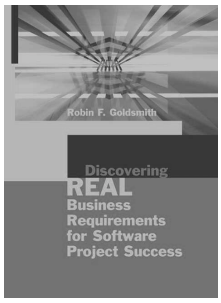
Prevent Showstopper Overruns with Risk-Based Proactive Testing™

Robin F. Goldsmith, JD

GO PRO MANAGEMENT, INC.
SYSTEM ACQUISITION & DEVELOPMENT
QUALITY/TESTING
PRODUCTIVITY
BUSINESS ENGINEERING
TRAINING

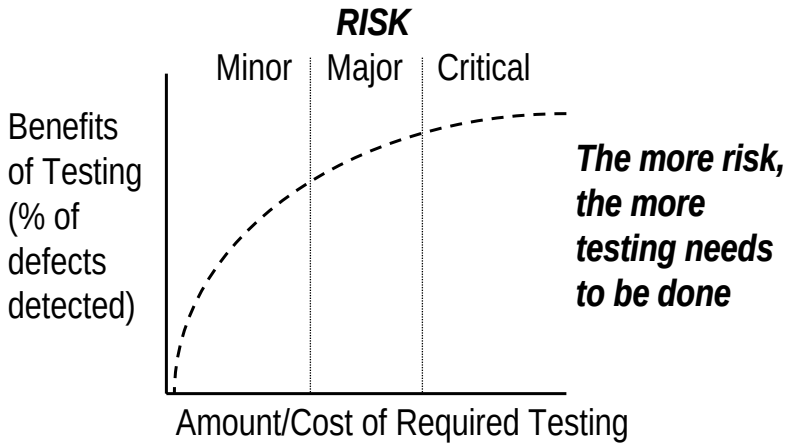
22 CYNTHIA ROAD
NEEDHAM, MA 02494-1412
INFO@GOPROMANAGEMENT.COM
WWW.GOPROMANAGEMENT.COM
(781) 444-5753 *VOICE/FAX*



Objectives

- Describe risk-based testing fundamentals.
- Identify limitations of traditional reactive testing approaches.
- Show how Proactive Testing™ continually refocuses on testing higher risks more and earlier.
- Demonstrate methods for identifying ordinarily-overlooked showstoppers and reducing overruns.

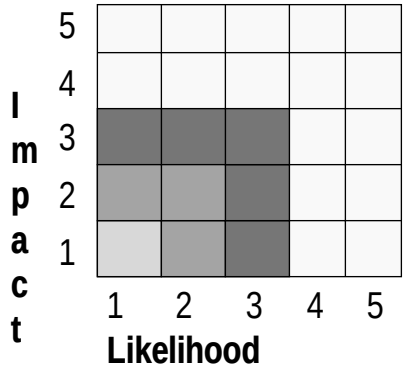
Testing Is Our Main Method of Reducing System Risks



Elements of Risk & Exposure

- Impact
  - Severity of damage
  - Number of people affected
  - Consequences, e.g., lose business
  - Cost, time, effort, ability to fix
  - Availability of workarounds
- Likelihood
  - Size, complexity
  - New technology
    - » In the world
    - » Here
  - Prior error history
    - » Product
    - » Producer
  - Other error indicators
    - » Lack of skill, motivation
    - » Inadequate methods

Typical Risk Exposure Scorings

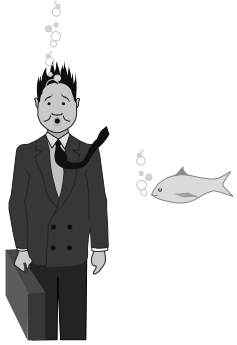


Lo	Med	Hi	Exposure
1	2	3	= 1 - 9

What would determine the scale used?
How about
1 3 5 = 1 - 25 ?
or
1 5 10 = 1 - 100 ?

