“Project Coverage”

www.adacore.com

Slides Available at

http://libre.adacore.com/coverage/Project_Coverage.pdf

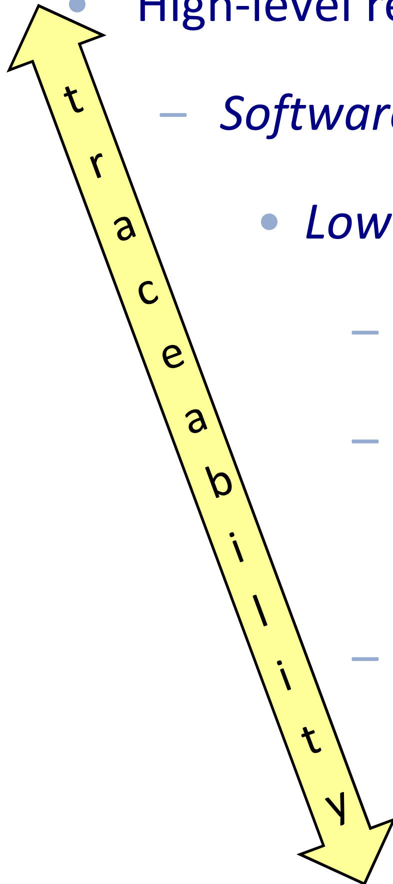
© AdaCore - www.adacore.com

Summary

- **The context**
- **Code coverage and FLOSS (Freely Licensed OSS)**
- **Code coverage approaches**
- **“Project Coverage”**
- **Background info on QEMU**

© AdaCore - www.adacore.com

DO-178B and Requirements-Based Testing



- High-level requirements / derived high-level requirements
  - *Software architecture*
    - *Low-level requirements / derived low-level requirements*
      - *Source code*
      - *Test cases*
        - » *Actual tests*
      - *Test coverage of high/low-level requirements*
        - » *Test coverage of software structure (code coverage)*

© AdaCore - www.adacore.com

DO-178B and Test Coverage of Software Structure

	Software Level			
Objective	**A**	**B**	**C**	**D**
Modified condition/decision	o			
Decision coverage	o	o		
Statement coverage	o	o	o	

© AdaCore - www.adacore.com

Code Coverage and FLOSS

- **No suitable FLOSS code coverage tool for DO-178B**
- **Would be nice to have cross-fertilization**
  - *DO-178B* ⇔ *Free Software/Open Source*
- **AdaCore**
  - *100% FLOSS company*
  - *DO-178B world*
  - *Very natural for us to look into this*

© AdaCore - www.adacore.com

Some Code Coverage Approaches

- **Instrument the source**
  - *Instrumentation added to source code to log coverage info*
- **Instrument the object**
  - *Instrumentation added by the compiler during compilation*
- **Extracting an execution trace**
  - *Via JTAG/Nexus probe, single stepping the CPU, …*

© AdaCore - www.adacore.com

Instrumenting the Source

- ☺ **Easy to map coverage info to source code**
- ☺ **Simple to put in place**

- ☹ **Programming-language specific**
- ☹ **Very intrusive**
- ☹ **Where do you save the coverage info log?**
- ☹ **May need to run the tests twice**
  - *On the host and the target un-instrumented + compare results*

AdaCore
The GNAT Pro Company

Instrumenting the Object

- **Special switch in GCC to do this**
  - *Information accrued in a log file (one per object)*
  - *GCOV reads the logs, consolidates the info, maps it back to the source*
- **☺ Programming language independent**
- **☹ Compiler dependent**
- **☹ Very intrusive**
- **☺ Only works when compiling for the host**
- **☺ Need to run the test twice**

AdaCore
The GNAT Pro Company

© AdaCore - www.adacore.com

Extracting the Execution Trace

- ☺ **Programming language independent**
- ☺ **Compiler independent**
- ☺ **Non intrusive**
- ☺ **Can obtain binary test coverage info on final application**

- ☹ **Can be slow or extra specialized hardware needed**
- ☹ **Need connetion between target and host to save the trace**
- ☹ **Relating the trace back to the original source can be hard**
  - *In the presence of advanced optimizations*

© AdaCore - www.adacore.com

“Project Coverage”

www.adacore.com

Starting Point

- **Nice to have:**
  - *FLOSS toolset for code coverage in a DO-178B context*
- **Interesting cross-fertilization opportunities**
  - *For safe/secure systems source code isn’t everything*

© AdaCore - www.adacore.com

“Project Coverage” Vision

- **Provide cutting-edge coverage tools**
- **FLOSS**
- **Cross-fertilization between communities**
  - *DO-178B ←→ Free Software/Open Source*
- **Extend the notion of Free Software beyond the source code**
  - *DO-178B artifacts (source code isn’t enough)*
- **Business model to create a self-sustainable technology**

