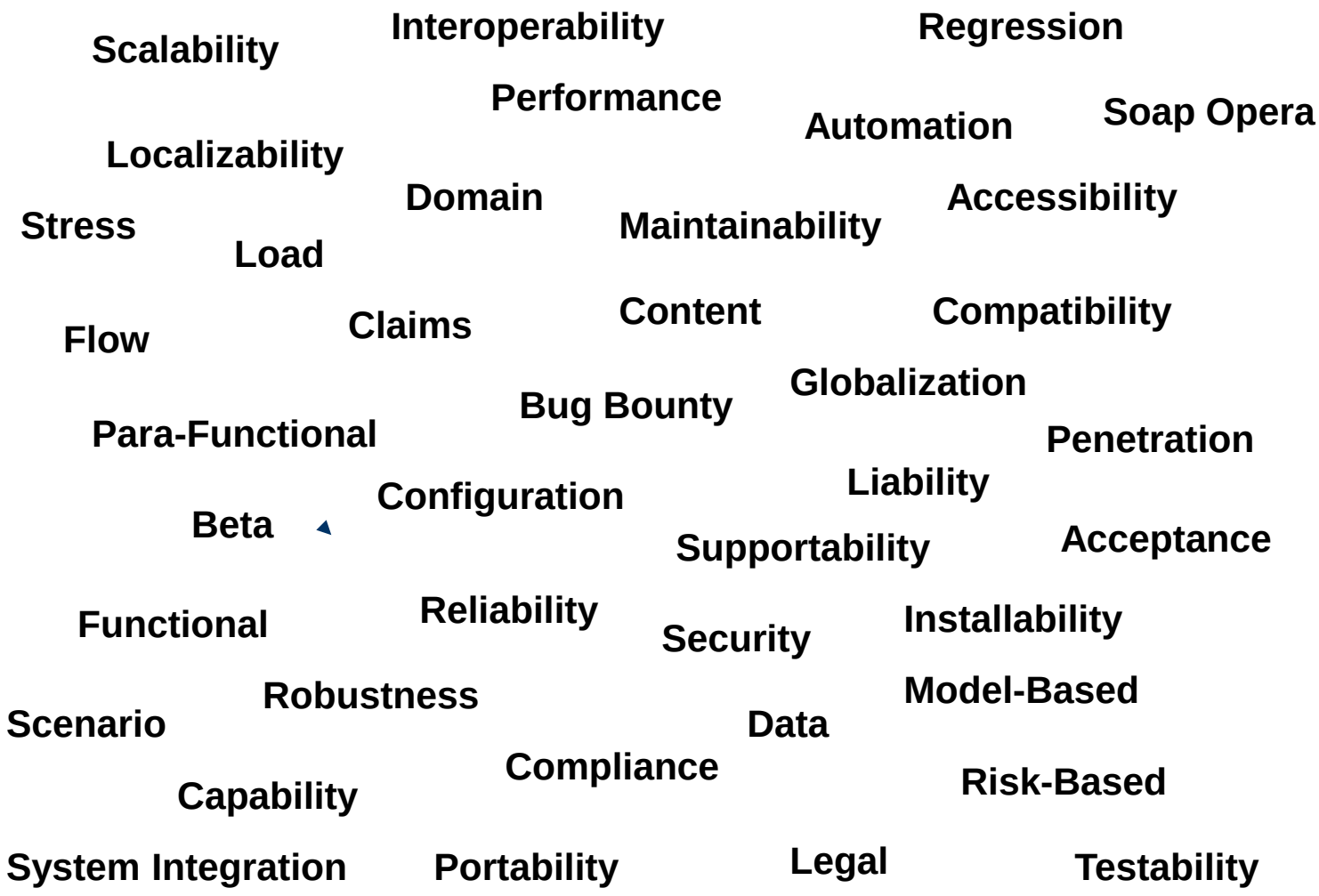
Do We Really Need All This Testing?

Jon Bach
Senior Consultant
Quardev Laboratories -- Seattle
jonb@quardev.com

Rose City SPIN
April 10, 2008


Scalability
Interoperability
Regression
Performance
Automation
Soap Opera
Localizability
Domain
Accessibility
Stress
Maintainability
Load
Content
Compatibility
Flow
Claims
Globalization
Bug Bounty
Para-Functional
Penetration
Liability
Configuration
Beta
Supportability
Acceptance
Reliability
Functional
Installability
Security
Model-Based
Robustness
Scenario
Data
Compliance
Risk-Based
Capability
System Integration
Portability
Legal
Testability

Logic

- Testing is expensive

- Testers can’t find all the bugs

- Developers can’t fix all the bugs

- Customers are waiting for your software

“Good enough” concept

- A way of thinking about quality in terms of utility and economy
- “To get a result that is good enough, although not necessarily the best” - *Herbert Simon, 1978 Nobel Laureate economist*
- Opposite of process formalist techniques (i.e. TQM and Six Sigma) which “strive for near perfection” http://mu.motorola.com/sixsigma.shtml
- Criteria that helps stakeholders decide utility of risk (benefits vs. problems)

