Design for Testability

Software Quality Consulting Inc.

Steven R. Rakitin
President

- Consulting
- Training
- Auditing

Phone: 508.529.4282
Fax: 508.529.7799

www.swqual.com
info@swqual.com

Topics

- **What is Design for Testability?**
  - **How does it apply to software?**
- **Design for Testability Techniques**
  - **Write Testable Requirements**
  - **Use Alternative Techniques for Expressing Requirements**
- **Storytelling as a critical tester skill**
- **Summary**

What Is Design for Testability?

- **Design for Testability (DFT) was developed by electrical engineers to improve the designs of circuit boards.**
  - **Objective: design boards with the intention of testing them using conventional automated testing equipment.**
  - **Example: By bringing certain test points out to the edge, test engineers can test more functions board is intended to perform, thus lowering failure rates and improving overall quality.**

How does DFT apply to software?

- **Brad Pettichord defines DFT as:**
  - **Testability is a design issue and needs to be addressed with the design of the rest of the system.**
  - **Projects that defer testing to the later stages of a project will find their programmers unwilling to consider testability changes by the time testers actually roll onto the project and start making suggestions.**
  - **Testability takes cooperation, appreciation and a team commitment to reliability.**

Pettichord, Bret, “Design for Testability”, presented at Pacific Northwest Software Quality Conference, October 2002.

Write Testable Requirements

- **Good writing improves comprehension and reduces ambiguity**
- **Good writing helps:**
  - **reduce confusion and misunderstanding**
  - **reduce time spent in reviews**
  - **improve testability**
  - **get it right the first time**
  - **eliminate avoidable rework**
- **Most people have had little or no training in good writing techniques…**

Write Testable Requirements

- **In your organization, what percent of people have excellent writing skills?**
- **Of those, how many understand the intricacies of writing requirements for software?**
- **Of those, how many are in a position where writing requirements is part of their job?**

Have you seen this problem?

English is not a good language for writing requirements...

Directions: Count the number of occurrences of the letter "e" and write your answer in the space provided.

"While inspections will not solve all problems, they are enormously effective. [Inspections] have been demonstrated to improve both quality and productivity by impressive factors. Inspections are not magic and they should not be considered a replacement for testing, but all software organizations should use inspections or similar technical review methods in all major aspects of their work. This includes requirements, design, implementation, test, maintenance, and documentation."

Answer: The letter "e" appears _____ times

Humphrey, W., Managing the Software Process, Addison-Wesley, 1989

Assessing Testability

- **Requirements Review is essential**
- **Role of testers at Requirements Review must be clear**
  - **Is every requirement testable?**
- **Raising testability issues early greatly increases the chances they will be addressed**

Write Testable Requirements

- **Use Short Sentences**
  - **Long, complicated sentences often indicate you aren't clear about what you want to say**
  - **Short sentences show clear thinking**
    - **Keep sentences to about 15-20 words**
  - **Short sentences convey complex information easier**
    - **Information in small, easier-to-process units is easier to grasp**
- **Express only one idea in each sentence**

Write Testable Requirements

- **Use Short Sentences**

  If the user does not enter anything into the field, a window should be displayed (before returning to the Request Addition to List Screen) saying that a value must be entered.

  1 sentence - 31 words

- **How can we rewrite this with fewer words?**

  If the user enters nothing, the "Value Must Be Entered" window is displayed before returning to the Request Addition to List Screen.

  1 sentence - 22 words

Write Testable Requirements

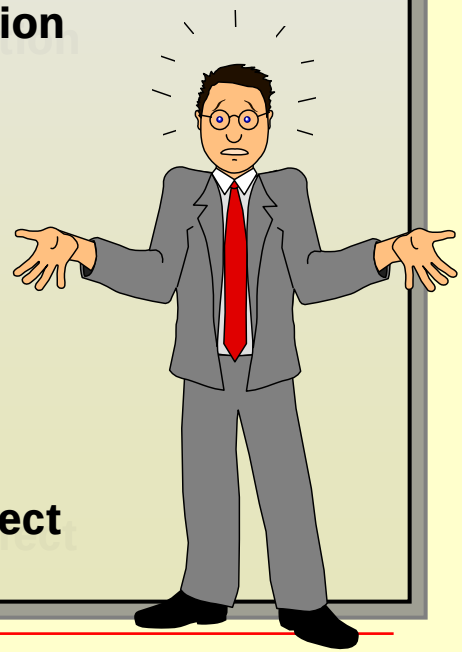
- **Use the simplest tense you can**
  - **Using present tense avoids clutter of compound verbs and clearly conveys what is required.**
    - **This table will be sortable by any column, and the user will have the option of saving the table as an Excel spreadsheet by clicking File, Save As.**
      1 sentence = 28 words
    - **This table is sortable by any column. The user can save the table as an Excel spreadsheet by clicking File, Save As.**
      2 sentences - avg sentence length = 10.5 words

