Better Software Testing through University-Industry Collaborations

Vahid Garousi, PhD, PEng
Assistant Professor of Software Engineering
Software Quality Engineering Research Group (SoftQual)
Department of Electrical and Computer Engineering
Schulich School of Engineering

Computer Chapter Chair, IEEE Southern Alberta Section

A talk for the SQDG
Calgary, AB
Sept. 20, 2011

Outline of the Talk

- Brief background about the speaker and his team in the UofC
- An overview of our SW testing projects
- Reviewing several selected projects
- Wrap-up: What can be gained from collaborative university/industry software testing projects?

Background of the Speaker

- Education:
  - PEng, 2008
  - PhD (Carleton University), 2006
  - MSc (University of Waterloo), 2003
  - BSc (Sharif University of Technology, Iran), 2000
- Has been with the UofC since September 2006
- Has worked in and with the software industry for 15+ years
- Focus Areas:
  - Software Engineering (in general)
  - Software Testing, and Software Test Engineering
  - UML-driven Development
  - Developing Scientific Software

Software Quality Engineering Research Group (SoftQual)

- www.softqual.ucalgary.ca
- Current students:
  - 1 research associate (co-supervised)
  - 1 PhD student
  - 2 MSc students
  - 1 U/G
- Alumni
  - 7 MSc
  - 17+ U/G
  - 1 research associate

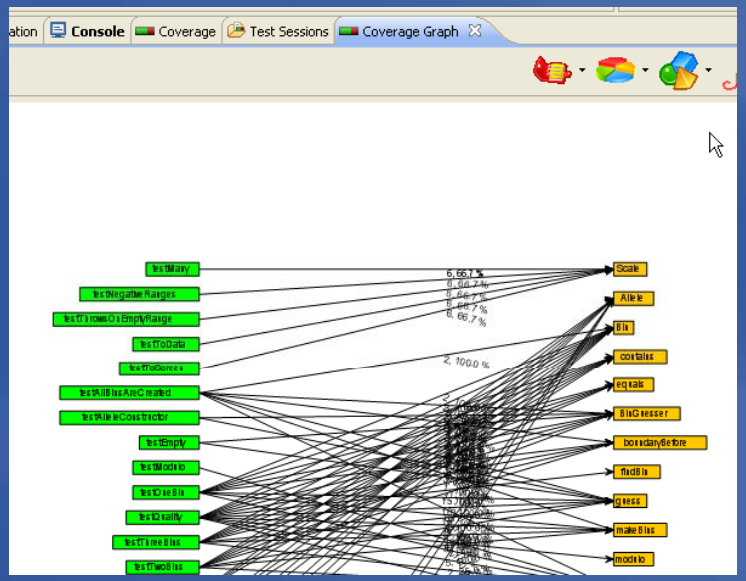
- All our projects are applied R&D projects
- Source of funding:
  - Governmental agencies such as NSERC, Alberta Innovates, etc.
  - Companies (via R&D project), e.g., IBM, Siemens, NovAtel, MR Control Systems, Analog Devices

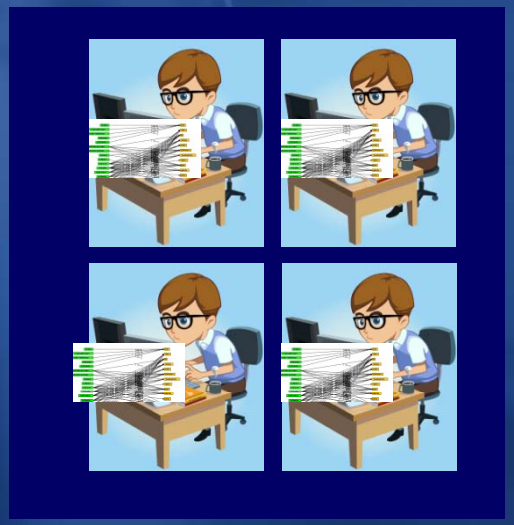
Outline of the Talk

- Brief background about the speaker and his team in the UofC
- An overview of our SW testing projects
- Reviewing several selected projects
- Wrap-up: What can be gained from collaborative university/industry software testing projects?

Projects: What do we really do?

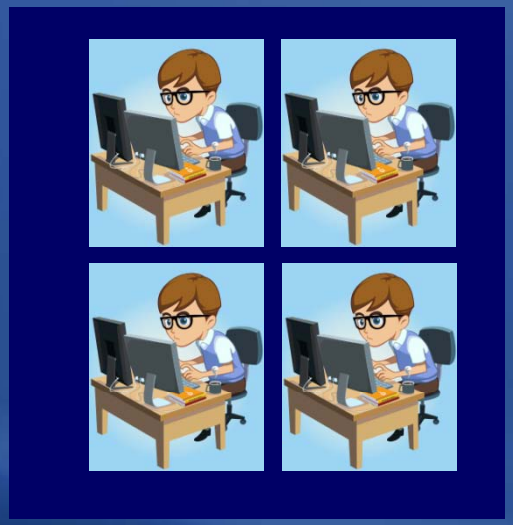
- Coming up with new methods to develop and/or test software
- Tool development (e.g., test coverage visualization)
- Empirically evaluating if a tool/method really works in practice or not
- More info: www.softqual.ucalgary.ca





versus…

Impact on SW maintenance productivity?



Projects - Active collaborations with the software industry

- In various capacities: R&D, consulting, training
- Reducing software maintenance (debugging) costs (2011-)
- Testing control software systems for the power industry (2009-)
- Testing Embedded Software (2008-2011)
- Engineering (developing) optimization software for the oil industry (2007-2011)
- Testing industrial real-time software systems (2006-2009)
- Survey of SW development and testing practices
- Improving SW testing and development by visualization of code coverage (2007-2012)

- More info: www.softqual.ucalgary.ca

Outline of the Talk

- Brief background about the speaker and his team in the UofC
- An overview of our SW testing projects
- Projects Selected for Discussions Today
  - Surveys of SW testing practices in Alberta and Canada
  - Improving SW testing and development by visualization of code coverage and traceability
  - Testing control software systems for the power industry
  - Testing Embedded Software
  - Optimization of software maintenance costs
  - Engineering (developing) optimization software for the oil industry
  - Testing industrial real-time software systems
- Wrap-up: What can be gained from collaborative university/industry software testing projects?

Surveys of SW testing practices in Alberta and Canada

- **Goal:** To get a broad picture of the SW testing practices in Alberta and Canada
- Our colleagues had done an earlier Alberta-wide survey in 2004
- We repeated it in 2009
- In 2010, we improved the questions and did it across Canada
- Some results next…
- Please see the articles for details.

The Journal of Systems and Software 83 (2010) 2251–2262

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

A replicated survey of software testing practices in the Canadian province of Alberta: What has changed from 2004 to 2009?

Vahid Garousi*, Tan Varma

Software Quality Engineering Research Group (SoftQual)[1], Department of Electrical and Computer Engineering, University of Calgary, 2500 University Drive NW, Calgary, AB, Canada T2N 1N4

ARTICLE INFO

ABSTRACT

Software organizations have typically de-emphasized the importance of software testing. In an earlier study in 2004, our colleagues reported the results of an Alberta-wide regional survey of software testing techniques in practice. Five years after that first study, the authors felt it is time to replicate the survey and analyze what has changed and what not from 2004 to 2009. This study was conducted during the summer of 2009 by surveying software organizations in the Canadian province of Alberta. The survey results reveal important and interesting findings about software testing practices in Alberta, and point out what has changed from 2004 to 2009 and what not. Note that although our study is conducted in the province of Alberta, we have compared the results to few international similar studies, such as the ones conducted in the UK, Turkey, Hong Kong and Australia. The study should thus be of interest to all testing professionals world-wide. Among the findings are the following: (1) almost all companies perform unit and system testing with a slight increase since 2004, (2) automation of unit, integration and systems tests has increased sharply since 2004, (3) more organization are using observations and expert opinion to conduct usability testing, (4) the choices of test-case generation mechanisms have not changed much from 2004, (5) JUnit and IBM Rational tools are the most widely used test tools, (6) Alberta companies still face approximately the same defect-related economic issues as do companies in other jurisdictions, (7) Alberta software firms have improved their test automation capability since 2004, but there is still some room for improvements, and, (8) compared to 2004, more companies are spending more effort on pre-release testing.

© 2010 Elsevier Inc. All rights reserved.

1. Introduction

A survey of software testing practices in Alberta

Un aperçu des pratiques d'essai de logiciel en Alberta

Adam M. Geras, M.R. Smith, and J. Miller*

1. Introduction

Surveys of SW testing practices in Alberta (2004, 2009)

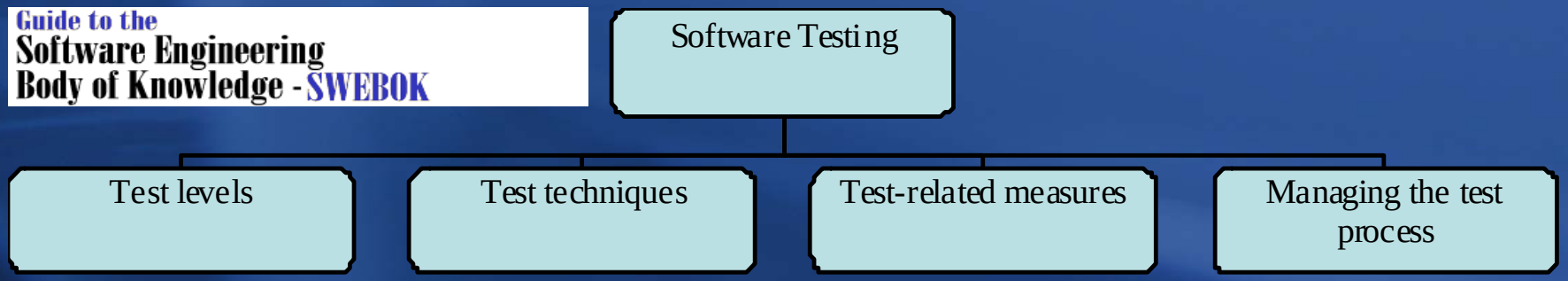
- Respondents:
  - 53 respondents
  - To get a measure of the sample size…
  - According to StatsCan, as of 2007, there were 2,947 SW developers (*publishers*) in AB

Statistics Canada Statistique Canada

Table 2
Summary statistics for the software publishers industry,

  - Assuming about a third of them are doing testing. Thus: about 1000 SW testers in AB
  - Thus sample size: 53/1000=5.3%
- Questions were developed and categorized using the IEEE SWEBOK

Guide to the Software Engineering Body of Knowledge - SWEBOK



Surveys of SW testing practices in Alberta (2004, 2009)

Aspect	Questions
Respondents Profiles	▪ Company profile: Please summarize your company and the type of projects you do in a few keywords. ▪ What best describes your current position? ▪ What is the size of your company (number of employees)? ▪ Which programming languages do you use in your company?

