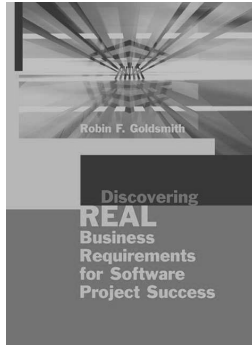
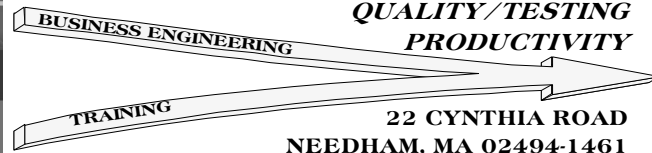


Use Cases for Requirements and Testing-- Facts and Follies



Robin F. Goldsmith, JD
GO PRO MANAGEMENT, INC.
SYSTEM ACQUISITION & DEVELOPMENT
QUALITY/TESTING
PRODUCTIVITY



22 CYNTHIA ROAD
NEEDHAM, MA 02494-1461
INFO@GOPROMANAGEMENT.COM
WWW.GOPROMANAGEMENT.COM
(781) 444-5753

Use Case— *How an Actor Interacts with the System*

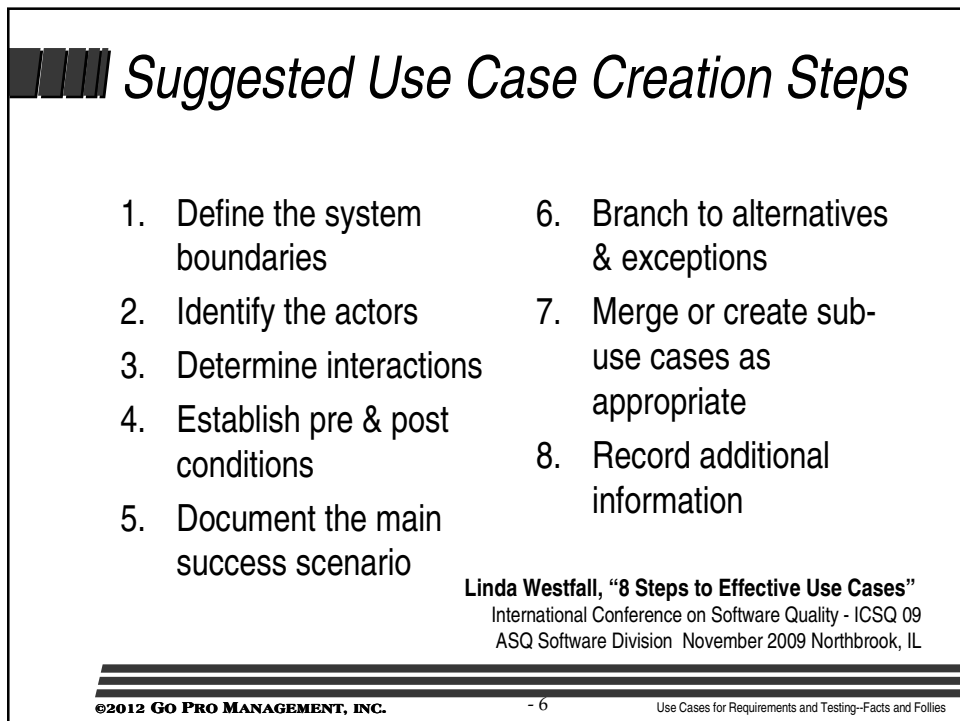
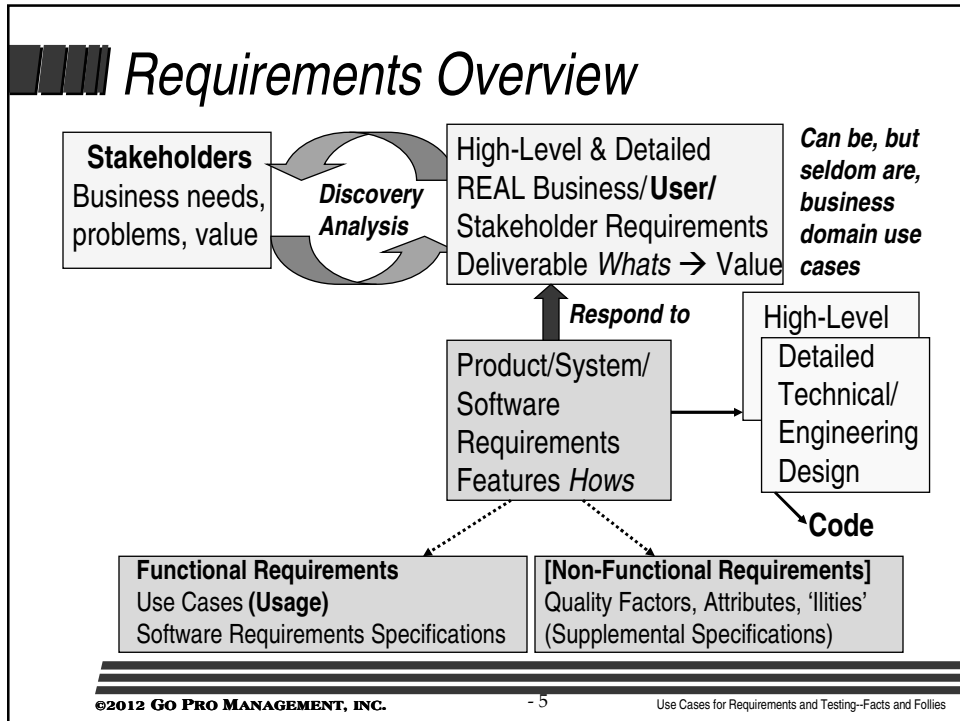
*For such a seemingly simple and
widely-used/advocated technique,
surprisingly little common
understanding/usage
and
lots of misunderstanding and
mis-usage*