© AdaCore - www.adacore.com

"Project Coverage": The Foundations

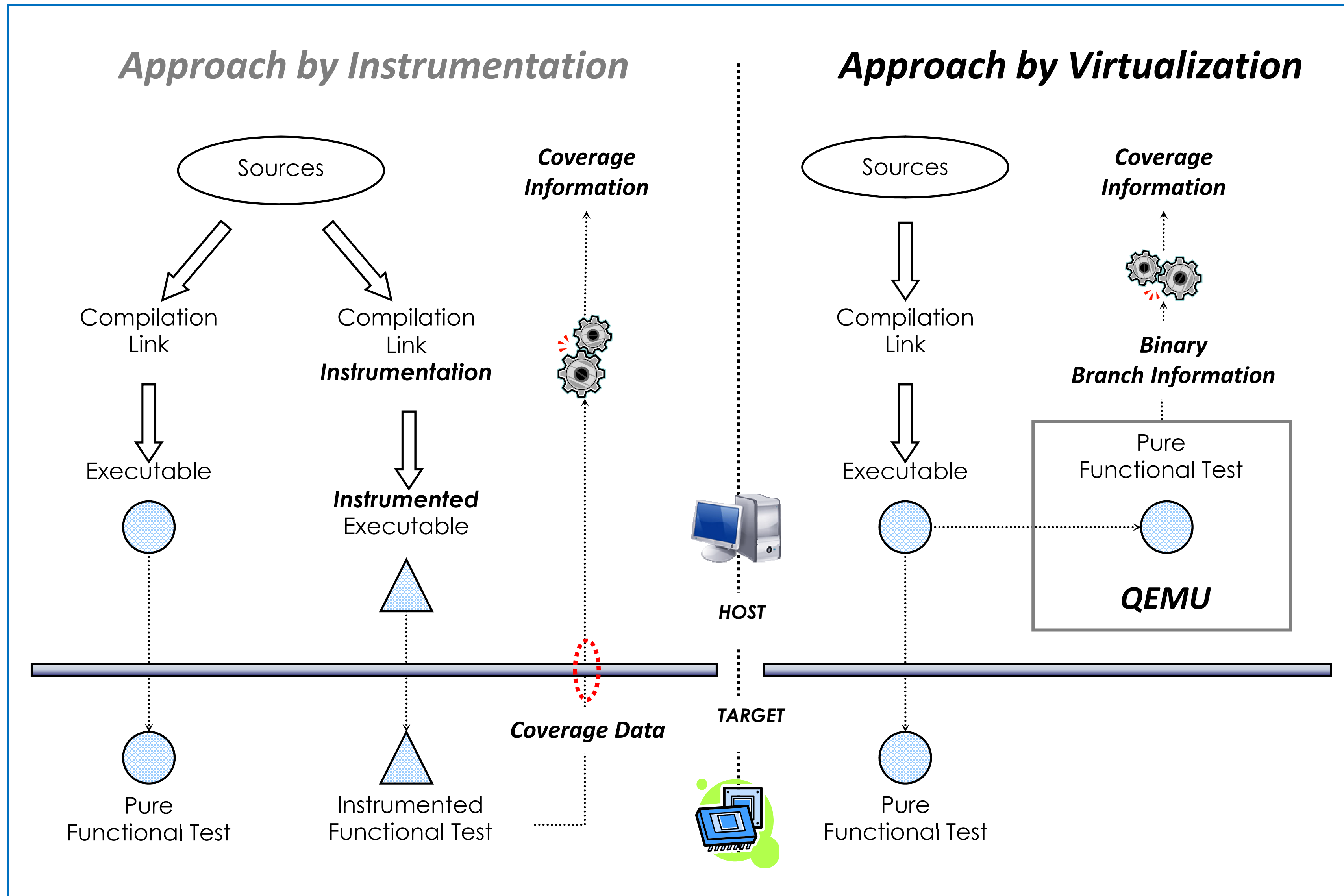
- **Working from execution traces looks appealing**
  - *How can we get rid of the hardware?*
- **Advances in hardware virtualization technology**
  - *Virtualization - VMware*
  - *Emulation – QEMU*
  - *Virtual machines*
- **Idea:**
  - *Don't instrument the code directly*
  - *Instrument the "thing" that executes the code*

"Project Coverage": How it Works

- **Run the target object code on the virtualized/emulated target**
  - *We use QEMU*
- **The emulator runs the target object on the host**
- **QEMU instrumented to collect object-level branch info**
  - *Branch info saved in a file on the host*
- **Branch info analyzed off-line and mapped back to the sources**
  - *Using the debug info (e.g. DWARF) available in the executable*

© AdaCore - www.adacore.com

“Project Coverage”



Consolidation and Capitalization

- **Pure unit testing**
  - *Need to consolidate coverage results for all object files comprising the final executable*
- **System-level testing + unit testing when needed**
  - *Need to capitalize/accrue coverage results between runs*
- **“Project Coverage” will provide support for**
  - *CONSOLIDATION*
  - *CAPITALIZATION*

