Next generation Cobol debugging

Petr Vacula – May 31 – 3.17

For Informational Purposes Only

Abstract

CA InterTest is more visual, easier to use, requires fewer steps, and less maintenance!

Ensure your critical mainframe applications are maintained and enhanced efficiently and effectively by NexGen developers who don't have the benefit of the collective insights gained by decades of experience. This session will describe how CA InterTest's Visual Debugger helps to expedite both problem resolution and general application understanding by leading developers down the same path the code takes using images easily translated to program paths. It will also show how CA InterTest will allow developers to use the TEST(SOURCE) NOLOAD compile option and then leverage the DWARF method of obtaining symbolic information for debug purposes, eliminating the requirement to match a load module with a listing and maintain a symbolic file.

Visual Debugger

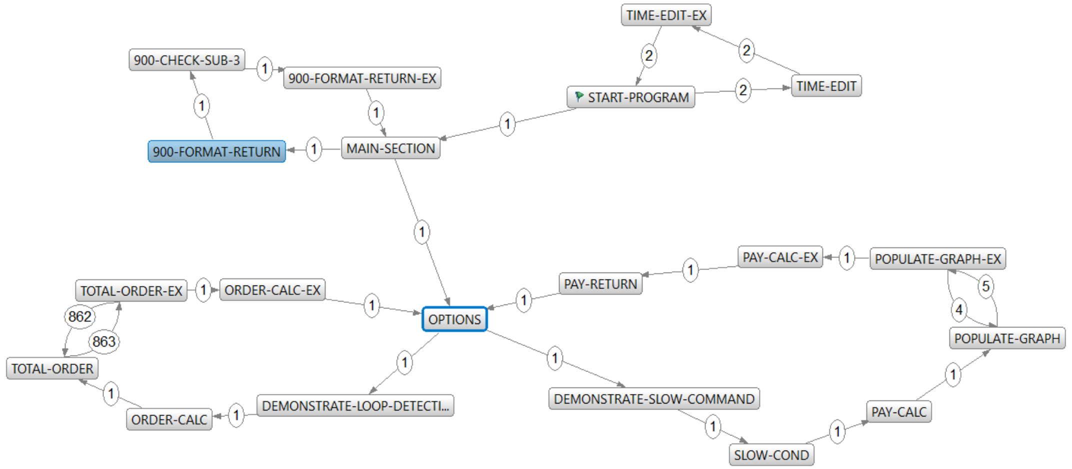
Legacy COBOL applications

Problems

- Experience workforce shortage
- Maintaining legacy applications requires understanding
- Enhancing legacy application almost impossible
- Learning curve for new mainframers is steep
- NextGen developers don’t find green screen cool enough

InterTest Visual Debugger

- Visual representation of a running COBOL (CICS/Batch)
  - *“A picture is worth a thousands words”*