Objectives

- Distinguish types of requirements use cases do and do not show effectively
- Describe common formats for representing use cases and issues they raise
- Explain conventional use case testing and its seldom-recognized limitations

Some Authors Say...

Use cases are the **user's** requirements
and **user** requirements are use cases

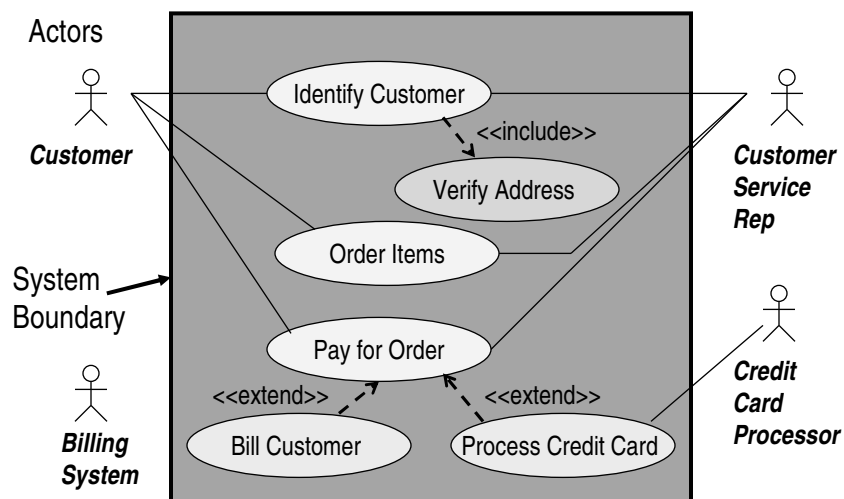


Actors Are Outside System Boundary-- Users, Other Systems, Equipment

- Who uses the system
- Who gets information from the system
- Who provides information to the system
- Who needs to be informed about certain occurrences within the system
- Who supports and maintains the system

Primary Actor's Goal provides a benefit, usually names the use case

Use Case Diagram




Some Points About Use Case Diagrams

- Diagrams often get disproportionately large amount of attention in use case books/courses
 - Main use case usage and value is the step-by-step textual elaboration of the use cases (inside the ovals)
 - Novices often draw ovals for each use case step
- Confusion because use case diagrams do not show elements that other common system diagrams show
 - Databases, files, outputs
 - Decisions and flow sequences
- Same diagram and event can represent multiple use cases, each from a different actor's perspective

