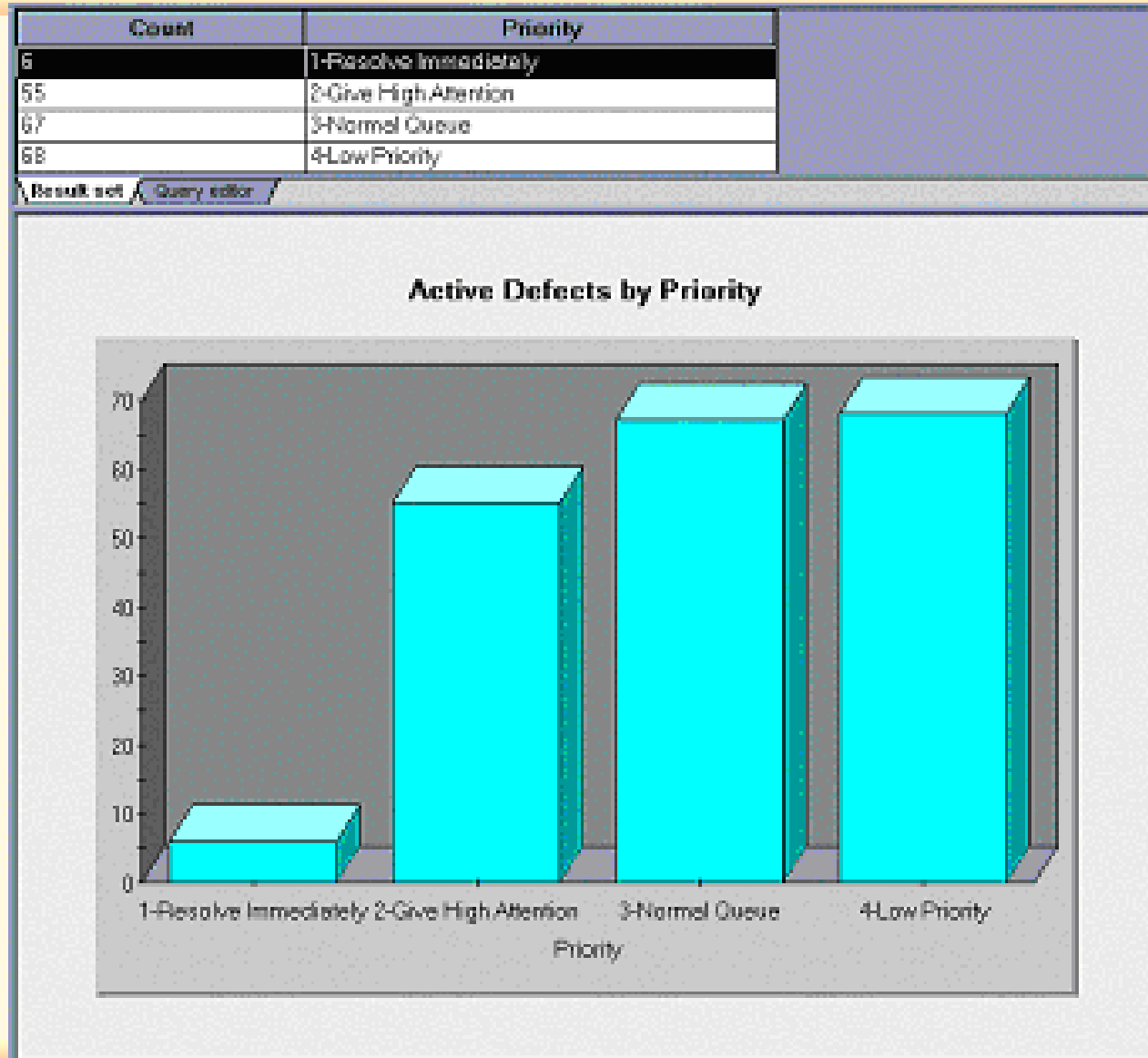
The screenshot displays a defect management application interface. At the top, there are several tabs: Iterations, Test Data, Environment, Unified Change Management, ClearCase, Requirements, Main, Notes, Resolution, Attachments, History, and PQC. The 'Main' tab is selected. The defect details are as follows:

