Code Change Impact Analysis for Testing Configurable Software Systems

Mithun Acharya
ABB Corporate Research
Raleigh NC USA
ABB: A power and automation company

>125 years, >100 nations, ~150,000 employees

Power products and electronics, Control Systems, Robotics, Smart Grid, Renewable Energy, …
ABB Corporate Research
Industrial Software Systems (ISS) research group

7 research centers worldwide

USA
Germany
Switzerland
Poland
Sweden
India
China

Raleigh, NC
## Software in ABB

<table>
<thead>
<tr>
<th>Hardware with software inside</th>
<th>Software with few hardware components</th>
<th>Pure Software</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Robot Arm" /></td>
<td><img src="image2.png" alt="Diagram" /></td>
<td><img src="image3.png" alt="Map and Screen" /></td>
</tr>
</tbody>
</table>

Software with hardware components inside, few hardware components outside, and pure software.
Software Evolution: A CSS constantly changes

Hundreds of such changes committed daily
Change Control Board meetings

Change impact visualizations for managers for decision making

Change impact at the code level for developers

Unit/module-level change impact for testers
Imp: Code change impact analysis for C/C++ programs

Impact database

IMP
BUILD
SERVER

CHECKOUT
NIGHTLY BUILD

IMP

Impact of C23567
Impact of C23586
Impact of C23712
Impact of changes since last nightly build
...

Version control
SERVER

C23567
C23586
C23712

CLIENT

CLIENT

CLIENT

Version control
SERVER

CLIENT

CLIENT

CLIENT
Quantifiable risk/cost analysis of changes to CSS

Automated Dependency Analysis
Will changes to foo.c affect Bob’s module? Dependency analysis

Automated Risk/Cost Analysis
3 days to release!!! Should I implement this feature or bug fix?

Automated What-If Analysis
What is the ‘best’ way to fix this bug or implement that new feature?

Overlay change impact with risky areas in code

Automated Regression Testing
Should I re-run ALL of my test suite for this change? New tests required?

3941 lines vs. 6 lines
Program and System Dependence Graphs for Slicing

```c
void main() {
    int i = 1;
    int sum = 0;
    while (i<11) {
        sum = add(sum, i);
        i = add(i, 1);
    }
    printf("sum = %d\n", sum);
    printf("i = %d\n", i);
}

static int add(int a, int b) {
    return(a+b);
}
```

Program Dependence Graph (PDG) for `main`

System Dependence Graph (SDG)

Code/Image Source: GrammaTech
Making impact analysis practical and useful
Making impact analysis practical and useful


Scaling beyond million lines
Making impact analysis practical and useful


Xiao Qu, Mithun Acharya, Brian Robinson. *Impact Analysis of Configuration Changes for Test Case Selection*. ISSRE 2011

- Scaling beyond million lines
- Testing configurable systems
Making impact analysis practical and useful


Scaling beyond million lines


Testing configurable systems


Xiao Qu, Mithun Acharya, Brian Robinson. *Impact Analysis of Configuration Changes for Test Case Selection*. ISSRE 2011

Regression test selection

Tingting Yu, Xiao Qu, Mithun Achayra, Gregg Rothermel. *Oracle-Based Regression Test Selection*. Under submission.
What configurations should we select for retesting?


Xiao Qu, Mithun Acharya, Brian Robinson. *Impact Analysis of Configuration Changes for Test Case Selection*. ISSRE 2011

Tingting Yu, Xiao Qu, Mithun Acharya, Gregg Rothermel. *Oracle-Based Regression Test Selection*. Under submission.
Regression testing of CSS with code change impact analysis

Automated Dependency Analysis
Will changes to foo.c, affect Bob’s module? Dependency analysis

Automated Risk/Cost Analysis
3 days to release!!! Should I implement this feature or bug fix?

Automated What-If Analysis
What is the ‘best’ way to fix this bug or implement that new feature?

Overlay change impact with risky areas in code

Automated Regression Testing
Should I re-run ALL of my test suite for this change? New tests required?

Test suite
Outline

Motivation

Approach

Implementation

Empirical Evaluation

Conclusions
Configurable Software Systems

- Software that can be customized through a set of options
- Example: Internet Explorer
Configurable Software Systems

- Software that can be customized through a set of options
- Example: Internet Explorer

Configurable option: “Pop-up Blocker”

Values: ON, OFF
## Internet Explorer Configurations

<table>
<thead>
<tr>
<th></th>
<th>C₁</th>
<th>C₂</th>
<th>…</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pop Up Blocker</td>
<td>ON</td>
<td>ON</td>
<td>…</td>
</tr>
<tr>
<td>Google Toolbar</td>
<td>Disabled</td>
<td>Enabled</td>
<td>…</td>
</tr>
<tr>
<td>Do Not Track</td>
<td>Yes</td>
<td>Yes</td>
<td>…</td>
</tr>
<tr>
<td></td>
<td>…</td>
<td>…</td>
<td>…</td>
</tr>
</tbody>
</table>
**Internet Explorer Configurations**

<table>
<thead>
<tr>
<th>Option</th>
<th>C₁</th>
<th>C₂</th>
<th>…</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pop Up Blocker</td>
<td>ON</td>
<td>ON</td>
<td>…</td>
</tr>
<tr>
<td>Google Toolbar</td>
<td>Disabled</td>
<td>Enabled</td>
<td>…</td>
</tr>
<tr>
<td>Do Not Track</td>
<td>Yes</td>
<td>Yes</td>
<td>…</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
</tr>
<tr>
<td>Option</td>
<td>Value</td>
<td></td>
<td></td>
</tr>
<tr>
<td>------------------------</td>
<td>--------</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>C₁</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>C₂</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Pop Up Blocker</strong></td>
<td>ON</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Google Toolbar</strong></td>
<td>Disabled</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Do Not Track</strong></td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Internet Explorer Configurations

