Continuous Change-Driven Build Verification (ChDT™)

Marc Hornbeek
Manager, SCM Systems
Spirent Communications

June 2011
INTRODUCTION
INTRODUCTION

Manager, SCM Systems, Spirent Communications
• Network Test and Measurement Systems
• Responsible for SCM Systems including automated BV
• Practical “How-to” guides for Engineering management professionals

Perforce Environment
• 1.6TB, 36 MLOC, 9 apps, 15 platforms, 18 releases/year, 100 builds /day
• 3000 tests/build: Build Test Cycle: 23 hours -> 4 hours
• Agile Manifesto:
  – Early and continuous delivery of valuable software
  – Deliver working software frequently
  – Working software is the primary measure of progress.

• Functional testing is the standard of proof.

• Testing for large scale systems is a bottleneck

• By dynamically testing code changes ChDT™ resolves this bottleneck.
ChDT™ Methodology
Automation allows the tests to be run automatically …right?…

..yes but… 😞

- time to run all the tests ?
- equipment to run enough tests in parallel ?
- time and effort to analyze all the results ?
The Change-Driven Test ChDT™ Methodology introduces a CODTEF™ Matrix into the code/test flow. The matrix is used by the ChDT™ system to automatically drive both test selection and test result report generation according to continuously adjusting code change-to-test-to-results correlation factors in the CODTEF™ Matrix.
“Empirical Study of Regression Test Selection Techniques – ACM Transactions”

1. Retest all - test everything regardless of changes
2. Hunches – human intuition
3. Randomly select tests
4. Minimalist - minimal code coverage
5. Dataflow technique - data interactions
6. Safe Techniques - cover each code segment deleted and added.

<table>
<thead>
<tr>
<th>Efficient?</th>
<th>Yes</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>Risk?</td>
<td>Low</td>
<td>High</td>
</tr>
<tr>
<td>Complexity?</td>
<td>Low</td>
<td>High</td>
</tr>
<tr>
<td>Platform?</td>
<td>Independent</td>
<td>Dependent</td>
</tr>
</tbody>
</table>

Change-driven: select tests based on specific code change information correlated to prior failure history of tests correlated to the changed modules
DYNAMIC ADJUSTMENT

- Code-to-test CODTEF™ correlation factors automatically adjust
  - Past test results automatically affect future test selections
  - Automatic adjustments of test timings
  - Automatic adjustment of test reports
  - Automatic adjustments of code and test performance data

Input -> Build/Test Process -> Output

Automatic Adjustment
IMPLEMENTATION FRAMEWORK
PRE_REQUISITES FOR ChDT IMPLEMENTATION

Technology:
• Source code management system
• Automated test framework

Process:
• Standalone automated test cases**

People:
• Tools developers (SCM Framework tools, DB design)

** Practical reference **
TSCRIPT™: A Practical System for Realizing Exemplary Automated Test Scripts
ChDT DATA FRAMEWORK

ChDT is a system of many variables

- Code variables
- Test variables
- Results variables
- System variables
- Computed variables

Framing the system around the product itself provides a logical framework.

There are choices:

1) Complete process-able definition of the product
2) Use attribute tags to define important (to test) characteristics
3) Tests themselves define the product behavior
4) Groups of modules and tests define major blocks of product behavior
5) Code itself describes the product behavior
**DEVELOPMENT PHASES**

**Phase 1**
Code/Test Group

- Break up test suite into groups. Identify code group – test group relationships.

- Tools: Automatically identify group code changes, Launch specific test groups, Report results according to group owners.

**Phase 2**
Refined Sub-Groups

- Identify code subgroups and initial test correlations. Identify and design specific tools changes.

- Tools: Implement remaining tools listed on prior chart except the advanced metrics.

**Phase 3**
Advanced Metrics

- Analysis and design: Determine which advanced metrics will be used by the organization.

- Tools: Create metric tools using the data collected during the Change-Driven loops.
TOOLS
ChDT™ ARCHITECTURE

Code Change Analyzer:
- Code Change DB
- Code deltas

Test Selector:
- Test Case Results DB
- CODTEF™ Matrix
- Test Case Selector

System Infrastructure:
- User inputs
- Resource DB
- Adjust CODTEF™ values

Reports and Metrics:
- Report Generators
- Metrics calculators

** Practical reference **

TCHANGE™: A Practical System For Implementing Change-Driven Test Methodology
CODTEF™ MATRIX

CODTEF™: Code/Test correlation Factor for each Code/Test combination

INITIALIZING CODTEF™
1) Manual Method: Coders/Testers fill in the CODTEF values in the matrix
   - Groups of code/tests
   - Individual tests
2) Automatic
   - Set all CODTEF factors to the average value of all CODTEF factors. ** (or 0.5 very initially until an average has been established)
3) Semi-automatic
   - Set some CODTEF factors using method 1, then let the system complete the rest using method 2

<table>
<thead>
<tr>
<th>CODTEF Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blank</td>
<td>unknown / not set</td>
</tr>
<tr>
<td>0.0</td>
<td>no correlation</td>
</tr>
<tr>
<td>0.5</td>
<td>some correlation</td>
</tr>
<tr>
<td>1.0</td>
<td>Direct correlation</td>
</tr>
</tbody>
</table>
CODTEF™ ALGORITHM

Adjust CODTEF after Pass/Fail/Inconclusive verdict is established for each test:

