

**TOO MUCH AUTOMATION  
OR NOT ENOUGH?**

**WHEN TO AUTOMATE  
TESTING**

Keith Stobie

[Keith.Stobie@microsoft.com](mailto:Keith.Stobie@microsoft.com)

# BRITTLE UI OR ROBUST MODEL

Badly automated UI tests

Test team says 2 weeks after 1 hour change

*Too much automation?*

Manually hacking a system

Attempting a few illegal values to break security

*Not enough automation? Fuzz testing*

Well automated model tests

Test team takes 15 minutes after 2 day change

# PRIMARY DRIVERS

## Cost to

1. Create - Depends on methodology
2. Run - Automated usually less expensive
3. Maintain - Depends on Design / Technology

# RETURN ON INVESTMENT

$$\begin{aligned} \text{ROI}_{\text{automation}}(\text{in time } t) &= \Delta B_a / \Delta C_a \\ &= \frac{\Delta(\text{Benefits from automation over manual})}{\Delta(\text{Costs of automation over manual})} \end{aligned}$$

$\Delta B_a$ : the incremental benefits from automated over manual testing.

$$\begin{aligned} \Delta B_a(\text{in time } t) &= \Sigma(\text{improvement in fixed costs of automated testing} \\ &\quad \text{times } (t/\text{Useful Life}) ) \\ &+ \Sigma(\text{variable costs of running manual tests } n_2 \text{ times during time } t) \\ &- \Sigma(\text{variable costs of running automated tests } n_1 \text{ times during time } t) \end{aligned}$$

$\Delta C_a$ : the incremental costs of automated over manual testing.

$$\begin{aligned} \Delta C_a(\text{in time } t) &= \Sigma(\text{increased fixed costs of automated testing} \\ &\quad \text{times } (t/\text{Useful Life}) ) \\ &+ \Sigma(\text{variable costs of creating automated tests}) \\ &- \Sigma(\text{variable costs of creating manual tests}) \\ &+ \Sigma(\text{variable costs of maintaining automated tests}) \text{ times } (n_1/N) \end{aligned}$$

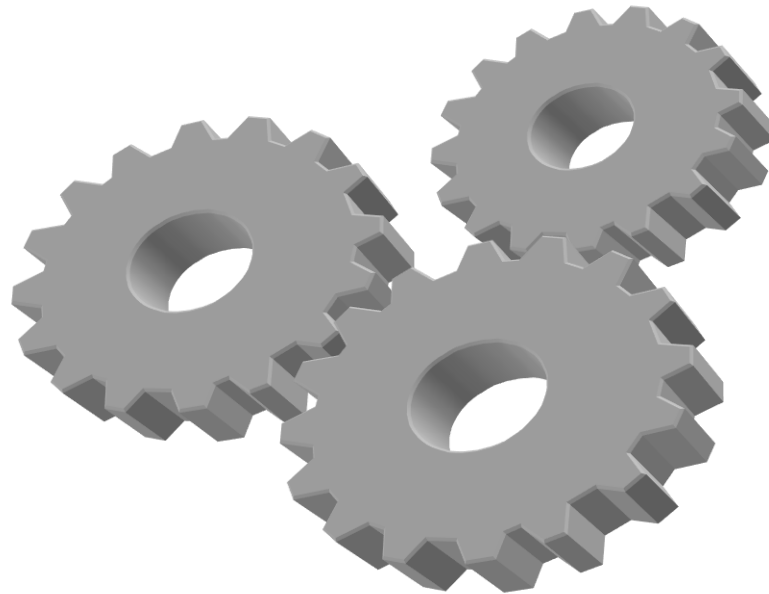
$n_1$ : Number of automated only test executions

$n_2$ : Number of manual test executions

$N$ : Average number of runs for automated tests before maintenance is needed

From Hoffman99 Cost Benefits Analysis of Test Automation

# MANUAL VS AUTOMATED



- ⦿ Ignoring other differences, like metrics.