<table>
<thead>
<tr>
<th>Option</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pop Up Blocker</td>
<td>ON, ON</td>
</tr>
<tr>
<td>Google Toolbar</td>
<td>Disabled, Enabled</td>
</tr>
<tr>
<td>Do Not Track</td>
<td>Yes, Yes</td>
</tr>
</tbody>
</table>

Configuration instance $C_1 = \{\text{ON, Disabled, Yes, \ldots}\}$
# Internet Explorer Configurations

<table>
<thead>
<tr>
<th>Option</th>
<th>Value</th>
<th>C₁</th>
<th>C₂</th>
<th>…</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pop Up Blocker</td>
<td>ON</td>
<td>ON</td>
<td>ON</td>
<td>…</td>
</tr>
<tr>
<td>Google Toolbar</td>
<td>Disabled</td>
<td>Enabled</td>
<td></td>
<td>…</td>
</tr>
<tr>
<td>Do Not Track</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>…</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
</tr>
</tbody>
</table>

Configuration instance $C₁ = \{\text{ON, Disabled, Yes, …}\}$

Configuration $C₁$
I have discovered that using the newly released IE 7 with Google Toolbar = Enabled can cause the right-click menu to lose the "Open In New Tab" option.
Impact of Configuration on System Behavior

To fix this, open IE7, click “Tools > Manage Add-ons > Disable Google Toolbar”
Test Case: Open IE7, Right Click a link on webpage

<table>
<thead>
<tr>
<th></th>
<th>C₁</th>
<th>C₂</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pop Up Blocker</td>
<td>ON</td>
<td>ON</td>
</tr>
<tr>
<td>Google Toolbar</td>
<td>Disabled</td>
<td>Enabled</td>
</tr>
<tr>
<td>Do Not Track</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

...
Test Case: Open IE7, Right Click a link on webpage

<table>
<thead>
<tr>
<th></th>
<th>C₁</th>
<th>C₂</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pop Up Blocker</td>
<td>ON</td>
<td>ON</td>
</tr>
<tr>
<td>Google Toolbar</td>
<td>Disabled</td>
<td>Enabled</td>
</tr>
<tr>
<td>Do Not Track</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
<td>…</td>
</tr>
</tbody>
</table>

Test: PASS
## Test Case: Open IE7, Right Click a link on webpage

<table>
<thead>
<tr>
<th></th>
<th>C₁</th>
<th>C₂</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Pop Up Blocker</strong></td>
<td>ON</td>
<td>ON</td>
</tr>
<tr>
<td><strong>Google Toolbar</strong></td>
<td>Disabled</td>
<td>Enabled</td>
</tr>
<tr>
<td><strong>Do Not Track</strong></td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>…</strong></td>
<td>…</td>
<td>…</td>
</tr>
</tbody>
</table>
Test Case: Open IE7, Right Click a link on webpage

<table>
<thead>
<tr>
<th></th>
<th>C₁</th>
<th>C₂</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pop Up Blocker</td>
<td>ON</td>
<td>ON</td>
</tr>
<tr>
<td>Google Toolbar</td>
<td>Disabled</td>
<td>Enabled</td>
</tr>
<tr>
<td>Do Not Track</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

A test case that passes with one configuration may fail with another
Configurations control system execution
Configurations control system execution

\[ C_1 = \{ \text{ON, disabled, Yes, ...} \} \]
Configurations control system execution

C₁ = {ON, disabled, Yes, …}
Configurations control system execution

Test: PASS

\[ C_1 = \{ \text{ON, disabled, Yes, …} \} \]

\[ C_2 = \{ \text{ON, enabled, Yes, …} \} \]
Configurations control system execution

Test: PASS

C₁ = {ON, disabled, Yes, …}

Test: FAIL

C₂ = {ON, enabled, Yes, …}
Configurations control system execution

Test: PASS

IE7

Test: FAIL

IE7

C₁ = {ON, disabled, Yes, …}

C₂ = {ON, enabled, Yes, …}

Can we statically approximate how configurations (options) control system execution?
Challenges for testing configurable systems
Challenges for testing configurable systems

n options $\rightarrow$ $2^n$ configurations
Challenges for testing configurable systems

\[ n \text{ options} \rightarrow 2^n \text{ configurations} \]

Product evolves

IE5

IE7
Challenges for testing configurable systems

- \( n \) options
- \( 2^n \) configurations

Product evolves from IE5 to IE7

- \( T = \{t_1, t_2, t_3, t_4, t_5\} \)
- \( T' = \{t_2, t_5\} \)

Test case selection
Challenges for testing configurable systems

- n options → 2^n configurations
- Product evolves
- Test case selection
- Configuration selection

T = \{t_1, t_2, t_3, t_4, t_5\} → T' = \{t_2, t_5\}

{C_1, C_2, C_3, C_4, C_5} → {C_1, C_5}
Configuration Sampling
Reducing the exponential number of configurations to a manageable size
Configuration Sampling
Reducing the exponential number of configurations to a manageable size

\[ \{C_1, C_2, C_3, C_4, C_5, \ldots\} \]
Exponentially large set

\[ T = \{t_1, t_2, t_3, t_4, t_5\} \]
Configuration Sampling
Reducing the exponential number of configurations to a manageable size

\[ \{C_1, C_2, C_3, C_4, C_5, \ldots\} \quad \text{Exponentially large set} \]

\[ T = \{t_1, t_2, t_3, t_4, t_5\} \]

\[ \text{Sampling} \]

\[ \{C_1, C_3, C_4\} \quad \text{Manageable size set} \]
Configuration Sampling
Reducing the exponential number of configurations to a manageable size

\[ \{C_1, C_2, C_3, C_4, C_5, \ldots\} \]
Exponentially large set

\[ T = \{t_1, t_2, t_3, t_4, t_5\} \]
No test case selection

Sampling

\[ \{C_1, C_3, C_4\} \]
Manageable size set

\[ T' = \{t_1, t_2, t_3, t_4, t_5\} \]
Configuration Sampling
Reducing the exponential number of configurations to a manageable size

Example: Configuration Interaction Testing (CIT)
Configuration Sampling
Reducing the exponential number of configurations to a manageable size

