Improving R&D Productivity by Triangulating Failures

Richard Léveillé
Senior Manager,
Corporate Quality & Program Management

Synopsys, Inc.
Copyright ©2014
Outline

- Where to focus to improve Developer’s productivity?
- Conditions for High Productivity
- Code Integration Model
- Case for Triangulation Tool: Tracer
- Detailed description of the Triangulation Tool
- Impact of the solution in developing predictable and high quality software releases
Helping Design the Chips Inside

Impacting Everyone, Everything, Everywhere, Every Day

Electronic Design Automation (EDA)
Many Factors affect Development Approach
Productivity: Context and Challenges

### Software Characteristics
- Complex Code (C, C++)
- Large applications: 
  \( \frac{1}{2} \text{ MLOC} \) to >20 MLOCs
  
  Avg. code change:
  \(~10\%\) every 9 months

### Code Integration Challenges
- Growing product dependencies
- Up to 250 developers on same code branch for one product
- Global Development Team

### Software Release
- 60+ Products simultaneously
- Predictable and Repeatable Releases

### Regression Testing
- \(~1\text{M test cases / day / platform}\)
- \(~30,000\text{ cpuHours / day}\)
- Avg. 5 platforms / product
Conditions for High Productivity

Stable Code

Quick Feedback Loop

- **Stable code** enables high quality and predictable releases
  - Build and Test failures affect all developers working on the same code branch
  - Regression Passing Rate must be maintained at a very high standard

- **Quick iterations** to support defect remediation by Developers
  - Need for efficient Find/Fix/Validate cycle for Developers
    - **Narrow down cause of failures**
    - Effective debug tools
    - Fast Turnaround Time for Build/Regression to validate fixes
Definition of “SUCCESS”
- 100% Daily builds Success Rate
- >98% Full Regressions Pass Rate

Developer’s discipline is key
- Strict check-in criteria are applied
- 1st task for Developer is to fix regression failures
Triangulation

Determine the location of a point from known points

Identify cause of failures

Determine the cause of failure from transactions on the code branch
Tracer Tool

- Identify “bad code check-ins” which may cause build or regression test failure
- Detect Build failure before Daily Build cycle
- Instant triaging of failures to notify the developer who can best address the issue

Integral part of the Code Integration Cycle
Build Triangulation

- Detect Build errors early and notify the developer who checked-in the code

- Triangulation – Build Tracer
  - Starts an incremental build at every changelist as soon as the code is checked in
  - Detects any build errors
  - Notifies success or failure to the developer who checked in and other stakeholders (e.g. integrator, branch manager)
  - Maintains record posted on web pages

Goal is 100% Daily Build Success
Build Tracer Workflow

1. Developer
   - Changelist code
2. SCM
   - Changelist
   - Notify build failure
3. Build Tracer
   - Changelist fix code
   - Changelist
   - Notify build pass
   - {build fail}
   - {build pass}
Build Tracer – User Viewpoint

<table>
<thead>
<tr>
<th>ID</th>
<th>Username</th>
<th>Time</th>
<th>Result</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>2631052</td>
<td>rhsu</td>
<td>14:41:00</td>
<td>success</td>
<td>Rebuild</td>
</tr>
<tr>
<td>2631042</td>
<td>minli</td>
<td>14:30:58</td>
<td>success</td>
<td>Rebuild</td>
</tr>
<tr>
<td>2631034</td>
<td>minli</td>
<td>14:10:26</td>
<td>failure</td>
<td>Rebuild</td>
</tr>
<tr>
<td>2631025</td>
<td>conos</td>
<td>14:00:00</td>
<td>success</td>
<td>Rebuild</td>
</tr>
</tbody>
</table>

**Diffs**

```
Diff 1:
//sta/prime/main/dev/lt/da/da/pGvars.hkk4
-181,6 +181,7
0181,6 (181,7) 0

Diffs 2:
```

**Change list comments**

Change 2631034 by minli@mainictedminli try on 2014/10/07 14:10:26

Enhancement: Added the instanace names to be printed in the timing report header

**List of Check-in Files**

- //sta/prime/main/dev/lt/da/da/daMaster.cc
- //sta/prime/main/dev/lt/da/da/psVShare.hh
- //sta/prime/main/dev/lt/te/tcpt/tcpt_time.c
### Build Tracer Detection Rate

