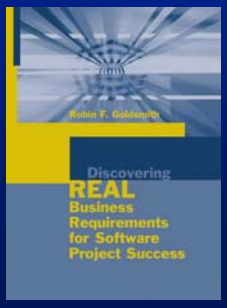
Testing Whether Requirements Are Right

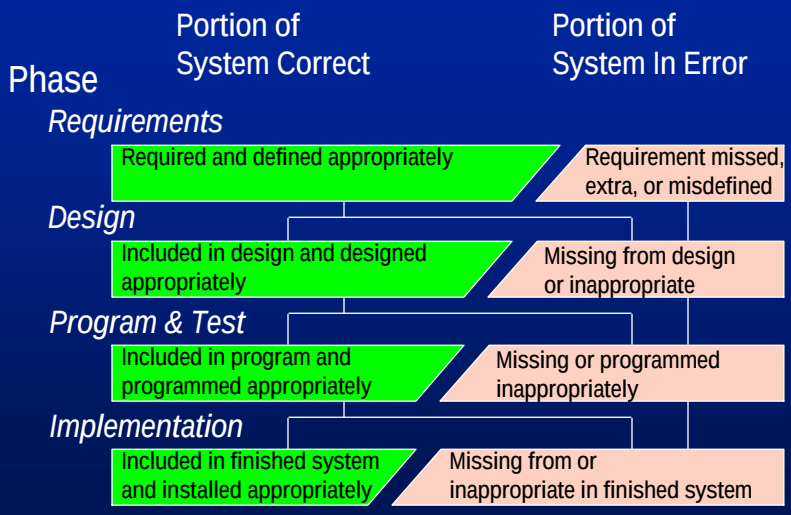


Robin F. Goldsmith, JD
GO PRO MANAGEMENT, INC.
SYSTEM ACQUISITION & DEVELOPMENT
QUALITY/TESTING
PRODUCTIVITY
BUSINESS ENGINEERING
TRAINING
22 CYNTHIA ROAD
NEEDHAM, MA 02494-1412
INFO@GOPROMANAGEMENT.COM
WWW.GOPROMANAGEMENT.COM
(781) 444-5753 *FAX* (781) 444-4913

Objectives

- Identify how creep and resulting budget/schedule overruns are largely due to (often unrecognized) inadequate definition of requirements
- Explain why requirements are seldom tested effectively for accuracy and completeness
- Introduce more than 21 methods for testing that requirements are right:
  - Form (including testability and clarity)
  - Finding overlooked requirements
  - Detecting incorrect requirements