Exponentially large set

Manageable size set

T = \{t_1, t_2, t_3, t_4, t_5\}
No test case selection
T' = \{t_1, t_2, t_3, t_4, t_5\}

Example: Configuration Interaction Testing (CIT)

We choose to test IE7 only under sampled configurations C_1, C_3, and C_4
and for each configurations we test IE7 with all tests \{t_1, t_2, t_3, t_4, t_5\}
vim: A configurable system
**vim**: A configurable system

$2^{90}$ configurations
vim: A configurable system

$2^{90}$ configurations $\rightarrow$ sampling $\rightarrow$ CIT selects 60 configurations
vim: A configurable system

$2^{90}$ configurations $\rightarrow$ sampling $\rightarrow$ CIT selects 60 configurations

Rerun the full test suite on each 60 configurations
**vim: A configurable system**

- $2^{90}$ configurations
- Sampling
- CIT selects 60 configurations

Rerun the full test suite on each 60 configurations

7 hours to execute the full test suite
Takes $7 \times 60 = 420$ hours (~2.5 weeks) to run all test cases under each configuration
Vim: A configurable system

2^{90} configurations → sampling → CIT selects 60 configurations

Rerun the full test suite on each 60 configurations

7 hours to execute the full test suite
Takes 7*60 = 420 hours (~2.5 weeks) to run all test cases under each configuration

Do we have to run all tests under each configuration?
Test case selection when configuration under test changes*

* Qu, Acharya, Robinson, “Impact analysis of configuration changes for test case selection”, ISSRE 2011
Test case selection when configuration under test changes*

* Qu, Acharya, Robinson, “Impact analysis of configuration changes for test case selection”, ISSRE 2011
Test case selection when configuration under test changes*

Source code DOES NOT change

Configuration under test changes

\[ C_1 = \{\text{ON, Disabled, Yes, \ldots}\} \quad C_2 = \{\text{ON, Enabled, Yes, \ldots}\} \]

* Qu, Acharya, Robinson, “Impact analysis of configuration changes for test case selection”, ISSRE 2011
Test case selection when configuration under test changes*

Source code DOES NOT change

Configuration under test changes

$C_1 = \{\text{ON, Disabled, Yes, …}\} \quad C_2 = \{\text{ON, Enabled, Yes, …}\}$

What test cases should I re-run for the new configuration?

$T = \{t_1, t_2, t_3, t_4, t_5\}$

* Qu, Acharya, Robinson, “Impact analysis of configuration changes for test case selection”, ISSRE 2011
Test case selection when configuration under test changes*

Source code DOES NOT change

Configuration under test changes

\[ C_1 = \{\text{ON, Disabled, Yes, …}\} \quad C_2 = \{\text{ON, Enabled, Yes, …}\} \]

What test cases should I re-run for the new configuration?

\[ T = \{t_1, t_2, t_3, t_4, t_5\} \quad T' = \{t_2, t_5\} \]

* Qu, Acharya, Robinson, “Impact analysis of configuration changes for test case selection”, ISSRE 2011
Test case selection when configuration under test changes*

Source code DOES NOT change

![IE7](image1)

![IE7](image2)

Configuration under test changes

\[ C_1 = \{ \text{ON, Disabled, Yes, …} \} \quad \rightarrow \quad C_2 = \{ \text{ON, Enabled, Yes, …} \} \]

What test cases should I re-run for the new configuration?

\[ T = \{ t_1, t_2, t_3, t_4, t_5 \} \quad \rightarrow \quad T' = \{ t_2, t_5 \} \]

For the ABB system analyzed, only about 20% of the tests had to be re-run for a configuration change

* Qu, Acharya, Robinson, “Impact analysis of configuration changes for test case selection”, ISSRE 2011
Product Evolution

Internet Explorer

IE1.0 & 2.0
IE5
IE6
IE7
Configuration prioritization for regression testing*

*Qu, Cohen, Rothermel, “Configuration-aware regression testing: An empirical study of sampling and prioritization”, ISSTA 2008
Configuration prioritization for regression testing*

\[ \{C_1, C_2, C_3, C_4, C_5\} \]

\[ T = \{t_1, t_2, t_3, t_4, t_5\} \]

*Qu, Cohen, Rothermel, “Configuration-aware regression testing: An empirical study of sampling and prioritization”, ISSTA 2008
Configuration prioritization for regression testing*

Source code CHANGES

\[\{C_1, C_2, C_3, C_4, C_5\}\]

\[T = \{t_1, t_2, t_3, t_4, t_5\}\]

*Qu, Cohen, Rothermel, “Configuration-aware regression testing: An empirical study of sampling and prioritization”, ISSTA 2008
Configuration prioritization for regression testing*

Source code CHANGES

IE5

IE7

\{C_1, C_2, C_3, C_4, C_5\} \rightarrow \text{Prioritization} \rightarrow \{C_1, C_4, C_3, \ldots\}

T = \{t_1, t_2, t_3, t_4, t_5\}

*Qu, Cohen, Rothermel, “Configuration-aware regression testing: An empirical study of sampling and prioritization”, ISSTA 2008
Configuration prioritization for regression testing*

{C₁, C₂, C₃, C₄, C₅} \rightarrow \text{Prioritization} \rightarrow \{C₁, C₄, C₃, ...\}

Reorder and run as many as you can

\[ T = \{t₁, t₂, t₃, t₄, t₅\} \]

*Qu, Cohen, Rothermel, “Configuration-aware regression testing: An empirical study of sampling and prioritization”, ISSTA 2008
Configuration prioritization for regression testing*

Source code CHANGES

IE5

{C₁, C₂, C₃, C₄, C₅} → Prioritization → {C₁, C₄, C₃, ...}

Reorder and run as many as you can

T = {t₁, t₂, t₃, t₄, t₅} No test case selection T' = {t₁, t₂, t₃, t₄, t₅}

*Qu, Cohen, Rothermel, “Configuration-aware regression testing: An empirical study of sampling and prioritization”, ISSTA 2008
Configuration prioritization for regression testing*