- ID: CLSIC00000089
- State: Submitted
- Headline: Remove item from shopping cart...it still displays
- Suite Project: Webshop
- UCM Project: V2
- Priority: 2-Give High Attention
- Severity: 3-Average
- Owner: terry
- Description: When you are on the Shopping Cart page, remove an item from your cart. Then hit the browser's Back button one time.  
Result: The Shopping Cart page redisplay and it shows the item that you removed from your cart!
- Keywords: web page behavior
- Symptoms: Unexpected Behavior

On the right side, there are buttons for 'Apply', 'Revert', and 'Print Record'. Below these is an 'Actions' dropdown menu which is open, showing the following options: Assign, Modify, Close, Duplicate, Postpone, Delete, and WorkOn. A blue callout box with the text 'Allowed actions for submitted defect' has two arrows pointing to the 'Assign' and 'WorkOn' options in the actions menu.

At the bottom of the interface, there is a navigation bar with left and right arrow buttons and the text 'ID : 00000089'.

# Keeping track of change requests



# 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*

# System features to use cases and use case requirements

The screenshot shows the RequisitePro interface with the following components:

- Relationships:** - direct only
- Feature List:**
  - FEAT2: Raw Ad Instance Extraction
  - FEAT3: Automatic Matching
  - FEAT4: Manual Matching
  - FEAT5: Manual Ad Instance Entry
  - FEAT6: Ad Verification
  - FEAT7: Ad Business Relationship...
  - FEAT8: Ad Rates and Demographics
  - FEAT9: Customer Report Generation
- Use Case List:**
  - UC10: Filter Displayed Data
  - UC18: Maintain Ad Duration...
  - UC19: Maintain Ad Links
  - UC46: Maintain Ads
    - UC46.1: Update An Ad
      - UC46.1.1: Update a TV, Radio...
      - UC46.1.2: Update an Outdoor...
      - UC46.1.3: Update a Print Ad
    - UC46.2: Display Ad Links For Ad
    - UC46.3: Display Raw Ads For...
    - UC46.4: Sort Ads
    - UC46.5: Blank Key Number
  - UC47: Maintain Advertiser Groups
    - UC47.1: Assigning Advertisers...
    - UC47.2: Adding Advertiser...
    - UC47.3: Deleting Advertiser...
    - UC47.4: Changing The Group...
    - UC47.5: Group Has Sub-Group
    - UC47.6: Group Has Advertisers
    - UC47.7: Group Or Sub-Group...
  - UC55: Maintain Agency Groups

The interface includes a menu bar (File, View, Requirement, Window, Help), a toolbar, and a status bar at the bottom showing "Ready" and "8 requirements".