© AdaCore - www.adacore.com

"Project Coverage" Limitations

- **Need debug info to map coverage holes back to the source**
  - *E.g. DWARF*
- **Certain restrictions imposed to do MC/DC**
  - *E.g. "and then" instead of "and" or "&&" instead of "&" in the source*
    - We are developing a code checker for Ada to enforce this
  - *Short circuit condition evaluation done by the compiler*

"Project Coverage" Benefits (1 of 2)

- ☺ **"Project Coverage" does not require specialized hardware**
  - *To extract coverage info*
- ☺ **"Project Coverage" tools are easy to use and deploy**
  - *They run on the host computer*
- ☺ **"Project Coverage" tools are non-intrusive**
  - *Can work on the final executable*
- ☺ **"Project Coverage" can extract coverage info quickly**
  - *Thanks to the difference in speed, memory, and file system requirements between the target and host computer*

© AdaCore - www.adacore.com

“Project Coverage” Benefits (2 of 2)

- ☺ **“Project Coverage” tools are language independent**
  - *Ada, C, …*
- ☺ **“Project Coverage” tools are compiler independent**
  - *To map coverage info back to the source DWARF debug info is assumed*
- ☺ **DO-178B qualification material**
- ☺ **“Project Coverage” tools will be FLOSS**
  - *Industrial users will have the option to purchase high-quality professional support*

Remarks

- **The approach taken by “Project Coverage” is general**
- **It works regardless of how you obtain branch info**
  - *Real hardware*
  - *Emulator*
  - *Virtualizer*
  - *Virtual machine*

AdaCore
The GNAT Pro Company

© AdaCore - www.adacore.com

Partners of "Project Coverage"

- **AdaCore** (industrial)
- **ENST** (academic)
  - *Generalization of the approach to distributed systems*
- **LIP6** (academic)
  - *Generalization to pure virtual machines (caML)*
- **Open Wide** (industrial)
  - *Provides avionics test bed, impact of MIL-STD-1553, ARINC 629, …*
- **http://libre.adacore.com/coverage**
- **Partly funded by French local and governmental institutions**

© AdaCore - www.adacore.com

Summary

- **“Project Coverage”: clever combination of several trends**
  - *FLOSS*
  - *Virtualization*
  - *DWARF*
  - *DO-178B and safety*
- **To produce a FLOSS code coverage solution**
  - *For safety-critical and non safety-critical developers*

QEMU

A Fast and Portable Dynamic Translator

QEMU (Quick EMUlator)

- **CPU emulator**
  - *x86, PowerPC, Sparc, …*
- **Available on several hosts**
  - *Linux, Windows, Sun, …*
- **QEMU supports full system emulation**
  - *A complete and unmodified OS is run in a virtual machine*
  - *E.g. Windows can be run on Linux and Linux on Windows*
- **Specific embedded devices can be simulated**
  - *By adding new machine descriptions and new emulated devices*

© AdaCore - www.adacore.com

QEMU Components

- **CPU emulator**
- **Emulated devices**
  - *VGA display, 16450 serial port, PS/2 mouse and keyboard, IDE hard disk, network card, ...*
- **Generic devices**
  - *e.g. block devices, character devices, network devices used to connect the emulated devices to the corresponding host devices*
- **Machine descriptions instantiating the emulated devices on the host**
- **...**

© AdaCore - www.adacore.com

QEMU and Portability

- **QEMU translates target instructions into an intermediate language**
- **Intermediate language compiled into host CPU instructions**
  - *Initially leveraged on GCC*
  - *Now moving to ~5,000 lines code generator*
  - *Job made simpler by the fact that x86 (Linux or Windows) is a universal host*

© AdaCore - www.adacore.com

QEMU: The Dynamic Translator

- **Performs a runtime conversion of the target CPU instructions**
  - *Into the host instruction set*
- **Leverages on off-line translation**
  - *Of the "basic blocks" in the target object code*
- **Resulting binary code is stored in a translation cache**
  - *So that it can be reused*
- **Advantage over an interpreter**
  - *Target instructions are fetched and decoded only once*
- **Advantage over a virtualizer**
  - *Instruction set of target and host can be different*

QEMU Dynamic Translation Mechanism

- **When QEMU first encounters a piece of target code, it translates it to host code up to the next jump**
  - *These basic blocks are called Translated Blocks (TBs)*
  - *A 16 MByte cache holds the most recently used TBs*
- **It then executes the TB on the host computer**
- **The hand is returned to QEMU at the end of the TB**

© AdaCore - www.adacore.com

Collecting Branch info with QEMU

- **QEMU is modified so that before each TB is executed PC and branch info are collected**
- **After each TB is executed QEMU uses the simulated PC to find the next TB**
  - *Uses a hash table*
- **If next TB not translated yet, translate it**
  - *By fetching the version that was translated offline*
- **Otherwise jump to the next TB**

Chaining TBs

- **In order to accelerate the most common case where the new simulated PC is known**
  - *E.g. for example after a conditional jump*
- **QEMU can patch a TB so that it jumps directly to the next one**

© AdaCore - www.adacore.com