



🕒 11 Avril 2017

9<sup>ème</sup> édition

JOURNÉE FRANÇAISE  
DES TESTS LOGICIELS



LE TEST LOGICIEL, MAÎTRISER L'ÉTAT  
DE L'ART DU TEST AVEC LE CFTL !

ISTQB  
International Software  
Testing Qualifications Board

# Smarter Testing with Artificial Intelligence

Stuart Reid PhD, FBCS

([stureid.test@gmail.com](mailto:stureid.test@gmail.com) / [www.stureid.info](http://www.stureid.info))

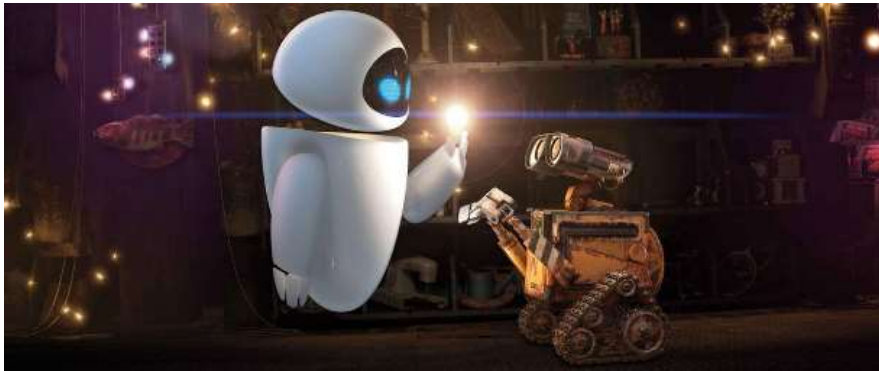
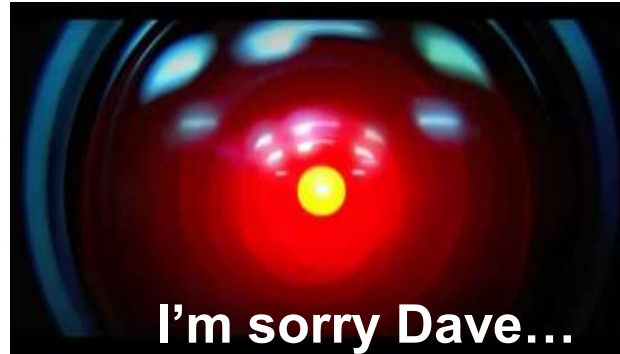
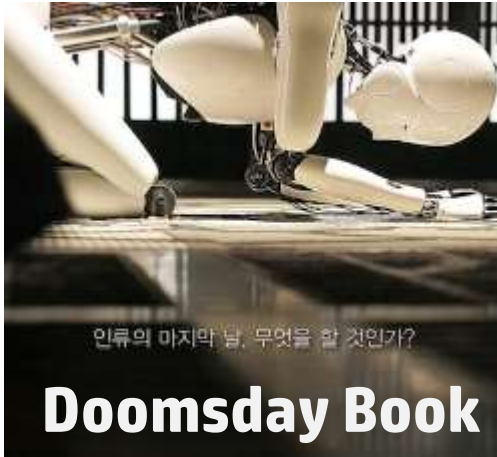
© Stuart Reid 2017

# Contents

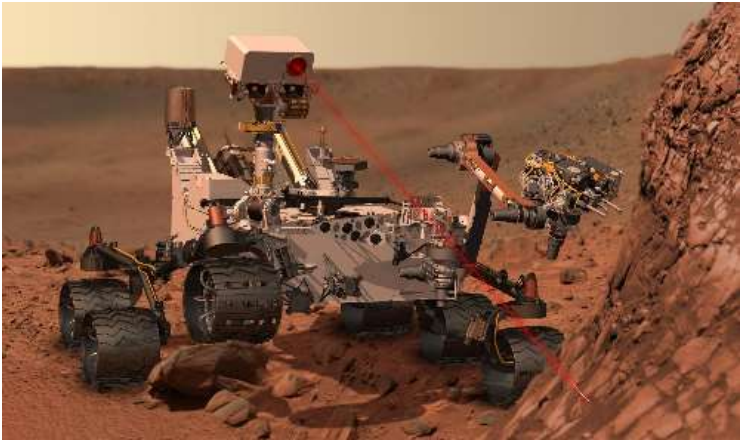
---

- **Artificial Intelligence and Testing**
- **Bug Prediction**
- **Static Analysis**
- **Regression Testing**
- **Automated Test Input Generation**
- **Automated Stress Testing**
- **Conclusions**

# Artificial Intelligence (AI) in the Cinema



# Artificial Intelligence (AI) Works!



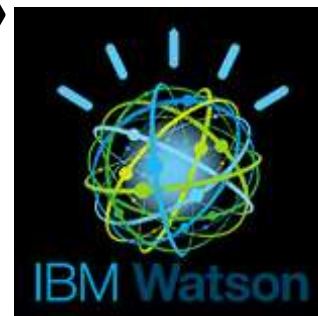
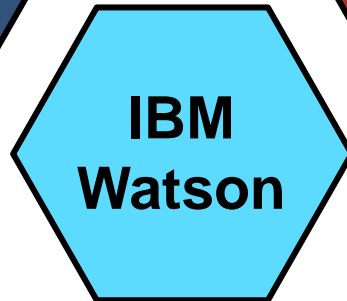
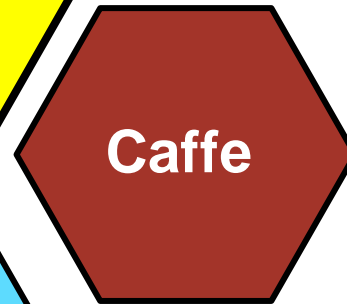
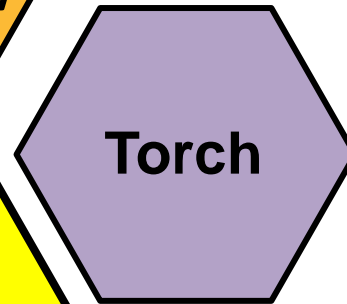
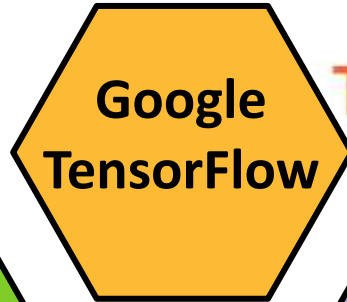
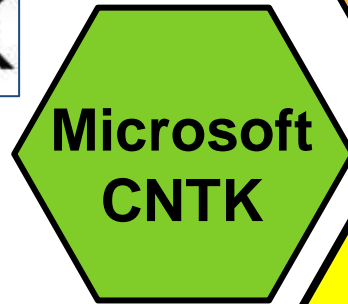
# Artificial Intelligence Techniques

---

- **Neural Networks**
- **Expert/Knowledge-based Systems**
- **Machine Learning**
- **Fuzzy and Probabilistic Logic**
- **Classification**
- **Search and Optimization**
  
- **Much of today's effective AI uses a variety of overlapping techniques**
  - and exploits the availability of processing power & storage