Surveys of SW testing practices in Alberta (2004, 2009)

Aspect	Questions
Test levels	▪ Have you received any formal software-testing related training? ▪ Does your organization have a training program that specifically targets any of the following? ▪ In your current or most recent software project, did the team conduct the following tests? ▪ In your current or most recent project, did the team automate any of the tests? ▪ What forms of usability testing are commonly employed in your organization?

Surveys of SW testing practices in Alberta (2004, 2009)

Aspect	Questions
Test techniques	▪ Which testing tools and frameworks do you use in your company? ▪ In your current or most recent software project, what mechanisms did the team use to generate test cases? ▪ Which of the following defect prevention techniques are regularly utilized in your organization?

Surveys of SW testing practices in Alberta (2004, 2009)

Aspect	Questions
Test-related measures	▪ In your current or most recent software project, did the team use any of the following measurements as a guide to planning or designing the tests?

Surveys of SW testing practices in Alberta (2004, 2009)

Aspect	Questions
Test process management	▪ In your current or most recent project, was the testing environment separate from the development or production environments? ▪ Please identify the ratio of developers to testers in your current or most recent project. ▪ What percentage of the pre-release work effort is spent on testing? ▪ Please rank the following defect types by the effort required to fix that type of defect. ▪ What criteria does your organization utilize to terminate the testing phase? ▪ What barriers do you believe prevents your company from adopting testing methodology and testing tools? ▪ What barriers do you believe prevents your company from providing software training to testing staff?

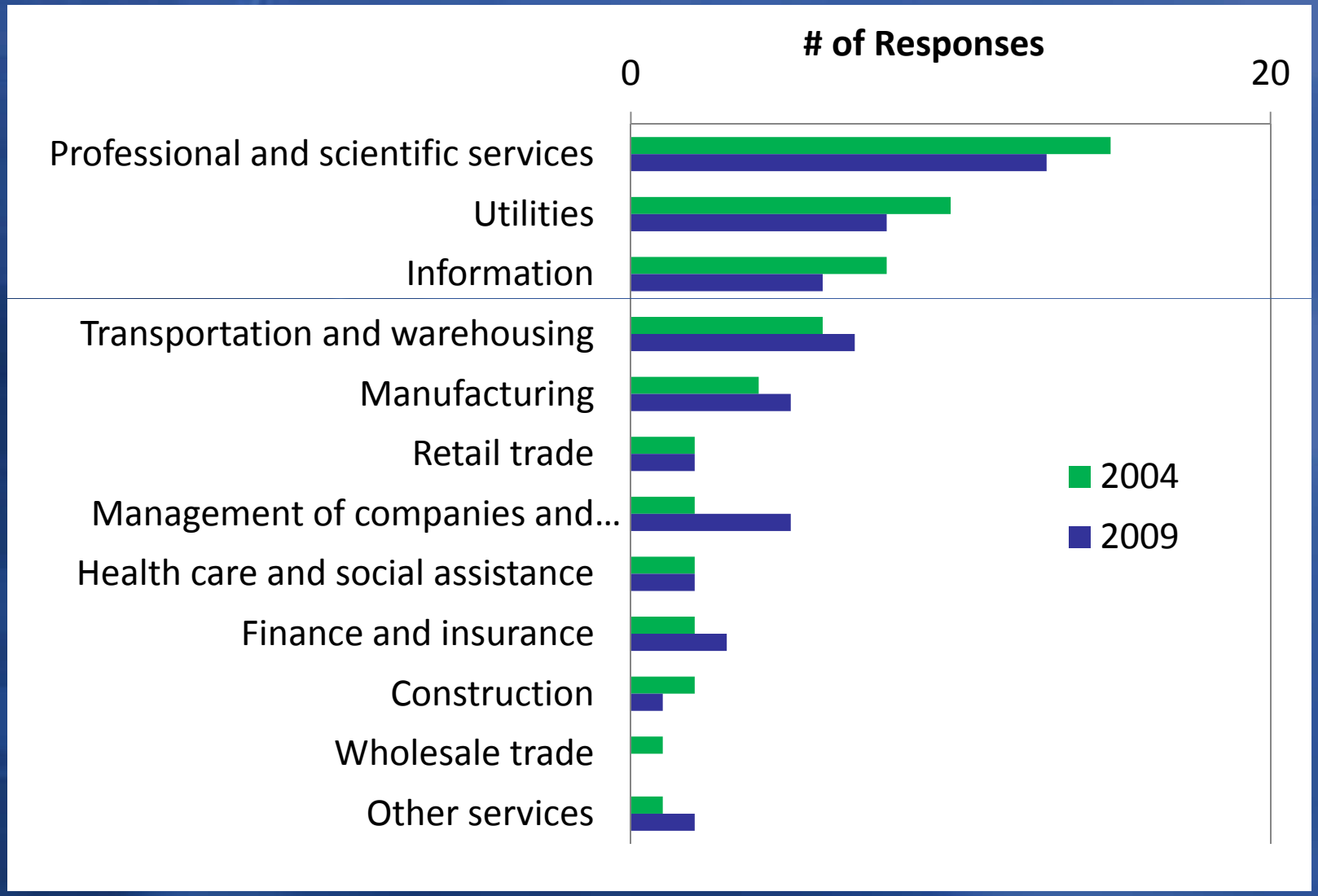
Surveys of SW testing practices in Alberta

Results for seven of the questions to be presented next…

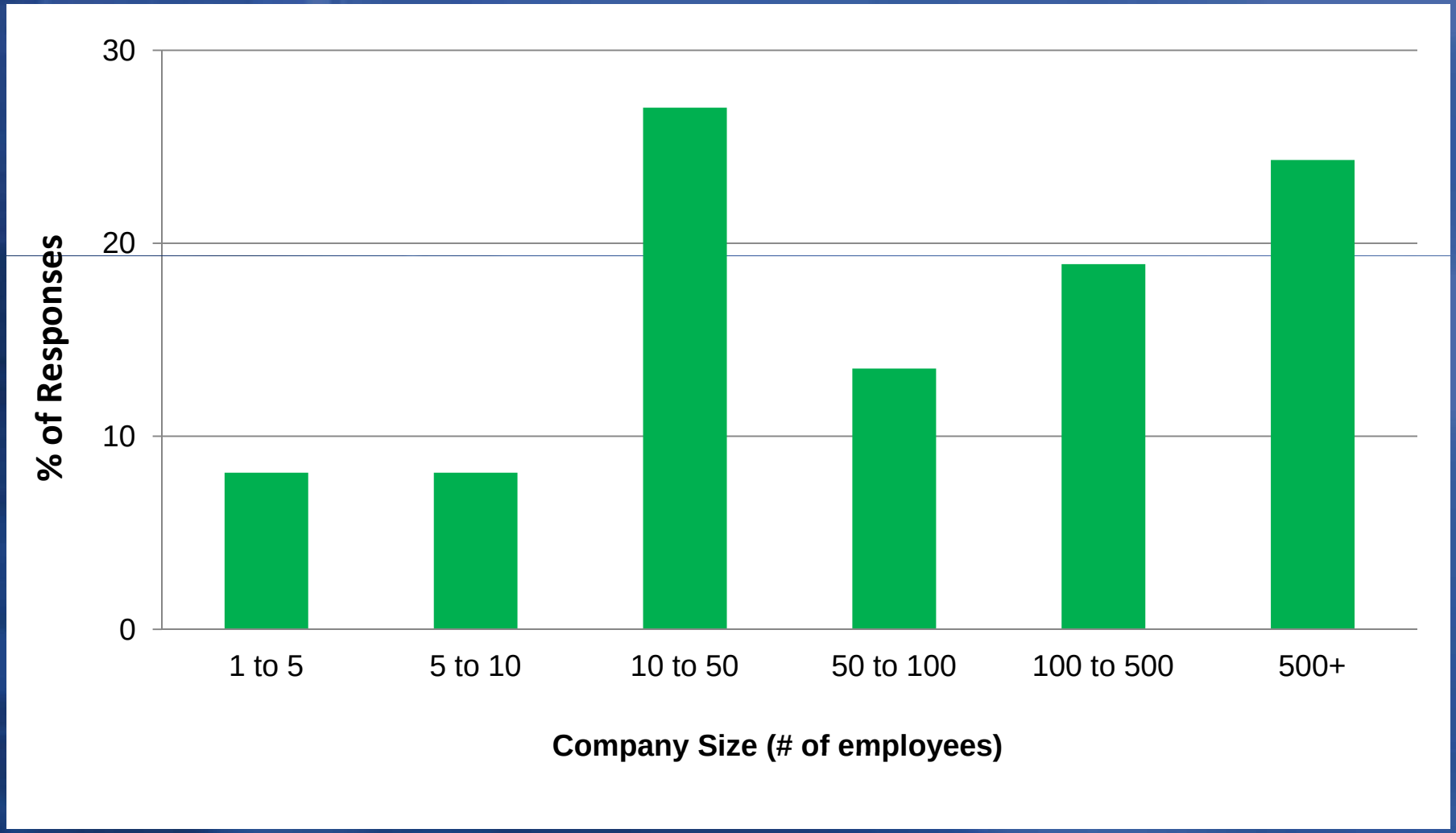
Aspect	Questions
Respondents Profiles	▪ Company profile: Please summarize your company and the type of projects you do in a few keywords. ▪ What best describes your current position? ▪ What is the size of your company (number of employees)? ▪ Which programming languages do you use in your company?
Test levels	▪ Have you received any formal software-testing related training? ▪ Does your organization have a training program that specifically targets any of the following? ▪ In your current or most recent software project, did the team conduct the following tests? ▪ In your current or most recent project, did the team automate any of the tests? ▪ What forms of usability testing are commonly employed in your organization?
Test techniques	▪ Which testing tools and frameworks do you use in your company? ▪ In your current or most recent software project, what mechanisms did the team use to generate test cases? ▪ Which of the following defect prevention techniques are regularly utilized in your organization?
Test-related measures	▪ In your current or most recent software project, did the team use any of the following measurements as a guide to planning or designing the tests?
Test process management	▪ In your current or most recent project, was the testing environment separate from the development or production environments? ▪ Please identify the ratio of developers to testers in your current or most recent project. ▪ What percentage of the pre-release work effort is spent on testing? ▪ Please rank the following defect types by the effort required to fix that type of defect. ▪ What criteria does your organization utilize to terminate the testing phase? ▪ What barriers do you believe prevents your company from adopting testing methodology and testing tools? ▪ What barriers do you believe prevents your company from providing software training to testing staff?