Source code CHANGES

IE5

IE7

$\{C_1, C_2, C_3, C_4, C_5\}$ → Prioritization → $\{C_1, C_4, C_3, \ldots\}$

Reorder and run as many as you can

$T = \{t_1, t_2, t_3, t_4, t_5\}$  No test case selection  $T' = \{t_1, t_2, t_3, t_4, t_5\}$

Increases rate of fault detection. But…

*Qu, Cohen, Rothermel, “Configuration-aware regression testing: An empirical study of sampling and prioritization”, ISSTA 2008
Configuration prioritization for regression testing*

Source code CHANGES

IE5

{C₁, C₂, C₃, C₄, C₅} ← Prioritization → {C₁, C₄, C₃, ...}

T = {t₁, t₂, t₃, t₄, t₅} No test case selection T' = {t₁, t₂, t₃, t₄, t₅}

Increases rate of fault detection. But…

Does not eliminate redundancy.
Does not detect all faults. Not safe.

*Qu, Cohen, Rothermel, “Configuration-aware regression testing: An empirical study of sampling and prioritization”, ISSTA 2008
Configuration prioritization for regression testing*

Source code CHANGES

{C₁, C₂, C₃, C₄, C₅} \rightarrow \text{Prioritization} \rightarrow \{C₁, C₄, C₃, ...\}

Reorder and run as many as you can

T = \{t₁, t₂, t₃, t₄, t₅\} \quad \text{No test case selection} \quad T' = \{t₁, t₂, t₃, t₄, t₅\}

Increases rate of fault detection. But...

Does not eliminate redundancy.
Does not detect all faults. Not safe.

Can we select a subset of \{C₁, C₂, C₃, C₄, C₅\} that is both non-redundant and safe?

*Qu, Cohen, Rothermel, “Configuration-aware regression testing: An empirical study of sampling and prioritization”, ISSTA 2008
Configuration selection for regression testing (Focus of this talk)
Configuration selection for regression testing (Focus of this talk)

{C₁, C₂, C₃, C₄, C₅}

T = {t₁, t₂, t₃, t₄, t₅}
Configuration selection for regression testing (Focus of this talk)

Source code CHANGES

\{C_1, C_2, C_3, C_4, C_5\}

\[T = \{t_1, t_2, t_3, t_4, t_5\}\]
Configuration selection for regression testing (Focus of this talk)

Source code CHANGES

IE5  
{C₁, C₂, C₃, C₄, C₅} → Selection → {C₁, C₅}

T = {t₁, t₂, t₃, t₄, t₅}
Configuration selection for regression testing (Focus of this talk)

Source code CHANGES

IE5

{C₁, C₂, C₃, C₄, C₅} → Selection → {C₁, C₅}

T = {t₁, t₂, t₃, t₄, t₅}  No test case selection  T' = {t₁, t₂, t₃, t₄, t₅}

IE7
Configuration selection for regression testing (Focus of this talk)

Source code CHANGES

IE5

IE7

\{C_1, C_2, C_3, C_4, C_5\} \rightarrow \text{Selection} \rightarrow \{C_1, C_5\}

T = \{t_1, t_2, t_3, t_4, t_5\} \quad \text{No test case selection} \quad T' = \{t_1, t_2, t_3, t_4, t_5\}

\{C_1, C_5\} \text{ is both safe (wrt retest-all configurations) and non redundant}
State of the Art in Configurable System Testing

<table>
<thead>
<tr>
<th>Problems</th>
<th>Single Version Testing</th>
<th>Regression Testing</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Configuration Level</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Selection</td>
<td>[11][15][18][25]</td>
<td>[18]</td>
</tr>
<tr>
<td>Prioritization</td>
<td>NA</td>
<td>[18]</td>
</tr>
<tr>
<td><strong>Test Case Level</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Selection</td>
<td>[17]</td>
<td>[22][23]^a</td>
</tr>
<tr>
<td>Prioritization</td>
<td>NA</td>
<td>[21]^a</td>
</tr>
</tbody>
</table>

*Focus of this talk*
State of the Art in Configurable System Testing

- Configuration sampling
- Single version
- No test case selection
- Example, CIT

**TABLE I. THE STATE OF THE ART IN CONFIGURABLE SYSTEM TESTING**

<table>
<thead>
<tr>
<th>Problems</th>
<th>Single Version Testing</th>
<th>Regression Testing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuration Level</td>
<td>Selection</td>
<td>[11][15][18][25]</td>
</tr>
<tr>
<td>Prioritization</td>
<td>NA</td>
<td>[18]</td>
</tr>
<tr>
<td>Test Case Level</td>
<td>Selection</td>
<td>[17]</td>
</tr>
<tr>
<td>Prioritization</td>
<td>NA</td>
<td>[22][23]</td>
</tr>
</tbody>
</table>

**Focus of this talk**
State of the Art in Configurable System Testing

- Configuration sampling
- Single version
- No test case selection
- Example, CIT

Test case selection [ISSRE ‘11]
- Single version
- Configuration under test changes
- Non-redundant
- Safe
### State of the Art in Configurable System Testing

- Configuration sampling
- Single version
- No test case selection
- Example, CIT

**TABLE I. THE STATE OF THE ART IN CONFIGURABLE SYSTEM TESTING**

<table>
<thead>
<tr>
<th>Problems</th>
<th>Single Version Testing</th>
<th>Regression Testing</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Configuration Level</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Selection</td>
<td>[11][15][18][25]</td>
<td></td>
</tr>
<tr>
<td>Prioritization</td>
<td>NA</td>
<td>[18]</td>
</tr>
<tr>
<td><strong>Test Case Level</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Selection</td>
<td>[17]</td>
<td>[22][23]</td>
</tr>
<tr>
<td>Prioritization</td>
<td>NA</td>
<td>[21]</td>
</tr>
</tbody>
</table>