# Test plans to test cases and test requirements

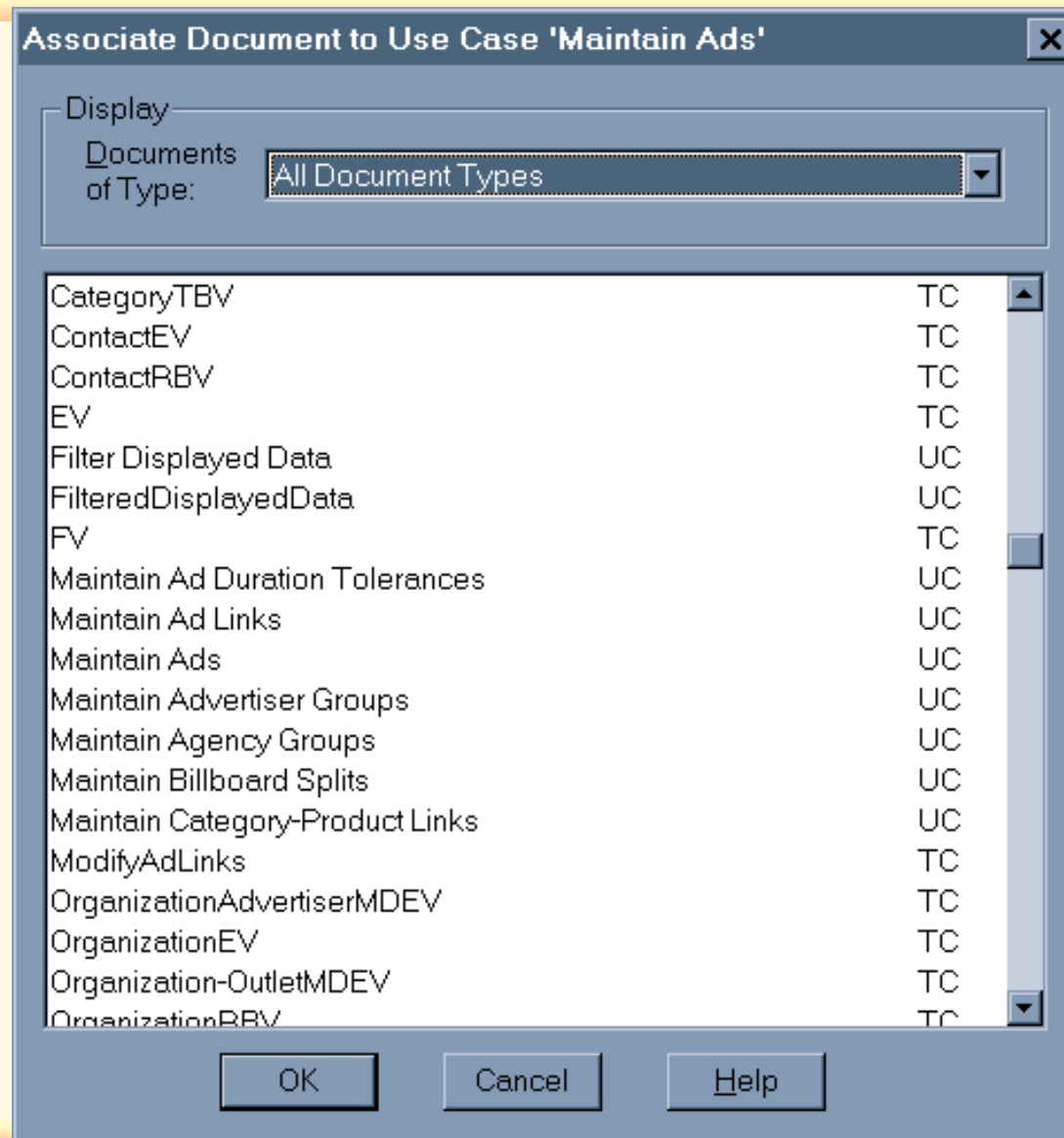
The screenshot displays the RequisitePro interface for 'Quality Control' test plans and requirements. The window title is 'RequisitePro Views - [TPL-TR: Quality Control]'. The menu bar includes 'File', 'View', 'Requirement', 'Window', and 'Help'. The toolbar contains icons for file operations and navigation.