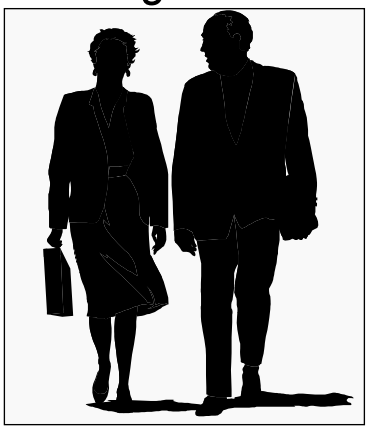
Risk Categorization Is Inconsistent: Focus on Effects--Business Impacts

- If this system/project fails, what is the impact on the business?
- At the least, we're out the time and money we've spent on it
- Opportunity cost--could have used the time and money for something more productive



Any other direct or indirect forms of injury?

Others Focus on Source: Management and Technical Risks



- Failure to plan, organize, direct, control, fund, staff adequately (seldom considered, e.g., Internet Time--after all, who is defining the risks?)
- Technical risk is that the technical complexity, newness, and instability cannot be made to work

Effects vs. Causes

Effect = Exposure (Impact x Likelihood)
Why we care, the bad outcomes
Business, Management, Technical

Cause(s) = What we can address to mitigate (reduce/eliminate) the effects

Testing is our main mitigation technique
Relevant to more than we often realize
But not relevant to everything

Conventional [Reactive] Approach to "Risk-Based" Testing

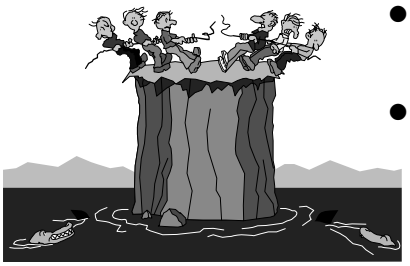
- For each product **feature** (or key function the product performs—also referred to as "requirements risk"—often each system menu choice is a feature) or **component** (technical pieces implementing it) identify
  - Vulnerabilities for failure, things likely to fail
  - Threats that trigger failure, situations where likely to fail
  - Victims impacted by failure, the damage if it fails
- Test the highest-risk things **most**
  - Things done most, things other things depend on
  - Things that are likely to fail—big, complex, new, changed
  - Things with biggest impact if they fail

Most Common Risk Method: Challenge of Determining High, Medium, or Low

- Most organizations rely on subjective judgment, usually by knowledgeable people, possibly with checklists, but provide no guidance for reliably differentiating high, medium, low
- Key to reliability and repeatability is defining objective characteristics indicating each level, e.g.
  - Low = interfaces with 0 other systems
  - Med = interfaces with 1-2 other systems
  - High = interfaces with 3 or more other systems
- Challenge remains that this one-at-a-time approach can still fail to distinguish among choices; prioritization is most effective when done by ranking among all choices

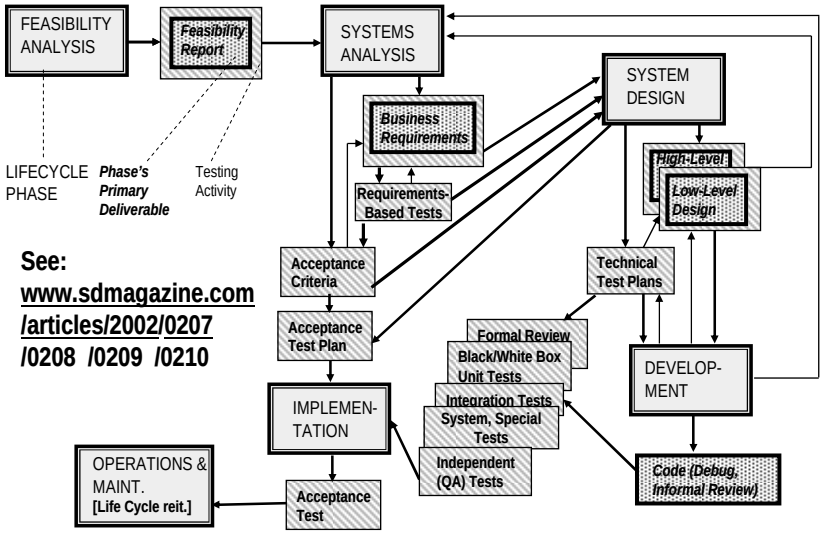
Biggest risk of either method: overlooking an important choice

