Systems & Software
Producibility
Collaboration and
Experimentation
Environment

Safe & Secure Systems & Software
Symposium (S5)
15 June 2011

Mr. Steven Drager
Advanced Computing Architectures Core
Technical Competency Lead
Principal Electronics Engineer, DR-IV
Air Force Research Laboratory/RIT

Integrity ★ Service ★ Excellence
Outline

• Motivation
• SPRUCE Overview
• Representative Challenge Problems
• What’s Next?
The Problem

• DoD systems are inherently complex, increasingly software intensive
• Difficult to introduce advanced systems & software engineering technologies into DoD acquisition programs
  – Insufficient evidence to prove capabilities
  – Immature prototypes lack stability, features, support, tool chain integrations
  – Challenging to evaluate technology against “realistic” DoD problems
  – Must manage restrictions of classified, proprietary and ITAR information

• Increasing use of software:
  F-4 (8%), F-22 (85%), F-35 (est. >90%)
• Rapidly changing hardware speed and architectures
• Huge scale: $10^5$ requirements, $10^7$ lines of code
• Huge number of component interactions
• Increasing use of COTS and 3rd party software
• Long (20-40) year product lifecycles
• Stringent certification standards
Root Cause

- *Ad hoc* collaboration within software engineering research community
  \( \Rightarrow \) “valley of disappointment” for DoD software research programs
  - Unable to find technologies that meet needs
  - Failure to adopt promising technologies
  - Software engineering problems encountered repeatedly across programs
  - Additional problem: “landing path” is not typically DoD programs
Software-Intensive Systems Producibility Initiative

• Software is the prime enabler of complex weapons systems and command and control infrastructure

• Software is the least well understood and the most problematic element of large-scale systems
  – Little underlying science
  – Minimal engineering knowledge base

• Software and software project failures dominate causes of
  – System cost and schedule overruns
  – Failures of systems to satisfy their requirements
  – Increasing numbers of costly and dangerous system failures

• Research to improve our ability to develop complex software-intensive systems is token, disjoint, and narrowly focused
  – Industry has no incentive to solve common problems
  – Academia sees no consistent funding stream
A Vision

• A common virtual meeting environment for program engineers & technology researchers to form collaborations based on posting/discovering challenge problems of today’s/tomorrow’s DoD systems

• Payoffs:
  – **Realistic artifacts to drive research and the community forward**
  – **Systematic process for transitioning software producibility technologies into practice**
An open, collaborative research, and development environment consisting of a portal, a repository of realistic problems and artifacts and dedicated experimentation resources on which to demonstrate, evaluate, and document the ability of novel tools, methods, techniques, and run-time technologies to yield affordable and more predictable production of software intensive systems

Brings together researchers, developers and domain experts from different communities to de-fragment the knowledge necessary to achieve SISPI research, development and technology transition

Capabilities

1. A repository of defined challenge problems and associated artifacts
2. A repository of proposed challenge problem candidate solutions – i.e., technologies/tools/methods that address some or all parts of a challenge problem
3. An experimentation testbed
   a) For conducting realistic experiments for assessing candidate solution utility
   b) For conducting repeatability/duplication experimentation
   c) For experimentation on novel tools, techniques and methods
4. A repository of test results documenting benefits to the DoD acquisition community
5. Focused communities of interest for collaboration on all of the above
SPRUCE Implementation = Customized collaboration portal + Repository of realistic problem artifacts + Dedicated experiment resources

Challenge Problems

Collaborations & Match-Making

Communities of Interest

SPRUCE Experimentation Testbed

SPRUCE Experimentation Portal

Define/conduct

Experiments

Validate Results

Solutions

Experiments

DISTRIBUTION STATEMENT A. Approved for public release; distribution is unlimited. PA Approval: #88ABW-2011-3205 dated, 06 Jun 2011.
## Representative Content