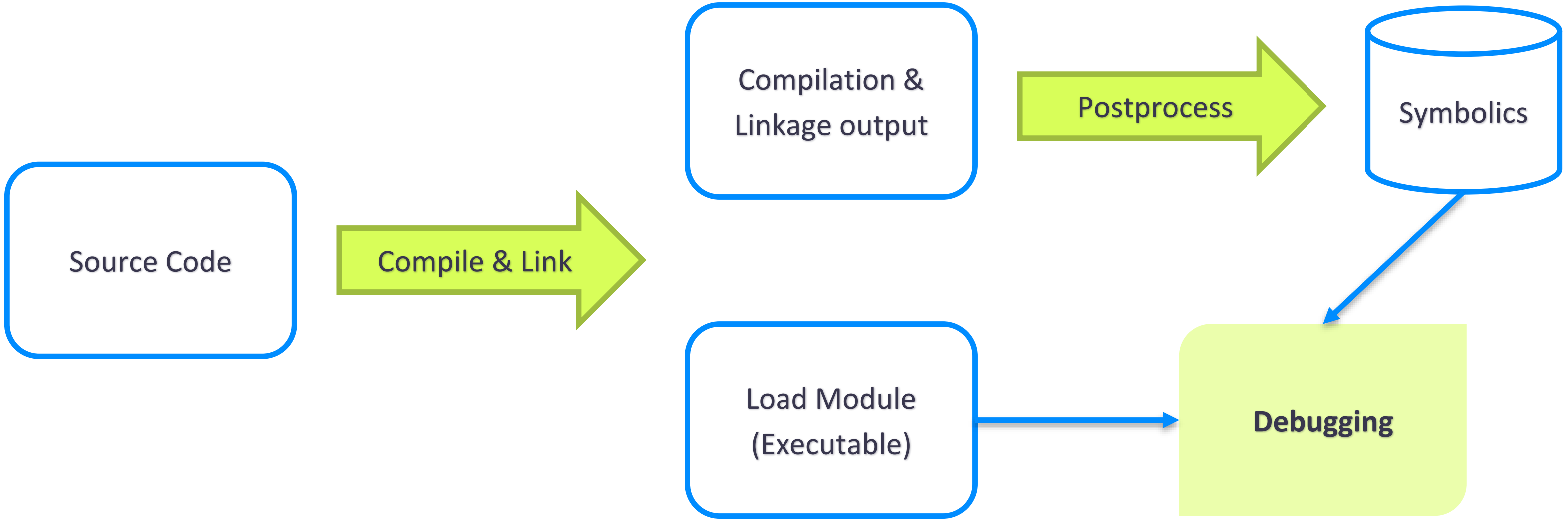
demo

DWARF Symbolic support

Symbolic Support definition

- Without Symbolic Support, debugging experience is reduced to stepping through processor instructions.
- Correlation between executable and source code is essential for meaningful debugging experience.
- Symbolic Support creates mapping between executable and source, making the debugging experience viable.

Debugging using Symbolics repository



Symbolic Support repository

Difficulties

- Need to maintain repositories
- Synchronize between LPARs
- Not matching symbolic

DWARF definition

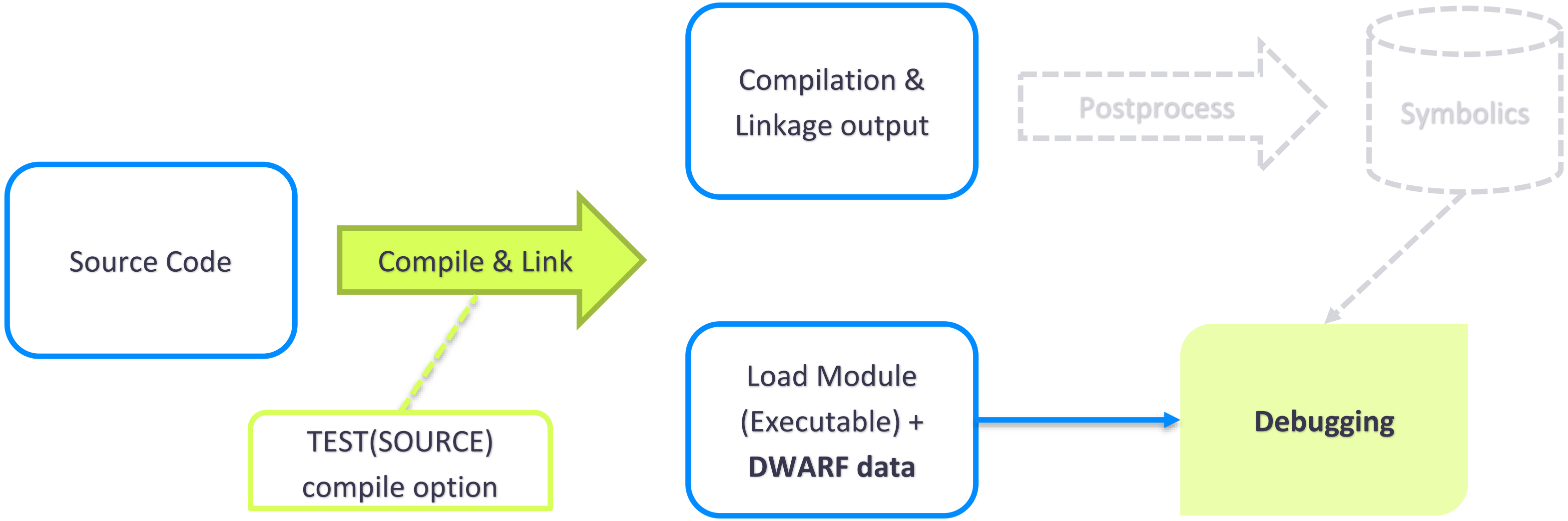
“DWARF is a widely used, standardized debugging data format.”

DWARF represents so called *extended source* that is included in a load module.

https://en.wikipedia.org/wiki/DWARF

Debugging using DWARF symbolics

~~Need to maintain repositories~~
~~Synchronize between LPARs~~
~~Not matching symbolic~~


Source Code
Compile & Link
TEST(SOURCE) compile option
Compilation & Linkage output
Postprocess
Symbolics
Load Module (Executable) + DWARF data
Debugging

Use case comparison

Symbolic Support repository

1. Compile & Link
2. Postprocess listings into repository
3. Distribute new repository
4. Distribute new load module
5. **Debug**

DWARF Symbolic Support

1. Compile with **TEST(SOURCE)** & Link
2. Distribute new load module
3. **Debug**

InterTest for CICS

Interactive debugger of CICS applications with Eclipse based GUI.

DWARF Symbolic Support currently under development

Initial drop will be available soon through validate.ca.com

Thank You.

Petr Vacula
Principal Product Owner
Petr.Vacula@ca.com

@cainc

slideshare.net/CAinc

linkedin.com/company/ca-technologies

ca.com

Call for Speakers Now Open

caWorld® '18 November 12-16, 2018 Miami Beach Convention Center

Share your experience. Be the teacher. Enhance your resume.

Register your session today!

Learn more: **ca.com/caworld**

ca technologies