Conventional Testing Is Typically Reactive and Often Poorly Received



- Tests are defined late in the development process
- Tests are based mainly on what developers have done, which often is not clearly or fully communicated
- Importance of doing even risk-based tests may not be meaningful to users, developers, or managers

Proactive Testing™ Life Cycle



Key Proactive Testing™ Concepts

Find WIIFMs

- Intertwine testing with each development deliverable
- Plan before acting at any point/level, independent paths
  - Acceptance testing first; users shouldn't do technical tests
  - Prepare test plans/designs during Design, promote reuse
  - Prioritize by level, from full choices, to avoid wasted effort
- Let testing drive development
  - Feedback test plans/designs to guide correct coding
  - Guide builds strategy to build and **test higher risks more *and earlier*** to reduce impact of and effort to fix errors

Test Plans

- Project plans for the Testing (sub) Project
- Objectives, strategies, guidelines, standards
- Identify testing tasks, resources, effort, duration →schedule, budget Based on:
  - The set of tests to demonstrate (detailed test plans in Master Test Plan, test design specifications in Detailed Test Plans)
  - Test support, environment, hardware, software, facilities
  - Ancillary and administrative activities

Test Design

- Identifies a set (list) of test cases (specifications) that taken together demonstrate the feature, function, or capability works
- Can be reusable or application-specific

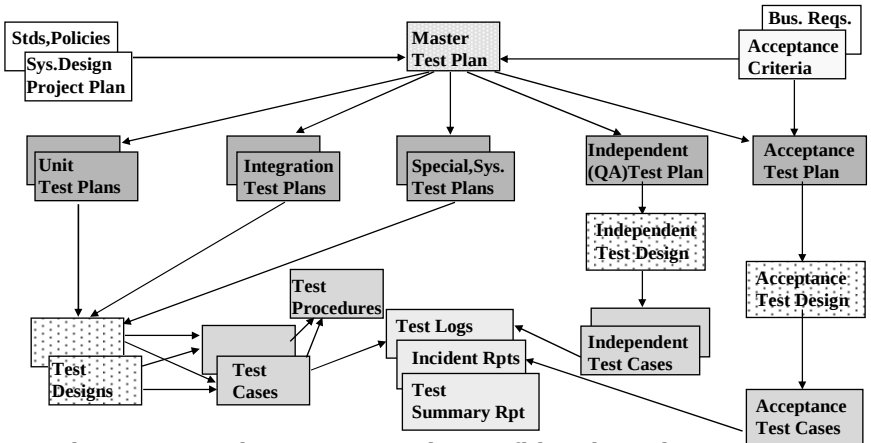
Test Case

- Input/condition and expected result
- What is executed
- Specification (in natural language) and data values (which actually are input and expected)
- Can be reusable, especially specification

Test Procedure

- Step-by-step instructions for executing test cases
- Includes set-up, establishing pre-conditions
- Can get to keystroke level
- Often embeds input and expected result data values, which increases maintenance difficulty

Testware--Test (Plan) Documentation per ANSI/IEEE Std. 829-1998



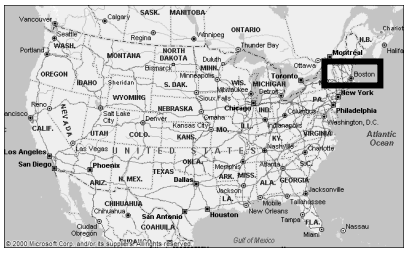
What must we demonstrate to be confident it works?

Testing Structure's Advantages

- ✓Organize and manage large set of test cases
- ✓Facilitate thorough test data recreation
- ✓Show the choices for meaningful prioritization
  - ✓Test the biggest risks earlier and more thoroughly
  - ✓Focus first on larger issues, drill down later to detail
  - ✓Leverage test skills; share ideas and detail work
- ✓Enable identification of reusable Test Design Specifications and Test Case Specifications

❶ Master Test Plan

- States positively how system will be tested
  - Defines detailed test plans which taken together demonstrate that full system works
  - Sets test priorities and strategy to address risk
  - Establishes default entry and exit criteria