# WHY REPEAT?

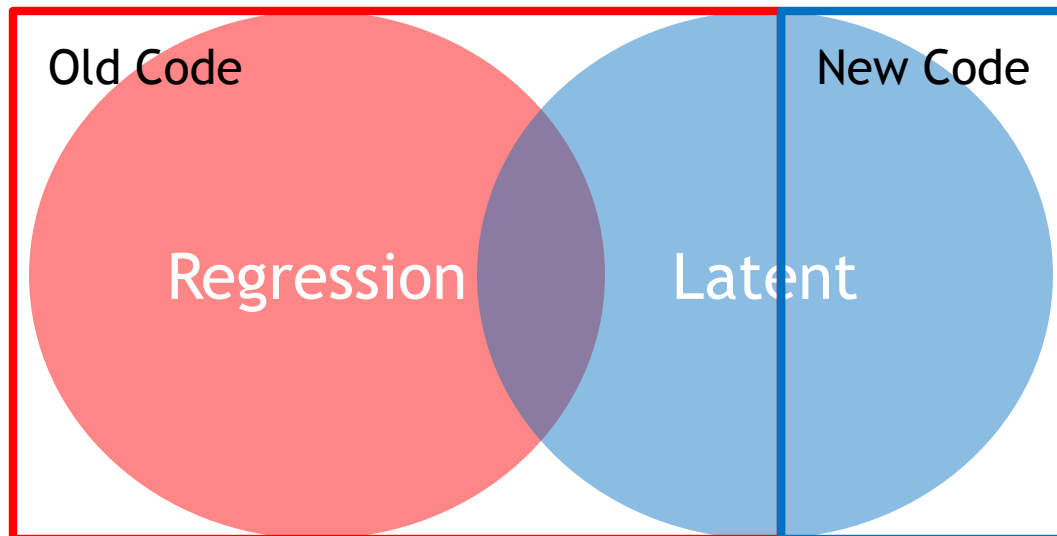
## ⦿ Change of

- Inputs (Data driven testing)
- Environment (Language, OS, Browser, etc.)
- Software (Regression)
  - Old feature no longer working