# AI Toolkits



# AI - Smart Testing Opportunities

Specification Testing

Bug Prediction

Regression Testing

Stress Testing

Specification-based Script Generation

Static Analysis

Test Input Generation

Smart Test Manager

Usage Profiling

Automated Exploratory Testing

Test Strategy Assistant

Interactive Dashboard

Defect Management

Crowd Testing

# ***Bug Prediction***



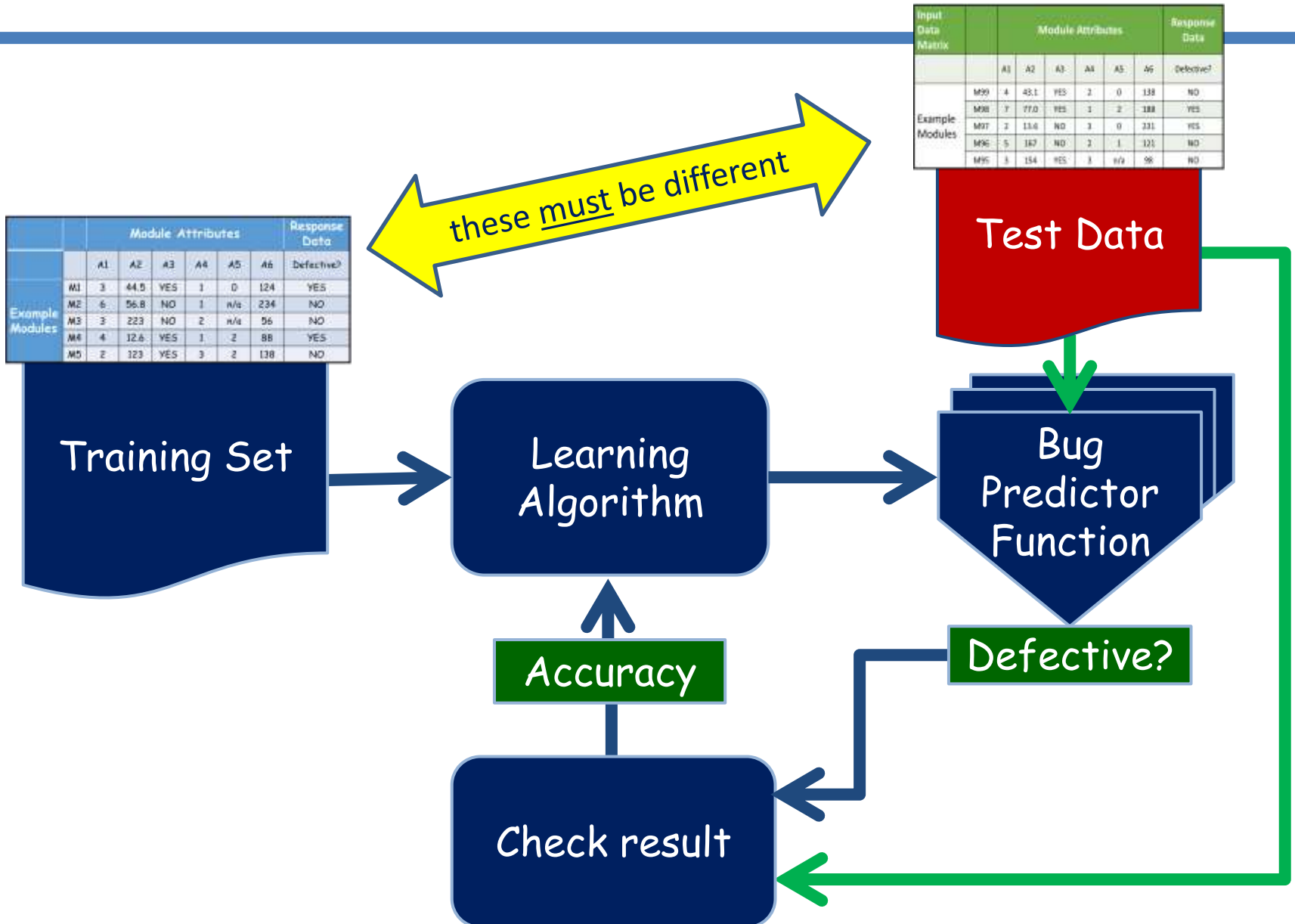




# Bug Prediction – Multiple Attributes

Input Data Matrix		Module Attributes						Response Data
		A1	A2	A3	A4	A5	A6	Defective?
Example Modules	M1	3	44.5	YES	1	0	124	YES
	M2	6	56.8	NO	1	n/a	234	NO
	M3	3	223	NO	2	n/a	56	NO
	M4	4	12.6	YES	1	2	88	YES
	M5	2	123	YES	3	2	138	NO

# Supervised Learning Process



# Bug Prediction Metrics

---

- **Source code metrics**
  - Lines of code
  - Number of comments
  - Cyclomatic complexity
  - Module dependencies
- **Process metrics**
  - Revisions made to module
  - Times refactored
  - Times fixed / when fixed
  - Lines of code changed (code churn)
  - Module age
- **People and organizational metrics**
  - Number of authors
  - Developer experience

# Bug Prediction Results

- **“87% detection rate achieved average with 26% false alarms”**
  - [Tosun, 2010]
- **“73%-95% of faults can be predicted in just 10% of files” (across 7 projects)**
  - [Kim, 2007]
- **Best predictors are:**
  - People and Organizational measures (84%)
  - Module change (80%)
  - Fixed recently (and connected modules)
  - Reused module are more error-prone than new modules
  - Module age