- Test case selection [ISSRE ‘11]
- Single version
- Configuration under test changes
- Non-redundant
- Safe

- Configuration prioritization [ISSTA ‘08]
- Source code changes
- Regression Testing
- No test case selection
- Redundant
- Not safe

Focus of this talk
State of the Art in Configurable System Testing

- Configuration sampling
- Single version
- No test case selection
- Example, CIT

**TABLE I. THE STATE OF THE ART IN CONFIGURABLE SYSTEM TESTING**

<table>
<thead>
<tr>
<th>Problems</th>
<th>Single Version Testing</th>
<th>Regression Testing</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Configuration Level</strong></td>
<td>Selection</td>
<td>[11][15][18][25]</td>
</tr>
<tr>
<td>Prioritization</td>
<td>NA</td>
<td>[18]</td>
</tr>
<tr>
<td><strong>Test Case Level</strong></td>
<td>Selection</td>
<td>[17]</td>
</tr>
<tr>
<td>Prioritization</td>
<td>NA</td>
<td>[22][23]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[21]</td>
</tr>
</tbody>
</table>

- Configuration selection [ICSM ‘12]
- Source code changes
- **Regression testing**
- Non-redundant
- Safe
- No test case selection

- Test case selection [ISSRE ‘11]
- Single version
- Configuration under test changes
- Non-redundant
- Safe

- Configuration prioritization [ISSTA ‘08]
- Source code changes
- Regression Testing
- No test case selection
- Redundant
- Not safe
Key Idea: Map configuration options to code
Key Idea: Map configuration options to code

Configuration options: \{pop-up-blocker, Google Toolbar, Do Not Track\}
Key Idea: Map configuration options to code

Configuration options: \{pop-up-blocker, Google Toolbar, Do Not Track\}
Key Idea: Map configuration options to code

Configuration options: \{pop-up-blocker, Google Toolbar, Do Not Track\}
Key Idea: Map configuration options to code

Configuration options: {pop-up-blocker, Google Toolbar, Do Not Track}

For ABB systems, configurable options (stored in a DB) maps to variables in the source code
Key Idea: statically compute configuration option impact

Configuration options: {pop-up-blocker, Google Toolbar, Do Not Track}
Key Idea: statically compute configuration option impact

Configuration options: {pop-up-blocker, Google Toolbar, Do Not Track}
Key Idea: statically compute configuration option impact

Configuration options: {pop-up-blocker, Google Toolbar, Do Not Track}
Key Idea: statically compute configuration option impact

Configuration options: {pop-up-blocker, Google Toolbar, Do Not Track}
Key Idea: statically compute impact of the changes

Configuration options: {pop-up-blocker, Google Toolbar, Do Not Track}
Key Idea: Intersect configuration impact with change impact

Configuration options: {pop-up-blocker, Google Toolbar, Do Not Track}
Key Idea: Intersect configuration impact with change impact

Configuration options: {pop-up-blocker, Google Toolbar, Do Not Track}

Select configuration option “Google Toolbar” for regression testing
Safely discard “pop-up blocker” and “Do Not Track”
Example Program
Mapping configurable options to source code

Configurable Options: \{P_1, P_2, P_3\}

```c
1. int f_1(int x){
2.     return ++x;
3. }
4. int f_2(int x){
5.     int s = -f_1(x);
6.     return s;
7. }
8. int f_3(int x){
9.     int s = f_1(x) % 4;
10.    return s;
11. }
12. }
13. }
14. void f_4(){ printf("f_4\n"); }
15. void f_5(){ printf("f_5\n"); }
16. void f_6(){ printf("f_6\n"); }
17. void f_7(){ printf("f_7\n"); }
18. void f_8(){ printf("f_8\n"); }
19. void f_9(){ printf("f_9\n"); }
20. }
21. void f_10(){ printf("f_10\n"); }
22. }
23. void f_11(){ printf("f_11\n"); }
24. }
25. //configurable options
26. unsigned int P_1;
27. unsigned int P_2;
28. unsigned int P_3;
29. void main(){
30.     int x;
31.     if (x == 0) {
32.         if(P_1)
33.             f_1(1);
34.         else
35.             f_2(2);
36.     } // end x==0
37.     else { /* x != 0 */
38.         f_3();
39.         if(x < 0){
40.             f_5();
41.             if(P_2)
42.                 f_4();
43.             } // end x < 0
44.         else { /* x > 0 */
45.             if(P_3){
46.                 if(P_2)
47.                     f_6();
48.                 else
49.                     f_7();
50.             }
51.             else
52.                 f_8(6);
53.         } // end x > 0
54.     } // end x != 0
55. }
56. }
57. }
58. }
```
Example Program

Mapping configurable options to source code

Configurable Options: \{P_1, P_2, P_3\}

```
1. int f_1(int x){
2.     return ++x;
3. }
4.
5. int f_2(int x){
6.     int s = -f_1(x);
7.     return s;
8. }
9.
10. int f_3(int x){
11.     int s = f_1(x)%4;
12.     return s;
13. }
14.
15. void f_4(){ printf("f_1\n"); }
16.
17. void f_5(){ printf("f_2\n"); }
18.
19. void f_6(){ printf("f_3\n"); }
20.
21. void f_7(){ printf("f_4\n"); }
22.
23. void f_8(){ printf("f_5\n"); }
24.
25. // configurable options
26. unsigned int P_1;
27. unsigned int P_2;
28. unsigned int P_3;
29.
30. void main(){
31.     int x;
32.     if (x == 0) {
33.         if (P_1)
34.             f_1(1);
35.         else
36.             f_2(2);
37.     } // end x==0
38.
39.     else { // x != 0
40.         f_3();
41.         if(x < 0){
42.             f_5();
43.             if (P_2)
44.                 f_4();
45.         } // end x < 0
46.         else{ // x > 0
47.             if (P_3){
48.                 f_6();
49.             } else
50.                 f_7();
51.         } // end x > 0
52.
53.     }
54.     else
55.         f_8(6);
56.     } // end x != 0
57. } // end x
```
Example Program
Mapping configurable options to source code