Surveys of SW testing practices in Alberta

Industry Sectors of the Respondents



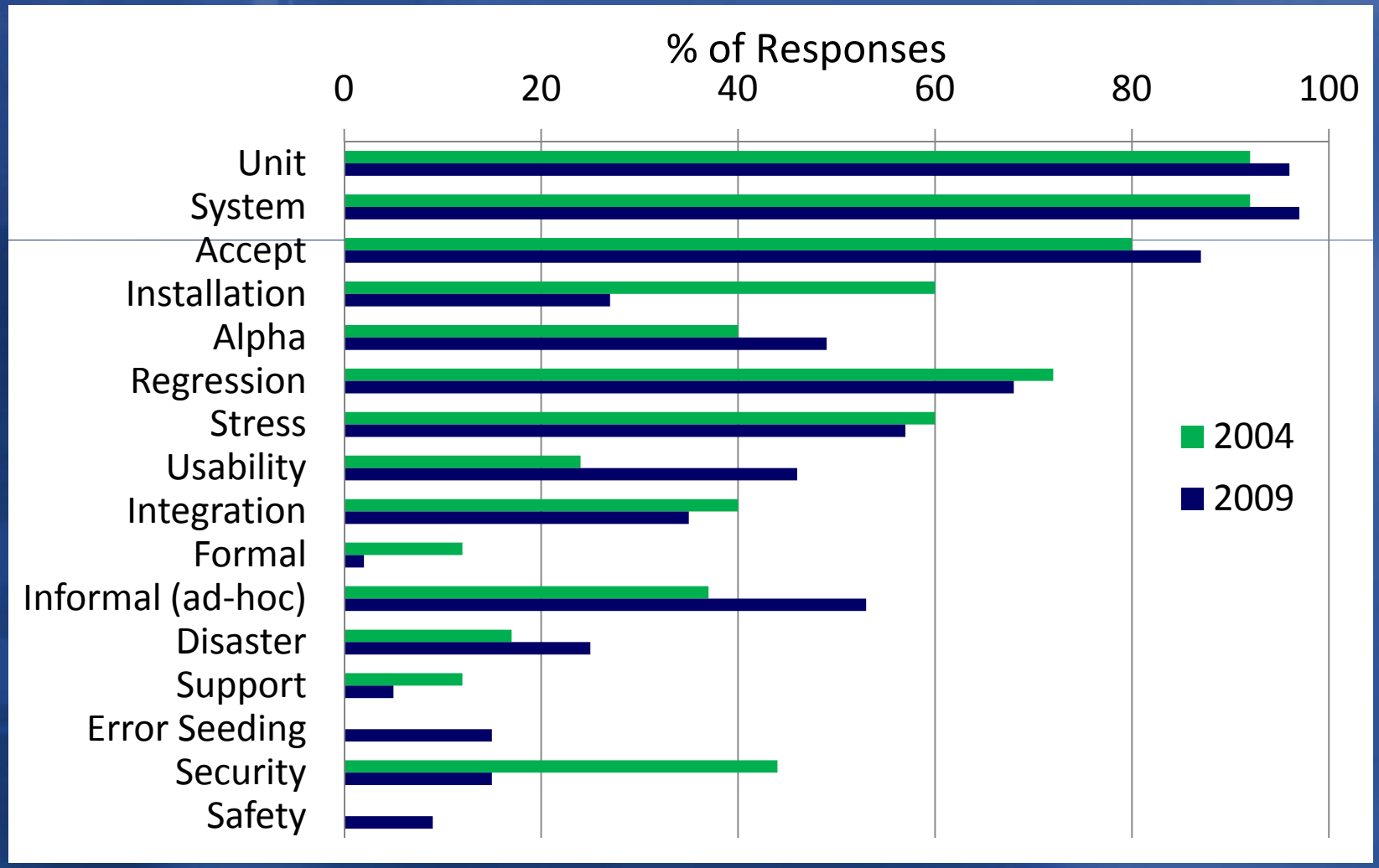
Surveys of SW testing practices in Alberta
Company Size



Surveys of SW testing practices in Alberta

Types of Testing (Test Levels)

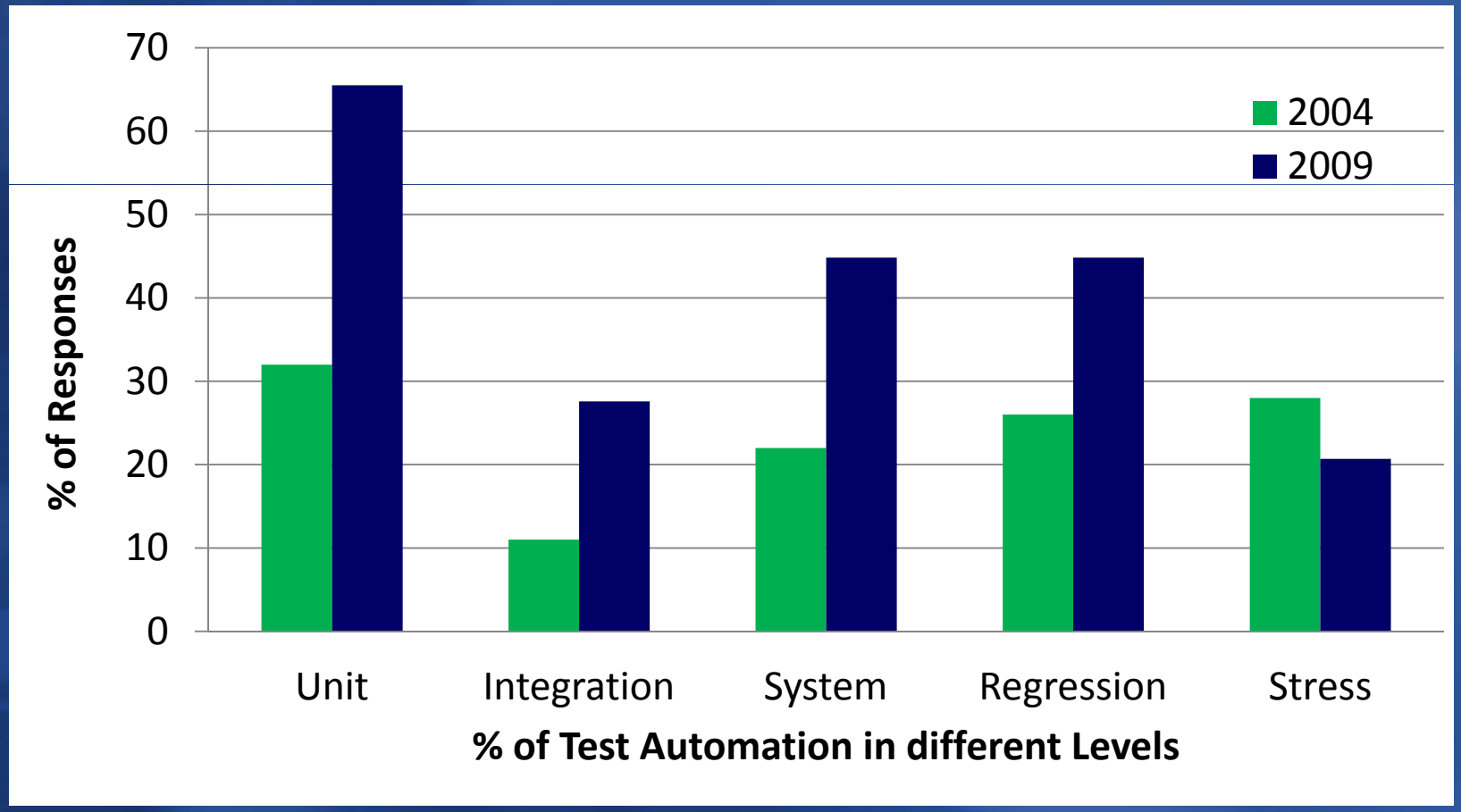
- Almost all companies perform unit and system testing.



Surveys of SW testing practices in Alberta

Test Automation

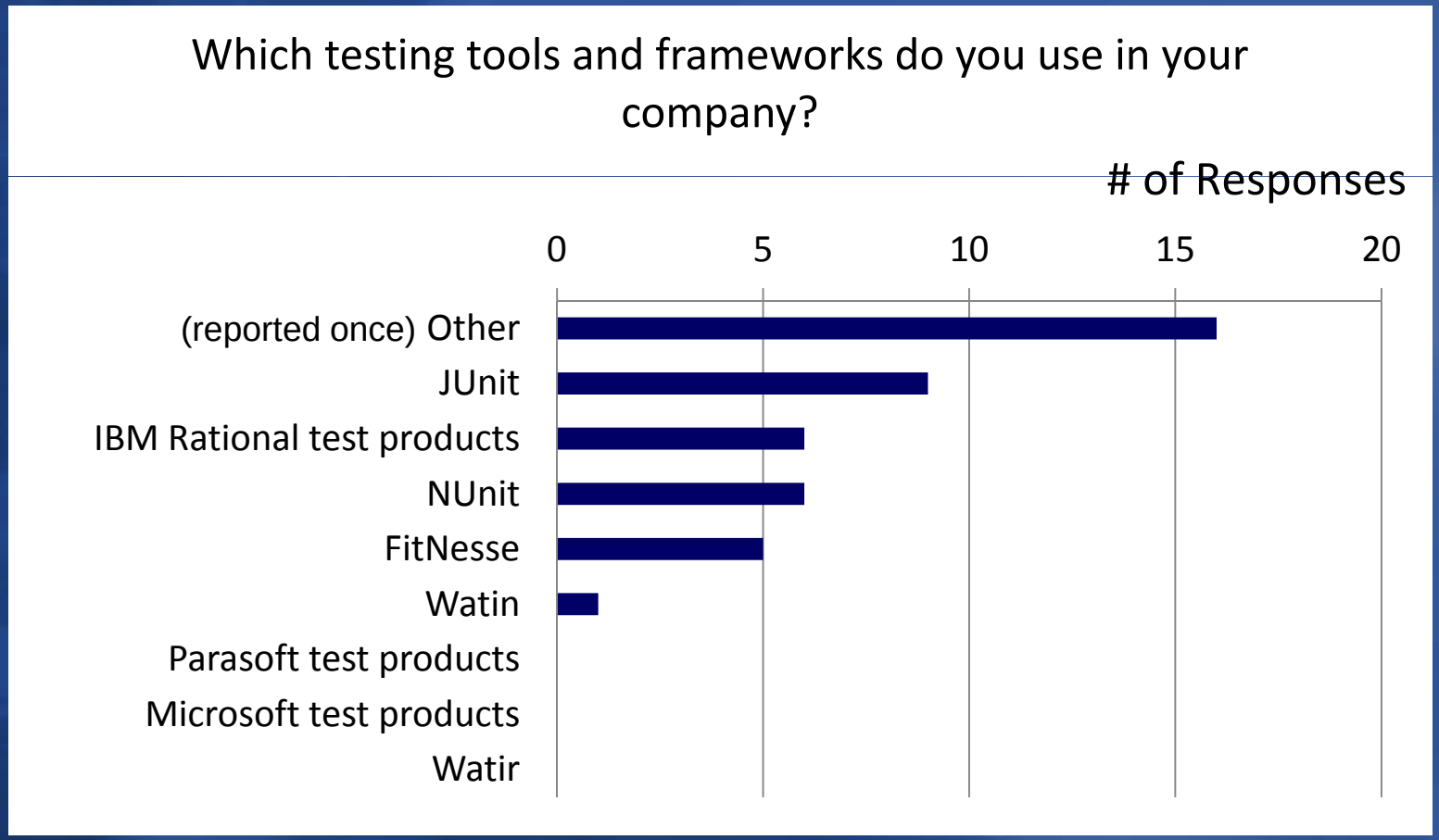
- Automation of unit, integration and systems tests has increased sharply since 2004.