Write Testable Requirements

- **Don't mix tenses**

  When equipment becomes available, the Standby request with the highest tier is filled (if two requests have the same tier, the request that was made first is filled). If this request was designated as an Autobook, a reservation will be made for the IMC and an email will be sent with the reservation number. If the request was designated as a Notify, a reservation is not booked for that IMC, but an email is sent letting the customer know that equipment is available. The next request designated as Autobook would have a reservation made.

  – **Tenses used: past, present, future, present perfect**



Write Testable Requirements

- **Define terms that affect requirements**
  - **SHALL, MUST**
  - **SHOULD, COULD, WOULD**
  - **WILL, MAY, MIGHT**
- **What do these words mean in a Requirements document?**
- **Create and use a glossary of technical terms**
  - **Use terms consistently**

Please try this at work...

List at least 5 nouns and 5 verbs that are specific to each group

Users

Developers

Stakeholders

Testers

Alternative Techniques for Expressing Requirements

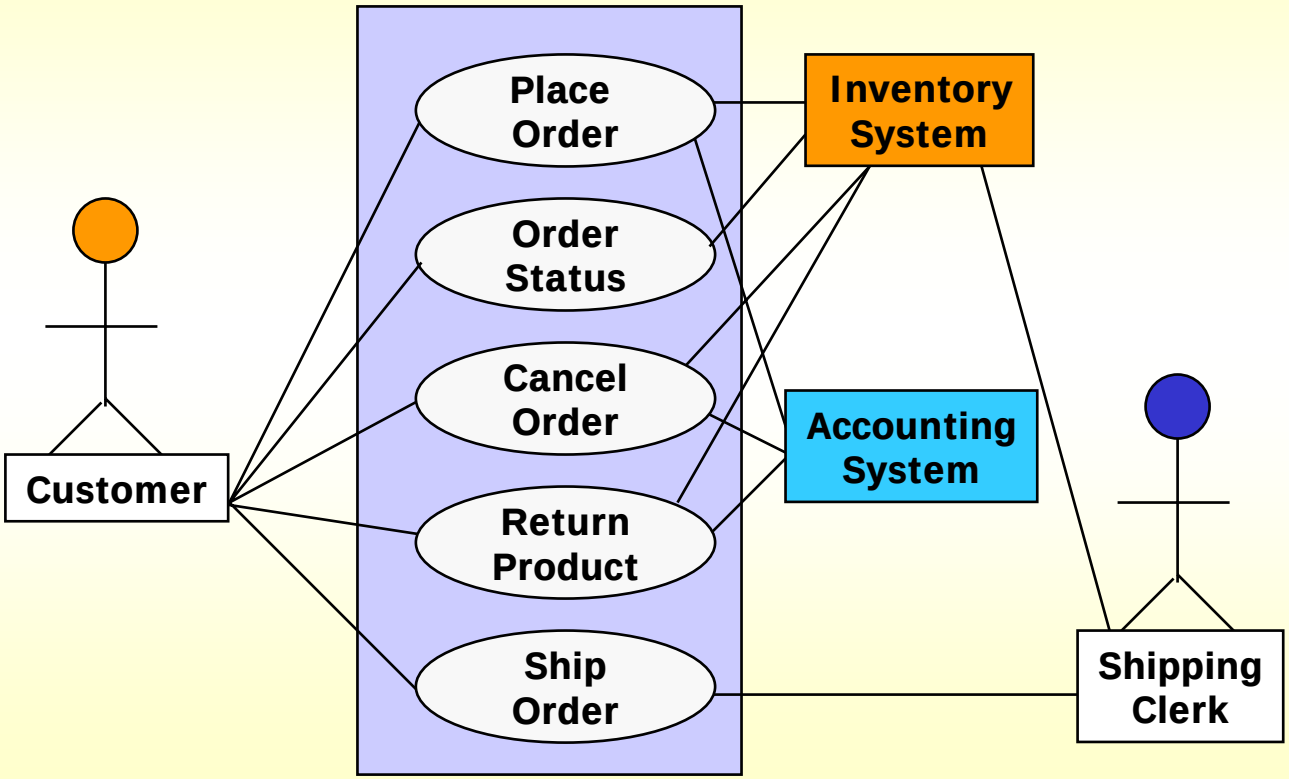
- **Reduce ambiguity by expressing requirements in a manner that leads to better understanding, a more coherent design, and more effective testing**
- **Some examples include:**
  - **Use Case Diagrams**
  - **Work Flow Diagrams**
  - **Flowcharts**
  - **Structured English**
  - **Truth Tables**
  - **Storytelling**