# ***Static Analysis***



# Static Analysis Tool - Facebook – Infer

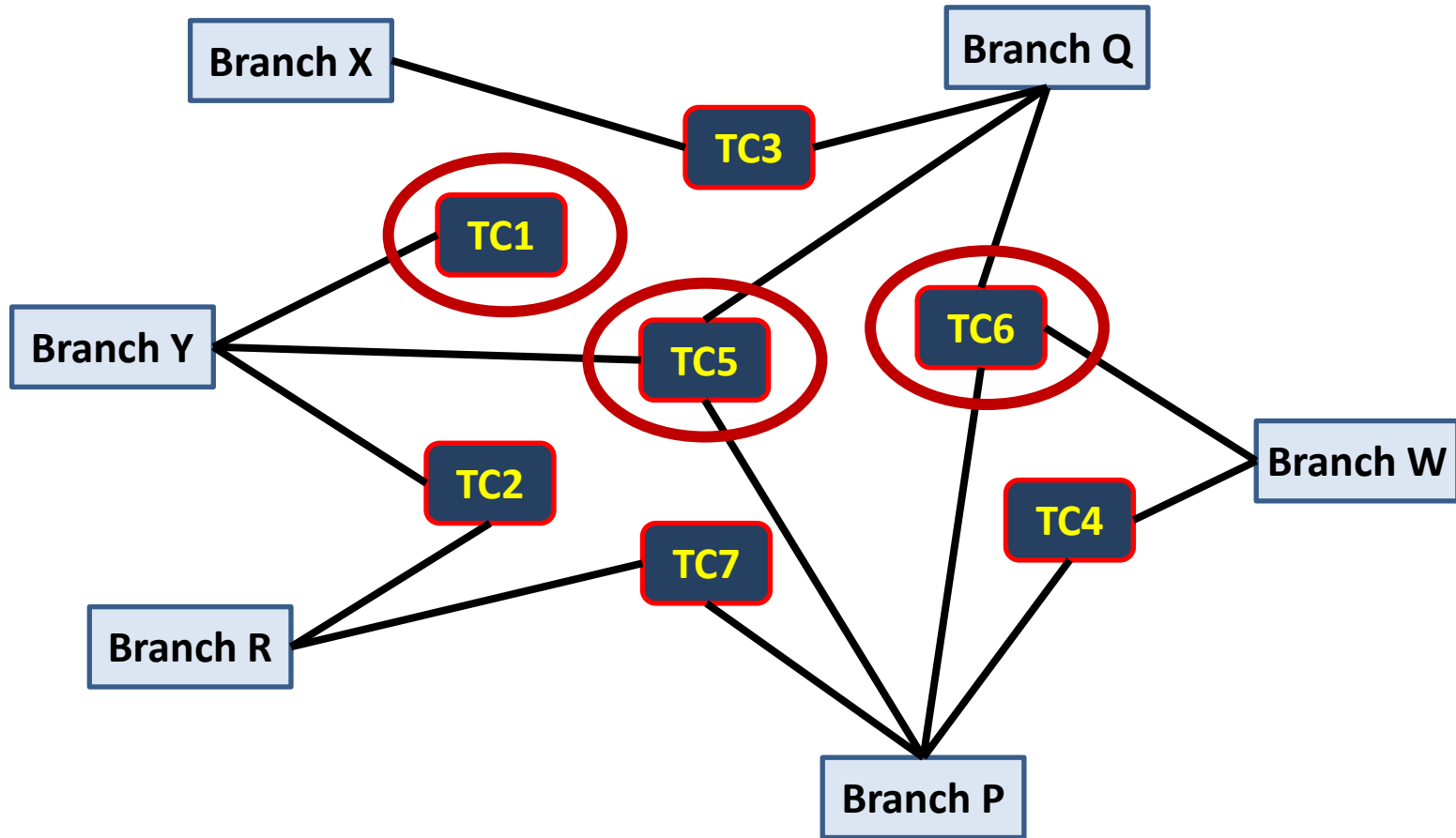


- **Open source**
- **Analyses C, Objective-C and Java**
  - on Android and iOS
- **Fast – can do millions of LOC in a few minutes**
  - ideal for continuous integration
- **Facebook claims that approximately 80% of raised issues are fixed (so are true faults)**
- **Also used by Instagram, Uber, Spotify, etc.**