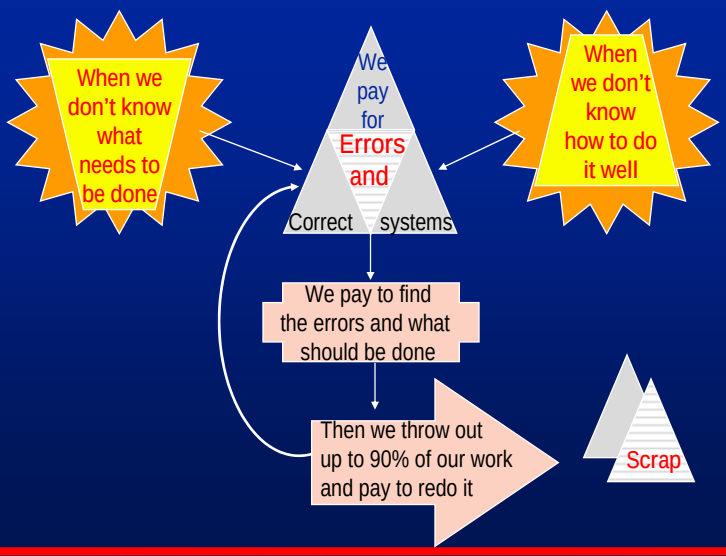
Error Sources by Phase



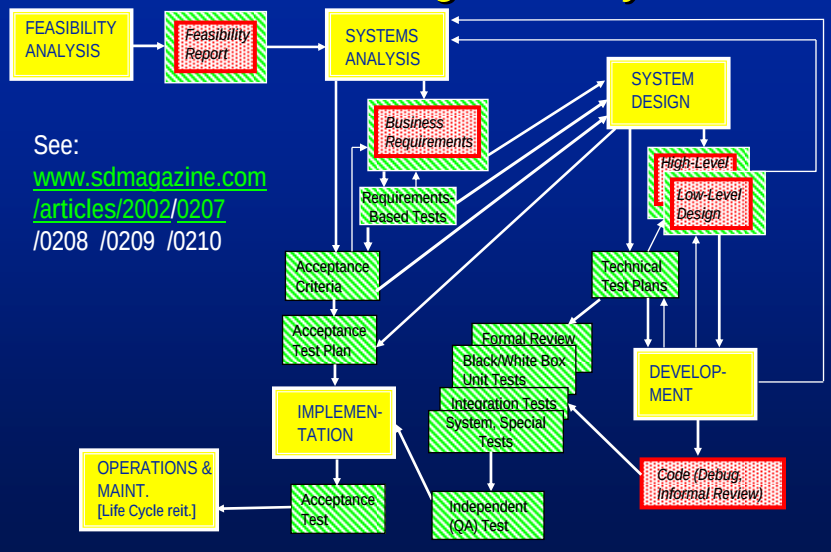
Quality Assurance Economics

- Maintenance is 66-90% of system cost
- Maintenance is mainly completion/ correction of development
- 2/3 of finished system errors are requirements and design errors
- Fixing a requirements error will cost
  - 10X+ during programming
  - 75X-1000X+ after installation (maintenance)

What Development Dollars Buy



Proactive Testing™ Life Cycle



See:
www.sdmagazine.com
/articles/2002/0207
/0208 /0209 /0210

Keys to Effective Testing

- Define correctness independently of actual results
- You must know what the "right answer" is
- Follow independent guidelines to be more thorough
- Systematically compare actual to expected results

Test Input	Actual Results	Expected Results
Cust. #123	John P. Jones	Jones, John P.
New Cust's name,address	Redisplays screen with fields cleared	"Added"
10 Widgets	$14.99	$14.99 $ .75 tax

Why Up-Front Testing Usually Is So Weak

- Unaware it can be done or how to do it
- No definition of "right answer"
- Person who defined it "tests" it
- Use limited or weak review methods
- Confuse design with requirements
- Don't manage overall software process
  - cost/consequences not measured/matched
  - activity and deadline driven

The "Regular Way"

- User review
- Management review
- Supervisory review
- Peer review
- QA Group review

Did you know:
What to do?
How to do it?
How to tell if you'd done it well?
How confident were you that you had done it well?

Much weaker than recognized passive review on which most organizations unwittingly over-rely

Strengthening the Review Process

- Use formal technical review
  - objective is to find all potential problems
  - preparation, participation, accountability
  - roles--moderator, recorder, presenter
  - written summary and detail issues reports
- Predefine topics & specifics to examine
  - Organization's prescribed format (e.g., IEEE)
  - Presumed functions and common functions

✪ *Making Sure They Are Requirements*

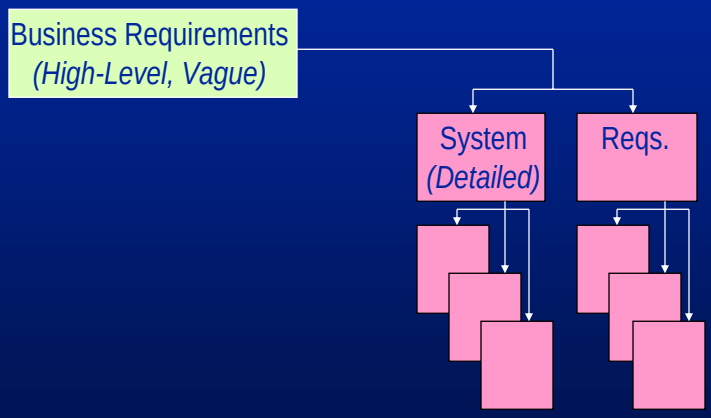
- In user/customer/business language
- What must be delivered to provide value
  - not how (design/technology) or desires, *except*
    - » required technical environment it must fit
    - » operational style preferences it should meet
    - » purposes, objectives, and expected benefits to clarify and place requirements in context
  - Qualitative characteristics

