Automated Performance Prediction for Model-Driven Engineering of Real-Time Embedded Systems

Dr. Connie U. Smith
Mark A. Smith
L&S Computer Technology, Inc.
Performance Engineering Services
(505) 988-3811
www.spe-ed.com

Overview

- Embedded Systems Modeling
- Software Performance Engineering (SPE) Overview
- Automating the Model-Driven Analysis
- Proof of Concept: Component-based RTES

Embedded Systems Performance

ES Software Industry Challenges

- ES revolution started in industry rather than universities
  - Common systems engineering problems haven't been scientifically addressed
- Shift from Hardware to Software ("softwareization")
- Dramatic increase in the complexity of functionality
  - Number of lines of code per function in aircraft systems was 10 in 1970, now 1,000,000
  - Increase in observable, controllable parameters
  - Trend to interoperability of ES in networks
- Growing gap between software size and developers' productivity

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE **MAY 2011**	2. REPORT TYPE	3. DATES COVERED **00-00-2011 to 00-00-2011**

4. TITLE AND SUBTITLE **Automated Performance Prediction for Model-Driven Engineering of Real-Time Embedded Systems**	5a. CONTRACT NUMBER 5b. GRANT NUMBER 5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER 5e. TASK NUMBER 5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **L&S Computer Technology, Inc,7301 Burnet Road # 102,Austin,TX,78757-2255**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S) 11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
Presented at the 23rd Systems and Software Technology Conference (SSTC), 16-19 May 2011, Salt Lake City, UT. Sponsored in part by the USAF. U.S. Government or Federal Rights License

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**11**	

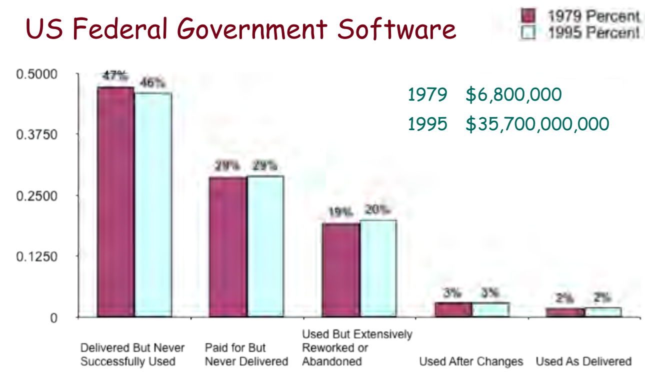
Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

Why Worry About Performance?

- Many systems experience performance problems on initial release
- Problems are often due to fundamental architecture or design rather than inefficient code
  - Introduced early in development
  - Not discovered until late
- "Tuning" code after implementation
  - Disrupts schedules and creates negative user perceptions
  - Results in poorer overall performance (than building performance into architecture)
  - May not be possible to achieve requirements with tuning
  - Increases costs

Value of Preventing Problems

US Federal Government Software

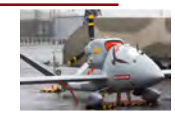


Lessons from History

Modernizing Telephone Switch Software

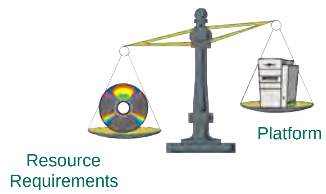
- Initial implementation of object oriented software resulted in significant performance problems
- Many OO telephony systems had the same performance problems (Software Performance Antipattern)
- Preventable with proper tools
- Risk of new technology and/or inexperienced personnel
- Problems likely to occur in initial MDE implementation for Embedded Systems

RTES/Analyzer Performance Modeling



- Automated assessment of software and systems architecture is essential
  - We cannot continue to build RTES with yesterday's methods
- RTES/Analyzer approach
  - Model interoperability
    - Automated translation of design models to performance models
    - Model solutions translated into meaningful results for developers
  - Adaptable, extensible evolution of tools

SPE Balance

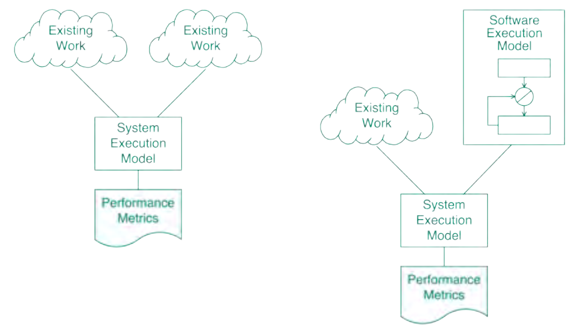


- Quantitative Assessment
- Begins early, frequency matches system criticality
- Often find architecture & design alternatives with lower resource requirements
- Select cost-effective performance solutions early
- Right-size the platform