For more information on this topic, see my Mar 2006 newsletter www.swqual.com/newsletter/Subscribe.htm

- **Tools for expressing requirements in a way that leads to improved testability…**

Use Case Diagrams

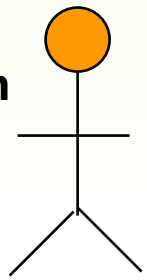
- **Use Cases...**



Schneider, G. and Winters, J., Applying Use Cases: A Practical Guide, Addison-Wesley, 1998.

Annotated Use Case Diagram...

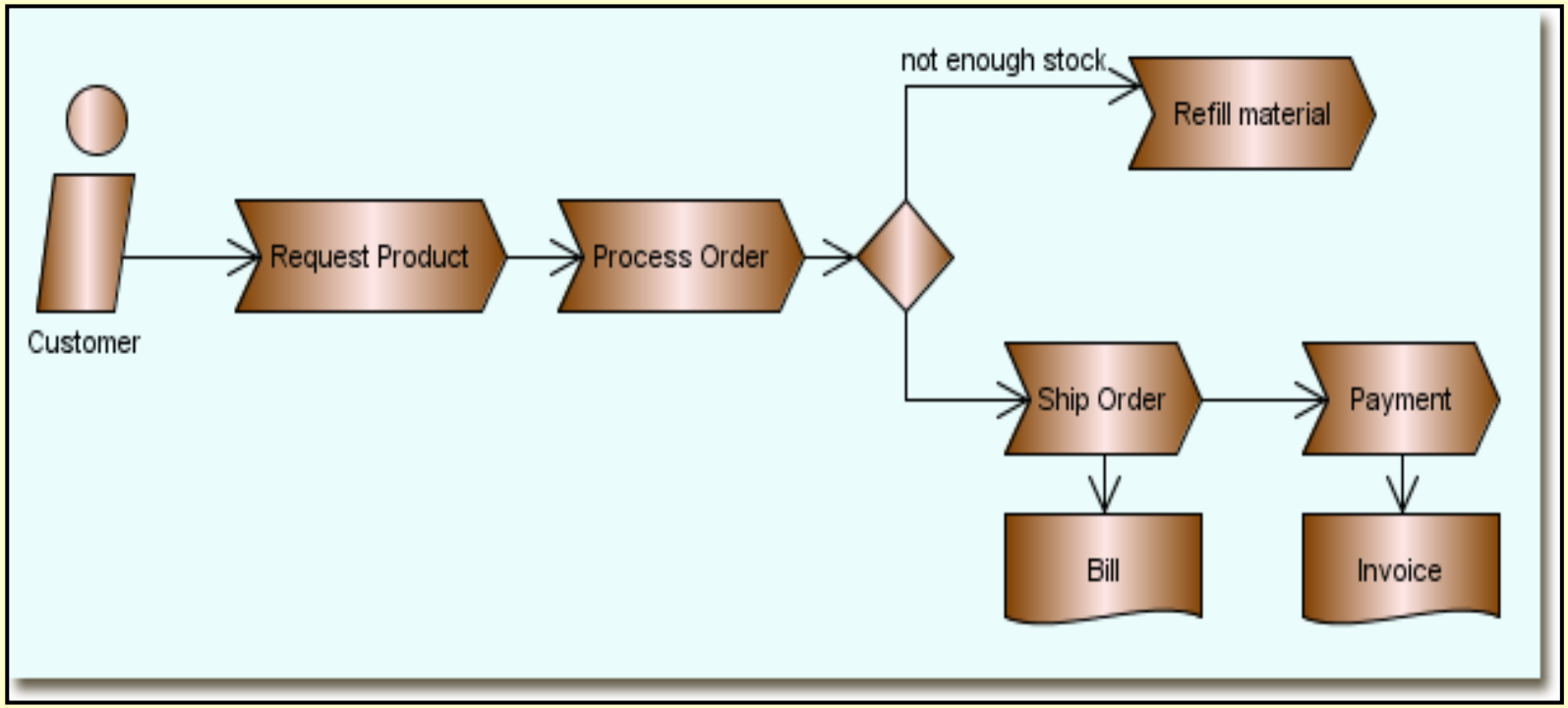
- **Place Order Process**
  - **Inputs and Outputs**
    - **Items ordered received as input from customer, along with bill to, ship to info, and customer payment details**
    - **Order confirmation, estimated ship date sent as output to customer**
    - **Order info sent as output to Inventory System**
    - **Customer billing info sent to Accounting System**
  - **When customer places an order, the following occurs:**
    - **Unique order number is generated**
    - **Pick list is generated representing the items ordered**
    - **Customer billing info entered into Accounting System**
    - **Order sent to Inventory System for fulfillment**
    - **Estimated ship date returned sent to customer**