Configurable Options: \{P_1, P_2, P_3\}

```c
1. int f_1(int x){
2.     return ++x;
3. }
4.
5. int f_2(int x){
6.     int s = -f_1(x);
7.     return s;
8. }
9.
10. int f_3(int x){
11.     int s = f_1(x) % 4;
12.     return s;
13. }
14.
15. void f_4(){ printf("f_4\n"); }
16.
17. void f_5(){ printf("f_5\n"); }
18.
19. void f_6(){ printf("f_6\n"); }
20.
21. void f_7(){ printf("f_7\n"); }
22.
23. void f_8(){ printf("f_8\n"); }
24.
25. // configurable options
26. unsigned int P_1;
27. unsigned int P_2;
28. unsigned int P_3;
29.
30. void main(){
31.     int x;
32.     if (x == 0) {
33.         if (P_1)
34.             f_1(1);
35.         else
36.             f_1(2);
37.     } // end x==0
38.     else { // x != 0
39.         f_3();
40.         if (x < 0){
41.             if (P_2)
42.                 f_5();
43.             else
44.                 f_4();
45.         } // end x < 0
46.         else{ //x > 0
47.             if (P_1){
48.                 if (P_2)
49.                     f_6();
50.                 else
51.                     f_7();
52.             } else
53.             f_8(6);
54.         } // end x > 0
55.     }
56.     else{ // x ! = 0
57.         f_8(0);
58.     }
59. }
```
Example Program

Mapping configurable options to source code

Configurable Options: \{P_1, P_2, P_3\}

```c
1. int f_1(int x){
2.     return ++x;
3. }
4. int f_2(int x){
5.     int s = -f_1(x);
6.     return s;
7. }
8. }
9. int f_3(int x){
10.    int s = f_1(x) % 4;
11.    return s;
12. }
13. }
14. void f_4(){ printf("f_4\n"); }
15. void f_5(){ printf("f_5\n"); }
16. void f_6(){ printf("f_6\n"); }
17. void f_7(){ printf("f_7\n"); }
18. void f_8(){ printf("f_8\n"); }
19. void f_9(){ printf("f_9\n"); }
20. void f_10(){ printf("f_10\n"); }
21. //configurable options
22. unsigned int P_1;
23. unsigned int P_2;
24. unsigned int P_3;
25. void main(){
26.     int x;
27.     if (x == 0) {
28.         if(P_1)
29.             f_1(1);
30.         else
31.             f_2(2);
32.     } // end x==0
33. }
34. else { // x != 0
35.     f_3();
36.     if(x < 0){
37.         f_3();
38.         if(P_1)
39.             f_1();
40.     } // end x < 0
41.     else{ //x > 0
42.         if(P_2)
43.             f_2();
44.         else
45.             f_7();
46.     } // end x > 0
47. }
48. else{ // end x != 0
49.     f_8(6);
50. }
51. }
52. ```
Example Program
Mapping configurable options to source code

Configurable Options: \{P_1, P_2, P_3\}

Function \(f_1\) changes

```
1. int f1(int x){
2.    return x + x;
3. }

5. int f2(int x){
6.    int s = -f1(x);
7.    return s;
8. }

10. int f5(int x){
11.    int s = f1(x)%4;
12.    return s;
13. }

15. void f3(){ printf("f1 "); }
16. void f4(){ printf("f2 "); }
18. void f6(){ printf("f5 "); }
20. void f7(){ printf("f6 "); }
22. void f8(){ printf("f7 "); }
24. void f9(){ printf("f9 "); }

25. // configurable options
26. unsigned int P1;
27. unsigned int P2;
28. unsigned int P3;
```