SPE Models

System Models

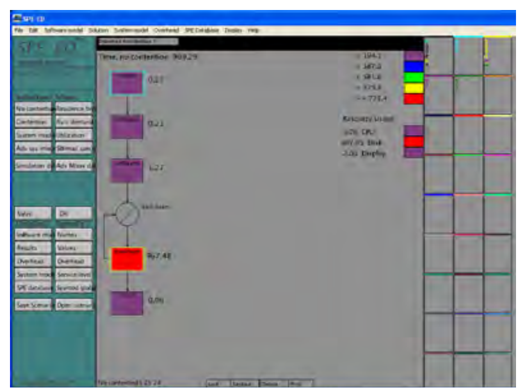
Software Prediction Models



SPE Model Requirements

- Low overhead
  - use the simplest possible model that identifies problems
- Goals:
  - initially distinguish between "good" and "bad"
  - later, increase precision of predictions
  - provide decision support

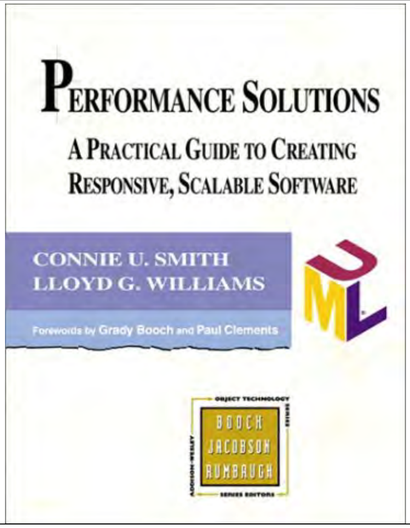
SPE·ED



- Tool for performance engineers
- Established technology
- Access to source code for R&D

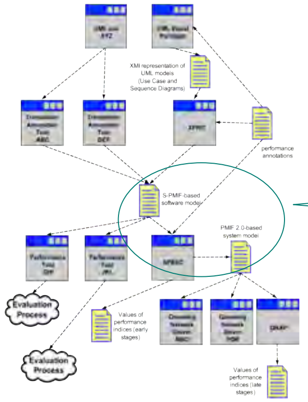
Additional SPE Topics

- Performance Principles
- Performance Measurement
- Performance Patterns & Antipatterns
- Architecture Assessment: PASA℠
- Business Case for SPE
- SPE Process



Part 2: Automating the Model-Driven Analysis

UML Design Models -> Performance Models



Model Interchange Formats (MIFs) streamline model interoperability process

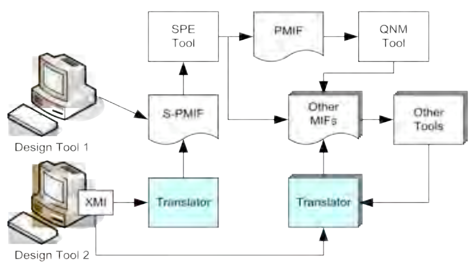
MIF Approach

- General approach to be used by a wide variety of tools
  - EIA EDIF/CDIF paradigm
  - Meta-model of information requirements
  - Transfer format based on meta-model
- XML implementation
  - Meta-model -> schema, transfer format in XML
  - Relatively easy to create
- Common interface
  - No need for n^2 customized interfaces between tools
  - Import/export can be external to tools with file interfaces

Our Model Interchange Research Results

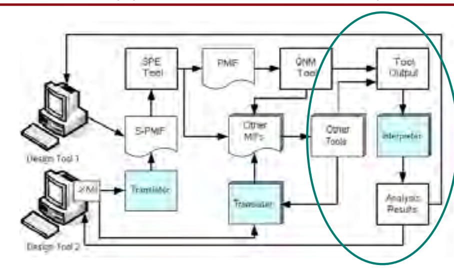
- Design tools to software performance models (S-PMIF)
- System performance models (PMIF)
- Model solutions
  - Experiments (Ex-SE)
  - Output metrics desired from experiments (Output-SE)
  - Transformation from output to tables and charts (Results-SE)