<table>
<thead>
<tr>
<th>Product</th>
<th># Changelists</th>
<th>% Faulty Changelists</th>
<th>Prevented Daily Build from Failure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product 1</td>
<td>592</td>
<td>4%</td>
<td>26%</td>
</tr>
<tr>
<td>Product 2</td>
<td>1675</td>
<td>1%</td>
<td>21%</td>
</tr>
<tr>
<td>Product 3</td>
<td>5771</td>
<td>2%</td>
<td>70%</td>
</tr>
<tr>
<td>Product 4</td>
<td>1826</td>
<td>2%</td>
<td>33%</td>
</tr>
<tr>
<td>Product 5</td>
<td>1398</td>
<td>1%</td>
<td>19%</td>
</tr>
<tr>
<td>Product 6</td>
<td>1121</td>
<td>4%</td>
<td>35%</td>
</tr>
<tr>
<td>Product 7</td>
<td>1121</td>
<td>2%</td>
<td>28%</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td></td>
<td><strong>2%</strong></td>
<td><strong>33%</strong></td>
</tr>
</tbody>
</table>
Regression Triangulation

Regression Triangulation determines “most likely” candidate to cause regression test failures

Triangulation – Regression Tracer
- Compares today’s daily regression test result with yesterday’s result
- Selects test cases to triangulate
- Invokes these test cases upon all executables built from each of the day’s changelists
- Notifies developer(s) who made the suspicious checkins

Goal is >98% Regression Pass rate during Development
Regression Tracer Workflow

1. **Developer**
   - Notify suspects of bad changelist

2. **Daily Regression**
   - Full regression test results

3. **Regression Tracer**
   - Identify delta test list
   - Get executables
   - Launch test case for each executable

4. **Executables storage**
   - Test results

5. **Regression run**
   - Identify suspects - bad changelist
Regression Tracer Workflow

- Compares today’s daily regression test result with yesterday’s result
- Selects test cases whose result worsened (e.g. pass → fail, fail → fatal) as “delta test case list”

<table>
<thead>
<tr>
<th>Daily build / Test Case Results</th>
<th>T1</th>
<th>T2</th>
<th>T3</th>
<th>T4</th>
<th>T5</th>
<th>T6</th>
<th>T7</th>
<th>T8</th>
<th>T9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yesterday’s results</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td>Fail</td>
<td>Fail</td>
<td>Fail</td>
<td>Fatal</td>
<td>Fatal</td>
<td>Fatal</td>
</tr>
<tr>
<td>Today’s results</td>
<td>Pass</td>
<td>Fail</td>
<td>Fatal</td>
<td>Pass</td>
<td>Fail</td>
<td>Fatal</td>
<td>Pass</td>
<td>Fail</td>
<td>Fatal</td>
</tr>
<tr>
<td>Cases to Triangulate</td>
<td>√</td>
<td>√</td>
<td></td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Only NEW failures are processed
Regression Tracer Workflow

- Invokes Delta test cases upon all executables built from each check-in
- Identifies all suspicious check-ins

<table>
<thead>
<tr>
<th>Regression Tracer launches test case for each executable</th>
</tr>
</thead>
<tbody>
<tr>
<td>(One Executable per Changelist)</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Exec2</td>
</tr>
<tr>
<td>--------</td>
</tr>
<tr>
<td>Test2</td>
</tr>
<tr>
<td>Test3</td>
</tr>
<tr>
<td>Test6</td>
</tr>
</tbody>
</table>

**Inference**: (One Executable per Changelist)
- Test2's failure is found within Executable4 corresponding to Changelist4
- Test3's failure is found within Executable5 corresponding to Changelist5
- Test6's failure is found within Executable5 corresponding to Changelist5
Regression Tracer – User Viewpoint