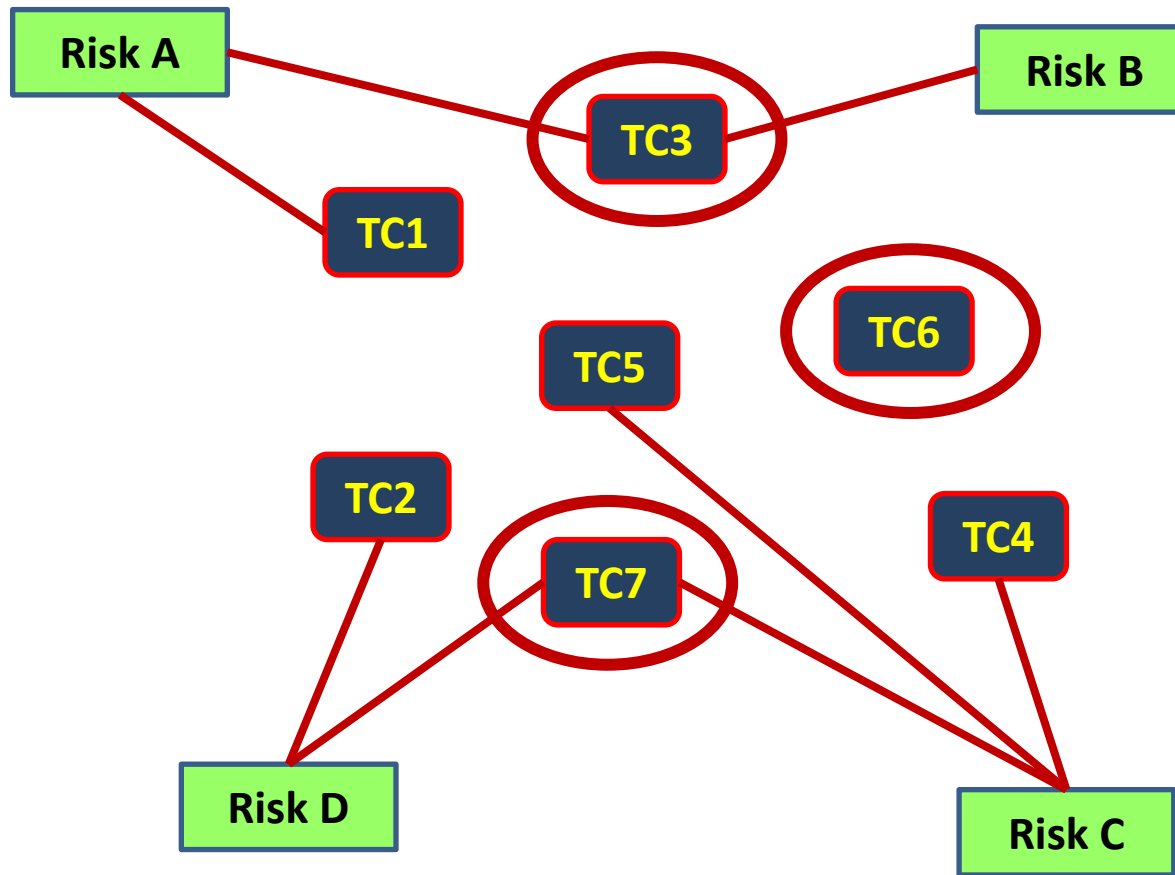


# ***Regression Testing***

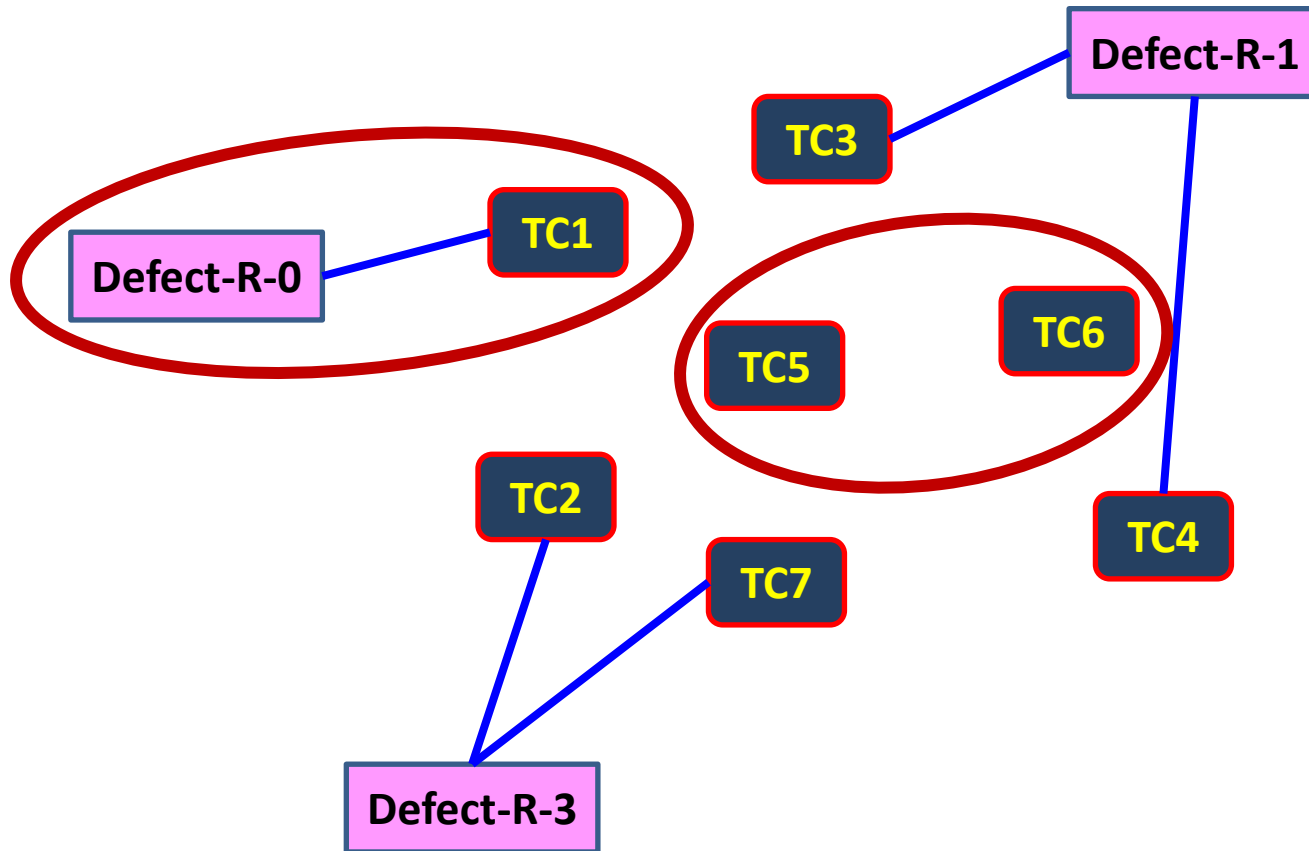
# Regression Test Optimization



# Regression Test Optimization



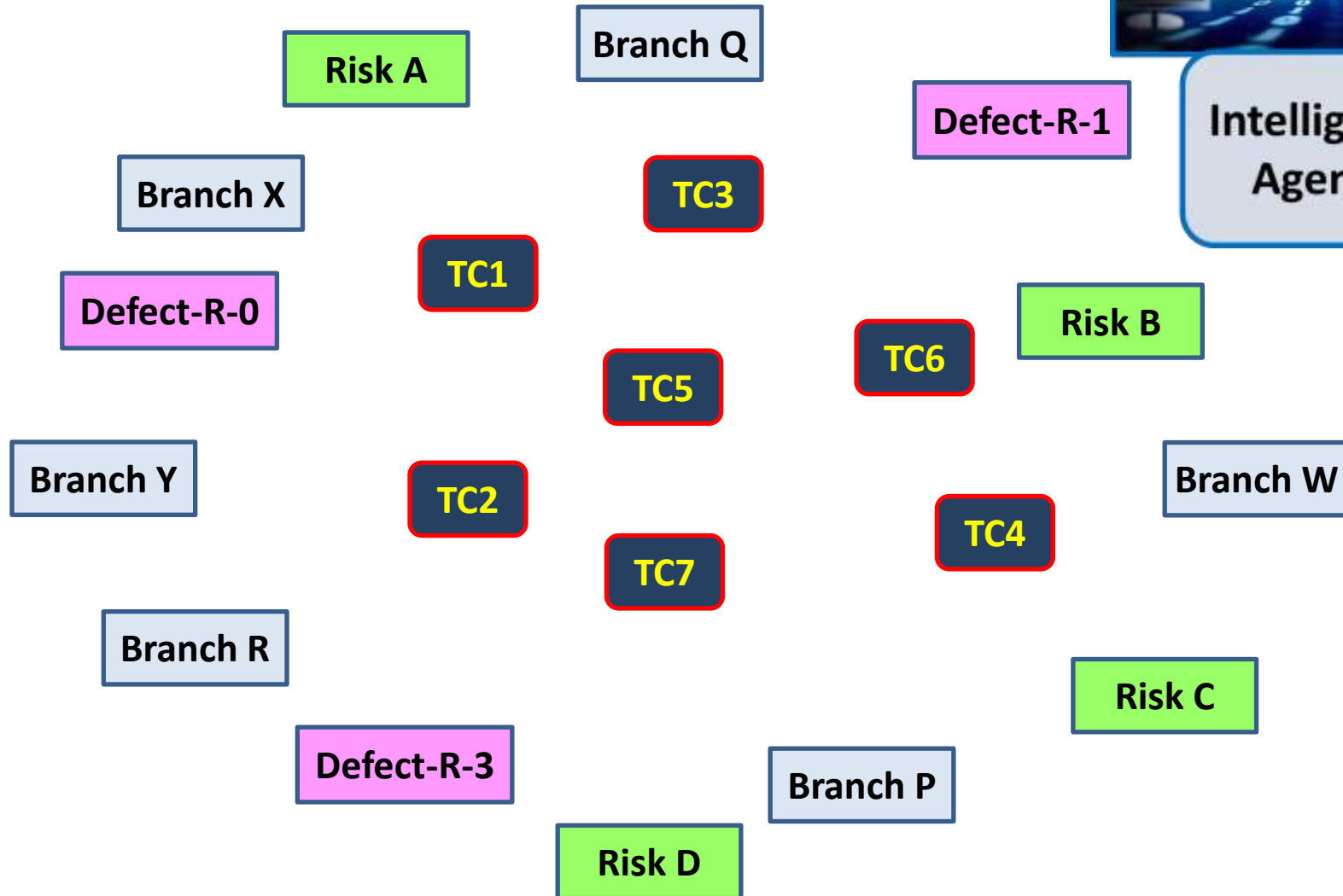
# Regression Test Optimization



# Regression Test Selection



Intelligent Agent



# Regression Test Prioritization



Intelligent  
Agent





# Regression Test Optimization Criteria

---

- Tests that found defects previously
- Tests that reduce execution time
- Reduce the number of tests needed
- Tests that achieve full coverage
- Test that exercise recently changed code
- Tests that address high risks
- etc.

