Code Coverage in Firmware Automation Testing

Spring 2017 UEFI Seminar and Plugfest
March 27 - 31, 2017
Presented by Liu Zhi (Intel Corporation)
Agenda

• Introduction
• Code Coverage
• Example Code Coverage Implementation w/ EDK II
• Tips & Tricks
• Summary / Q&A
Introduction

• Test automation is common in firmware environments
• Typically automation is designed to replace tests with human interaction
• However, automation can cover a larger set of scenarios for firmware
Firmware Test Challenges

• Simulating user input in automation
  – Presented as USB HID (keyboard/mouse)
• Adjusting test cases to output
  – LED status, screen output, text recognition, etc.
• Persistence across platform reset
• Determining coverage of validation plans
• This session will focus on the last item
Agenda

• Introduction
• Code Coverage
• Example Code Coverage Implementation w/ EDK II
• Tips & Tricks
• Summary / Q&A
Code Coverage

• Code coverage is a measure of the degree source code is tested by a test suite
  – Identifies what areas of the code are exercised during program execution
  – High code coverage = more thorough testing

• Code coverage reports direct users to add tests for uncovered code & find dead code

• Best with a full automation solution
Code Coverage in Firmware

• Code coverage is commonly used in application, but not as much in firmware
• Firmware use cases typically require changes in configuration
  – Change setup menu options
  – Add/remove peripherals (USB, SATA, …)
• Implement without source changes
Metrics for Code Coverage

• Measure percentages for three specific types of execution coverage:
  – Functions
  – Conditions
  – Lines

• Metrics can be general (entire firmware project) or granular (focus on one file)
## Example Report

### Directory Coverage

<table>
<thead>
<tr>
<th>Directory</th>
<th>Branches</th>
<th>Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>s:\MdeModulePkg\Core\Dxe\Dispatcher/</td>
<td>32% (143/444)</td>
<td>81% (18/22)</td>
</tr>
<tr>
<td>s:\MdeModulePkg\Core\Dxe\DxeMain/</td>
<td>34% (77/222)</td>
<td>68% (11/16)</td>
</tr>
<tr>
<td>s:\MdeModulePkg\Core\Dxe\Event/</td>
<td>67% (153/228)</td>
<td>100% (22/22)</td>
</tr>
<tr>
<td>s:\MdeModulePkg\Core\Dxe\PwVol/</td>
<td>46% (116/249)</td>
<td>76% (16/21)</td>
</tr>
<tr>
<td>s:\MdeModulePkg\Core\Dxe\PwVolBlock/</td>
<td>36% (28/76)</td>
<td>36% (4/11)</td>
</tr>
<tr>
<td>s:\MdeModulePkg\Core\Dxe\Gcc/</td>
<td>44% (291/655)</td>
<td>84% (33/39)</td>
</tr>
<tr>
<td>s:\MdeModulePkg\Core\Dxe\Hand/</td>
<td>58% (453/798)</td>
<td>100% (36/36)</td>
</tr>
<tr>
<td>s:\MdeModulePkg\Core\Dxe\Image/</td>
<td>37% (155/411)</td>
<td>60% (9/15)</td>
</tr>
<tr>
<td>s:\MdeModulePkg\Core\Dxe\Library/</td>
<td>50% (13/26)</td>
<td>100% (3/3)</td>
</tr>
<tr>
<td>s:\MdeModulePkg\Core\Dxe\Mem/</td>
<td>47% (284/604)</td>
<td>80% (20/25)</td>
</tr>
</tbody>
</table>
Agenda

• Introduction
• Code Coverage
• Example Code Coverage Implementation w/ EDK II
• Tips & Tricks
• Summary / Q&A
Example Code Coverage Implementation w/ EDK II

• This example is based on an EDK II project from tianocore.org
• Test automation is implemented using Intel® Intelligent Test System (Intel® ITS)
• Code coverage is an add-on package for Intel ITS, using Bullseye
Overall Workflow

Build
- UEFI Code
- CodeCoverageBinPkg
- Build Tool
- Firmware Image
  - Null .COV

Runtime
- Updated .COV
- Dump Tool
- Target Platform Test Execution
  - Code coverage data is collected in memory during boot process

Analysis
- Report Tool
  - Data is extracted from memory for coverage reports
Configuring Code Coverage

- Add CodeCoverageBinPkg
- Patch EDK II build tools
  - Build\Win32
  - Removes need to modify code
- Edit platform config files to add service drivers (PEI, DXE & Reset)
- Contains additional tools
  - Libraries for Visual Studio
  - Patches for Bullseye & two EDK II reference platforms (NT32, OVMF)
  - Tools for report generation
Build with Code Coverage

• Build project to enable code coverage
  – Example for NT32 (UEFI emulator)
    subst s: c:\Edk2
    s:\> CodeCoverageBinPkg\Build\Win32\CovEdkSetup.bat
    s:\> set ITSCOVTOOLCHAIN=VS2013
    s:\> build -t VS2013x86 -E
    CodeCoverageBinPkg\Config\Nt32Pkg.ini
    -D ENCOV -D ITSCOV_VS2013_ENABLE=TRUE -Z

• Boot platform to execute test plan and gather code coverage data
Dump Code Coverage Data

Code coverage data is collected in memory during boot & dumped under one of three conditions …

1. Collected from UEFI Shell (ucovdump.efi)
2. Stored to USB drive on system reset
3. Stored to USB drive at ExitBootServices()
   (not supported in NT32)

UEFI Plugfest – March 2017

UEFI.org
ucovreport creates coverage reports

Use these results to find testing gaps

Please see the details below:

**ITS Code Coverage Report**

<table>
<thead>
<tr>
<th>Directory</th>
<th>Branches</th>
<th>Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>z:\KdeModulePkg\Universal</td>
<td>31 %</td>
<td>117/372</td>
</tr>
<tr>
<td></td>
<td>0 %</td>
<td>28/1190</td>
</tr>
<tr>
<td>z:\KdePkg\library/</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Agenda

• Introduction
• Code Coverage
• Example Code Coverage Implementation w/ EDK II
• Tips & Tricks
• Summary / Q&A
Tips & Tricks

• Based on current implementation of Intel® ITS using Bullseye
  – Boot speed when code coverage is enabled
  – Build issues integrating code coverage
  – Code coverage in SEC & PEI phases
  – Changes in firmware image size
  – Code coverage for assembly code
Tips & Tricks

• Boot speed when code coverage is enabled
  – There is some overhead for code coverage entry/exit routines (increases boot time)
  – Intended for test environments only

• Build issues integrating code coverage
  – Make sure you are using the patched build tools (build.exe and GenFds.exe)
  – Note base tools version for your compiler will differ from EDK II master tree version
Tips & Tricks

• Code coverage in SEC & PEI phases
  – Not supported at this time

• Changes in firmware image size
  – Compiled image size will be increased, due to additional support for code coverage

• Code coverage for assembly code
  – Not supported at this time
Summary

• Test automation is an important method for validating firmware solutions
• Code coverage should be applied to firmware, as well as application code
• Analysis helps identify dead code and untested code functions/branches
• Code coverage solutions are available for UEFI and EDK II
Q&A
Thanks for attending the Spring 2017 UEFI Seminar and Plugfest

For more information on the UEFI Forum and UEFI Specifications, visit http://www.uefi.org