Workflow Diagrams

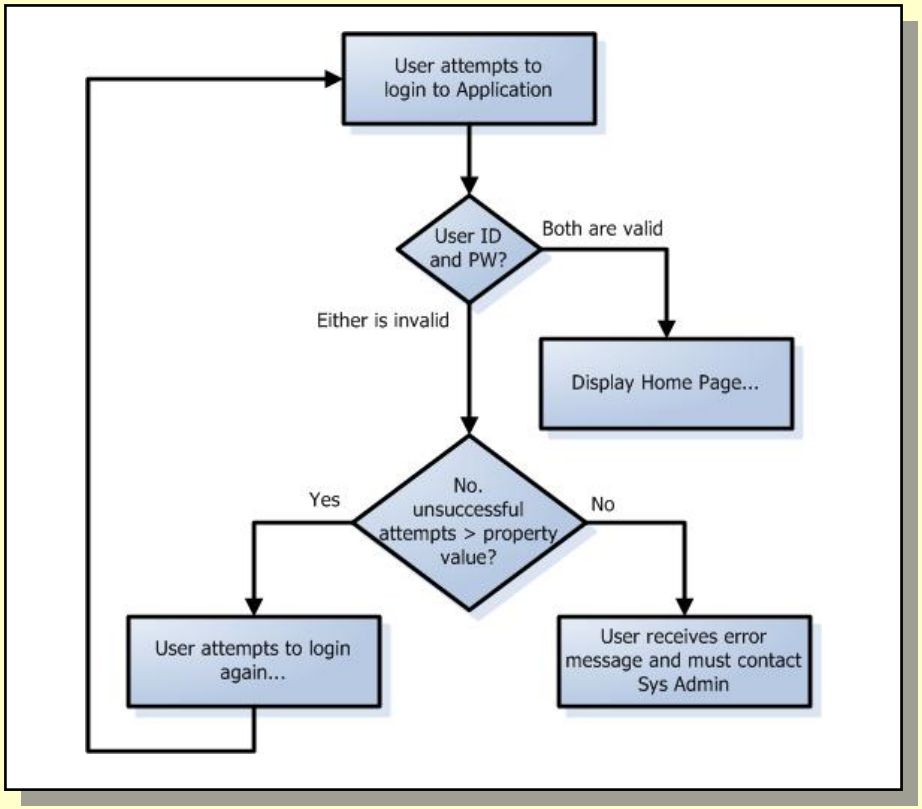
Express requirements in ways that facilitate testing

Workflow Diagrams	
When to use	Expressing user interface flow, user decisions points, and general workflow…
How to use	Highlight decisions, criteria, and response (messages, actions, etc.)