**Relationships: - direct only**

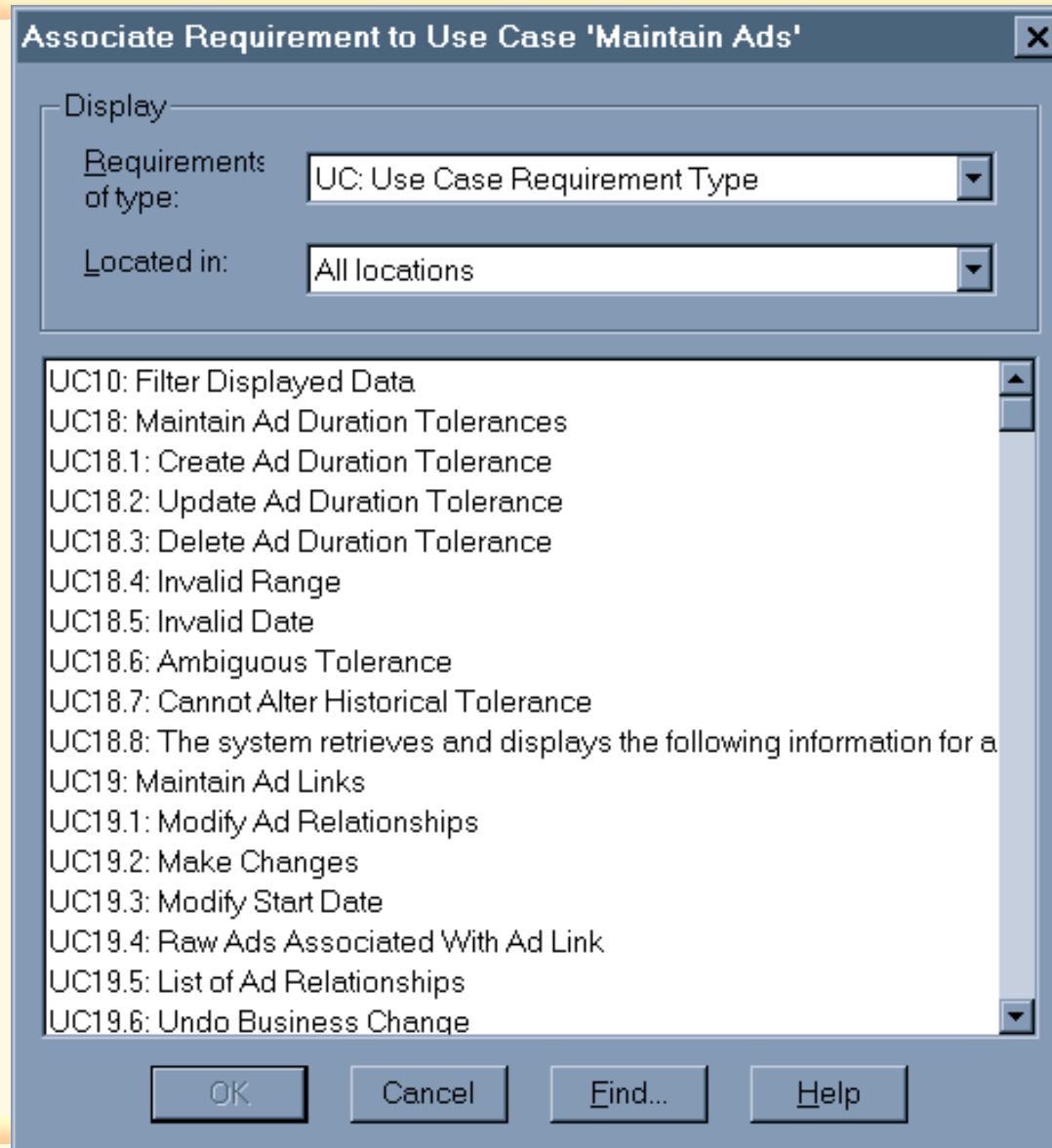
| Test Plan (TPL)                       | Test Requirement (TR)              | Relationship |
|---------------------------------------|------------------------------------|--------------|
| TPL3.31: Outlet Edit                  | TR28.1: Modify Ad Links (incl...)  |              |
| TPL3.32: Organisation Edit            | TR28.2: Insert, Update and...      |              |
| TPL3.33: Organisation-Advertiser Edit | TR28.3: Each of the three menu...  |              |
| TPL3.34: Organisation-Outlet Edit     | TR28.4: Modify Ad Links (MAL)...   |              |
| TPL3.35: Outlet Row Browse            | TR28.5: Fields that cannot be...   |              |
| TPL3.36: Link Restrictions Row Browse | TR28.6: MAL window does not...     |              |
| TPL3.37: Relink Raw Ads               | TR28.7: On entry, the system...    |              |
|                                       | TR28.8: If the Quality Control...  |              |
|                                       | TR28.9: If the Quality Control...  |              |
|                                       | TR28.10: When Business...          |              |
|                                       | TR28.10.1: the system...           |              |
|                                       | TR28.10.2: The Quality Control...  |              |
|                                       | TR28.10.3: if any of the...        |              |
|                                       | TR28.11: In both Correction and... |              |
|                                       | TR28.11.1: if the Quality...       |              |
|                                       | TR28.11.2: The Quality Control...  |              |
|                                       | TR28.11.3: if the Quality...       |              |
|                                       | TR28.11.4: The Quality Control...  |              |

The bottom status bar shows 'Ready' and '44 requirements'.

# UML diagrams to documents



# UML diagrams to requirements



# Use case requirements to test requirements

The screenshot shows a software interface titled "UC-TR: Quality Control". It features a tree view on the left and a grid on the right. The tree view is expanded to show "UC18: Maintain Ad Duration Tolerances", which includes sub-items like "UC18.1: Create Ad Duration...", "UC18.2: Update Ad Duration...", "UC18.3: Delete Ad Duration Tolerance", "UC18.4: Invalid Range", "UC18.5: Invalid Date", "UC18.6: Ambiguous Tolerance", "UC18.7: Cannot Alter Historical...", and "UC18.8: The system retrieves and...". The "UC18.8" item is selected. The grid on the right shows relationships between these use cases and test requirements (TR2). The grid has columns for "TR2: Ad Duration Edit" and "TR2.16.1: The name of this...". The "UC18.8" row has four blue arrows pointing to the right, indicating relationships with the first four TR2 items in the second column. The status bar at the bottom displays the text: "UC18.8: The system retrieves and displays the following information for all ad duration tolerances: media" and "TR2.16.1: The name of this field is 'Ad gets charged at [value] % of standard rate'".