Surveys of SW testing practices in Alberta

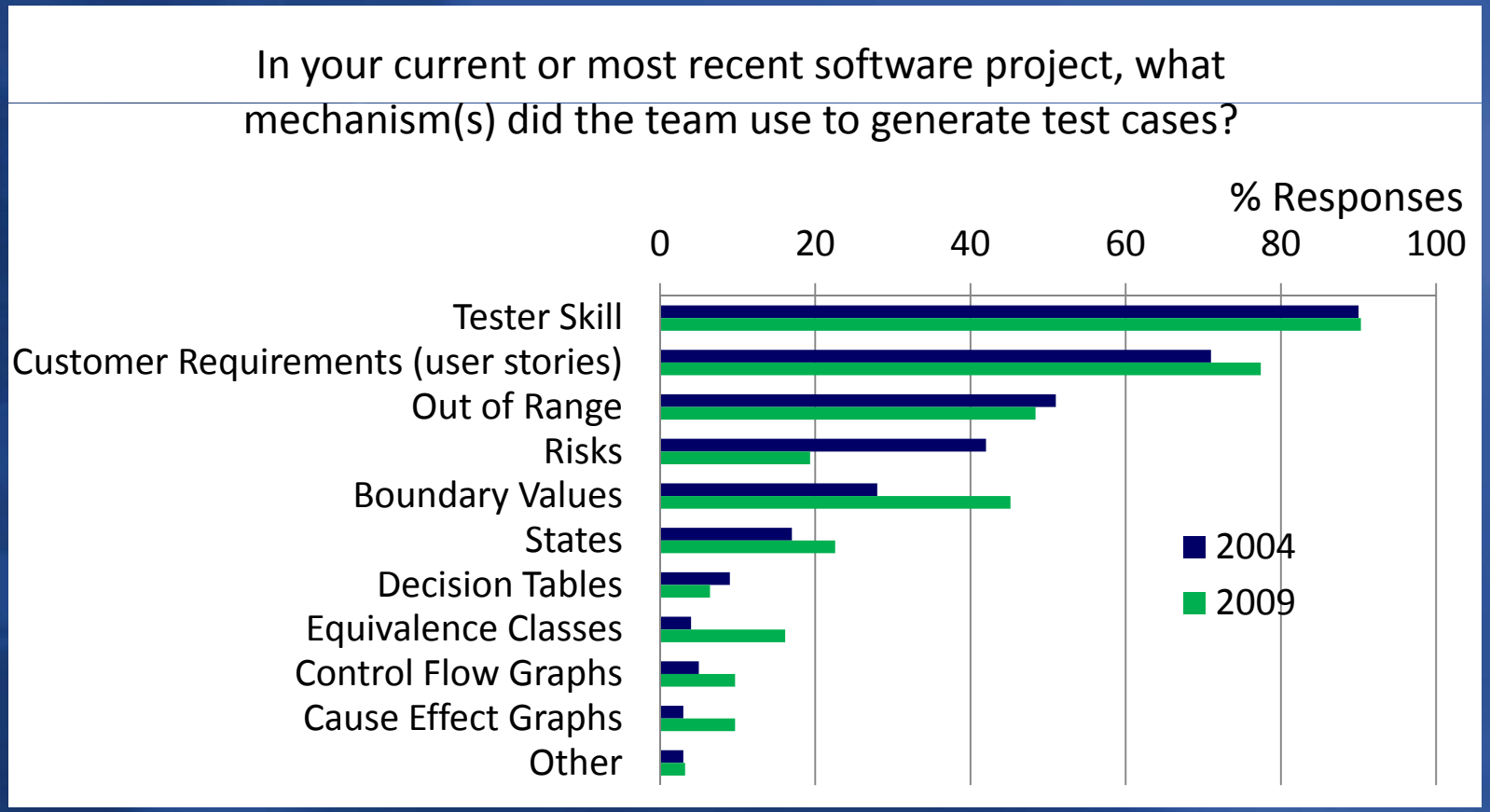
Test tools and frameworks

- JUnit and IBM Rational tools are the most widely used test tools.



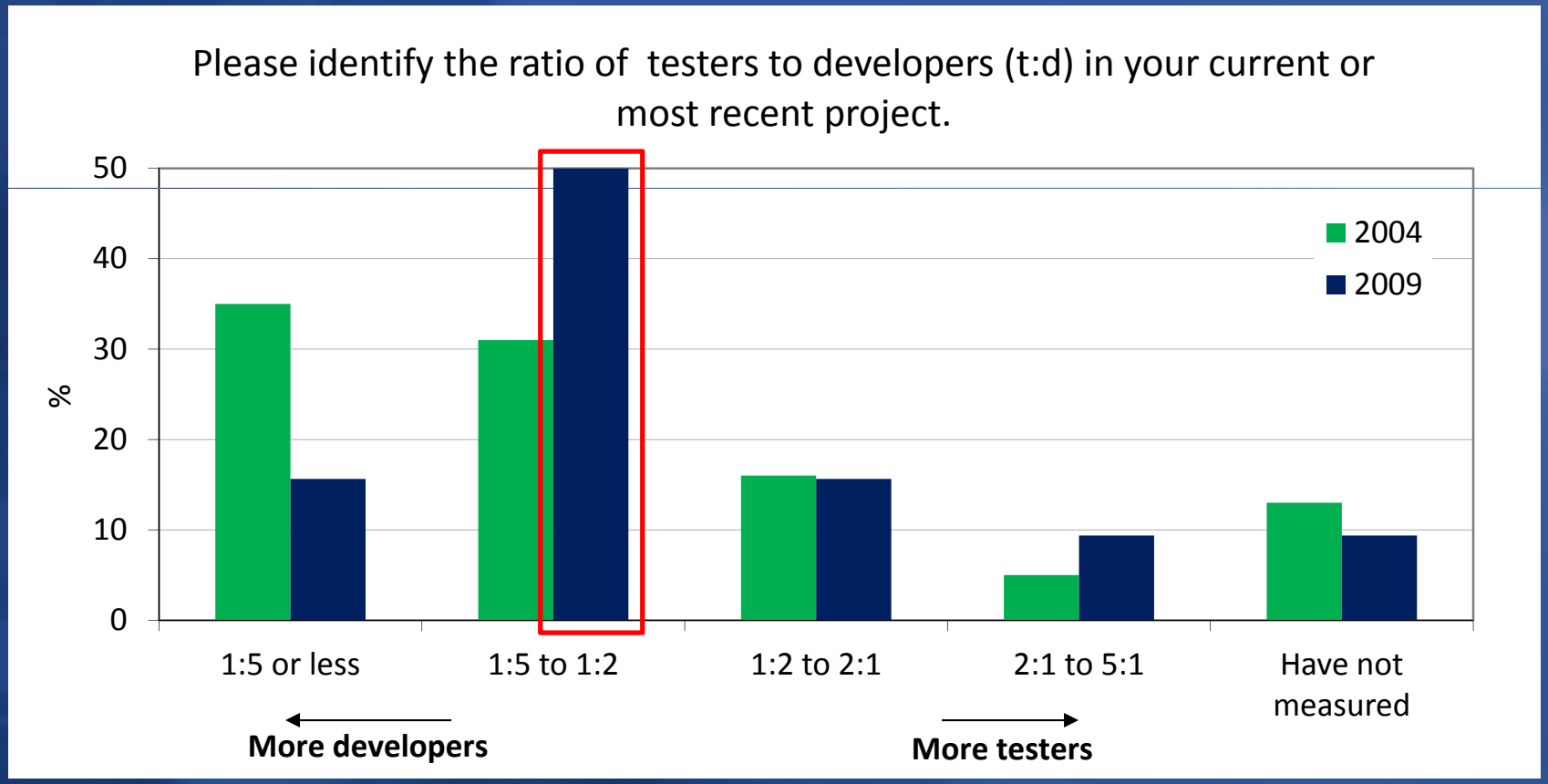
Surveys of SW testing practices in Alberta
Techniques for identifying test cases

- The choices of test-case generation mechanisms have not changed much.
- Systematic techniques are used less.



Surveys of SW testing practices in Alberta
Ratio of testers to developers (t:d)

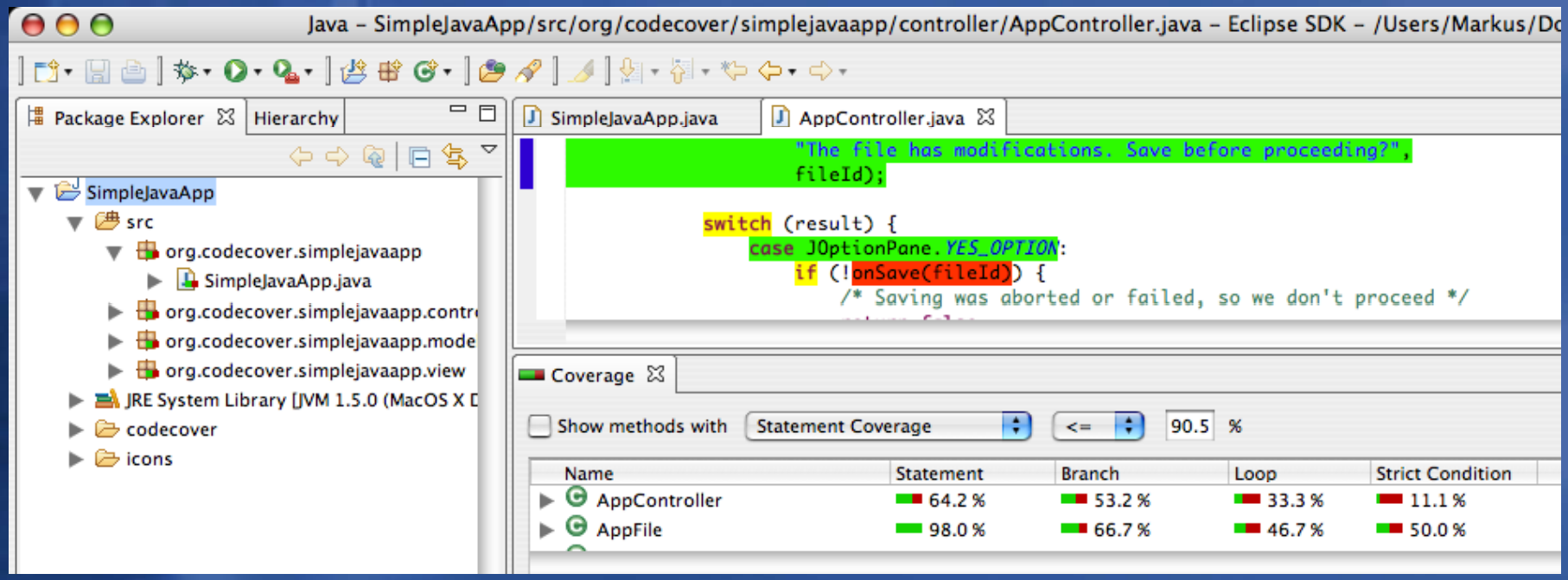
- In most companies, # of testers < # of developers