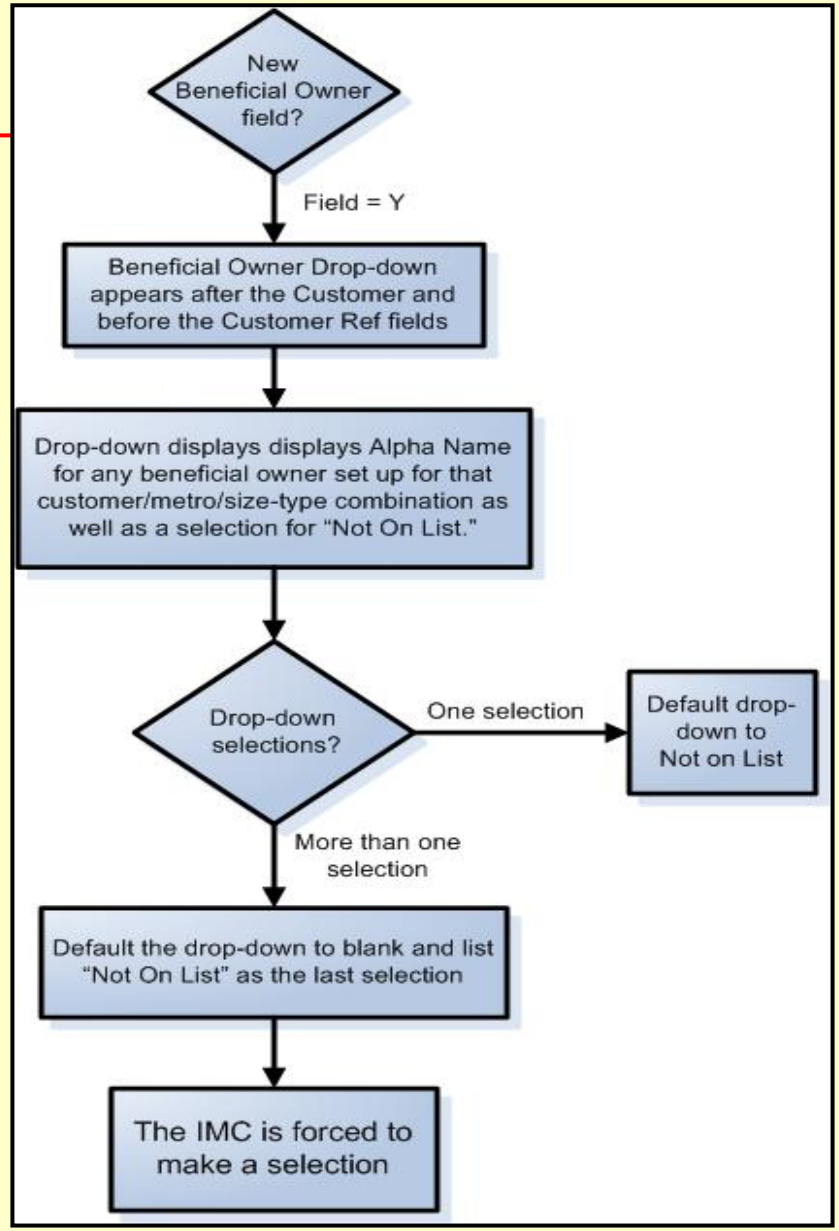
Flowcharts

Flowcharts	
When to use	Expressing user interface flow, user decisions points, and general workflow…
How to use	Highlight workflow, decision points, and response (messages, actions, etc.)