Test Failure Details for multi_scenario/merged_reporting
/ms_global_timing_native_pba.tcl

Status: fail on Change 2631372
Assignee: euric
Suspect: euric, @2630607 Tracer
Test Owner: NA

Branch: //sta/prime/main/dev
Suite: cit
Platform: x86_64
Operation Purpose: nightly
Client: main_ct
Gut Report: Report

Test details:
- Test: multi_scenario/merged_reporting
- Triangulate: yes
- Suspect: euric
- Test owner: NA
- Time: 39.86 Secs
- CPU: 4.06 Secs
- Memory: 898.29 MB
- Changelist: 2630607
- Reference: Pass
- Target: log
- Age: <1 days
- Previous: Fail
- Current: Fail

Changes:
- Test failed (fail) Oct 7, 2014
- Ownership changed from NA to euric by tracer Oct 7, 2014 21:34
- Tracer identified euric as a suspect for changelist @2630607 Tracer details Oct 7, 2014 13:00
- Test pass on Change 2630155 Oct 6, 2014 21:13

Suspect Found: No
Build Failed: No
Pass: Yes
## Regression Tracer Results

### Are we Detecting any Failures?

<table>
<thead>
<tr>
<th>Product</th>
<th>Volume of regression tests</th>
<th>Volume of regression tests to triangulate</th>
<th>Found Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product 1</td>
<td>8200</td>
<td>22</td>
<td>42%</td>
</tr>
<tr>
<td>Product 2</td>
<td>7900</td>
<td>26</td>
<td>51%</td>
</tr>
<tr>
<td>Product 3</td>
<td>7000</td>
<td>3</td>
<td>54%</td>
</tr>
<tr>
<td>Product 4</td>
<td>4200</td>
<td>5</td>
<td>67%</td>
</tr>
<tr>
<td>Product 5</td>
<td>22000</td>
<td>35</td>
<td>58%</td>
</tr>
<tr>
<td>Product 6</td>
<td>4200</td>
<td>8</td>
<td>63%</td>
</tr>
<tr>
<td>Product 7</td>
<td>5100</td>
<td>18</td>
<td>70%</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td></td>
<td></td>
<td><strong>58%</strong></td>
</tr>
</tbody>
</table>
## Positive Impact to Productivity

### Is the Code Branch Stable?

<table>
<thead>
<tr>
<th>Product</th>
<th>Daily Build Success Rate</th>
<th>Daily Regression Passing Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(12-month Average)</td>
<td></td>
</tr>
<tr>
<td>Product 1</td>
<td>98%</td>
<td>95.6%</td>
</tr>
<tr>
<td>Product 2</td>
<td>97%</td>
<td>98.3%</td>
</tr>
<tr>
<td>Product 3</td>
<td>95%</td>
<td>99.6%</td>
</tr>
<tr>
<td>Product 4</td>
<td>100%</td>
<td>99.2%</td>
</tr>
<tr>
<td>Product 5</td>
<td>99%</td>
<td>98.0%</td>
</tr>
<tr>
<td>Product 6</td>
<td>96%</td>
<td>99.4%</td>
</tr>
<tr>
<td>Product 7</td>
<td>96%</td>
<td>93.7%</td>
</tr>
<tr>
<td>Average</td>
<td>97%</td>
<td>98%</td>
</tr>
</tbody>
</table>

**Goal**

- **100%**
- **>98%**

© 2014 Synopsys, Inc. All rights reserved.
• The Tracer tool has been in use for ~2 years by many teams
  – Reduced effort to dispatch failures to all stakeholders
  – Reduced effort by each developer in parsing reports and running multiple tests to locate the point of failure
  – Shortened the defect resolution cycle
  – Improved stability of the code development branch

• Accountability shifts from the person who owns the regression test to the one who makes regression fail
Conclusions

• R&D productivity has been at the forefront of Synopsys development strategy.

• Automation through tools, like Tracer, is one successful approach towards optimizing the effort from our development resources

• Tracer contributes to the stability of code during development and predictable release cycle
Thank You