## ⦿ New feature not working?

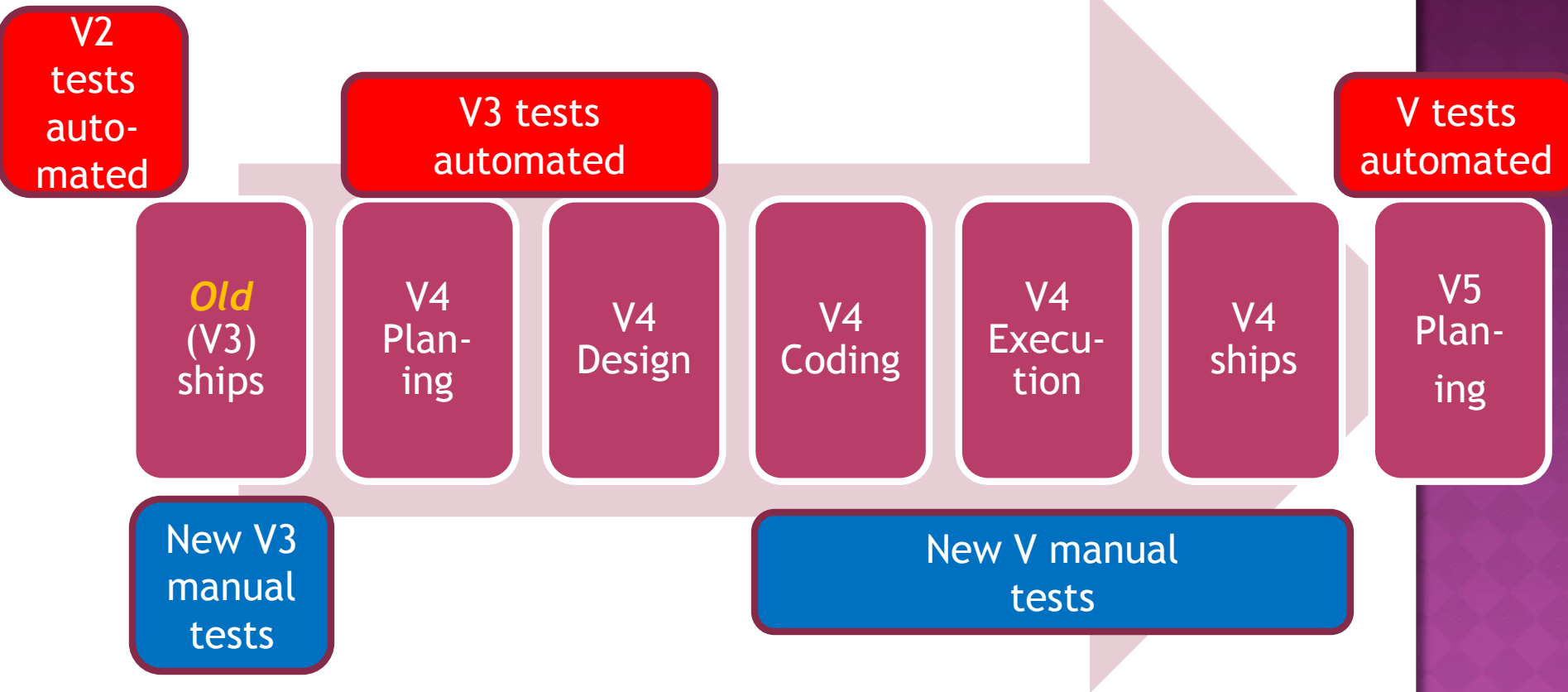
# REGRESSION VS LATENT

- ◉ Writing and running tests both cost.
- ◉ Run same tests, reduce regression defects. Automation mostly helps here
- ◉ Create new tests, reduce latent defects. Manual testing can most help here