<table>
<thead>
<tr>
<th>Community of Interest</th>
<th>Challenge Problem</th>
<th>Artifacts</th>
</tr>
</thead>
<tbody>
<tr>
<td>System of systems</td>
<td>Model Driven Architecture and Design</td>
<td>70+ image files with UML diagrams, description of two specific challenges and process issues</td>
</tr>
<tr>
<td>Static code analysis</td>
<td>Automated Test Code Generation for Static Code Analysis</td>
<td>130+ C code artifacts, illustrating various flaws (e.g., 291 overflow scenarios like off-by-one etc)</td>
</tr>
<tr>
<td></td>
<td>Determination of Ground Truth in Large Scale Benchmarks for Evaluating Static Analysis Tools</td>
<td>Six benchmark applications, each with source code (e.g., one application has 140+ files 38,000+ lines of code), thousands of generated warnings in raw and normalized format, and a template for ground truth</td>
</tr>
<tr>
<td>Multi-core architectures</td>
<td>Cache False-Sharing in Multi-Core Architectures</td>
<td>Source code, build instructions, results and discussion of benchmarking experiments</td>
</tr>
<tr>
<td></td>
<td>Distributed Integration Strategies for Functional Components on Multi-Core Architectures</td>
<td>A PDF file containing a reference to the problem and artifacts</td>
</tr>
<tr>
<td>Software forensics</td>
<td>Assessing the Evolution of Software Features</td>
<td>Pointer to source code for Firefox, Gaim and Openoffice</td>
</tr>
<tr>
<td></td>
<td>Finding the Functionally Representative Areas of a Software System</td>
<td>Pointer to source code for Firefox, Gaim and Openoffice</td>
</tr>
<tr>
<td>Real-time and Embedded Systems</td>
<td>Multi-dimensional Resource Optimization for Publisher Subscriber-based Avionics Systems</td>
<td>Topology of avionics application, with applications, nodes and with data on 10,000+ message types exchanged among them</td>
</tr>
<tr>
<td></td>
<td>Fiber Channel Protocol Optimization in Real-time Embedded Systems</td>
<td>Above artifacts + definition of protocol used in cross connect switch aboard aircraft</td>
</tr>
</tbody>
</table>
Multi-Core Cache False Sharing

- Motivation: DoD platform upgrade challenge
- Challenge: how to minimize the negative performance impact of caching effects in multi-core architectures
  - Migration to multi-core already happening, is infrastructure ready?
  - Naïve migration of legacy software does not yield the expected benefits w.r.t. computational performance and power consumption
  - Cache false sharing: multiple threads running on different cores share the same cache line, causing degraded performance due to cache misses

- Artifacts
  - Suite of C programs that produce different caching effects

Data writes induce cache copies to be invalidated, evicted and subsequently cause cache misses on other copies.
Multi-Core Cache False Sharing

- SPRUCE experimentation testbed enables reproducibility of the challenge / observed behavior

Experiment Details

The benchmark environment used to generate these results was:
- Dual Quad-core Intel Xeon Processors E5320 @ 1.86 GHz
- Linux Fedora Core 5 (2.6.20-6-1.2944.fc6 emulate-1 #4 SMP - from uname -a)
- ThreadPairsCacheFalseSharingCode
- gcc 4.1.1 20070105
- Compiler flags: -O0
- Using nptl (native pthread library)
- Program Execution Time metric is measured by pairs programs and written to stdout.

The benchmark environment is available through the LMATLProject in the SPRUCE testbed.

1. Browse [http://www.isislab.vanderbilt.edu](http://www.isislab.vanderbilt.edu)
2. Login using your SPRUCE identity (Not currently implemented; use must have a separate ISISLab Identity)
3. Browse LMATLProject (Will create a new CacheFalseSharingPairExp project in the future)
4. Swap-in SPRUCEtest experiment (will only work if experiment is NOT currently active)
5. ssh to pc8 (the Dual Quad-core node)
6. cd /proj/LMATLProject/spruce/pairs
7. run *-exe programs for one sample statistics.
6. run run_pairs.py to generate 5000 sample statistics.