Everyone says they do this, but most miss a critical point …

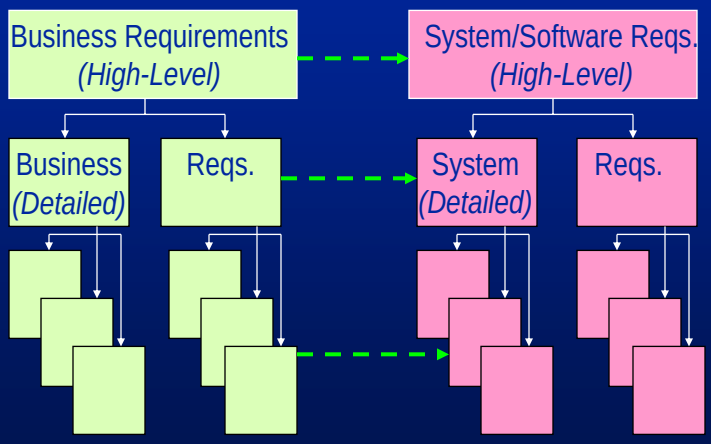
➲ Two Types of Requirements:

Business/User	System/Software
• User view & language, conceptual • Serves business objectives • *What* business results must be delivered to solve a business need (problem, opportunity, or challenge) and provide value when delivered Many possible ways to accomplish	• Technical/product operational view, language • Product *design* of one of the possible ways, *How*, to deliver the business results • *What* external functions each piece of the product must accomplish for it to work as designed (Functional Specifications)

Even Requirements "Experts" Think the Difference is Detail



When Business/User Requirements Are Detailed First, Creep Is Reduced

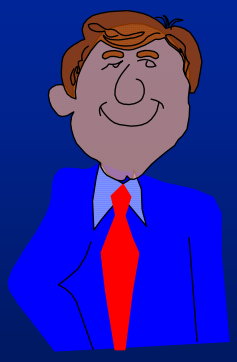


Other Tests of Requirements Form

- Deliverable, attainable in the world (not a budget issue)
- Testable (write test cases)
- Reviewable
- Reasonably understandable in the business community
- Clear and structurally complete
  - Stated as positive (can't prove absence of a negative)
  - Terms: known meaning, identifiable, consistent
  - Assumptions documented
  - Stand on own without internal contradictions, logic flaws, or vagueness/ambiguities
- Consistent with (suitable) objectives
- Identifies major functions, limits
- Alternative consequences defined
- [In prescribed format, e.g., IEEE Std. 830-1998 for Software Requirements Specifications, use cases]
- [Magic words--must, shall, will (no TBD)]
- Hierarchical itemized business deliverable whats, traceable

These, or Just a Subset (e.g., Testability or Use Cases), Are All Most "Experts" Address

- Often resisted as nitpicking
- Still can miss important form issues, even with checklists and procedural rigor
- Contrary to presumptions, unlikely to spot *content* issues
  - Overlooked requirements
  - Incorrect requirements

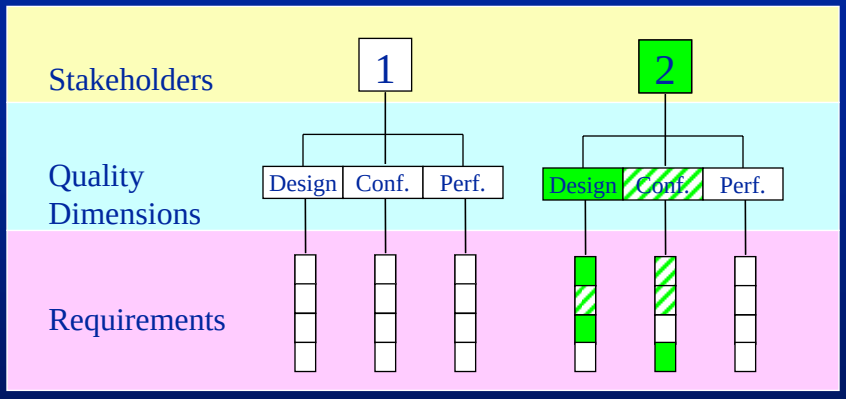


Tests of form can be performed by people with minimal subject area knowledge, e.g., many of US