The problem

- There is bad software in the world, blamed on processes that aren't "certified"
- Clinical definition is limiting: *"Stopping a search for alternatives by choosing the first option that exceeds one's aspiration level."*
- It's the opposite of "optimal", so it's wrongly considered a synonym for substandard, or mediocre – reinforced by a social definition that connotes sloppiness:

  "Good enough for government work"

The Four Noble Truths

1) **Sufficient benefits**

2) **No critical problems**

3) **The benefits outweigh the problems**

4) **All things being equal, further design and testing is more harmful than helpful**

The answer must be "Yes" to all four in order to ship

Criteria #1 – Sufficient Benefits

The software helps users solve problems or meet needs by:

Increasing their productivity

Providing entertainment

Helping them compete in the marketplace

Establishing / enhancing reputation

Meeting standards

Criteria #2 – No critical problems

The software should not exhibit anything that is deemed by stakeholders to be “critical”, which could include:

Embarrassing typos

Legal issues

Failures or faults

Poor feature set

Localization: insults

Criteria #3 – Benefits outweigh problems

Risk analysis – what's the likelihood of something bad happening, and how bad would it be?

Customers don't notice (or forgive) the faults

The bugs are all low severity

The feature set is competitive and appropriate

It's awesome (fun, cool, timely, useful)

If sued, we can make enough back to afford it

Criteria #4 – Testing is more harmful then helpful

If we keep testing or designing…

…we'll miss our ship window

…we'll break our budget

…we'll lose market share

…we'll go out of business

…staff will quit / revolt / burn out

When to use it

- If prone to changing requirements, behind schedule
- If your project is understaffed, with ill-defined test processes
- If time to market is crucial
- If people may not care if the quality is “perfect”
- If it can help reach an acceptable level of RISK
- Replaces “blind perfectionism with vigilant moderation”
- Danger: we may cut too many corners
- “We ship when we believe the risks to be acceptably low”
- Helps know the difference between important and unimportant, necessary and unnecessary
- It isn’t the number of bugs that matters, it’s the effect of those bugs – aka Triage
- Alternative to counting LOC -- “no more than 3.4 defects per million opportunities in any process, product, or service” --

The Price is Right?

Some software…

Street Maps USA
Frog Frenzy Safari
Exotic Classic Cars
God Bless America
Professional Resume Plus
1000 Best Solitaire Games
Tarot / Lotto Magic

What would you pay for all of this?

$??.??

And the actual retail price of your showcase IS…

$10.06

Contextual considerations

- Good enough for who?
  - Good enough for what?
    - Good enough for when?

Good enough for who?

- Its intended users
- People in the marketplace
- Beta participants
- CEO / Product Manager
- Trade show attendees
- Trainees
- Interview Candidates
- Testers

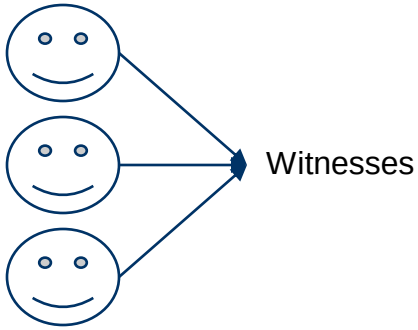
Good enough for what?

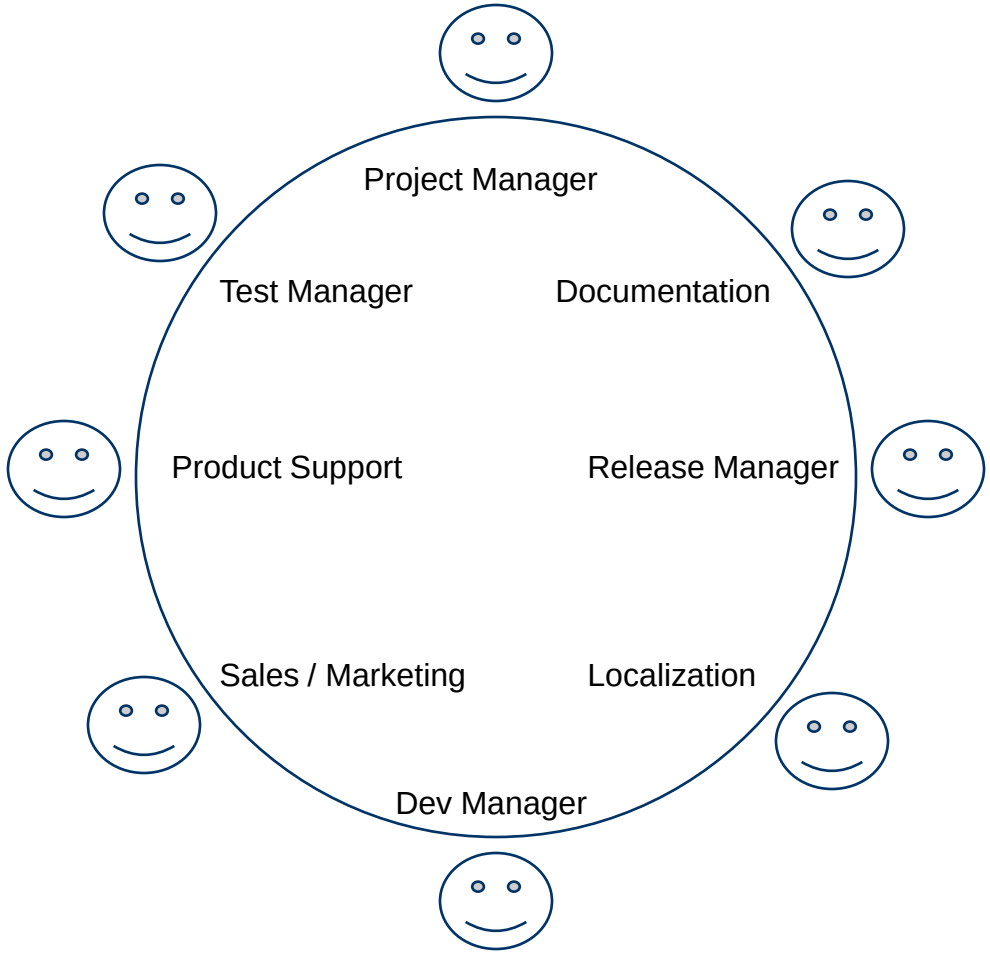
- Its intended purpose
- To compete in the marketplace
- Beta
- Proof-of-concept
- Trade show demo
- Class exercise
- Interview test
- Testing