Alternatively, you can define a new experiment in ISISLab using the benchmark environment machine (pc8). The entire pairs code tree is available as an artifact from the related experiment page.
Multi-Dimensional Resource Optimization

- **Motivation:** Avionics domain challenge

- **Challenge:** How to optimize allocation of processing tasks to processors, such that:
  - Processing tasks meet deadlines
  - Number of processors is minimized
  - Network bandwidth is minimized
  - Negative impact of internal bus I/O is minimized

- **Artifacts:** Sanitized versions of avionics domain artifacts
  - Software workload
  - Network traffic
  - ~40 processors, ~100 software components and ~14,000 messaging interactions

- **Problem and artifacts provided by Lockheed Martin Aeronautics**
Multi-Dimensional Resource Optimization

• The Solution: particle swarm optimization
• Solution provide by Vanderbilt University

6 fewer processors needed than expected
13.9 $\times 10^7$ Bytes = 24% network bandwidth reduction
Solution found in ~9s on average
Predictive Cache Modeling & Analysis (PCMA)

**Problem**

- Random task order in leads to highly variable execution times
- Embedded system needs predictably high performance
- Early measurements indicate this variability is due to changes in cache behavior from different interleavings
- Solving the problem can not only yield predictable performance, but possibility to optimize throughput

**Approach**

- Generate a set of synthetic application tasks
  - Tasks have known memory sharing characteristics
  - Place maximum pressure on memory subsystem by saturating all integer & floating point pipelines
- Construct a portable embedded scheduler which mimics key characteristics of the systems of interest (embedded avionics, auto, cell phone, …)
- Investigate advanced heuristic optimization techniques to produce task orderings expected to reduce execution time variability and yield higher performance
- Validate the orderings on a variety of embedded and commercial processor architectures

**Results to Date (October 2010)**

- Synthetic application generator developed & working
- First-order heuristic optimization techniques tested using a non-real time scheduler; early results show that task ordering affects variability by 5%-10%
- Portable embedded scheduler emulator in progress

```c
void RunRateN()
{
    while (running) {
        decr(semaphoreN);
        #include "scheduleRateN.txt"
    }
}

void RunRateN2()
{
    while (running) {
        decr(semaphoreN2);
        #include "scheduleRateN2.txt"
    }
}
```

• Reduction in Variability
Multi-Modeling

- **Motivation:** Lack of integrated and cohesive early analysis of properties (e.g., reliability, thermal, QoS, etc.) of the whole system
- **Challenge:**
  - Semantic precision of inter-model data exchange
  - Consistency among interdependent models
  - Mechanisms to capture model interdependencies
  - Assistance in the selection of components
  - Correct composition of components
  - Automated support for component integration analysis
  - Scalability and ‘Big Data’ challenges
- **Artifacts:**
  - Requirements (e.g., DOORS, RequisitePro)
  - Specifications (e.g., UML, AADL, SysML, etc.)
  - Software (e.g. source code, Matlab, VAPS, etc.)
  - Tests (e.g. manually developed and automated)
  - Characteristics (Security; Synchronous/Asynchronous; Event/Time Driven; Performance/processor load; QoS; Fault Tolerance)
### MDA&D Grand Challenge

#### Problem Highlights
- Result of Raytheon's lead system integrator experience with software design and development for the DDG 1000 platform
- Applicable to many other software intensive systems
- Potential to spawn several sub-challenge problems and related research areas

#### Synopsis
This challenge problem focuses on one specific technical challenge of a large Program of Record: the difficulty of efficiently managing interface requirements to maintain consistency of the interfaces across the System of Systems (SoS) throughout the SoS lifecycle.

#### Specific Challenge
Provide capabilities (tools and techniques) to deal with modeling large scale systems across multiple major release levels concurrently, while maintaining consistency, as well as the ability to assure adherence to the non-functional technical performance measures; these technical performance measures support a “living” model with an ability to address multiple levels of detail, while assuring architectural consistency, correctness, and performance characteristics.