Projects Selected for Discussions Today

- Surveys of SW testing practices in Alberta
- **Improving SW testing and development by visualization of code coverage and traceability**
- Testing control software systems for the power industry

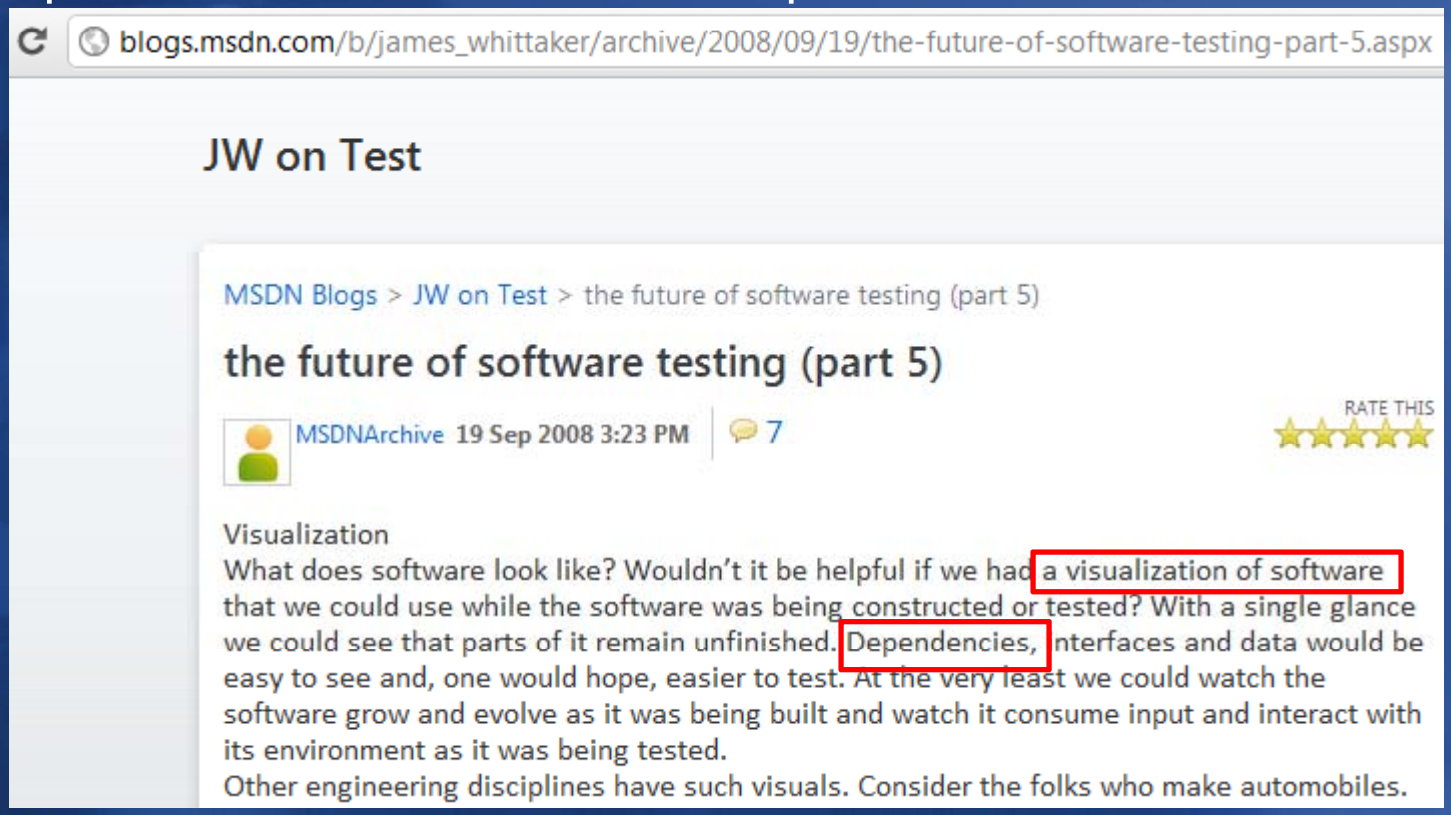
Can code (test) coverage be shown more visually?



- e.g., the CodeCover plug-in for the Eclipse IDE
- Conventionally, test coverage values are shown in percentages and are visualized by progress-bar-like green/red boxes
- But are they helpful for <u>all</u> development, testing and maintenance needs/tasks?

Can code (test) coverage be shown more visually?

- Increasing size and complexity of production and test code (e.g., JUnit)
- High-profile test engineers such as James Whittaker mention the need for test visualization explicitly
- Analysis of dependencies: If I change a function in the production code, which parts of the test code should I update?



blogs.msdn.com/b/james_whittaker/archive/2008/09/19/the-future-of-software-testing-part-5.aspx

JW on Test

MSDN Blogs > JW on Test > the future of software testing (part 5)

the future of software testing (part 5)

MSDNArchive 19 Sep 2008 3:23 PM 7

RATE THIS

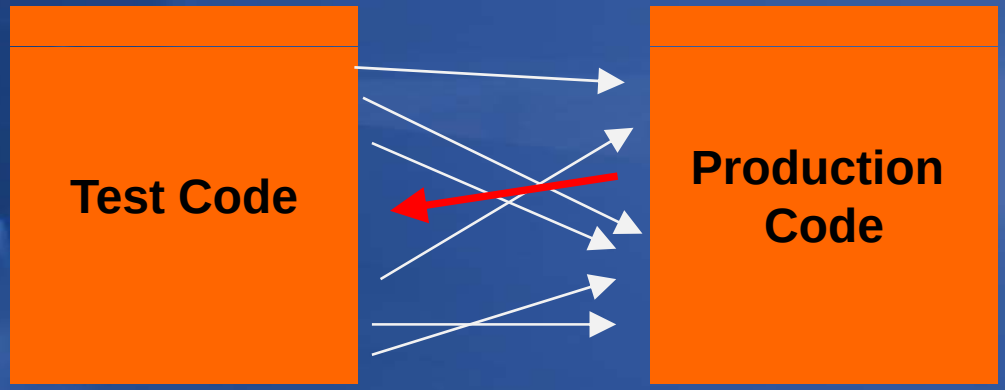
Visualization

What does software look like? Wouldn't it be helpful if we had a visualization of software that we could use while the software was being constructed or tested? With a single glance we could see that parts of it remain unfinished. Dependencies, interfaces and data would be easy to see and, one would hope, easier to test. At the very least we could watch the software grow and evolve as it was being built and watch it consume input and interact with its environment as it was being tested.

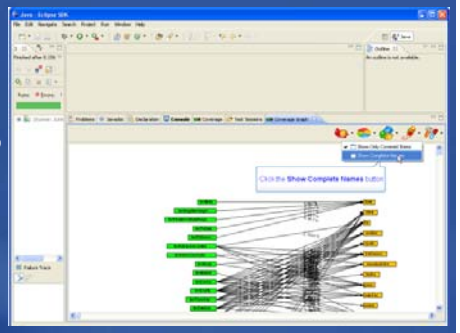
Other engineering disciplines have such visuals. Consider the folks who make automobiles.

Can code (test) coverage be shown more visually?

- We have also seen the need in discussions with testers from our partners such as IBM, and NovAtel
- Analysis of dependencies: If I change a function in the production code, which parts of the test code should I update?



- Existing code coverage tools help us with ⟶ , but not ⟶
- We have built an Eclipse plug-in for this purpose
- And have recently ported it to embedded systems
- Let's see a video demo:

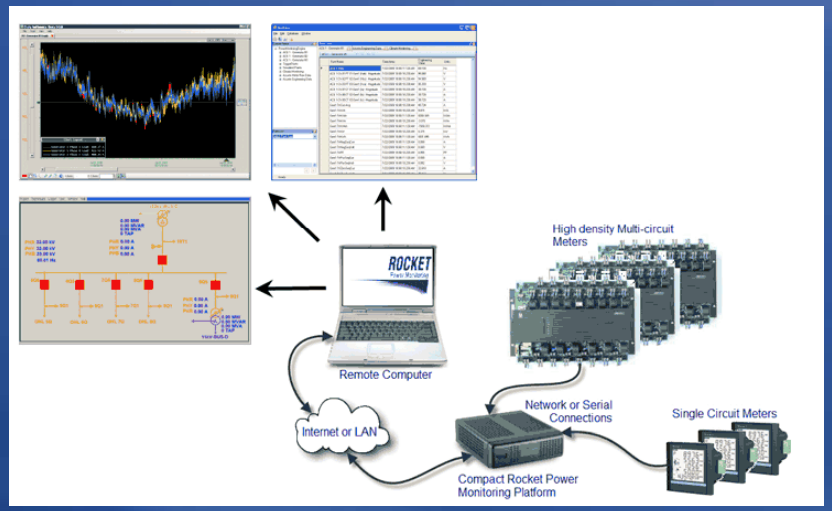


Projects Selected for Discussions Today

- Surveys of SW testing practices in Alberta
- Improving SW testing and development by visualization of code coverage
- Testing control software systems for the power industry

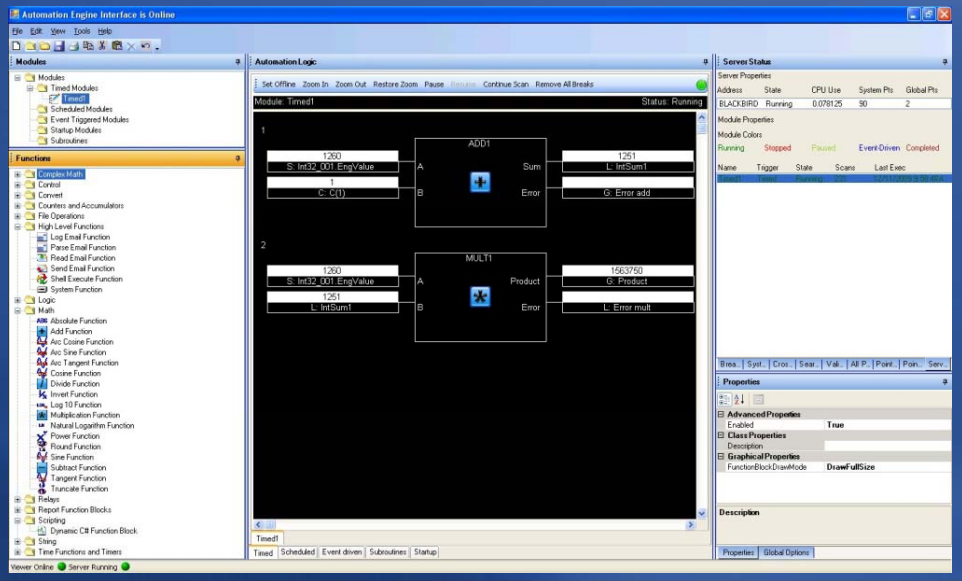
Testing control software systems for the power industry

- Software under test: A commercial large-scale Supervisory Control and Data Acquisition (SCADA) software system
- Is called *Rocket*
- Has been developed using Microsoft Visual Studio C#
- Developed using the iterative development process (but not strictly Agile)
- Has now been deployed in several locations across Canada and the US.