UC-TR: Quality Control

Relationships:  
- direct only

TR2: Ad Duration Edit

- TR2.1: EV window consists of a...
- TR2.2: EV window does not...
- TR2.3: There are three...
- TR2.4: The name of a specific...
- TR2.5: The Record menu in the...
- TR2.6: EV window is modal, i.e...
- TR2.7: OK button applies the...
- TR2.8: Cancel button ignores all.
- TR2.9: Command buttons are...
- TR2.10: Pressing ENTER...
- TR2.11: Navigation between...
- TR2.12: Information entered in...
- TR2.13: There are three Ad...
- TR2.14: The following fields are...
- TR2.15: Duration, lower limit...
- TR2.16: Portion of standard rate
- TR2.16.1: The name of this...**
- TR2.17: Insert Ad Duration...
- TR2.17.1: The system...

UC18: Maintain Ad Duration Tolerances

- UC18.1: Create Ad Duration...
- UC18.2: Update Ad Duration...
- UC18.3: Delete Ad Duration Tolerance
- UC18.4: Invalid Range
- UC18.5: Invalid Date
- UC18.6: Ambiguous Tolerance
- UC18.7: Cannot Alter Historical...
- UC18.8: The system retrieves and...**

UC19: Maintain Ad Links

UC18.8: The system retrieves and displays the following information for all ad duration tolerances: media

TR2.16.1: The name of this field is "Ad gets charged at [value] % of standard rate"

# Test requirements to defects

The screenshot shows the Rational ClearQuest interface. The main window displays a table of defects with the following data:

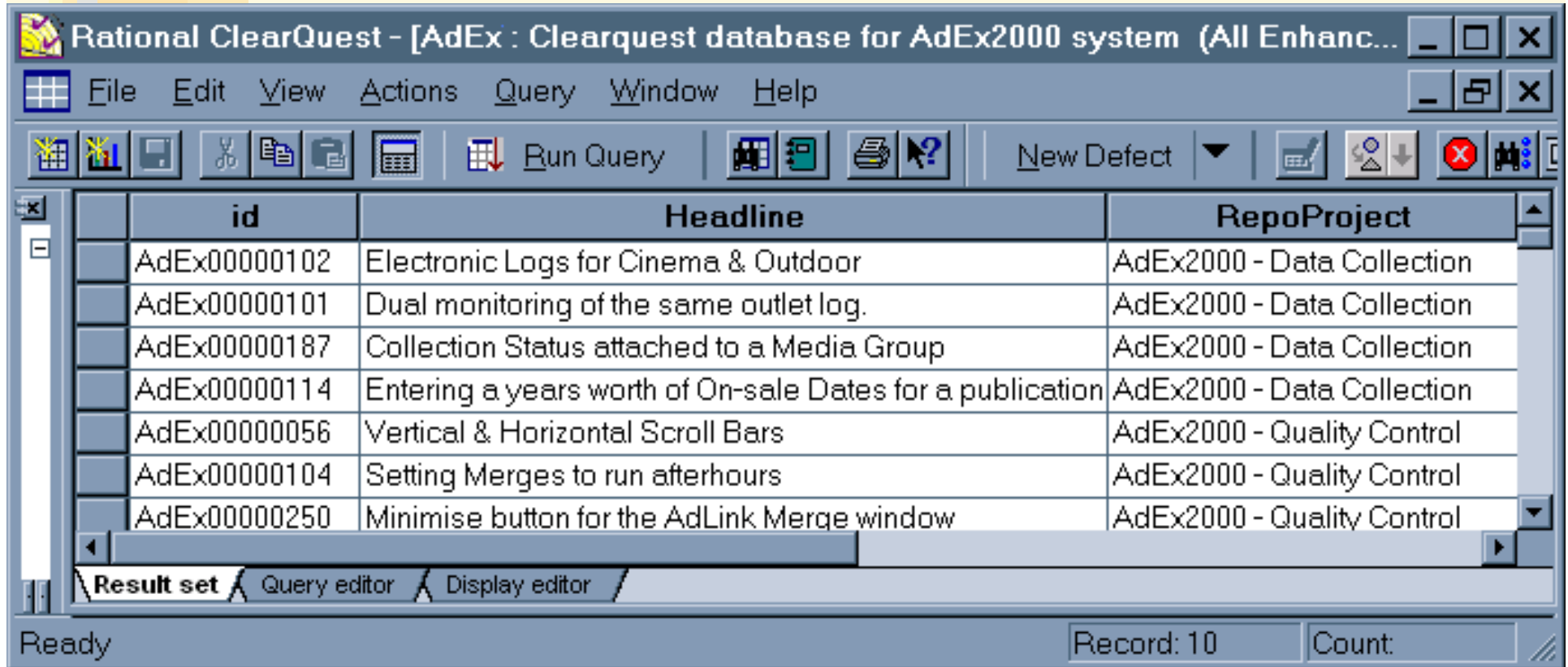
| id           | Headline                          | RepoProj   | SubSy                         |
|--------------|-----------------------------------|------------|-------------------------------|
| AdEx00000132 | Modify Ad Links screen labels     | AdEx2000 - | Modify Ad Link Edit View (E   |
| AdEx00000138 | Report too large to print message | AdEx2000 - | Product Split Report          |
| AdEx00000139 | Incorrect Specifications          | AdEx2000 - | Split Billboard Edit View (EV |
| AdEx00000140 | AdPart Numbers / Billboards       | AdEx2000 - | Split Billboard Edit View (EV |
| AdEx00000141 | AdPart Numbers / Billboards       | AdEx2000 - | Undo Billboard Split Edit Vie |
| AdEx00000142 | Adding a new column               | AdEx2000 - | Ad Links Row Browse View      |
| AdEx00000143 | Column width                      | AdEx2000 - | Ad Links Row Browse View      |