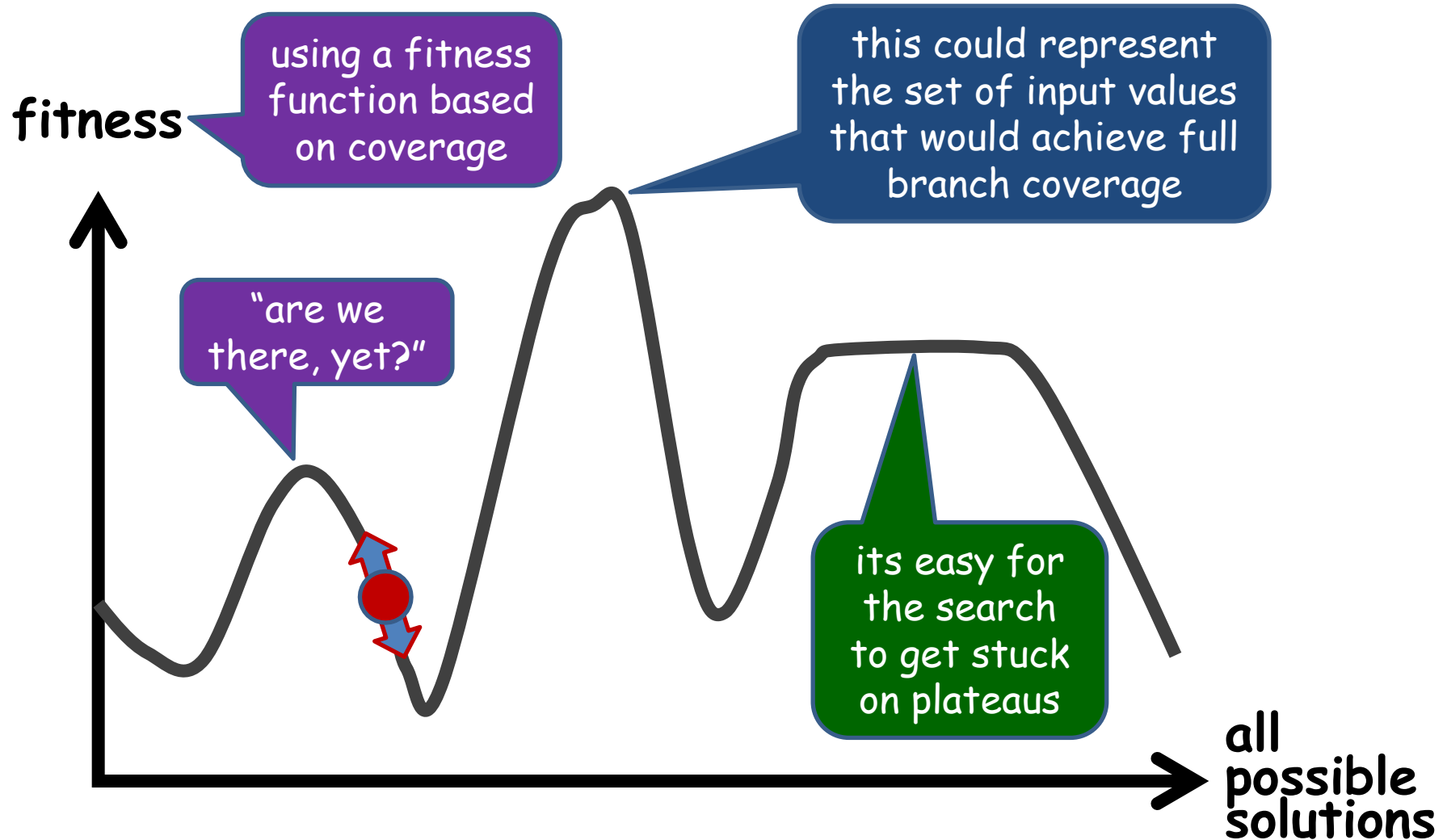
# Regression Test Optimization Results

- **The algorithm reduces the test suite data by approximately 50%**
  - [Rai, 2014]
- **The techniques are 40-50% more effective in uncovering the first failure of the changed program**
  - [Jiang, 2009]
- **Average reduction in test suite size of 94% while achieving requirements-based coverage**
  - implemented in:
    - a continuous integration env't with 30 seconds run time
    - implemented at Cisco, Norway
  - [Gotlieb, 2016]

