Automated Verification Support in Producible Adaptive Model-based Software

Sumit Ray : BAE Systems, sumit.ray@baesystems.com
Kevin M. McNeill : BAE Systems, kevin.mcneill@baesystems.com
Gabor Karsai : Vanderbilt University, gabor.karsai@vanderbilt.edu
James Donlon : DARPA, James.Donlon@darpa.mil
Agenda

• Overview of Producible, Adaptive Model-Based Software (PAMS) / Disruptive Manufacturing Technology (DMT) DARPA program
  • Motivation, approach and goal

• PAMS model-driven technologies developed to address the producibility problem

• Addressing the verification problem using PAMS technologies
Overview of DMT PAMS

Initiative: Disruptive Manufacturing Technologies (DMT)
Program: Producible, Adaptive Model -based Software (PAMS)

• DARPA sponsored program
  • 3-Year program starting in 2007
  • PAMS is the sole software program in the DMT initiative

• Program Team
  • BAE Systems (Prime), Vanderbilt University Institute for Software Intensive Systems (ISIS) and MIT Computer Science and Artificial Intelligence Laboratory (CSAIL)

• Program goal is to reduce development time by 70% and cost by 90% for software changes to long-lived platforms
Overview of DMT PAMS technology

- **Problem:** *In danger of reaching the producibility limit* given the dramatic increase in complexity, size and adaptivity required for modern software intensive systems
  - Rapidly shifting requirements result in disillusionment with and abandonment of inadequate software methodologies
  - Software not easily configured for purpose and not adaptable to changing conditions

- **PAMS Strategy:** *Adapt to changing requirements throughout the s/w lifecycle*
  - At design time to evolving requirements as well as to evolving domain concepts
  - At load time to mission, context, resources
  - At runtime to mission changes and unforeseen conditions

**PAMS Approach:** Develop an innovative *methodology* and a *suite of tools* that enable rapid, model supported adaptation
Program Approach

Three new software producibility techniques

July 2007 - September 2010

**Design Time Adaptation (DTA)**
- **Tooling**: Support the spiral development process with tools that adapt while carrying forward existing engineering products (models, designs, etc.)

**Load Time Adaptation (LTA)**
- **Configuration**: Enable rapid adaptation of system to new platforms or mission profiles

**Run Time Adaptation (RTA)**
- **Control**: Create system software that can rapidly adapt to changing mission requirements, environmental events, and faults in the system

**Goal**: Reduce Development Time By 70% And Cost By 90%

**Diverse Experimental Domains**
- Electronic Flight Control System (FCVMS)
- Software Defined Radio

**Team**: BAE Systems (Prime); Vanderbilt University (ISIS); MIT (CSAIL)
Program Approach

Three new software producibility techniques

July 2007 - September 2010

**Design Time Adaptation (DTA)**

Enable rapid adaptation of system to new platforms or mission profiles.

**Load Time Adaptation (LTA)**

Create system software that can rapidly adapt to changing mission requirements, environment changes, and faults in the system.

**Run Time Adaptation (RTA)**

Goal: Reduce Development Time By 70% And Cost By 90%

Focus of this presentation

**Diverse Experimental Domains**

**Tooling**: Support the spiral development process with tools that adapt while carrying forward existing engineering products (models, designs, etc.)

Team: BAE Systems (Prime); Vanderbilt University (ISIS); MIT (CSAIL)

Electronic Flight Control System (FCVMS)

Software Defined Radio
Design-Time Adaptation Producibility Problem

• Typical model-based environment for software development:
  • Modeling environments are used to design, analyze, and automatically generate code and executables
Design-Time Adaptation Producibility Problem

• **Problem:**
  • After significant modeling, an aspect of the application arises that cannot be adequately modeled

• **Underlying reasons:**
  • Modeling languages have limited expressivity
  • Long-running software projects do not adequately prepare for the evolution of the problem domain, tools, and processes
  • Very difficult to adapt languages & development tools or to integrate new tools

• **End results:**
  • Models are abandoned; developers move to handwritten code modification
  • Significant cost increase relative to full lifecycle use of model-based development
  • Alternatively, expensive rework is required to reproduce what has already been done once

![Diagram showing the comparison between Model-Based Development and Non-Model-Based Development]

- **Model-Based Development**
  - Modeling Language
  - Analysis tools
  - Modeling tool
  - Domain models
  - Code Generator
  - Executable components
  - Models abandoned
  - Changes made in code

- **Non-Model-Based Development**
  - Design inputs
  - Non-Model-Based Development
  - Manual development
  - Continued Development with Model Abandonment
PAMS Approach to the Producibility Problem

- **PAMS Solution:**
  - Co-evolve modeling language and model-based development tools
  - Result: Enable use of model-based development throughout lifecycle

- **PAMS Approach:**
  - Develop model change language (MCL) to specify modeling language changes
  - Develop migration tools that use these change models to synthesize software that automatically transform the existing engineering artifacts (models), and tools

- **Result:** Significantly reduced lifecycle costs
PAMS Tools for the Producibility Problem

- Flight-control domain specific meta-model / language (FCSL) is a multi-aspect modeling environment

Legend

- Created manually
- Created automatically
- Evolution
- GME-based PAMS tool

FCSL Meta-model created using Vanderbilt University’s Graphical Modeling Environment (GME)

Multi-aspect models
- System architecture
- Deployment modeling
- Fault modeling
- Specification modeling
PAMS Tools for the Producibility Problem

• Flight-control domain specific meta-model / language (FCSL) is a multi-aspect modeling environment