Flowcharts

- A Beneficial Owner drop-down should appear after the Customer and before the Customer Ref Number if the new Beneficial Owner field is “Y”. The drop-down should display the Alpha Name for any beneficial owner set up for that customer/metro/size-type combination as well as a selection for “Not On List.” If the drop-down only has one selection (i.e. “Not On List”), default the drop-down to this selection. If the drop-down has more than one selection, default the drop-down to blank and list “Not On List” as the last selection (the IMC will be forced to make a selection).
- **avg 24.7 words/sentence**



Structured English

Structured English	
When to use	Expressing complex logic, decision constructs, and iterations
How to use	Write requirement using keywords: **IF, THEN, ELSE, DO, DO WHILE, DO UNTIL, AND, OR, TRUE, FALSE**,…

Example	**(OP = Old Password NP = New Password)**
If the OP is correct, the NP and Confirm NP are the same, NP follows configuration rules, and has not been used during the prior two changes, confirm the change with a successfully changed password message.	**IF** Old PW correct AND New and Confirm PW same AND New PW follows configuration rules AND New PW not used prior two changes **THEN** confirm change with message… Password successfully changed

Truth Tables

Truth Tables	
When to use	Use when dealing with **discrete variables that are related**...
How to use	Create a **table** that lists all possible values for each variable.

Truth Tables

Truth Table Example… (OP = Old Password NP = New Password)

User enters new PW. Application confirms if password meets the following configuration rules.

1. If OP is correct, NP and Confirm NP are same, pass configuration edit, and has not been used during prior two changes, confirm change with a successfully changed password message. An “OK” button brings user to Home page. Error Message = Password successfully changed.

2. If OP is correct and NP and Confirm NP match but do not conform to configuration settings, a message describing error is displayed. OP, NP and Confirm NP are blank after user selects “OK” on error message. “OK” button brings user to Change Password screen. Error Message = Password you entered does not conform to format specified by your system administrator. Enter a valid password.

3. If OP is valid and NP and Confirm NP do not match, a message describing error is displayed. OP, NP and Confirm NP are blank after user selects “OK” on error message. “OK” button brings user to Change Password screen. Error Message = NP and Confirm NP entries do not match. Try again.

4. If OP is correct and NP and Confirm NP match and pass configuration but has been used prior by user during previous two changes a message is displayed. OP, NP and Confirm NP are blank after user selects “OK” on error message. “Ok” button brings user to Change Password screen. Error Message – Password has been used too recently. Try again.

5. If NP and Confirm NP match and pass configuration but OP is invalid, a message is displayed. OP, NP and Confirm NP are blank after user selects “OK” on error message. “Ok” button brings user to Change Password screen. Error Message – OP entered is invalid. Try again.

Truth Tables

OP Confirmed	NP and Confirm NP match	Password Rules followed	NP not used in last two changes	Password change successful?	Display Message
TRUE	TRUE	TRUE	TRUE	Yes	1
TRUE	TRUE	FALSE	DON'T CARE	No	2
TRUE	FALSE	DON'T CARE	DON'T CARE	No	3
TRUE	TRUE	TRUE	FALSE	No	4
FALSE	DON'T CARE	DON'T CARE	DON'T CARE	No	5

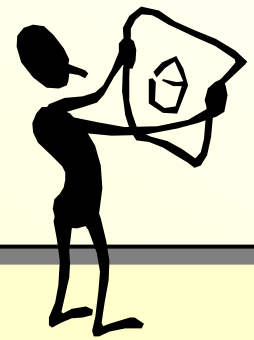
Messages:

1. "Password successfully changed"
2. "The password entered does not conform to the format specified by your sys admin. Enter a valid password."
3. "New Password and Confirm New Password entries do not match. Try again."
4. "The password you entered has been used recently. Try again."
5. "The Old Password you entered is invalid. Try again."

How many tests are required to test these requirements?

Other Techniques to Reduce Ambiguity

- Use pictures to illustrate work flow and functionality…
- Create a glossary and use terms consistently
- Every requirement clearly identified and numbered
  - "The software shall…"
- Avoid ambiguous words:
  - could, would, should, might, may, can, user-friendly, etc…
- Use a good document outline…
  - IEEE 830 is a good starting point…