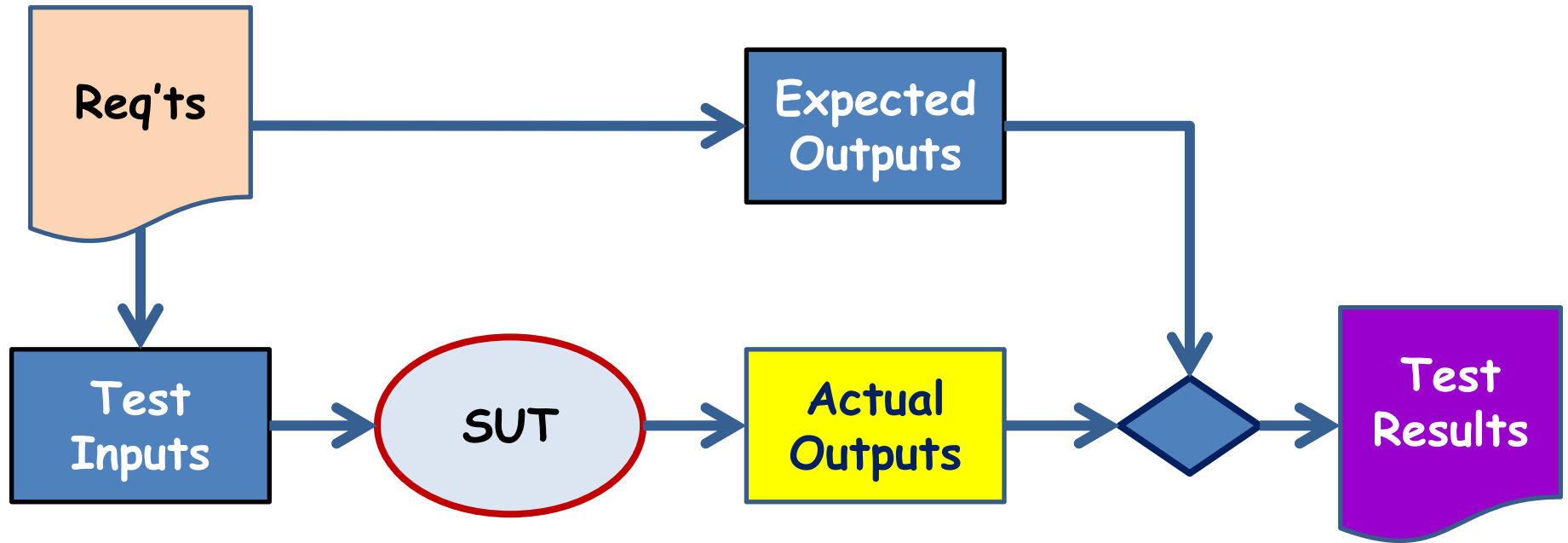
# ***Automated Test Input Generation***

# Example

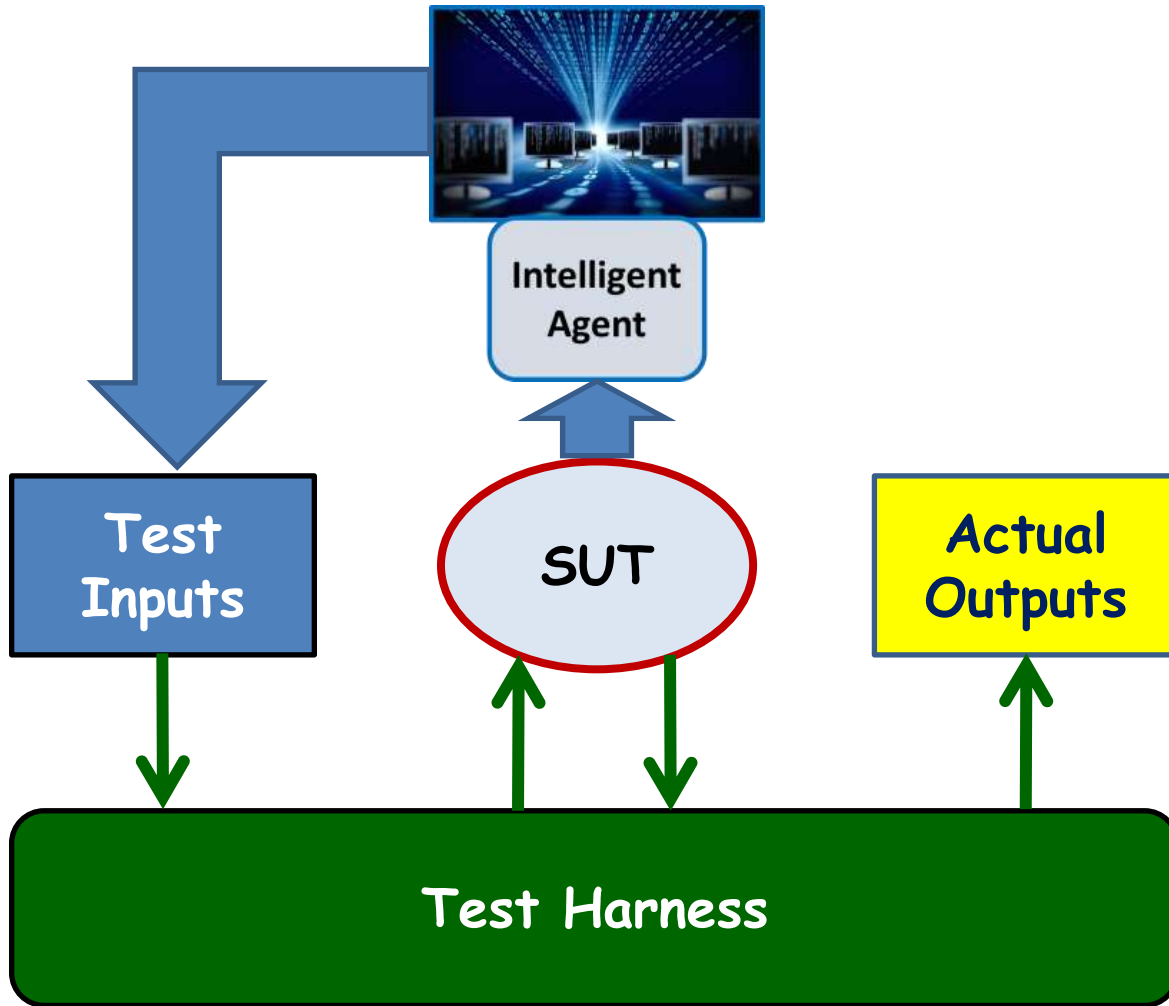
## – Searching using a 'Hill Climb' Algorithm