• Model-based interpreters graphically describe usage of information encoded in models

Interpreter synthesis based on Vanderbilt University's model-based graph re-writing tool GReAT

Auto-generated artifacts
■ Analysis reports
■ Synthesized software
■ Wrapper generators
  ▪ Code composition
  ▪ Tool integration

Legend
• Created manually
• Created automatically
E Evolution
GME-based PAMS tool

AFRL/RBCC – 2nd Annual Safe and Secure Systems and Software Symposium (S5), 15-17 June 2010
Distribution Statement “A” (Approved for Public Release, Distribution Unlimited)
PAMS Tools for the Producibility Problem

- Flight-control domain specific meta-model / language (FCSL) is a multi-aspect modeling environment
- Model-based interpreters graphically describe usage of information encoded in models

**PAMS tools:**
- MCL is a simple but expressive modeling language to specify migration rules describing meta-model evolution
- MCL interpreters synthesize software to automate model evolution based on MCL rules
- Universal Model Migrator (UMM) is a GUI-based model evolution engine
PAMS Tools for the Producibility Problem

- Flight-control domain specific meta-model / language (FCSL) is a multi-aspect modeling environment
- Model-based interpreters graphically describe usage of information encoded in models

**PAMS tools:**
- MCL is a simple but expressive modeling language to specify migration rules describing meta-model evolution
- MCL interpreters synthesize software to automate model evolution based on MCL rules
- Universal Model Migrator (UMM) is a GUI-based model evolution engine
- Interpreter Evolver (IE) is a GUI-based interpreter evolution synthesis engine

Legend:
- Created manually
- Created automatically
- Evolution
- GME-based PAMS tool
PAMS Tools for the Producibility Problem

• Flight-control domain specific meta-model / language (FCSL) is a multi-aspect modeling environment

• Model-based interpreters graphically describe usage of information encoded in models

• PAMS tools:
  • MCL is a simple but expressive modeling language to specify migration rules describing meta-model evolution
  • MCL interpreters synthesize software to automate model evolution based on MCL rules
  • Universal Model Migrator (UMM) is a GUI-based model evolution engine
  • Interpreter Evolver (IE) is a GUI-based interpreter evolution synthesis engine

Legend
• Created manually
• Created automatically
• Evolution
• GME-based PAMS tool

Co-evolve models and tools in sync based on simple rules
PAMS Tools Applied to the Verification Problem

• **Software verification problem:**
  - Largest cost and schedule driver for high-confidence software development programs
  - Powerful, state-of-the-art software verification tool chains and methodologies exist but have not been fully embraced by industry

• **Underlying reasons:**
  - Modeling the software abstraction for verification is an “additional / overhead” activity that typically require highly trained specialists
  - Automated verification tools generate an overwhelming volume of artifacts that are difficult to review, maintain and evolve

• **Result:**
  - Modeling, review and configuration management overheads may overwhelm automation benefit
PAMS Tools Applied to the Verification Problem

**PAMS approach:**
- Develop the behavior modeling language (BML) aspect of FCSL to model system specifications in the SCR formalism
- Apply BML to create our behavior models
- Develop the interpreter modeling language (TTM-meta) targeting the T-VEC wrapper representation
- Model the rules to create the BML2TTM interpreter that ingests BML models and produces T-VEC input files

**Result:**
- Tool chain that ingests SCR behavior specification models and then drives the T-VEC theorem-provers to automate test vector generation

---

The standard Software Cost Reduction (SCR) formalism is implemented by the T-VEC™ automated verification tool suite

T-VEC™ Tabular Model (TTM) is an extended concrete representation of the SCR syntax

Rapidly integrate existing tool chains effectively synthesizing wrappers for existing tools

Legend:
- Created manually
- Created automatically
- Evolution
- GME-based PAMS tool

AFRL/RBCC – 2nd Annual Safe and Secure Systems and Software Symposium (S5), 15-17 June 2010
Distribution Statement “A” (Approved for Public Release, Distribution Unlimited)
PAMS Tools Applied to the Verification Problem

- **PAMS evolutions:**
  - **Language evolutions to**
    - Improve comprehension
    - Manage complexity
  - **Tool chain evolution to reduce modeling effort**
    - Introducing the active pattern technology
    - Automating generation of model to artifact links
    - Supporting legacy code bases (in progress)
    - Developing generalized evolution technology by synchronously evolving related meta-models (in progress)

  **Language evolutions:**
  - Improve comprehension: Evolve BML to add the capabilities to link, group and hierarchically abstract models
  - Manage Complexity: Evolve BML to allow modeling constraints for the theorem-prover that guide the exploration and address the vector generation explosion problem

  **Tool chain evolutions:**
  - Pattern tool: Enable rapid reuse by allowing any model to be used as a parameterized template or pattern
  - Automated linking: Improve review efficiency through interpreter supported automatic model to artifact linkage
  - Legacy support: Apply DSML and source-to-source transformation technologies to automate transformation of existing code bases into models
Conclusion

• **PAMS success:**
  • Using the evolved tool chain develop reviewable verification artifacts automatically and efficiently
  • The tool chain and models remain flexible to continuous evolution

• **Next steps:**
  • Better support for evolving legacy code bases
  • Maturing the tool chains to improve understanding of the evolution impacts

---

**Program Is On Track To Meet Our Metric Goals**
Our customers

THANK YOU

We Protect Those Who Protect Us®

DISCLAIMER: The views, opinions, and/or findings contained in this article/presentation are those of the author/presenter and should not be interpreted as representing the official views or policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the Department of Defense.