Previous Approach - Several Distinct Steps



- A proof of concept has been implemented for each step
- Each step was a separate, independent program
- Modeling expertise required limits usefulness for developers

Automated Approach for Developers



- Want to automate the end-to-end analysis steps:
  - Transformations, validation, experiment definition, and tool invocation,
  - Collect and present result data to developers for problem identification and diagnosis

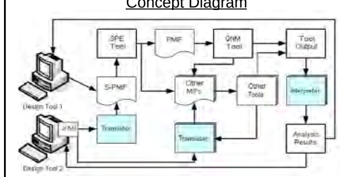
Vision: Developers Do Robust Engineering

- Explore options using familiar tools & notations (UML, Eclipse)
- Select candidate designs for exploration
- RTES/Analyzer
  - Select metrics
  - Specify analysis conditions and select tools
  - Quantitative predictions from multiple tools
  - Environment invokes analysis tool(s), collects output, prepares results in user-friendly format
  - Identify performance antipatterns
- Bring in performance specialists for serious problems

Robust Engineering of Large Distributed RTES

L & S Computer Technology

Concept Diagram



Objective

- •Robust Framework for automatic performance assessment of RTES
- •Translate designs to performance models
- •Define and execute experiments
- •Convert output metrics to meaningful results
- •Compare results from multiple tools
- •Ability to extend Framework with new analysis capabilities for developers
- •Automated studies (scalability, sizing, sensitivity, etc.)
- •Identify problematic design features and performance antipatterns

Approach

- •Based on Performance Model Interchange Formats (PMIF and S-PMIF) and tool import and export interfaces
- •Interoperability of heterogeneous software design and performance analysis tools
- •Develop Prototypes
- •Representative tools and services for end-to-end analysis from design to meaningful results
- •Real-time embedded system (RTES) focus
- •Ability to evaluate partial designs & combine designs
- •Demonstration – representative DoD RTES
- •Marketing, sales, commercialization plan

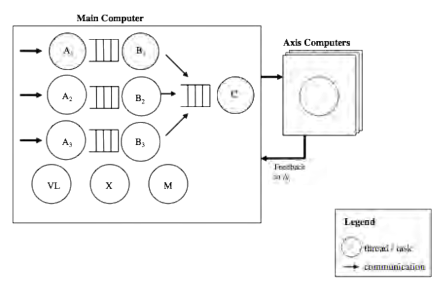
Impact/Milestones

- •FY09
- •Enabling technology complete
- •Architecture complete
- •Improved analysis capabilities can cut up to 95% of time required for (manual) performance analysis of designs
- •Automatically keep design and performance models in sync
- •Performance models keep pace with design changes
- •Eliminates manual comparison and re-creation of models
- •Ease of use increases likelihood of conducting performance studies early in lifecycle
- •Result: Better performing systems with optimally sized networks and platforms reduces hardware costs

Technology & Target Market: Analysts and Developers of Real-Time Embedded Systems (RTES)

Case Study

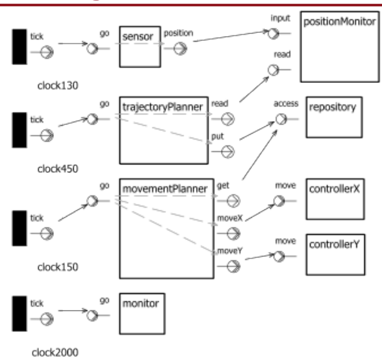
Robot Controller SEI Model Problem



- Main computer generates work orders
- Decomposed into subwork orders to axis computer(s)
- Interpreted by device drivers for movement of robot arms

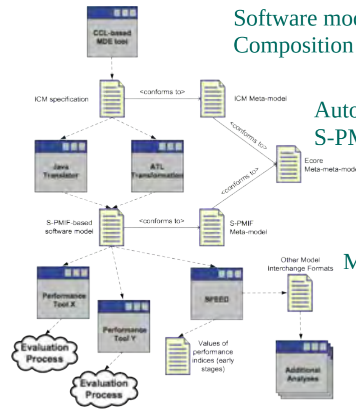
 24

Controller Design



- Movement planner cannot find repository empty
- Planners cannot miss deadlines at end of period