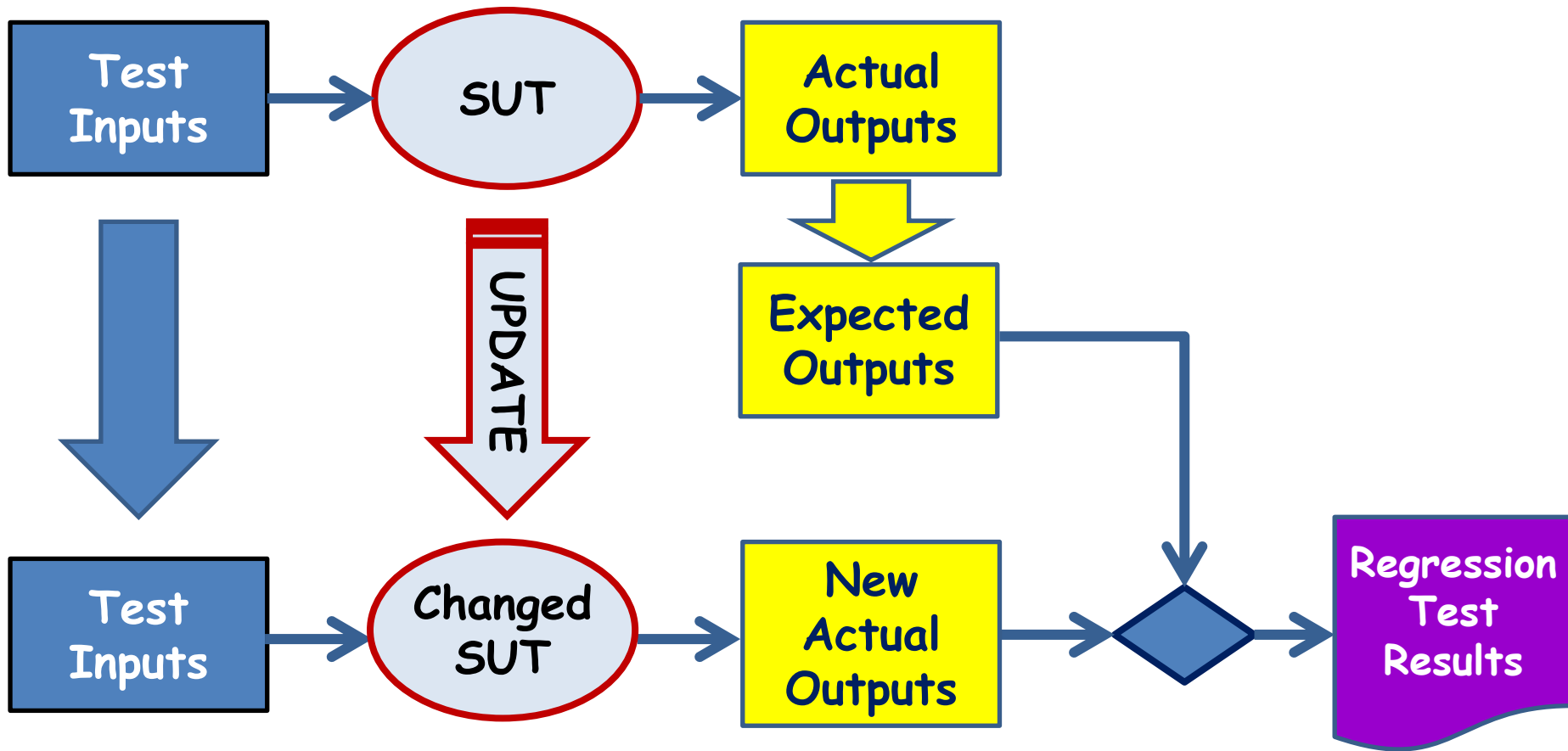
# Manual Test Process



# Automated Test Input Generation

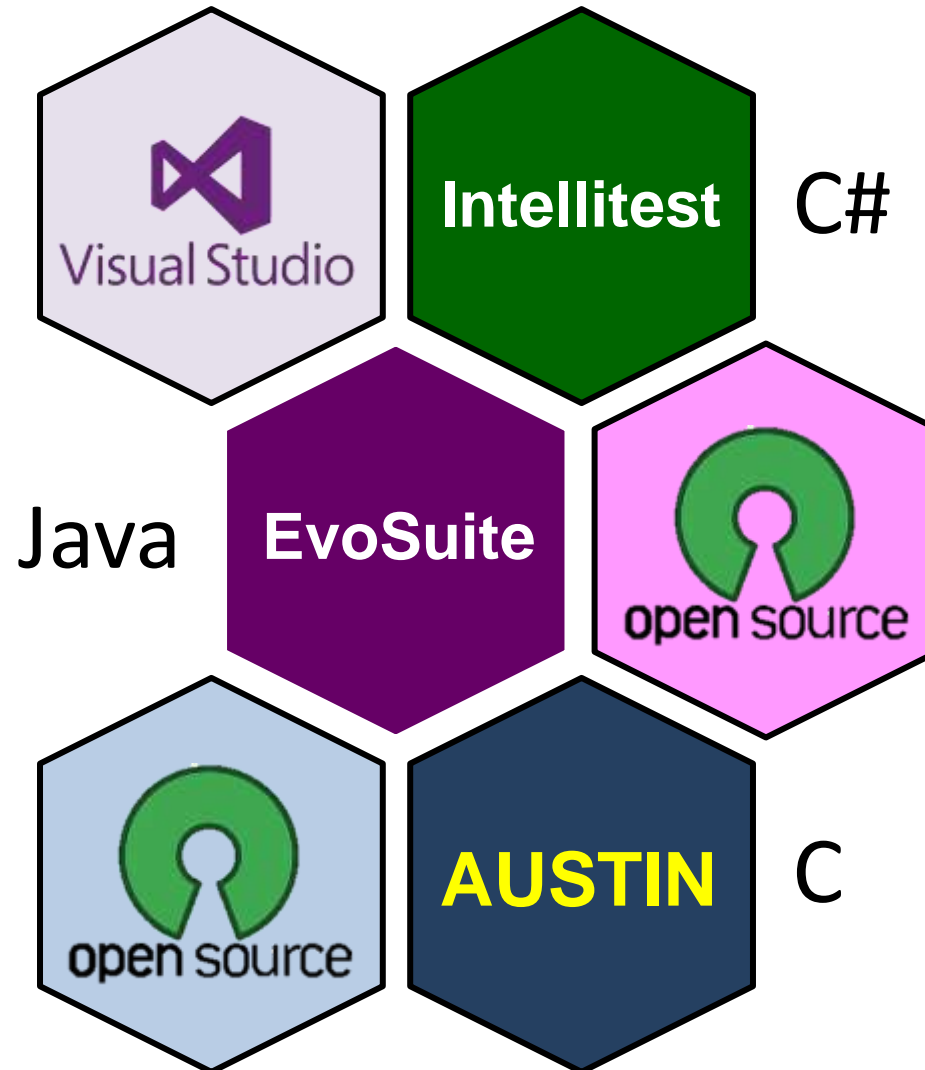


# Automated Regression Test Case Generation





# Example Tools



# Automated Test Input Generation - Summary

- **Empirical studies have shown:**
  - tool support can lead to improvements in code coverage of up to 300%
  - that there is no measurable improvement in the number of bugs actually found by developer/testers – even though more branches are covered
- **But, a set of automatically-generated regression tests providing full coverage is an excellent starting point when you change or refactor the code**
- **Danger!!!**
  - testers rely on the tool → little or no black box testing
  - testers use the tool to meet safety-related test standards

# ***Automated Stress Testing***

# Automated Stress Testing Tools

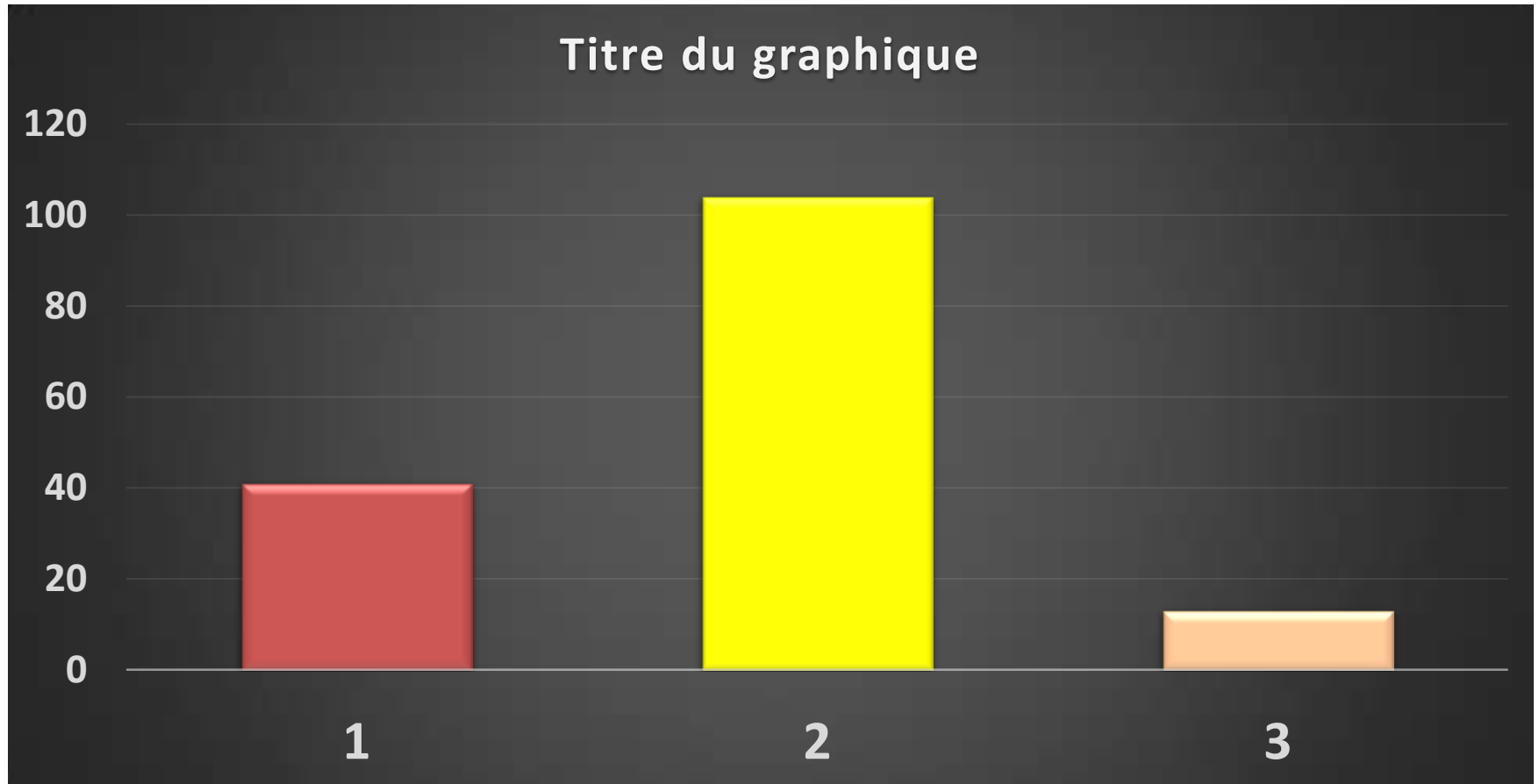
---

- **Generate pseudo-random streams of user events such as clicks, touches, or gestures, as well as a number of system-level events**
  - they pretend they are a ‘stupid’ tester
- **Aim to cause an ANR (‘Application Not Responding’) or for the app to simply crash**
  - so test result is easy to observe
- **Require little tester input**
  - except to check-out the reported failures