Below the table, the 'Associated Requirements' section is visible, containing a table with the following data:

| Tag       | Requirement   |
|-----------|---|
| TR23.23.6 | The new ad link is identified as a separate part of the split billboard |
| TR23.25.1 | Enter a legitimate, complete record. Verify that the transaction is e.. |

The interface also shows a menu bar (File, Edit, View, Actions, Query, Window, Help), a toolbar with icons for various actions, and a status bar at the bottom indicating 'Ready', 'Record: 67', and 'Count:'.

# *Use case requirements to enhancements*



The screenshot shows the Rational ClearQuest application window. The title bar reads "Rational ClearQuest - [AdEx : Clearquest database for AdEx2000 system (All Enhanc...". The menu bar includes "File", "Edit", "View", "Actions", "Query", "Window", and "Help". The toolbar contains various icons, including "Run Query" and "New Defect". The main area displays a table with three columns: "id", "Headline", and "RepoProject". The table contains seven rows of data. At the bottom of the window, the status bar shows "Ready", "Record: 10", and "Count:".

| id           | Headline  | RepoProject                |
|--------------|---|----------------------------|
| AdEx00000102 | Electronic Logs for Cinema & Outdoor                      | AdEx2000 - Data Collection |
| AdEx00000101 | Dual monitoring of the same outlet log.                   | AdEx2000 - Data Collection |
| AdEx00000187 | Collection Status attached to a Media Group               | AdEx2000 - Data Collection |
| AdEx00000114 | Entering a years worth of On-sale Dates for a publication | AdEx2000 - Data Collection |
| AdEx00000056 | Vertical & Horizontal Scroll Bars                         | AdEx2000 - Quality Control |
| AdEx00000104 | Setting Merges to run afterhours                          | AdEx2000 - Quality Control |
| AdEx00000250 | Minimise button for the AdLink Merge window               | AdEx2000 - Quality Control |

# Summary

- **Testing and change management** span the development lifecycle
- Testing has two dimensions
  - It is a reactive (post factum) activity when used as a **quality control** mechanism
  - It can, however, be a very proactive quality assurance activity when used within the framework of the **test driven development**
- Testing and change management assume that the **traceability** links between system artifacts exist and have been properly maintained during the development
- **Testing** divides into the testing of **system services** and the testing of **system constraints**
- **Test requirements** are identified in the test case documents and linked to the use case requirements in the use case documents
- A **change request** is normally either a **defect** or an **enhancement**