Greater Knowledge and Added Methods Are Needed to Find Missed Requirements

- Focusing on what *has been defined* actually makes it harder to realize what's been overlooked
- Typically find *groups* of requirements of which we were unaware
- Quickest way to gain support for requirements review

How Requirements Get Missed



- Overlooked
- Misdefined
- Defined Correctly

✪ What are business and temporal event triggers?

When There's No Definition of "Right," Create a "Strawman"

- Reasonable guess of what requirements might be--no assurance they are right
- Match to what has been defined
  - Reasonably present confirms it has been addressed, assume appropriately
  - Not reasonably present indicates need to examine more fully
  - Sample: One problem may signal more
- Forcing concrete examples challenges assumptions that we've addressed them

Greatest Knowledge and Structure Are Needed to Detect Wrong Requirements

- Like a strawman in identifying the likely "right answers" to compare to
- Structured methods focus knowledge on important and error-prone areas
- Unlike strawman, informed judgments do evaluate correctness

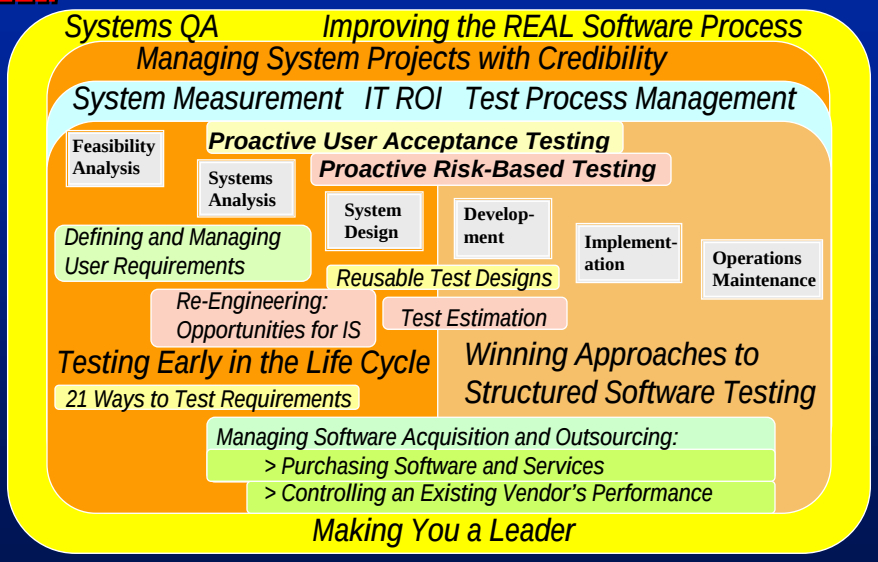
Some Methods for Testing Accuracy

- Application feature Engineered Deliverable Quality™
- Cross-application
  - Guidelines & conventions
  - Engineering standards
- Conflicts and tradeoffs
- Working out implications
- Prototyping & simulation
- Walkthroughs
- Joint Application Development (JAD)
- Defining acceptance criteria
- Mini-definitions and "canned" requirements
- Expert review
- Two independent reviews

Summary

- Requirements creep mainly because system/software requirements (functional specification) and use cases for the expected *product* don't meet the REAL, business/user requirements--*whats* that provide value when delivered.
- Requirements are hard to test because there is no prior definition of "right" to compare to
- There are 21+ ways to test that requirements are right
  - Tests of form, including testability, clarity, and prescribed formats, are all that most people know about, easiest, and weakest
  - Finding overlooked requirements takes more knowledge
  - Identifying requirements errors takes most knowledge/structure

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*