Component Architecture -> Performance Models



Software model: Construction & Composition Language (CCL)

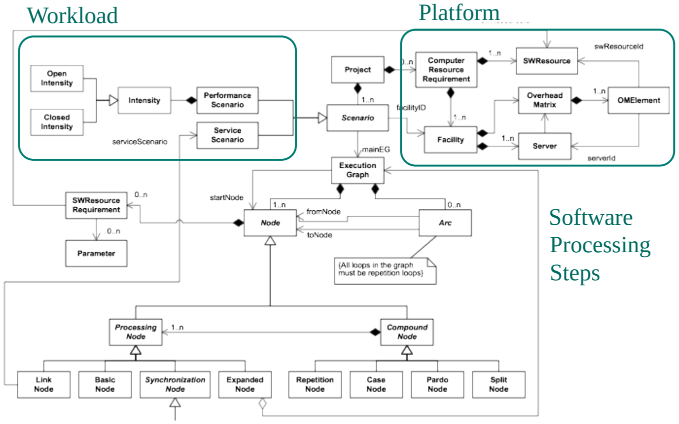
Automated Transformation to S-PMIF Performance Model

Model Solutions

S-PMIF MetaModel

Workload

Platform



Software Processing Steps

S-PMIF Excerpt

```
<PerformanceScenario InterarrivalTime="450.0"
MainEG="trajectoryPlanner.go" Priority="4"
ScenarioId="trajectoryPlanner.go"
ScenarioName="trajectoryPlanner.go" SWmodelfilename="icm">

<ExecutionGraph EGId="trajectoryPlanner.go"
EGname="trajectoryPlanner.go"
StartNode="N_trajectoryPlanner.go">

<BasicNode NodeId="N_trajectoryPlanner.go"
NodeName="N_trajectoryPlanner.go">
<SWResourceRequirement SWResourceId="R_CPU"
UnitsOfService="89.665066"/>
</BasicNode>
<SynchronizationNode
NodeId="N_trajectoryPlanner.read_positionMonitor.read"
NodeName="N_trajectoryPlanner.read_positionMonitor.read"
myType="SynchronousCall" partnerId="N_positionMonitor.read"
partnerScenario="positionMonitor.read"/>
```

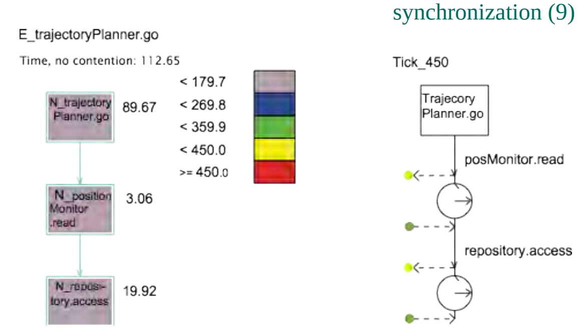
Performance Analysis

- Best and worst case analysis
- Simple model and advanced model with synchronization
- Multiple tools
  - Worst case latency - PSK performance-reasoning framework on linear sequence of actions
    - MAST tool - RMA technique
    - Discrete event simulator
  - SPE-ED tool

Software Performance Models

Simple: scenario per event source (4)

Advanced: scenario per thread with synchronization (9)



Model Results

Transaction	Best	Average	Worst
RMA Analytic			
clock130.tick	15.04		98.04
clock450.tick	112.65		262.77
clock150.tick	60.02		79.94
clock2000.tick	0.32		278.14
DE Simulation			
clock130.tick	15.04	33.71	75.08
clock450.tick	247.73	259.49	262.83
clock150.tick	60.02	60.00	60.04
clock2000.tick	0.32	103.08	278.20
SPE-ED Results			
clock130.tick	15.04	33.78	99.07
clock450.tick	112.65	259.67	262.77
clock150.tick	60.02	60.02	60.02
clock2000.tick	0.32	71.61	278.14

Results

- Simulation solutions comparable, not exact
  - DE simulation does not include contention
  - In best case, response to clock450.tick preempted twice by clock150.tick -> higher response time than no contention best case
- Simple, best case is optimistic
  - Identifies problems that must be corrected
  - Then proceed to more precise evaluations