#### Artifacts
- 70+ UML sequence diagrams and class diagrams. Two representative end-to-end scenarios encompassing at least 2 subsystems, and the respective Technical Performance Measures. Representative list of organizational, system and software change process and tooling issues.

#### Performer
Raytheon
MDA&D Grand Challenge
Constituent Problems

- Five constituent, focused challenge problems to the MDA&D Grand challenge

- Currently, collaborations among AFRL, NRL, Raytheon, QinetiQ and Lockheed Martin
# Software Quality Challenge Problems

## Synopsis

These challenge problems seek to improve current state-of-the-art for evaluating static code analysis tools.

## Specific Challenge

**Static analysis tool accuracy:** Develop tools to quantitatively determine the accuracy of a given static analysis tool. Preferred method is to first determine the actual vulnerabilities in the selected programs. Compare tool outputs against this “gold standard”.

**Static analysis tool evaluation:** Develop code generation tools to evaluate a given static analysis tool’s ability to detect various defects.

## Artifacts (From NIST)

- **Tool accuracy artifacts:** six benchmark applications, each with source code (e.g., one application has 140+ files 38,000+ lines of code, another with 388 files constituting 53,847 lines of code), tens of thousands of generated warnings, and a template for ground truth.
- **Tool evaluation artifacts:** 1164 auto-generated buffer overflow tests in 130+ C code artifacts; a collection of 20 auto-generated tests to detect errors ranging from memory leaks and stack overflows to OS injection; a pair of tests for an auto-generated test case for resource injection and a hand-corrected version.

## Performer

LM ATL with help from NIST

## Problem Highlights

- NIST SAMATE project supports DHS Software Assurance Tools and R&D Requirements Identification Program
- Widely applicable in cyber security and software engineering
- Potential for solution providers to showcase tool capabilities
End-to-End Performance Analysis
Challenge Problem

• Motivation: avionics platform upgrade
• Challenge: predict best & worst-case end-to-end message delays under architecture change
  – Legacy system satisfies constraints on end-to-end message delays
  – Migrate to new processor platform, too costly to re-certify code ⇒ can’t recompile
  – Group legacy tasks into partitions, run multiple partitions/node using hierarchical periodic scheduling
  – Are end-to-end constraints still met?

• Artifacts: Excel spreadsheet
  – Example system description
  – 65 interacting tasks
  – 20 message chains
Quantum Verification Challenge Problem

- Motivation: aircraft safety standards
- Challenge: explore revolutionary approach – application of “quantum simulator” – to verify system operation
  - Exponential sensor input state space
  - Must validate behavior under all combinations
  - “Quantum simulator” offers possibility of parallelizing huge state space searches
- Motivated by 1982 Feynman paper “Simulating Physics with Computers”

- Artifacts
  - State machine diagram of triple-redundancy vehicle management system
  - Condition/action table defining operation
  - C program simulating operation
  - Sample sensor input data file
  - Sample violated requirement
What's Next for SPRUCE?

• Continue to grow SPRUCE content and community

• Immediate focus on three areas (communities of interest)
  – Multi-/many-core
  – Multi-modeling / model integration / model consistency
  – Cyber physical systems

• Introducing
  – Moderators (for each of the above CoIs)
  – Follow us on LinkedIn, Facebook, Twitter
  – Computer science grand challenges
  – Improved user experience
Revolutionizing the way that systems and software engineering technologies are identified, developed, and evaluated

Steven Drager  
Air Force Research Laboratory/RITA  
525 Brooks Rd.  
Rome, NY 13441  
TEL: +1.315.330.2735  
Email: Steven.Drager@rl.af.mil

William McKeever  
Air Force Research Laboratory/RITA  
525 Brooks Rd.  
Rome, NY 13441  
TEL: +1.315.330.2897  
Email: William.McKeever@rl.af.mil

Dr. Richard Buskens  
Lockheed Martin  
Advanced Technology Laboratories  
3 Executive Campus, 6th Floor  
Cherry Hill, NJ 08002  
TEL: +1.856.792.9756  
Email: rbuskens@atl.lmco.com