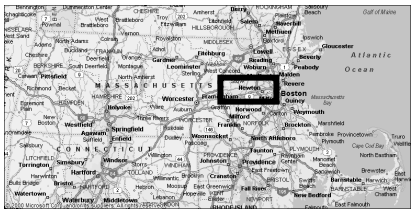


United States

- Project/system-level, becomes part of project plan
- Management agreement between customer and technical executives, understandable to both

❷ *Detailed Test Plan*

- States positively how piece will be tested
  - Defines set of features, functions, and capabilities (can be a Test Design Specification for each) which taken together demonstrate that it works
  - Identifies exceptions to Master Test Plan defaults
  - Sequences, data sources

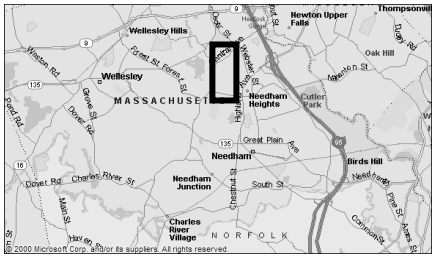


Massachusetts

- One per unit, integration, special, system, independent QA, and user acceptance test
- Technical document
- Basis for detailed workplan and estimates

❸ *Test Design Specification*

- States positively how feature etc. will be tested
  - Defines conditions that must be demonstrated to be confident it works
  - Identifies set of Test Cases that taken together demonstrate conditions
  - May define procedures
  - Can be formal or informal

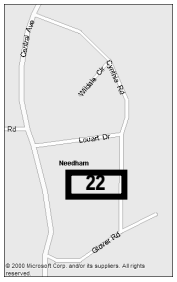


Needham

- One per feature, function, and capability—can be consolidated for economy and practicality
- Valuable intermediate level, often overlooked
- Potential for reuse

❹ *Test Case Specification*

- States positively how test will be executed
  - Identifies input/situation (environment), expected result, and procedure
  - Includes Specification (in words) and Data Values
  - Common formats are script (especially when automated) and matrix



Cynthia Road

- Lowest-level, can be simple (one input, one result) or complex (series of inputs-results)
- Actually executed
- Potential for reuse

Example: Proactive Testing™ Master Test Planning Risk Analysis for

Web Quote Personal Auto Insurance

(as described in Software Testing class by attendees)

- For use by independent agents
- 1. Ascertain who client is, kind of cars, drivers, driving records, location, marital, sex, age, VIN, usage, driver training, grades, types of coverage, deductibles.
- 2. If passes initial scrutiny, find out about liens on the vehicle, additional insured, billing plan, payment type.
- Calculate and provide premium quote.
- Print application form to be signed, returned with payment.

Exercise: System-Level Risks

What Can Go Wrong	What Must Go Right
That would prevent the system in operation from functioning effectively	**For the system in operation to function effectively**
Can't print application form	*Find/add applicant in database*

Major Cause of Overruns: Late, Unplanned Redesign and Rework

- From showstoppers that testing did not anticipate, at the worst possible time
- Often up to 80 percent of total project effort--all unanticipated
- Throwing out and replacing significant amounts of completed work

The Stomach-Ache Metric

WIIFM
Proactive?

Proactive vs. Reactive

- What percent of these tests would we ordinarily have overlooked?
- Which would be showstoppers?
- Could any users, developers, or managers not understand these showstoppers' significance?
- Ordinarily, when would most of them be tested?
- Would there be any WIIFMs from letting testing drive development so they are tested earlier?

Let Testing Drive Development

- ✓ Define Unit, Integration, and Special tests to test the high risks early, before other work has been done which will have to be redone if the risk comes true
- ✓ Development builds just enough to allow early testing of these high risks
- ✓ Once these significant redesign/rework risks are reduced, build and test the remaining pieces

Risks to the System in Operation that Testing Should Address

[*identified by author*]
1 car, 1 driver; more cars than drivers; more drivers than cars
Age groups; accidents and tickets
Order Motor Vehicle Record Rates Agents' use Flow to in-house system
[*added by others*] Data validation and editing Lose connections, session continuity
Hardware capacity and performance Compatibility—browser, O/S, platform Viruses