```
28. void main(){
29.    int x;
31.    if (x == 0) {
32.        if(P1)
33.            f1();
34.        else
35.            f2();
36.        } // end x==0
37.    else { // x != 0
38.        f3();
39.        if(x < 0){
41.            f5();
42.            if(P3)
43.                f1();
44.        } // end x < 0
45.        else{ //x > 0
46.            if(P1){
47.                f6();
48.                if(P2)
49.                    f5();
50.                else
51.                    f7();
52.            }
53.            else
54.                f8();
55.        } // end x > 0
56.    } // end x != 0
58. }
```
Example
Example

### Configurable Options

<table>
<thead>
<tr>
<th>Options</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_1$</td>
<td>True</td>
</tr>
<tr>
<td>$P_2$</td>
<td>True</td>
</tr>
<tr>
<td>$P_3$</td>
<td>True</td>
</tr>
</tbody>
</table>
Example

Configurable Options

<table>
<thead>
<tr>
<th>Options</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_1$</td>
<td>True</td>
</tr>
<tr>
<td></td>
<td>False</td>
</tr>
<tr>
<td>$P_2$</td>
<td>True</td>
</tr>
<tr>
<td></td>
<td>False</td>
</tr>
<tr>
<td>$P_3$</td>
<td>True</td>
</tr>
<tr>
<td></td>
<td>False</td>
</tr>
</tbody>
</table>

Configurations by pair-wise CIT

<table>
<thead>
<tr>
<th></th>
<th>$P_1$</th>
<th>$P_2$</th>
<th>$P_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_1$</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>$C_2$</td>
<td>True</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>$C_3$</td>
<td>False</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>$C_4$</td>
<td>False</td>
<td>False</td>
<td>True</td>
</tr>
</tbody>
</table>
Example
Simplified dependency graph

Configurable Options

<table>
<thead>
<tr>
<th>Options</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>(P_1)</td>
<td>True</td>
</tr>
<tr>
<td></td>
<td>False</td>
</tr>
<tr>
<td>(P_2)</td>
<td>True</td>
</tr>
<tr>
<td></td>
<td>False</td>
</tr>
<tr>
<td>(P_3)</td>
<td>True</td>
</tr>
<tr>
<td></td>
<td>False</td>
</tr>
</tbody>
</table>

Configurations by pair-wise CIT

<table>
<thead>
<tr>
<th></th>
<th>(P_1)</th>
<th>(P_2)</th>
<th>(P_3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(C_1)</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>(C_2)</td>
<td>True</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>(C_3)</td>
<td>False</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>(C_4)</td>
<td>False</td>
<td>False</td>
<td>True</td>
</tr>
</tbody>
</table>
Impact of configuration option $P_1$
$f_1$, $f_2$, $f_6$, $f_7$, and $f_8$

<table>
<thead>
<tr>
<th>Options</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_1$</td>
<td>True</td>
</tr>
<tr>
<td></td>
<td>False</td>
</tr>
<tr>
<td>$P_2$</td>
<td>True</td>
</tr>
<tr>
<td></td>
<td>False</td>
</tr>
<tr>
<td>$P_3$</td>
<td>True</td>
</tr>
<tr>
<td></td>
<td>False</td>
</tr>
</tbody>
</table>

Configurable Options

Configurations by pair-wise CIT

<table>
<thead>
<tr>
<th>Options</th>
<th>$P_1$</th>
<th>$P_2$</th>
<th>$P_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_1$</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>$C_2$</td>
<td>True</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>$C_3$</td>
<td>False</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>$C_4$</td>
<td>False</td>
<td>False</td>
<td>True</td>
</tr>
</tbody>
</table>
Impact of configuration option P₂ f₇ and f₈

Configurable Options

<table>
<thead>
<tr>
<th>Options</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>P₁</td>
<td>True</td>
</tr>
<tr>
<td>P₂</td>
<td>True</td>
</tr>
<tr>
<td>P₃</td>
<td>True</td>
</tr>
</tbody>
</table>

Configurations by pair-wise CIT

<table>
<thead>
<tr>
<th>C₁</th>
<th>P₁</th>
<th>P₂</th>
<th>P₃</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>C₂</td>
<td>True</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>C₃</td>
<td>False</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>C₄</td>
<td>False</td>
<td>False</td>
<td>True</td>
</tr>
</tbody>
</table>
Impact of configuration option $P_3$

### Configurable Options

<table>
<thead>
<tr>
<th>Options</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_1$</td>
<td>True</td>
</tr>
<tr>
<td></td>
<td>False</td>
</tr>
<tr>
<td>$P_2$</td>
<td>True</td>
</tr>
<tr>
<td></td>
<td>False</td>
</tr>
<tr>
<td>$P_3$</td>
<td>True</td>
</tr>
<tr>
<td></td>
<td>False</td>
</tr>
</tbody>
</table>

### Configurations by pair-wise CIT

<table>
<thead>
<tr>
<th></th>
<th>$P_1$</th>
<th>$P_2$</th>
<th>$P_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_1$</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>$C_2$</td>
<td>True</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>$C_3$</td>
<td>False</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>$C_4$</td>
<td>False</td>
<td>False</td>
<td>True</td>
</tr>
</tbody>
</table>
Impact of changed function $f_1$
$f_1$, $f_2$, and $f_6$

Program Entry
(input int x)

Y
N

X == 0

P1

f3

P1

f1, f2

f5

X < 0

N
Y

P3

f6

P1

f7

P2

f8

Program Exit

Configurable Options

<table>
<thead>
<tr>
<th>Options</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>True</td>
</tr>
<tr>
<td>P2</td>
<td>True</td>
</tr>
<tr>
<td>P3</td>
<td>True</td>
</tr>
</tbody>
</table>

Configurations by pair-wise CIT

<table>
<thead>
<tr>
<th></th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>True True True</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C2</td>
<td>True False False</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C3</td>
<td>False True False</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C4</td>
<td>False False True</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Select option $P_1$ and safely discard $P_2$ and $P_3$

<table>
<thead>
<tr>
<th>Configurable Options</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Options</strong></td>
</tr>
<tr>
<td>---</td>
</tr>
<tr>
<td>$P_1$</td>
</tr>
<tr>
<td>$P_2$</td>
</tr>
<tr>
<td>$P_3$</td>
</tr>
</tbody>
</table>

**Configurations by pair-wise CIT**

<table>
<thead>
<tr>
<th></th>
<th>$P_1$</th>
<th>$P_2$</th>
<th>$P_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_1$</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>$C_2$</td>
<td>True</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>$C_3$</td>
<td>False</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>$C_4$</td>
<td>False</td>
<td>False</td>
<td>True</td>
</tr>
</tbody>
</table>