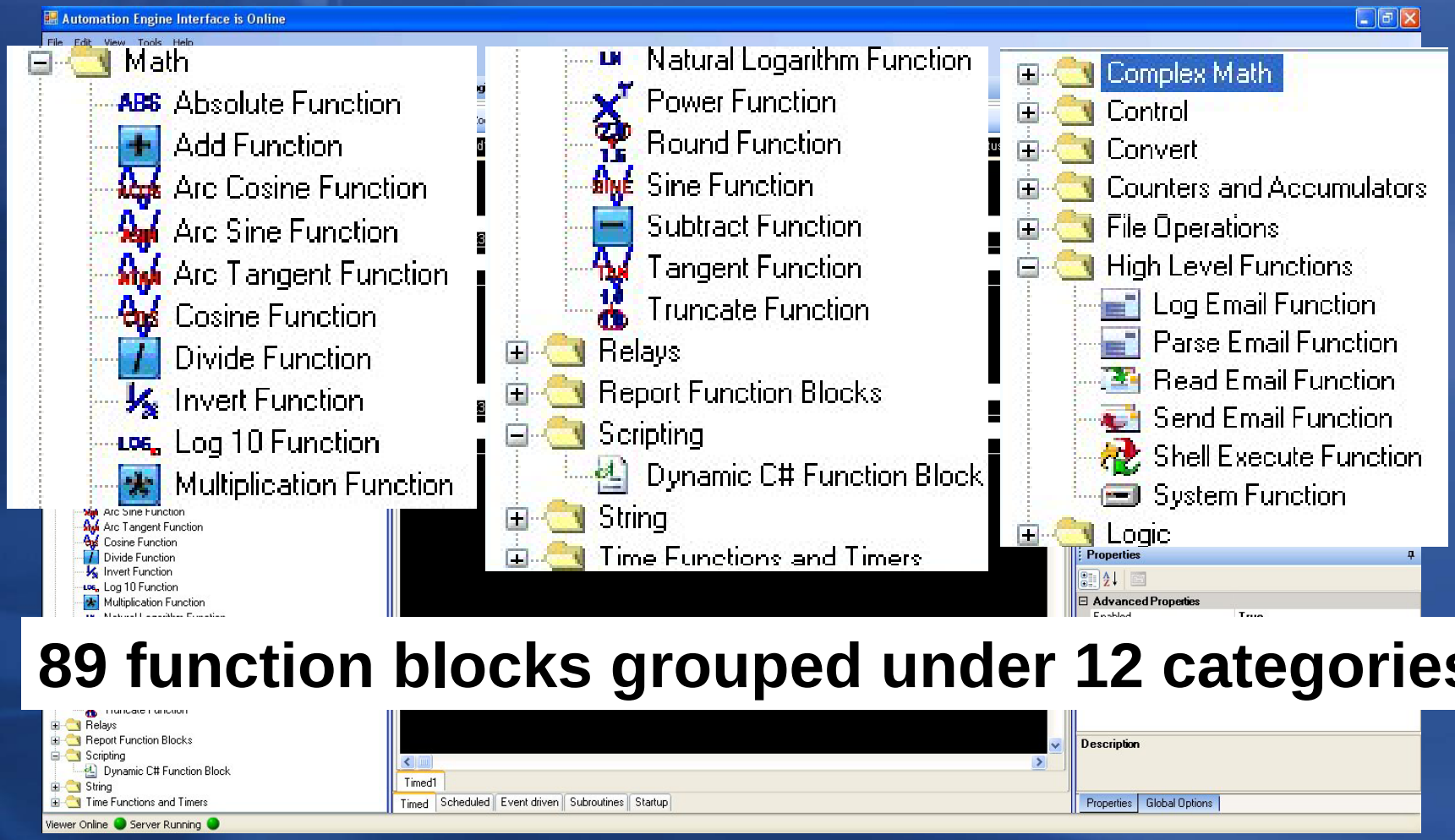
Testing a control software system

- The SUT has only been tested manually during its development iterations.
- Towards the end of the project, importance of automated testing was felt
- Thus, a collaboration between my team and the company
- Our goal: to conduct automated software testing on the *Rocket* system.
- We discuss the SUT next…



Our First Step: Black-box Unit Testing

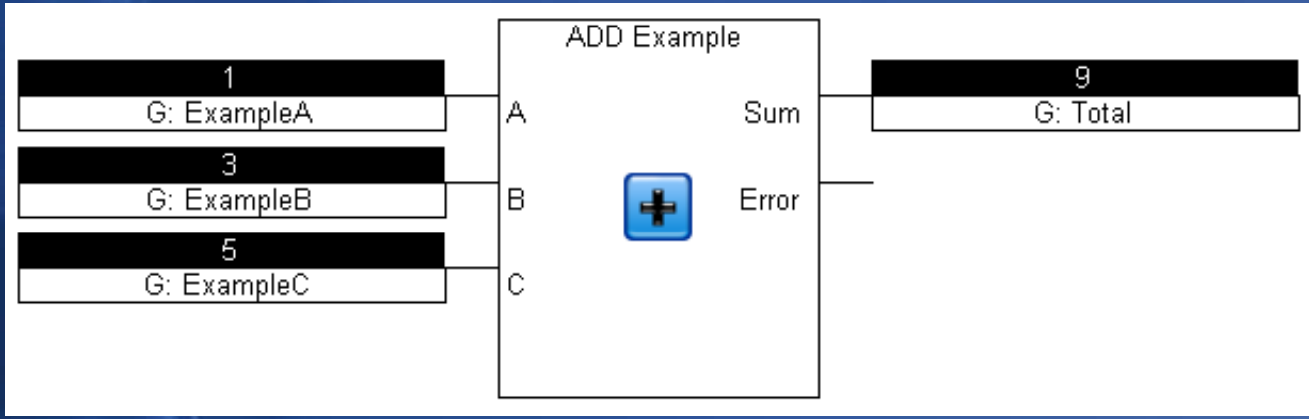
- Units Under Test



89 function blocks grouped under 12 categories

Black-box Unit Testing (BBUT): Challenges

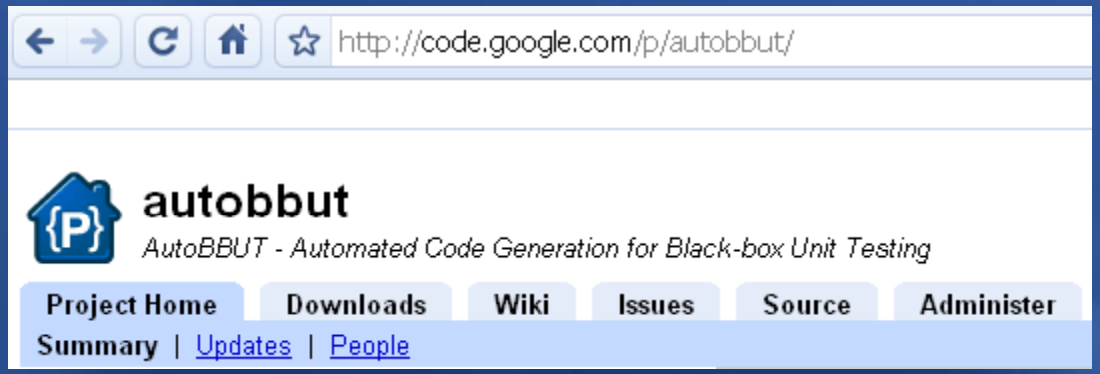
- The Add function block



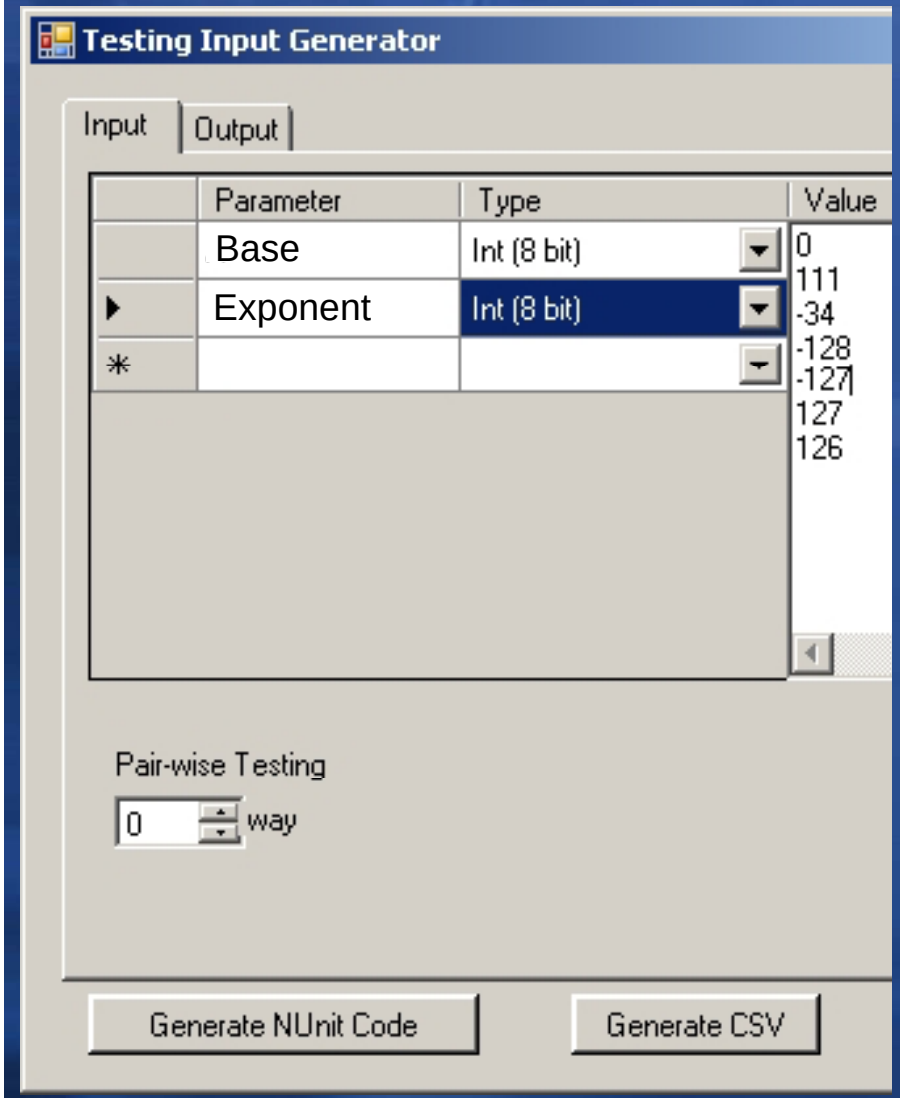
- If we apply the equivalence classing, we will get 19,683 test cases for only this function block. Bad news ;(
- Challenge 1: Coding of test cases (in NUnit): Too much effort
- Challenge 2: Coupling of test cases to test input data
- Challenge 3: Manual generation of expected outputs (test oracle)