[from author & others]	*New customers,*
Printing quotes and app forms Underwriting rules	*Existing custs, mult D/B records*
Send in signed printed application;	*Multiple requests for quotes*
check accompanies if paying by check	*Reports on types of quotes,*
Track applications, tie back to ones not sent in	*quotes vs. purchases*
Minimum set of data Calculations	*multiple quotes for same person*
Security Firewalls, anti virus	*Compare web applications to*
Order credit scores, receive back for calculations	*phone, mail applications*
Validating payment with credit card, not approved	*Purging records*
3rd party system down	*Interactions with other systems*

These represent a combination of Detailed (unit, integration, or special) Test Plans and Test Design Specifications—Also Identify Design Issues

Functionality Matrix—Identify Test Designs

Technical View

User View (Use Cases)	Create	Retrieve	Update	Delete	Commun.	Interface	Logic	ChgState	PerfLevel	Constraint
Find applicant by driver's license		X			X				X	
Found and confirmed					X	X		X		
Found but not confirmed					X	X		X		
Not found					X	X		X		
Search for applicant by name		X			X		X		X	
Search for applicant by address		X			X		X		X	
Select applicant from search list *					X	X		X		
Quit the search *					X	X		X		
Add applicant to database	X				X	X	X	X	X	X
Quit					X	X		X		

Test Design: What Must We Demonstrate to Be Confident "Find an applicant by driver's license" Works?

Assumptions: License number is fixed-length number

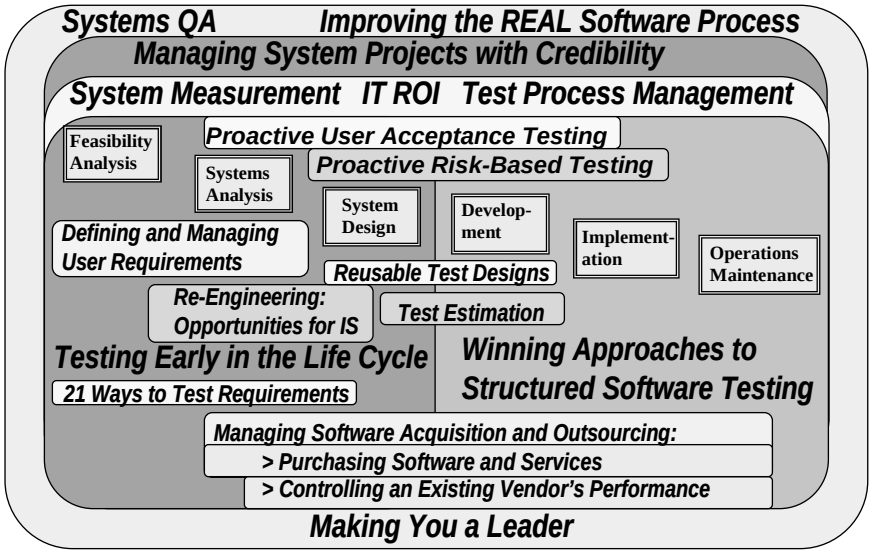
Valid
Actual number for my state
Actual number for a different state

Invalid
Invalid length, too long, too short
Number of proper length for my state, not a license
Number of proper length for a different state, not a license
Valid number for my state but indicated for a different state where not a license
No state, invalid state
Alphabetic, special characters

Summary

- Testing is our primary means of reducing risk, which is impact times likelihood of problems.
- Traditional reactive testing evaluates risks of features and components and tests the higher risks **more** but often comes too late and tests only what is designed.
- Proactive Testing™ identifies ordinarily-overlooked risks, including those that cause showstoppers, and continually refocuses so higher risks are tested more and earlier, when they are easiest to fix.

Go Pro Management, Inc. Seminars--Relation to Life Cycle



Robin F. Goldsmith, JD

robin@gopromanagement.com (781) 444-5753

www.gopromanagement.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.
- Previously a developer, systems programmer/DBA/QA, and project leader with the City of Cleveland, leading financial institutions, and a "Big 5" 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.
- Chair of BOSCON 2000 and 2001, ASQ Boston Section's Annual Quality Conferences.
- Member ASQ Software Division Methods Committee.
- Admitted to the Massachusetts Bar and licensed to practice law in Massachusetts.
- Author of book: ***Discovering REAL Business Requirements for Software Project Success***