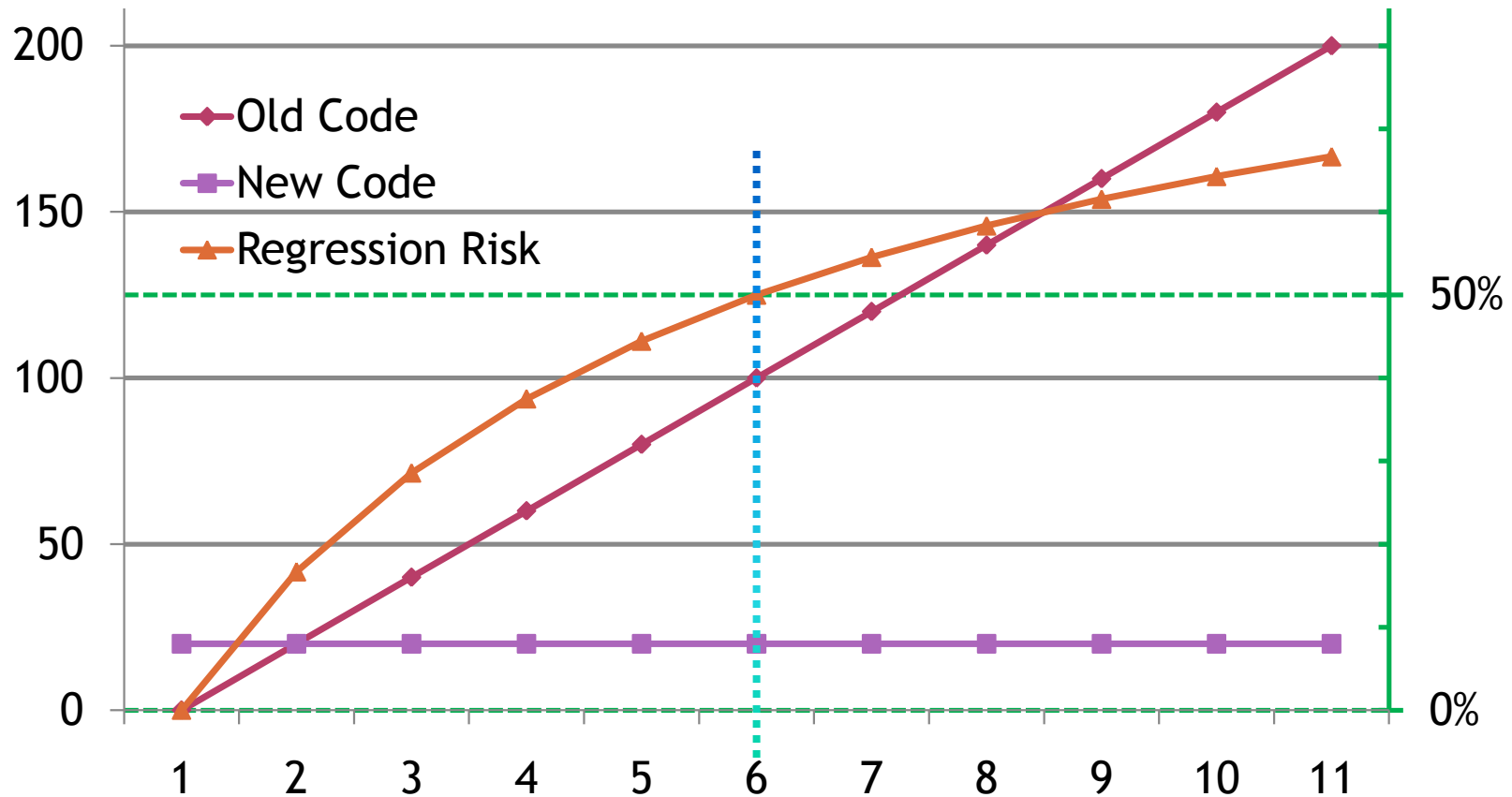
# HYBRID EXAMPLE

## VERSIONS 2, 3, 4, 5



# REGRESSION RISK VS CODE BASE

- Example for 10 iterations when new code has 10% errors and old code 2% regression errors.



# TEST CREATION

Level of detail?

- ◉ Manual : High level (e.g. Exploratory Charters)  
detailed scripts (at level of code)
- ◉ Automated: Model based testing  
keyword action testing  
scripting and compiled code

Less detail is cheaper to create and maintain.

Designed for Change? Coding to Design Patterns

## Charter

- Login

## Outline

- Enter Username and Password

## Specific

- Click mouse in username field to place cursor in username field. Type “testuser”.
- Click mouse in password field to place cursor in password field. Type “Pa\$\$w0rd”
- Click on login button

# Action


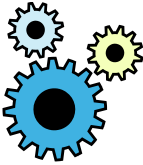




- `LoginEnteringUsernameAndPassword( string username, string password)`

## AUTOMATED DETAILS

# Specific

- `Mouse.Position(username).Click`  
`Username.Enter("testuser")`
- `Mouse.Position(password).Click`  
`Username.Enter("Pa$$w0rd")`
- `LoginButton.Click`

# S.E.A.R.C.H.

- ◉ **Setup** Environment config or check 
- ◉ **Execution** Running the software 
- ◉ **Analysis** Verification with Oracle 
- ◉ **Reporting** Roll ups of Analysis 
- ◉ **Cleanup** Teardown, restoration, etc. 
- ◉ **Help** Doc to Run, Analyze & Maintain 

You can automate or leave manual each part.

# ANALYSIS

- ⦿ Automated:  
Failure analysis by log comparison
- ⦿ Manual:  
UI comparisons - blink comparisons

# UI COMPARISONS

A



**NATIONAL ALERT REGISTRY** *Be Aware • Be Alert • Be Safe*

[ABOUT US](#) [CONTACT US](#) [HELP](#)

Does a sexual offender live in **your** neighborhood?