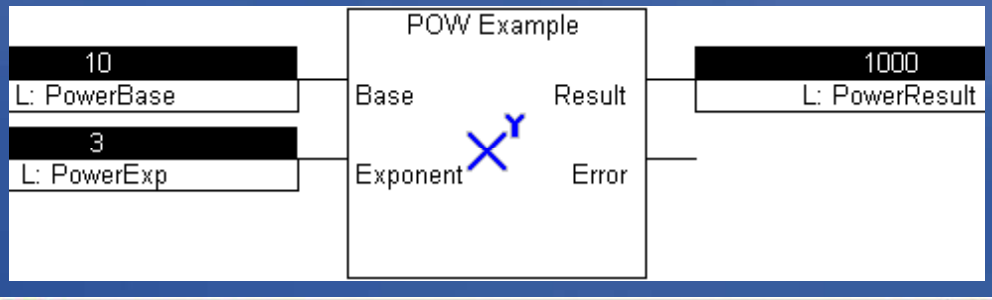
Black-box Unit Testing (BBUT): Challenges

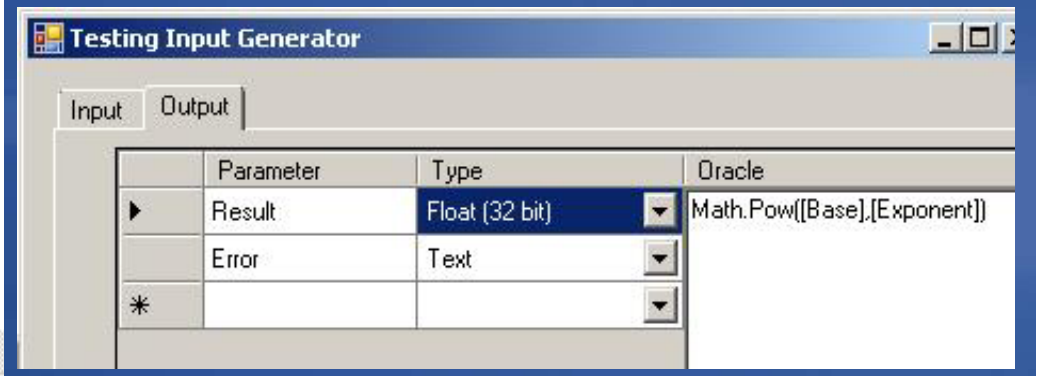
- One possible solution → Automated generation of NUnit test code
- There are some tools out there:
  - Microsoft Pex, JML-JUnit, JUB (JUnit test case Builder), TestGen4J, JCrasher, NModel
- After evaluating them for our purpose, unfortunately none was suitable (details in our article)
- Decision: to implement our own test tool!



AutoBBUT - GUI and Features







AutoBBUT - Example Usage

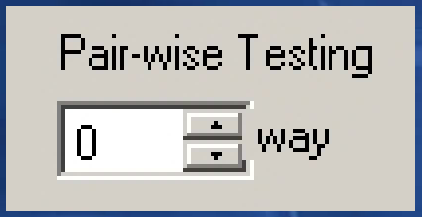
Reminder: Test code is automatically generated, saving many hours

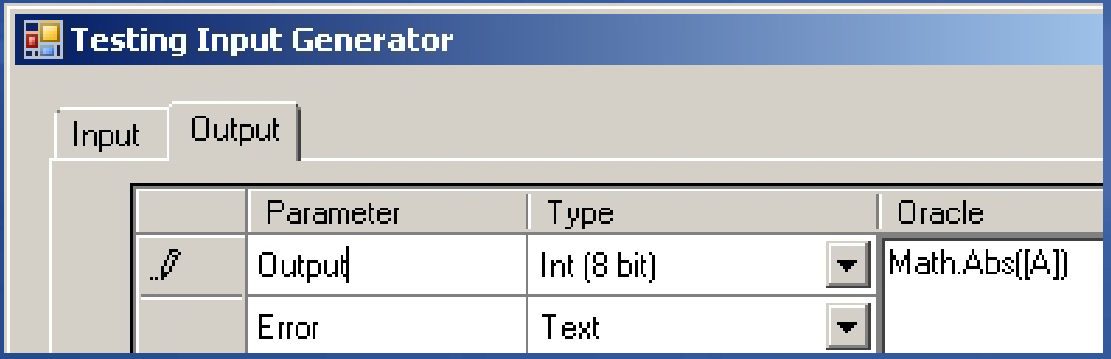
TestProject.PowerFBTest | Test61642af496604639a6bc3223c659c542()

```
[TestMethod]
public void Test61642af496604639a6bc3223c659c542()
{
    TD.setInputParameter(FunctionBlockName, "Base", "Int (8 bit)", "-128");
    TD.setInputParameter(FunctionBlockName, "Exponent", "Int (8 bit)", "-128");

    RocketParameter resultParam = TD.setOutputParameter(FunctionBlockName, "Result", "Float (32 bit)");
    RocketParameter errorParam = TD.setOutputParameter(FunctionBlockName, "Error", "Text");

    TD.execute(FunctionBlock, FunctionBlockName);

    Assert.AreEqual(float.Parse("1.8929E-270"), float.Parse(TD.getOutputByName(resultParam.PointName)), 0.0001);
    Assert.AreEqual("", TD.getOutputByName(errorParam.PointName));
}

[TestMethod]
public void Test6e134866ed144f3d9c2370bc20cda45f()
{
    TD.setInputParameter(FunctionBlockName, "Base", "Int (8 bit)", "-127");
    TD.setInputParameter(FunctionBlockName, "Exponent", "Int (8 bit)", "-127");

    RocketParameter resultParam = TD.setOutputParameter(FunctionBlockName, "Result", "Float (32 bit)");
    RocketParameter errorParam = TD.setOutputParameter(FunctionBlockName, "Error", "Text");

    TD.execute(FunctionBlock, FunctionBlockName);

    Assert.AreEqual(float.Parse("-6.5604E-268"), float.Parse(TD.getOutputByName(resultParam.PointName)), 0.0001);
    Assert.AreEqual("", TD.getOutputByName(errorParam.PointName));
}
```

AutoBBUT – Implementation Details

- Technologies (libraries) used…
- To generate all the n-way test cases, we used a recently-introduced Test API from Microsoft, called **`Microsoft.Test.VariationGeneration`**





- For the development of automated test oracle generation, we used a utility available in the .NET framework class library, called **`System.CodeDom.Compiler`**
  - (CodeDOM: Code Document Object Model)

AutoBBUT - Implementation Details

- Developed in C# .Net platform. 875 LOC.

InputGenerator.TestInputGenerator | generateTestSuiteCode()

```
// loop over all pairwise test cases
foreach (Variation v in combinatoryModel.GenerateVariations(pairwise_way, 1234))
{
    inputList = new Dictionary<string, RocketParameter>();
    Guid guid = Guid.NewGuid();
    // remove -'s from the random string
    string guidstring = Regex.Replace(guid.ToString(), "-", "");
    // start generating the test method one by one
    testSuiteCode += "[TestMethod]" + Environment.NewLine;
    // test method signature
    testSuiteCode += "public void Test" + guidstring + "()" +
        Environment.NewLine + "{" + Environment.NewLine;

    // feeding the input parameters in to the test code
    foreach (Parameter param in combinatoryModel.Parameters)
    {
        RocketParameter input = new RocketParameter();
        testSuiteCode += "TD.setInputParameter(FunctionBlockName,\"" + input.Name + "\", \"" +
            input.Type + "\", \"" + input.Value + "\" );" + Environment.NewLine;
        inputList.Add(input.Name, input);
    }
    testSuiteCode += Environment.NewLine;

    // feeding the output parameters in to the test code
    foreach (DataGridViewRow outputRow in outputGridView.Rows)
```