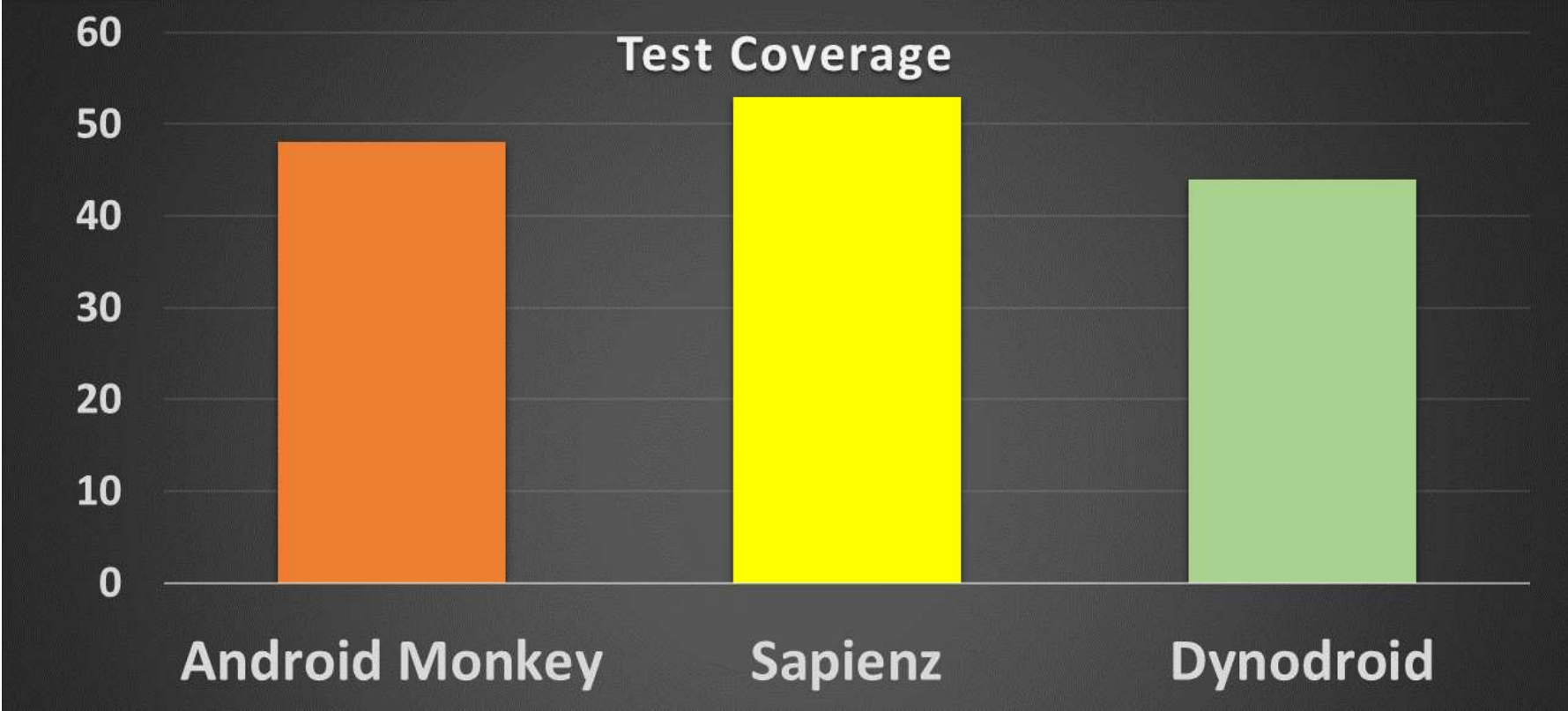
# Example - Android Stress Testing Tools

- **Google Monkey**
  - built into the Android development platform - free
  - fuzz testing tool – random inputs
- **Sapienz**
  - open source
  - search-based testing tool
  - when applied to the top 1,000 Google Play apps, Sapienz found 558 unique, previously-unknown faults
- **Dynodroid**
  - open source
  - allows interleaving of human and tool
  - when applied to the top 1,000 Google Play apps, Dynodroid found 6 unique, previously unknown faults

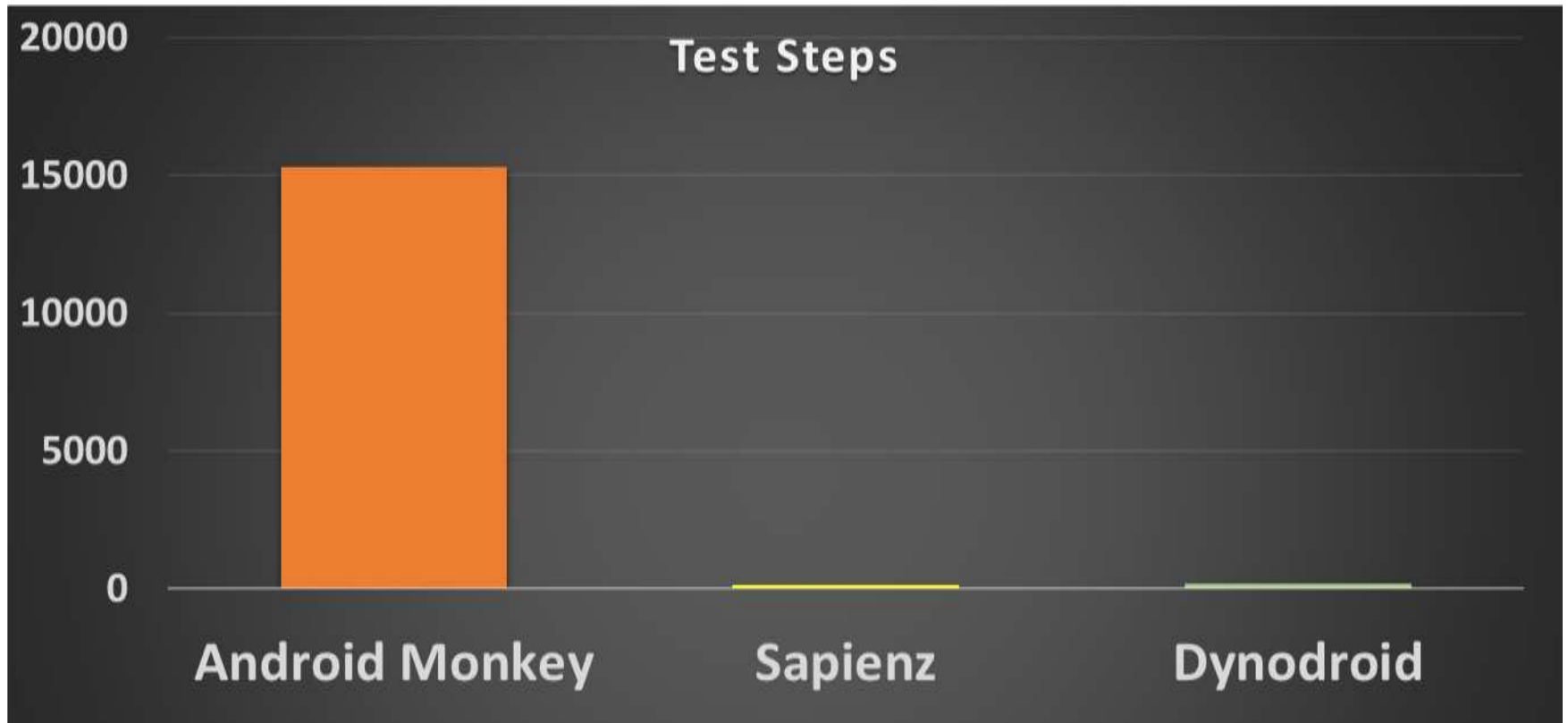
# Defect Detection Effectiveness



# Test Coverage



# Fault Revealing Steps





# Conclusions

---

- **Artificial Intelligence and Testing**
- **Bug Prediction**
- **Static Analysis**
- **Regression Testing**
- **Automated Test Input Generation**
- **Automated Stress Testing**

Thank you for listening 😊

---

**Any Questions?**