**Free** search for sex offenders in your area:

Registered Sex Offender Search

Enter Zip Code:

Email Address:

(average search takes less than 5 seconds)

**You have the right to know**

B



**NATIONAL ALERT REGISTRY** *Be Aware • Be Alert • Be Safe*

[ABOUT US](#) [CONTACT US](#) [HELP](#)

Does a sexual offender live in **your** neighborhood?

**Free** search for sex offenders in your area:

**Free** Registered Sex Offender Search

Enter Zip Code:

Email Address:

(average search takes less than 5 seconds)

**You have the right to know**

Source: Marketing Experiments

<http://www.marketingexperiments.com>

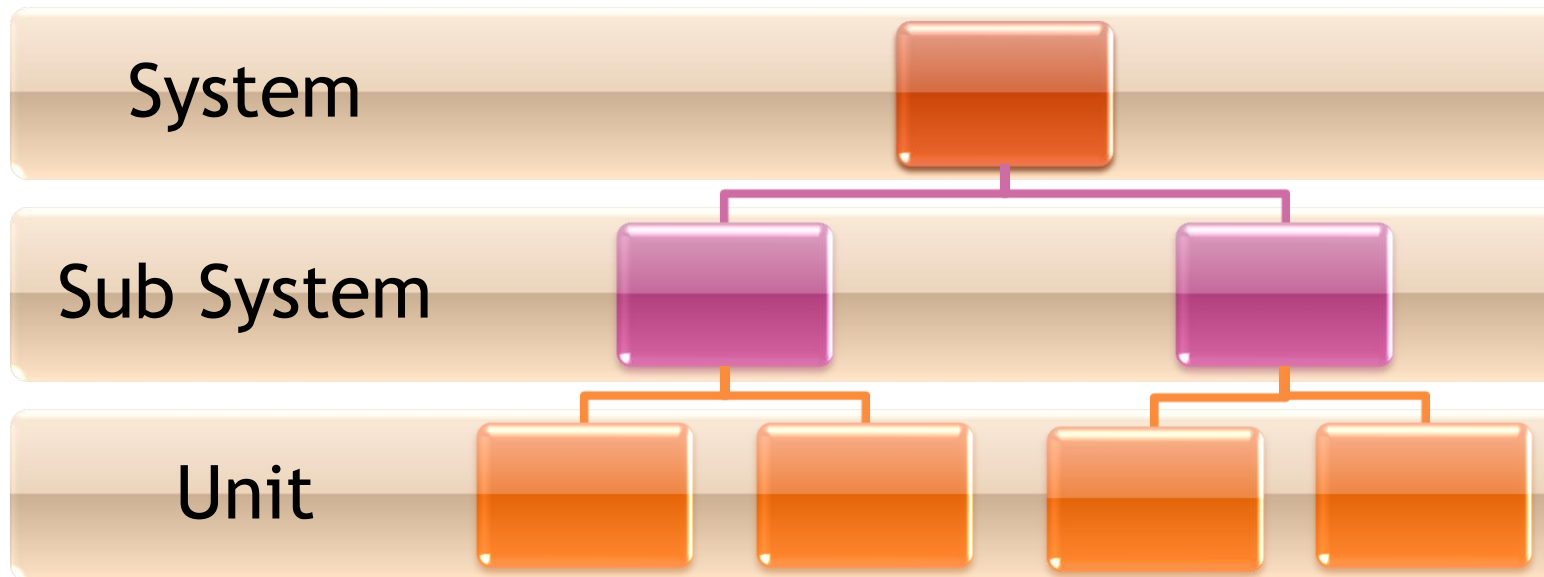
Should you Automate? - Keith Stobis, Microsoft 2009-08-20

# SEARCH COMBO EXAMPLES

- Automated Setup/Cleanup,  
Virtual machines auto scheduled in lab  
Manual Execution and Analysis  
Exploratory testing with Charter
- Manual Setup/Cleanup,  
Hand craft unique environment,  
USB/Net disconnects? Power? Etc.  
Automated Execution and Analysis  
Coded tests with expected results
- Automated Execution with Manual Analysis  
Frame capture for review