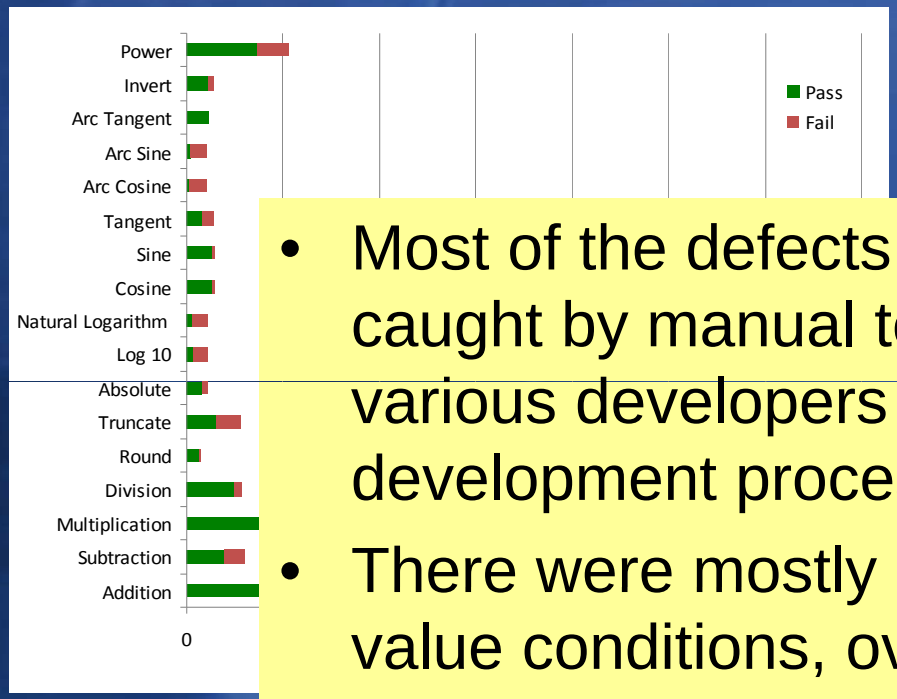
A Case Study - Using the AutoBBUT tool

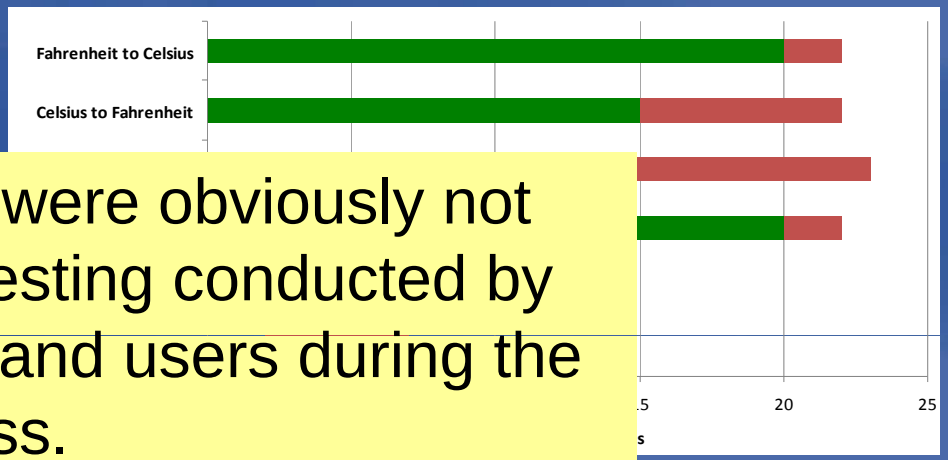
- We generated 1,962 NUnit test cases (methods) for automated black-box unit testing of 58 function blocks
- The total size of the NUnit test suite is currently 15,906 test LOC.

TestProject.PowerFBTest | Test61642af496604639a6bc3223c659c542()

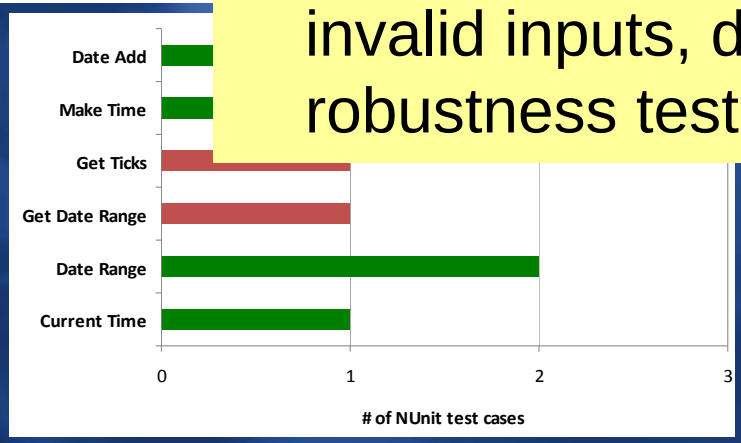
```
[TestMethod]
public void Test61642af496604639a6bc3223c659c542()
{
    TD.setInputParameter(FunctionBlockName, "Base", "Int (8 bit)", "-128");
    TD.setInputParameter(FunctionBlockName, "Exponent", "Int (8 bit)", "-128");

    RocketParameter resultParam = TD.setOutputParameter(FunctionBlockName, "Result", "Float (32 bit)");
    RocketParameter errorParam = TD.setOutputParameter(FunctionBlockName, "Error", "Text");

    TD.execute(FunctionBlock, FunctionBlockName);

    Assert.AreEqual(float.Parse("1.8929E-270"), float.Parse(TD.getOutputByName(resultParam.PointName)), 0.0001);
    Assert.AreEqual("", TD.getOutputByName(errorParam.PointName));
}
```

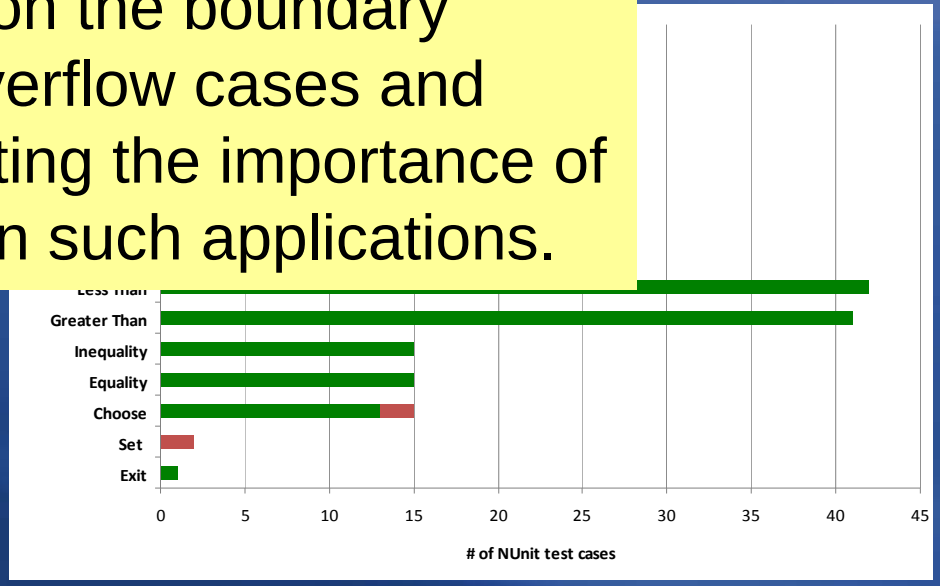
Effectiveness in Detecting Defects





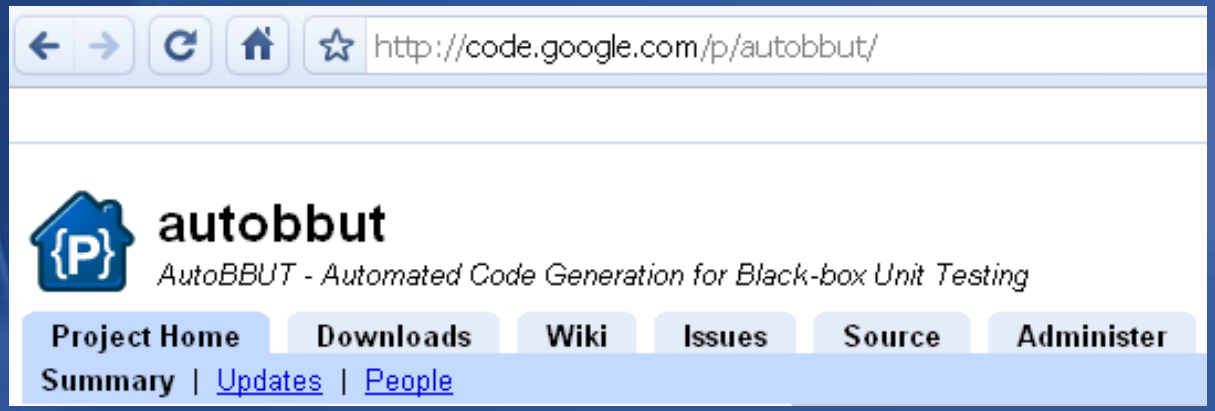
- Most of the defects were obviously not caught by manual testing conducted by various developers and users during the development process.
- There were mostly on the boundary value conditions, overflow cases and invalid inputs, denoting the importance of robustness testing in such applications.





AutoBBUT – Conclusions and Impact

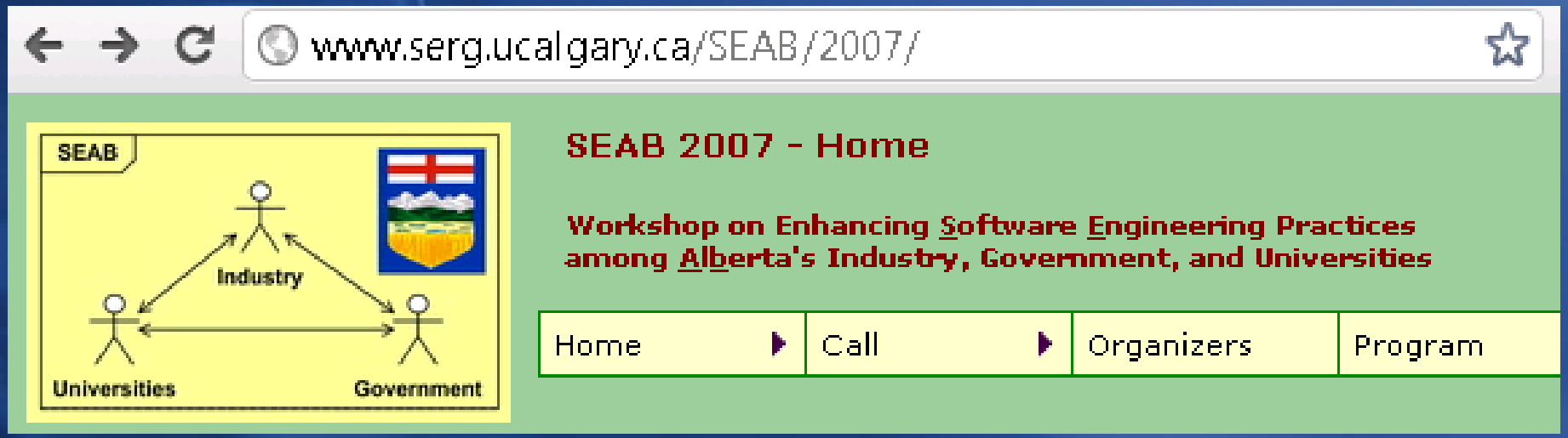
- An email from the MRCSI's CEO:
  - *"Many thanks for your efforts. I reviewed your [defect] report. It looks complete and clear. I and am very pleased with the results. We will include all identified bugs to the list and will try to address them."*



- Open source
- A lot of effort has been spent to have a clean design for it which makes it easily extensible and adaptable to other platforms (e.g., JUnit) by other testers.

Outline of the Talk

- Brief background about the speaker and his team in the UofC
- An overview of our SW testing projects
- Reviewing several selected projects
- **Wrap-up: What can be gained from collaborative university/industry software testing projects?**



Wrap-up: What can be gained from collaborative university/industry software testing projects?

- It is indeed a win-win case for both sides: companies and university teams

- Companies win: Have their software testing/quality issues addressed in least expen$ive way… Compare it to contractors or in-house testing teams. Opportunities to hire top-quality graduates
- University teams win: Learning and growth environment for graduate students and researchers

- We are very interested to talk to you folks about your testing needs

Outline of the Talk

- Brief background about the speaker and his team in the UofC
- An overview of our SW testing projects
- Reviewing several selected projects
- Wrap-up: What can be gained from collaborative university/industry software testing projects?

Q/A