Use Case Example 1 of 2: *Customer Orders and Eats Meal*

- **Customer asks to be seated**
- **Waiter shows customer to a table and gives him a menu**
- **Customer gives food order to the waiter**
- **Waiter writes down order and gives it to kitchen**
- **Waiter serves food when it is ready in the kitchen**
- **Customer eats meal**
- **When customer finishes, waiter gives customer the bill**
- **Customer pays the bill and leaves a tip**
- **Waiter takes the payment to the cashier and keeps tip**




Use Case Example 2 of 2:

Waiter Takes Order and Serves Meal

- Waiter shows customer to a table and gives him a menu
- Customer gives food order to the waiter
- Waiter writes down order and gives it to kitchen
- Waiter serves food when it is ready in the kitchen
- Customer eats meal
- When customer finishes, waiter gives customer the bill
- Customer pays the bill and leaves a tip
- Waiter takes the payment to the cashier and keeps tip

©2012 GO PRO MANAGEMENT, INC. - 11 Use Cases for Requirements and Testing-Facts and Follies



Typical Use Case Template

- Use case name
- Summary description
- Primary Actor
- Secondary Actors
- Trigger event that initiates use case
- Pre-conditions that must be present to start use case
- Post-conditions that must be present before use case can end successfully
- Priority, frequency of use, risk
- Special requirements
- Scenario steps of primary Actor's interaction with the system from primary Actor's perspective
 - Main success flow ("Happy Path")
 - Alternate success flows
 - Exception flows (don't end with success)

©2012 GO PRO MANAGEMENT, INC. - 12 Use Cases for Requirements and Testing-Facts and Follies

User Stories—Agile Requirements

As an order processing clerk,
I need to enter orders
So the company can sell things

As an order processing clerk,
I need to identify the customer
So the company can sell them things

As an order processing clerk,
I need to find or add the customer
So the company can sell them things

***Which is the requirement? Other issues?
Where do use cases come in?***

Find/Add Customer *Is this a use case?* 1 of 2

To place an order, a customer must be identified in the system. Once in the system, each customer has a unique Customer ID. When the Customer ID is entered, the specific customer's record should be retrieved. Some customers have registered a credit card which they prefer to use. If the Customer ID is not known, the customer's record can be located by entering the credit card number.

If the customer's record cannot be found by exact match of Customer ID or credit card number, the customer's record can be searched for alphabetically by the customer's name. The name should be entered in last, first, middle name sequence. The name to be searched for (search argument) can be full or partial. The program will display a list of customers starting with the customer whose name is equal to or next greater than whatever has been entered as a search argument. There is no wild card logic. One can scroll backward and forward alphabetically through the list of names and addresses and select the record that is the customer's.

Find/Add Customer *Is this a use case?* 2 of 2

If the customer's record cannot be located, the customer may be added to the database and assigned a Customer ID consisting of the first three characters of the customer's last name and the next sequential three-digit tiebreaker, starting with 000 for the first customer with those three characters. When adding a customer, the customer's name, address, home/business phone numbers, and (optionally) a credit card number must be entered. Phone numbers should be 10 digits. The address should have a five- or nine-digit postal Zip code and a valid two-character state abbreviation.

Once the customer's record has been retrieved/created and confirmed, go to the item entry routine.

Issues with this format?

Identify the "Happy Path"

One-column alternating "round trip" format

1. **User** enters the customer's Customer ID
2. **System** retrieves and displays the customer's record
3. **User** confirms customer is correct
4. **System** requests entry of items to be bought

Issues with this format?

One-Column Formats from Actor's View

1. System displays menu choices. Actor selects Find by Customer ID.

or

1. Actor selects Find by Customer ID from the displayed menu choices.

Issues with this format?

Other issues?

Identify Alternate Success Paths

1. User enters the customer's Customer ID

1.1.1 System cannot find customer's record by ID

1.1.2 User enters customer's credit card number

1.1.3. Return to Happy Path step 2 (System retrieves and displays the customer's record)

1.1.2.1 System cannot find customer's record by credit card no.

1.1.2.2 User enters customer name to search for

1.1.2.3 System display list of customers

1.1.2.4 User selects customer from list

1.1.2.5 Return to Happy Path step 2 (System retrieves and displays the customer's record)

1.1.2.4.1 User does not select customer from list

1.1.2.4.2 User adds customer to database

1.1.2.4.3 Return to Happy Path step 2 (System retrieves and displays the customer's record)

Issues with this format?