# TEST LEVEL

Many unique conditions, perhaps  
test manually only once



Unit tests frequently repeated - thus automated

# MAJOR CONSIDERATIONS

- ⦿  $\Delta$  Rate of change (SW, Env, etc.)
- ⦿  $\omega$  frequency of Execution
- ⦿ Purpose of automation (Latent vs Regression)

Thus:

Automated	Unit Tests	(? $\Delta$ , High $\omega$ , Regression)
Automated	Build Verification Tests	(High $\Delta$ , High $\omega$ , Regression)
Manual	Usability tests	(medium $\Delta$ , medium $\omega$ , Latent)

Automated API tests - natural environment for automation

Automated Performance & Stress - High amount of execution

# RISK COVERAGE

- Prioritize Product and release goals
  - More stable release or focus on new features?
- Work around known errors
  - making progress while waiting for fixes
- Anticipate likely errors
  - Easy to automate error finding  
vs Likely to fail or Important if fail
- Choose evaluation methods [Kaner 2001]
  - Comparison with previous results or with specifications
  - Self-verifying data
  - Oracle-Based Testing
  - Heuristic consistency

# SUMMARY

- **How** repeat? Humans or Software
- **When** repeat? How Often?
- **Why** repeat? Change of Env? SW?  
Data (e.g. boundaries)?  
Variance to give confidence
- **What** repeat? Same Env? SW?  
Data (e.g. perf)?  
Same to give confidence

# Q&A

# REFERENCES

Bergman M., and K. Stobie, 1992 *How to Automate Testing-the Big Picture*, Quality Week 1992 [http://keithstobie.net/Documents/TestAuto\\_The\\_BigPict.pdf](http://keithstobie.net/Documents/TestAuto_The_BigPict.pdf)

Dustin, E., et al 1999, *Automated Software Testing: Introduction, Management, and Performance*, Addison-Wesley Professional 1999. ISBN: 978-0201432879

Gamma, E., et al 1994, *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley Professional, 1994 ISBN: 978-0201633610

Hoffman, D. 1999, *Cost Benefits Analysis of Test Automation*, Proceedings of the Software Testing Analysis and Review Conference (STAR East). Orland, Florida. May 1999  
<http://www.softwarequalitymethods.com/Papers/Star99%20model%20Paper.pdf>

Kaner, C., J. Bach and B. Pettichord. 2001, *Lessons Learned in Software Testing*, Wiley 2001

Marick, B. 2000, *Testing For Programmers*, (pages 43-47)  
<http://www.exampler.com/testing-com/writings/half-day-programmer.pdf>

Marick, B. 1999, *Maybe Testers Shouldn't Be Involved Quite So Early*  
<http://www.exampler.com/testing-com/writings/testers-upstream.pdf>

# REFERENCES 2

Marick, B. 1998, *When Should a Test Be Automated?*, Quality Week 1998

<http://www.exampler.com/testing-com/writings/automate.pdf>

Mosley, D. and B. Posey 2002, *Just Enough Software Test Automation*, Prentice Hall PTR, 2002. ISBN: 978-0130084682

Saran, c 2008, *Autistic people prove valuable in software testing*, ComputerWorld, 15 Feb 2009, <http://www.computerweekly.com/Articles/2008/02/15/229432/autistic-people-prove-valuable-in-software-testing.htm>

Staneff, G., 2007, *Test Logging and Automated Failure Analysis*, SASQAG September 2007. [http://www.sasqag.org/pastmeetings/9-2007/SASQAG\\_9-20-2007.ppt](http://www.sasqag.org/pastmeetings/9-2007/SASQAG_9-20-2007.ppt)

Travison, D. and Staneff, G. 2008, *Test Instrumentation and Pattern Matching for Automatic Failure Identification*, 1st International Conference on Software Testing, Verification, and Validation, 2008 ISBN: 978-0-7695-3127-4

Whittaker, J. 2002, *How to Break Software: A Practical Guide to Testing*, Addison Wesley 2002  
ISBN: 978-0201796193