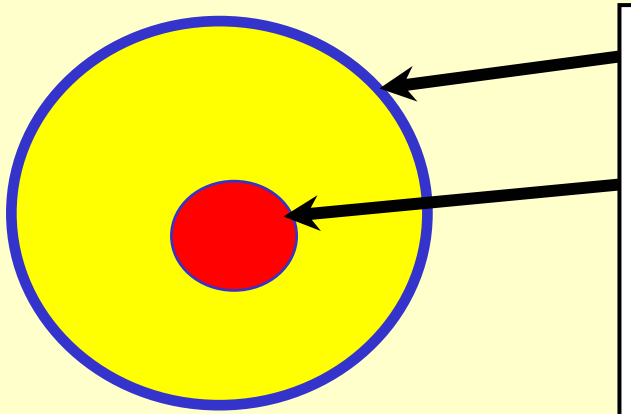
Storytelling as a Critical Testing Skill

- **Ability to tell a compelling story about how customers might use your software is critical**
- **Story telling can help testers do a better job of Acting Like a Customer…**
- **Soap Opera Testing…**

Buwalda, H., “Soap Opera Testing”, *Better Software*, February 2004.

Storytelling as a Testing Skill

Act Like a Customer Testing™



Software has lots of defects

Customers typically find a small percentage of the total

Focus testing efforts on finding those defects Customers are likely to find

Act Like a Customer Testing is a trademark of Software Quality Consulting, Inc.

An example of a story...

- **A customer rents a car for a three-day business trip. Midway through the rental, he extends the rental for another week. This, by the way, gives him enough rental points to reach Preferred status. Several days later, he calls to report that the car has been stolen. He insists that the Preferred benefit of on-site replacement applies, even though he was not a Preferred customer at the start of the rental. A new car is delivered to him. Two days after that, he calls to report that the "stolen" car has been found. It turns out he'd forgot where he'd parked it. He wants one of the cars picked up and the appropriate transaction closed. Oh, and one other thing - the way he discovered the missing car was by backing into it with its replacement, so they're both damaged.**
- **What test scenarios can you identify from this story?**

This story is from Brian Marick and appears in "Soap Opera Testing", *Better Software*, February 2004, by Hans Buwalda

Storytelling as a Testing Skill

- **Stories can help identify aspects of complex systems that need to be tested…**
- **Scenario tests have several characteristics:**
  - **Scenario tests are based on a story about what customers might do**
  - **Stories are motivating and credible**
  - **Stories often involve complex use of software, environment, or data**

For more information on this topic,
see my Oct 2005 newsletter
www.swqual.com/newsletter/Subscribe.htm

Kaner, C., "Scenario Testing", June 2003.

Scenario Testing

- **Cem Kaner identifies ways to create good scenarios:**
  - Write life histories for objects in the system.
  - List possible users, analyze their interests and objectives.
  - Consider disfavored users: how do they want to abuse your system?
  - List “system events.” How does the system handle them?
  - List “special events.” What accommodations does the system make for these?
  - List benefits and create end-to-end tasks to check them.
  - Interview users about famous challenges and failures of the old system.
  - Work alongside users to see how they work and what they do.
  - Read about what systems like this are supposed to do.
  - Study complaints about the predecessor to this system or its competitors.
  - Create a mock business. Treat it as real and process its data.
  - Try converting real-life data from a competing or predecessor application.

Cem Kaner, “Scenario Testing”, June 2003.

Summary

- **Design for Testability concept applies to software**
- **Design software to be more "testable" from the outset**
- **Focus on requirements…**
  - **People who write requirements often have little or no training**
  - **Use variety of techniques to improve testability**
- **Storytelling as a tester skill is critical for improving quality**

Additional Workshops from Software Quality Consulting

- **Writing and Reviewing Software Requirements**
- **Building Realistic Project Schedules from Software Requirements**
- **Software Verification & Validation**
- **Accurate Estimating and Scheduling Using Yellow Sticky Method**
- **Predictable Software Development**
- **Peer Reviews and Inspections**
- **Improving the Effectiveness of Testing**
- **Risk Management for Embedded Software Development**

- **For more information, please visit**
  - **www.swqual.com/training/on_site.html**

- **Subscribe to SQC e-newsletter…**
  - **www.swqual.com/newsletter/Subscribe.htm**

Thank you…

…for taking time to attend this seminar.

If you have any questions, please call or e-mail…

Software Quality Consulting Inc.

Steven R. Rakitin
President

- Consulting
- Training
- Auditing

Phone: 508.529.4282
Fax: 508.529.7799

www.swqual.com
info@swqual.com