1. Inconclusive: Do not adjust CODTEF

2. FAIL:

New CODTEF = Prior CODTEF + δF(1 - CODTEF),
Maximum=1
where δF = average test failure rate (E.g. 0.03)

3. PASS:

New CODTEF = Prior CODTEF - δP(1 - CODTEF),
Minimum=0
where δP is the change rate of CODTEF for PASS that will match the change rate of FAIL if the system performs at the average level over time. δP= δF(δF/(1 - δF)) (E.g. 0.03x (0.03/(1-0.03)) = 0.001 )
SYSTEM DATA

R : Resources available for test
T : Time available for test
Q : Quality level = minimum CODETEF™ value
M : Must-do tests
TE : Test Exclusion guard band
   • % of tests allowed to be excluded from each level if T does not allow selection of all of them.
Test Selection = TS1 + TS2 + TS3 + TS4 =

= Test Selection Level 1 = Must-do tests
  • Regardless of CODTEF™

+ Test Selection Level 2 = Recently Failed tests
  • Regardless of CODTEF™

+ Test Selection Level 3 = Best CODTEF™ tests
  • CODTEF™ > Q

+ Test Selection Level 4 = Additional tests if time allows
  • CODTEF™ < Q
RESULTS AND METRICS
METRICS DERIVED FROM CODTEF™ VALUES

Testers
– High yield tests to be re-factored (high average CODTEF™)
– Low yield tests to be pruned (low average CODTEF™)

Code Owners
– CODTEF™ is Q or higher
  • Eliminates redundant reports of test results to irrelevant owners
– Code reliability: re-factor error prone code modules (modules with high average CODTEF™ values)

QA Managers
– Increasing average CODTEF™ values indicate quality erosion
## RETURN ON INVESTMENT (ROI)

<table>
<thead>
<tr>
<th></th>
<th>Benefit</th>
<th>Cost</th>
<th>ROI</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>250 Developers</td>
<td>$120,000 ChDT specific Tools labor (one time)</td>
<td>16 to 1 return</td>
</tr>
<tr>
<td></td>
<td>3,150,000 LOC</td>
<td>$21,000 Reconfigure existing tests (one time)</td>
<td>$3,208,500 net cash equivalent returned</td>
</tr>
<tr>
<td></td>
<td>4,550 test cases</td>
<td>$7,500 ChDT sever (one time capital expense)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>36 hours to run all tests</td>
<td>$21,000 Annual maintenance</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4 hours average cycle test with ChDT</td>
<td>$211,500 3 year cost</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3,800 bugs fixed per year</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4 hours to fix bug found during ChDT cycle</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>20 hours to fix bug found after ChDT cycle</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>30% fewer bugs escape integration after ChDT implements</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$130,000 loaded labor rate $/year = $62.50 $/hour</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$14,250,000 Cost of bugs before ChDT (3 years)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$10,830,000 Cost of bugs after ChDT (3 years)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$3,420,000 Cost savings of CHDT derived from lower cost of bugs (3 years)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

** Practical reference **

TCASE™: A Practical System for Creating $uccessful Test Automation Business Cases
EXAMPLE SYSTEM
EXAMPLE SSCC RESULT FILE

```xml
<?xml version="1.0" ?>
<Report branch="mainline" startbuild="3.50.4330" endbuild="3.50.4810">
  <FileGroup name="IPS" owner="Marc Hornbeek">
    <FilePath name="PortConfigs" changes="0" />
    <FilePath name="Rfc2544Back2Back" changes="0" />
    <FilePath name="Rfc2544Common" changes="0" />
    <FilePath name="Rfc2544FrameLoss" changes="0" />
    <FilePath name="Rfc2544Latency" changes="0" />
    <FilePath name="Rfc2544Throughput" changes="0" />
  </FileGroup>
  <FileGroup name="Routing" owner="Owner Name">
    <FilePath name="bfd" changes="0" />
    <FilePath name="bfd" changes="8" />
    <FilePath name="eoam" changes="0" />
  </FileGroup>
</Report>
```

SSCC = Smartest Source Code Counting tool
SUMMARY

Continuous Change-Driven Build Verification (ChDT™) methodology using the CODTEF™ matrix is a practical, platform independent, efficient and scalable solution to the build / verification loop dilemma.

Faster Cycles
Meaningful Metrics
Improved Quality
RELATED READING

• **TCHANGE™**: A Practical System For Implementing Change-Driven Test Methodology, Marc Hornbeek, 2010

• **A Systematic Review on regression test selection techniques**; Emelie Engström, Per Runeson, Mats Skoglund, Department of Computer Science, Lund University, SE-221 00 Lund, Sweden

• **An Optimized Change-Driven Regression Testing Selection Strategy for Binary JAVA Applications**; Sheng Huang, Yang Chen, Jun Zhu, Zhong Jie Li, Hua Fang Tan; IBM China Research Laboratories, and Department of Computer Science, Tsinghua University Beijing

• **An Empirical Study of Regression Test Selection Techniques**; Todd L. Graves, Mary Jean Harrold, Jung-Min Kim, Adam Porter, Gregg Rothermel; Ohio State University, Oregon State University, and University of Maryland

• **Market Survey of Device Software Testing Trends and Quality Concerns in the Embedded Industry**, Wind River, June 2010
THANK-YOU!

Thank-you for attending this session.

Marc Hornbeek

☑ Please fill out an evaluation form.
Thank-YOU!!

Follow-up comments, questions and suggestions?
Marc.Hornbeek@Spirent.com

The End