Option

- Replace X and Y controllers with controllers that also provide position feedback to position monitor
  - Simple model: changes Clock150.tick to make +2 calls
  - Advanced model: changes ControllerX and ControllerY threads to make asynchronous calls to PositionMonitor.input

Revised Results

Transaction	Best	Average	Worst
RMA Analytic			
clock130.tick	15.04		124.06
clock450.tick	112.65		496.91
clock150.tick	86.03		109.02
clock2000.tick	0.32		431.24
DE Simulation			
clock130.tick	15.04	52.18	115.99
clock450.tick	314.80	347.63	431.04
clock150.tick	86.03	89.57	105.99
clock2000.tick	16.19	220.18	431.36
SPE-ED Results			
clock130.tick	15.04	46.51	208.16
clock450.tick	112.65	305.60	317.88
clock150.tick	86.03	90.08	192.65
clock2000.tick	0.32	128.68	413.30

Worst-case times differ:
SPE•ED computed average time for all calls to positionMonitor.input
RMA distinguishes between calls from different “clocks” - each has different response time due to pre-emption

Proof of Concept

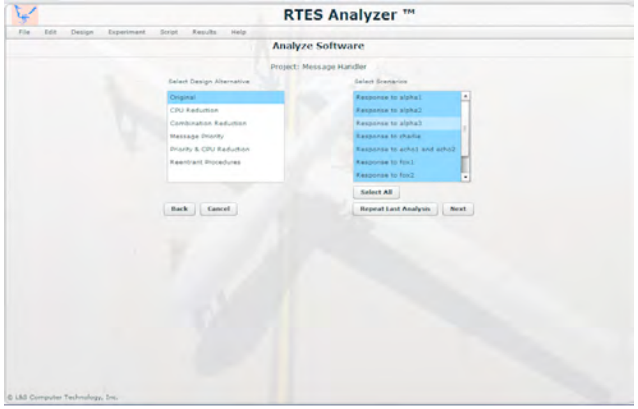
- Demonstrates viability of model interchange approach
- Builds on work in component-based systems, SPE, and model interchange
- Helpful to compare solutions from different software performance modeling tools
- Automation of steps simplifies performance assessment

Case Study Conclusions

- S-PMIF transformations can be procedural (custom code) or declarative Model to Model (M2M) transformations
- Enables performance analysis of CCL specifications with additional analysis tools without special integration efforts
- Demonstrates viability and ease of using S-PMIF with multiple design notations in addition to UML

RTES/Analyzer: Sample User Interface

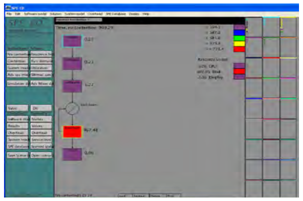
UI Demonstration



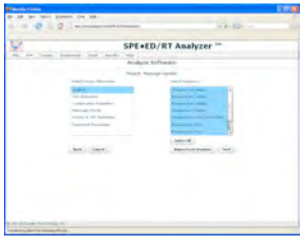
UI Demonstration

- Demonstrates ease of use for developers
- Selection of designs and experiments
- Meaningful results
- Flashbuilder foundation for Phase 2 implementation

SPE·ED -> RTES/Analyzer



->



- SPE·ED
  - Users are performance experts
  - Primarily IT systems
- RTES/Analyzer
  - Target developers as users
  - Focus on Real-Time & Embedded System market sector

RT/Analyzer Addresses Future Needs

- Cost
  - Ability to predict performance of designs reduces cost of re-work due to late discovery of problems
  - Up to 100 times more expensive to fix it later
- Quality
  - Systems meet performance requirements
- Automated Analysis
  - RT/Analyzer early detection of problems, performance ranking of solutions
  - Less expertise and shorter time for analysis
- Productivity
  - Quicker to build-in performance
  - Resources can be devoted to development rather than re-work

 41

Status

- RTES/Analyzer architecture and enabling technology are positioned for future development
- SBIR Phase II funding approved
- Developing prototype RTES/Analyzer to demonstrate the viability of automatic generation and evaluation of performance models, and presentation of quantitative results useful for developers
- Seeking comprehensive case study data
- Seeking partners to create commercial products

 42

Summary

- Embedded Systems Modeling
- Software Performance Engineering (SPE) Overview
- Automating the Model-Driven Analysis
- Proof of Concept: Component-based RTES

cusmith@spe-ed.com
http://www.spe-ed.com/