Select option $P_1$ and safely discard $P_2$ and $P_3$
Research Questions

1. How does our selection compare to retest-all, in terms of fault detection?
2. What percentage of configurations is discarded as redundant by our selection?
3. How much regression time can our selection save?

Effectiveness   Efficiency
Research Questions

1. How does our selection compare to retest-all, in terms of fault detection?
2. What percentage of configurations is discarded as redundant by our selection?
3. How much regression time can our selection save?
Subjects

- **Make** (Software Infrastructure Repository)
  - V3.77 to v3.78.1
  - LOC: \( \approx 15k \) LOC
  - Code changes: selects 60 from 869
  - Seeded 15 faults
  - Configurable options: 11 (binary) \( \rightarrow \) 7 configurations

- **Grep**
  - V1.0 to V2.0
  - LOC: \( \approx 8k \) LOC
  - Code changes: 15
  - Seeded 15 faults
  - Configurable options: 14 (binary) \( \rightarrow \) 7 configurations
### Results

#### Fault Detection Ability

<table>
<thead>
<tr>
<th></th>
<th>Make</th>
<th>Grep</th>
</tr>
</thead>
<tbody>
<tr>
<td>Retest-all</td>
<td>8/15</td>
<td>6/15</td>
</tr>
<tr>
<td><strong>Our selection</strong></td>
<td>8/15</td>
<td>6/15</td>
</tr>
<tr>
<td>Random selection</td>
<td>3/15</td>
<td>5/15</td>
</tr>
</tbody>
</table>
## Results

### Fault Detection Ability

<table>
<thead>
<tr>
<th></th>
<th>Make</th>
<th>Grep</th>
</tr>
</thead>
<tbody>
<tr>
<td>Retest-all</td>
<td>8/15</td>
<td>6/15</td>
</tr>
<tr>
<td><strong>Our selection</strong></td>
<td>8/15</td>
<td>6/15</td>
</tr>
<tr>
<td>Random selection</td>
<td>3/15</td>
<td>5/15</td>
</tr>
</tbody>
</table>

Our approach is safe wrt retest-all configurations
Research Questions

1. How does our selection compare to retest-all, in terms of fault detection?

2. What percentage of configurations is discarded as redundant by our selection?

3. How much regression time can our selection save?

Effectiveness

Efficiency
Subject

ABB1

- LOC: 1.18 MLOC
- Number of Functions: 20,432 functions
- Code changes: 203
- Configurable options: 545 (number of values range from 2 to 9) → 159 configurations
- Among the 203 changes, we selected three sets of 30 changes for analysis
Results
Percentage of configurations selected

NUMBER OF CONFIGURABLE OPTIONS SELECTED

<table>
<thead>
<tr>
<th></th>
<th>Change set 1</th>
<th>Change set 2</th>
<th>Change set 3</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Retest-all</td>
<td></td>
<td></td>
<td></td>
<td>545</td>
</tr>
<tr>
<td>Selected</td>
<td>167</td>
<td>161</td>
<td>161</td>
<td>163</td>
</tr>
<tr>
<td>reduction</td>
<td>69%</td>
<td>70%</td>
<td>70%</td>
<td>70%</td>
</tr>
</tbody>
</table>

NUMBER OF CONFIGURATIONS SELECTED

<table>
<thead>
<tr>
<th></th>
<th>Change set 1</th>
<th>Change set 2</th>
<th>Change set 3</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Retest-all</td>
<td></td>
<td></td>
<td></td>
<td>159</td>
</tr>
<tr>
<td>Selected</td>
<td>120</td>
<td>120</td>
<td>120</td>
<td>120</td>
</tr>
<tr>
<td>reduction</td>
<td>25%</td>
<td>25%</td>
<td>25%</td>
<td>25%</td>
</tr>
</tbody>
</table>
Research Questions

1. How does our selection compare to retest-all, in terms of fault detection?

2. What percentage of configurations is discarded as redundant by our selection?

3. How much regression time can our selection save?
## Results

### Testing time savings

<table>
<thead>
<tr>
<th></th>
<th>grep</th>
<th>make</th>
<th>ABB1</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Testing time</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Retest-all</td>
<td>70m</td>
<td>700m</td>
<td>795h</td>
</tr>
<tr>
<td>Our approach</td>
<td>60m</td>
<td>300m</td>
<td>600h</td>
</tr>
<tr>
<td><strong>Overhead of selection</strong></td>
<td>5.2m</td>
<td>13m</td>
<td>28h</td>
</tr>
<tr>
<td><strong>Time savings</strong></td>
<td>5m</td>
<td>387m</td>
<td>167h</td>
</tr>
</tbody>
</table>

**50%**  
**55%**  
**21%**
## Results

### Testing time savings

<table>
<thead>
<tr>
<th></th>
<th>grep</th>
<th>make</th>
<th>ABB1</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Testing time</strong> Retest-all</td>
<td>70m</td>
<td>700m</td>
<td>795h</td>
</tr>
<tr>
<td>Our approach</td>
<td>60m</td>
<td>300m</td>
<td>600h</td>
</tr>
<tr>
<td><strong>Overhead of selection</strong></td>
<td>5.2m</td>
<td>13m</td>
<td>28h</td>
</tr>
<tr>
<td><strong>Time savings</strong></td>
<td>5m</td>
<td>387m</td>
<td>167h</td>
</tr>
<tr>
<td></td>
<td><strong>50%</strong></td>
<td><strong>55%</strong></td>
<td><strong>21%</strong></td>
</tr>
</tbody>
</table>

Our configuration selection approach saves about 20-55% of testing time wrt retest-all configurations.
Better than random, safe wrt retest-all

1. How does our selection compare to retest-all and random in terms of fault detection? **Safe**

2. What percentage of configurations is discarded as redundant by our selection? **15 – 60%**

3. How much regression time can our selection save? **20 – 55%**

Effectiveness  
Efficiency
First configuration selection approach for regression testing
First configuration selection approach for regression testing

Source code CHANGES

IE5

IE7
First configuration selection approach for regression testing

Source code CHANGES

IE5 \[\{C_1, C_2, C_3, C_4, C_5\}\] \rightarrow \text{Selection} \rightarrow \{C_1, C_5\}

IE7
First configuration selection approach for regression testing

Source code CHANGES

IE5

{C₁, C₂, C₃, C₄, C₅} → Selection → {C₁, C₅}

T = {t₁, t₂, t₃, t₄, t₅} No test case selection T' = {t₁, t₂, t₃, t₄, t₅}
First configuration selection approach for regression testing

Source code CHANGES

IE5

IE7

\{C_1, C_2, C_3, C_4, C_5\} \rightarrow \text{Selection} \rightarrow \{C_1, C_5\}

T = \{t_1, t_2, t_3, t_4, t_5\} \quad \text{No test case selection} \quad T' = \{t_1, t_2, t_3, t_4, t_5\}

\{C_1, C_5\} \text{ is both safe (wrt retest-all configurations) and non redundant}
First configuration selection approach for regression testing

Source code CHANGES

IE5

IE7

\{C_1, C_2, C_3, C_4, C_5\} \rightarrow \text{Selection} \rightarrow \{C_1, C_5\}

\{t_1, t_2, t_3, t_4, t_5\} \rightarrow \text{Selection} \rightarrow \{t_1, t_2, t_3, t_4, t_5\}

\{C_1, C_5\} \text{ is both safe (wrt retest-all configurations) and non redundant}

In our experiments, 15-60% of configurations were discarded as redundant saving 20-55% of regression testing time
Power and productivity for a better world™