Good enough for when?

- RTM
- Today's Bug Bash
- Beta
- Until we get a bad review
- E3 / Comdex / SASQAG
- Until we patch it
- Until we get a competitor
- Until another Y2K

The context factory: “Triage”





The triage meeting is a key tool of “good enough” software.

If you have one, you’re doing it.

Testing G.E. – Bugs to triage

- On the splash screen, the title of the software is misspelled
- Blue screen in driver.vxd when opening CD during app setup
- Resumes written in Arial font print in Courier
- Vegas-style game does not load or launch
- 20 Jefferson St. in Newburyport, MA is missing
- ScreenSaver GIFs look pixilated in 640 X 480
- 2000 Lotto numbers picked for me did not win anything

Measuring “good”-ness

Which ISP is the Best?						
(based on ranking of 1-10)						
		Criteria				
		Cost	Speed	Dial-up sites		
Options	MSN	6	5	8		
	AOL	7	5	10		
	No-charge	10	5	3		
	Speakeasy	4	5	3		
	Earthlink	5	5	6		
Weights		1	2	3		
						Ranking
Results	MSN	6	10	24		**6.67**
	AOL	7	10	30		**7.83**
	No-charge	10	10	9		**4.83**
	Speakeasy	4	10	9		**3.83**
	Earthlink	5	10	18		**5.50**

Helping stakeholders

A few of the ways testers help stakeholders make “good enough” (economic) decisions:

- Feature comparison with competing products
- Performance in different configurations
- Assigning severity to a bug
- Talking to PSS
- Simulating different users
- Compatibility with past products
- Beta programs
- Hosting Playtest sessions
- Our past experience
- Our own “gut feelings”

Good Enough gone wrong: part 1

How software ships with too little quality:

- **Complacency**
- **Denial**
- **Irrational Exuberance**
- **Inadequate analysis of risk**
- **Pressure to ship (economic, cultural, emotional, political)**
- **Misunderstanding of testing**
- **Monopoly**
- **Test did not raise the right questions**

Good Enough gone wrong: part 2

How software ships with too much quality:

- **Complacency**
- **Denial**
- **Irrational Exuberance**
- **Inadequate analysis of risk**
- **Pressure to ship (economic, cultural, emotional, political)**
- **Misunderstanding of testing**
- **Monopoly**
- **Test did not raise the right questions**

Key Ideas

- In a market-driven economy, you can't think about fixing a bug without also thinking about the cost to fix it.

- If the answer to "Can we fix this?" is "Yes", the next question should be: "should we fix this?"

- Ensures the right quality at the right price -- not too little, not too much -- considering the contexts (good enough for who, what , and when)

Was this talk good enough?

(Context: for this audience, for the purpose of enlightening other professionals, at this time)

- ❑ **Sufficient benefits**
- ❑ **No critical problems**
- ❑ **Benefits outweigh problems**
- ❑ **Cost of improvement is too high**

Questions may be an indicator that my talk does not have good enough quality

Sources / More info

James Bach
"The Challenge of Good Enough Software"
http://www.satisfice.com/articles/gooden2.pdf

BJ Rollison
SASQAG April 2003 presentation
http://www.sasqag.org/pastmeetings/rollison_quality.ppt

SixSigma.org
http://www.six-sigma.org/why.html

Motorola
http://mu.motorola.com/sixsigma.shtml

Michael Byron
"Satisficing and Optimality"
http://mbyron.philosophy.kent.edu/pubs/satisficing.html