Identify Exception Failure Paths

- 1.1.2.4.1 User does not select customer from list
- 1.1.2.4.1.F1 User quits looking

Issues with this format?

Use Case

Strengths

- Step-by-step format is easy to understand
- Developers can develop systems that work the way users expect
- Readily translates into tests—execute each use case path

Weaknesses

- Format is not content, usage view may mask REAL requirements
- Can be *as is* rather than *should be* model
- Often includes only things expected to be automated
- Often misses paths other than Happy Path
- May actually create complexity

Use Cases Are Not Well-Suited for:

- Business rules
- Algorithms, formulas, and calculations
- Quality factors (non-functional requirements), including usability, performance, and security
- Configuration issues
- States, timing, and concurrency
- Data base contents and structure

Omitted or kept in Supplementary Requirements Specifications can interfere with understandability

Other Format Variations

- One-column full-view (as opposed to separate Happy Path, Alternative Success Paths, and Exception Failure Paths)
- Two-column
- Graphical

And, volume of text varies widely

Functionality Matrix

Technical View

User View (Use Cases)	Create	Retrieve	Update	Delete	Commun.	Interface	Logic	ChgState	PerfLevel	Constraint
Find by Cust. ID (exact match)		X			X				X	
Customer is not found *					X	X		X		
Cust. is found and confirmed*					X	X		X		
Cust. is found, not confirmed*					X	X		X		
Find by credit card no. (exact)		X			X				X	
Search by cust. Name (partial)		X			X		X		X	
Select cust. from search list					X	X		X		
Quit the search					X	X		X		
Add new customer to database	X				X	X	X	X	X	X
Quit					X	X		X		

Functionality Matrix: Identifies & Can Test Requirements Use Cases Miss

User view, step-by-step

Technical view, what's happening "under the covers"

Create, add, insert a new record in a file

Retrieve, read, query an existing record in a file

Update, modify, change an existing record in a file

Delete, remove, scratch an existing record in a file

Communicate with an external device

Interface to another piece of software

Perform **logic** or calculations

Change state

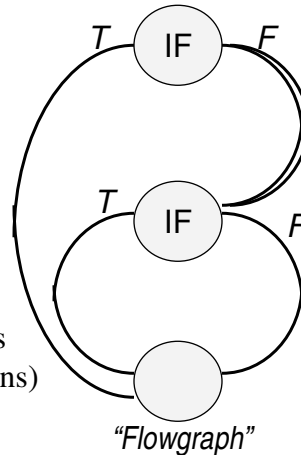
Meet a specified **performance** level

Comply with an external **constraint**

Each user/technical view intersection should be addressed in a Test Design Specification (can split or consolidate)

Mapping Logic Path Not Just for Program Code

- Flowgraph: **Node**
 - ❑ Module's Entry, Exit
 - ❑ Where logic branches
 - ❑ Junctions (logic returns)
- Flowgraph: **Edge**
 - ❑ all logic between nodes



Two-Column Use Cases

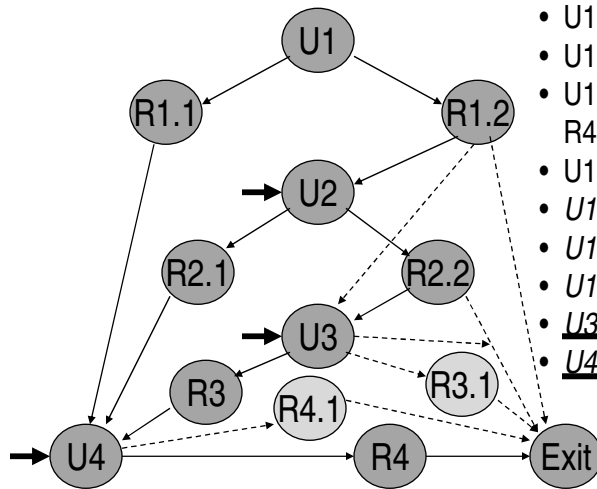
Look Even More Like Test Cases

Left column is Actor/User action. Right column is System Response. Multiple possible responses are indicated, and flow to next step can be described. Numbering can be sequential or differentiated by Action/Response as shown.

Conventional wisdom=one test per use case path/scenario

U1. Enter customer number	R1.1. Customer is found (U4)
	R1.2. Customer is not found (U2)
U2. Enter customer name	R2.1. Select customer from list (U4)
	R2.2. Customer is not in list (U3)
U3. Add customer	R3. Customer is added (U3)
U4. Enter order	R4. Order is entered (Exit)

Flowgraph of Use Case



- U1-R1.1-U4-R4-Exit
- U1-R1.2-U2-R2.1-U4-R4-Exit
- U1-R1.2-U2-R2.2-U3-R3-U4-R4-Exit
- U1-R1.2-U3-R3-U4-R4-Exit
- U1-R1.2-Exit
- U1-R1.2-U2-R2.2-Exit
- U1-R1.2-U2-R2.2-U3-Exit
- U3-R3.1-Exit
- U4-R4.1-Exit

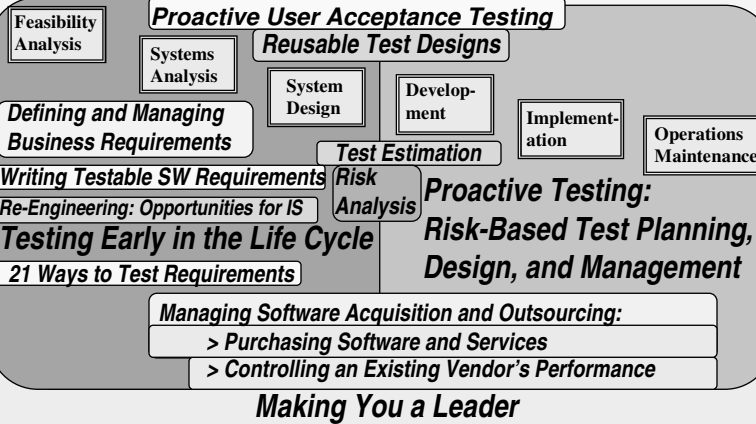
Conditions to test:
U1-R1.2
R3-R3.1

Objectives

- Distinguish types of requirements use cases do and do not show effectively
- Describe common formats for representing use cases and issues they raise
- Explain conventional use case testing and its seldom-recognized limitations

**Systems QA Software Quality Effectiveness Maturity Model
Credibly Managing Projects and Processes with Metrics**

Software, Test Process Measurement & Improvement



Robin F. Goldsmith, JD

robin@gpromanagement.com www.gpromanagement.com

- President of Go Pro Management, Inc. consultancy since 1982, working directly with and training professionals in business engineering, requirements analysis, software acquisition, project management, quality and testing.
- Partner with ProvelT.net in REAL ROI™ and ROI Value Modeling™.
- Previously a developer, systems programmer/DBA/QA, and project leader with the City of Cleveland, leading financial institutions, and a "Big 4" consulting firm.
- Degrees: Kenyon College, A.B.; Pennsylvania State University, M.S. in Psychology; Suffolk University, J.D.; Boston University, LL.M. in Tax Law.
- Published author and frequent speaker at leading professional conferences.
- Formerly International Vice President of the Association for Systems Management and Executive Editor of the *Journal of Systems Management*.
- Founding Chairman of the New England Center for Organizational Effectiveness.
- Member of the Boston SPIN and SEPG'95 Planning and Program Committees.
- Attendee Networking Coordinator for STAR, Better Software, and Test Automation Conferences.
- Chair of record-setting attendance BOSCON 2000 and 2001, ASQ Boston Section's Annual Quality Conferences.
- Member IEEE Std. 829 for Software Test Documentation Standard Revision Committee.
- Member IEEE P1805 working group to develop a standard for Requirements Capture Language (RCL).
- Member IEEE P730 standard for Software Quality Assurance Revision Committee.
- International Institute of Business Analysis (IIBA) Business Analysis Body of Knowledge (BABOK) subject expert.
- TechTarget SearchSoftwareQuality.com requirements and testing expert.
- Admitted to the Massachusetts Bar and licensed to practice law in Massachusetts.
- Author of book: **Discovering REAL Business Requirements for Software Project Success**