Proving the Shalls: The Future of Requirements?

Steven P. Miller, Ph.D.
Advanced Computing Systems
Rockwell Collins
400 Collins Road NE, MS 108-206
Cedar Rapids, Iowa 52498
spmiller@rockwellcollins.com

Who We Are

A World Leader In Aviation Electronics And Airborne/ Mobile Communications Systems For Commercial And Military Applications

Portfolio Composition

2002 Sales: $2.5 Billion

30% 25% 45%
Core Capabilities

- Communications
- Navigation
- Automated Flight Control
- Displays / Surveillance
- Aviation Services
- In-Flight Entertainment
- Integrated Aviation Electronics
- Information Management Systems

Formal Methods at Rockwell Collins

- Participants in the MCC Formal Methods Transition Study 1991
- Formal Specification of the µReal Time Executive in RAISE 1992
- Formal Verification of Microprocessors 1993 - 2003
  - AAMP5 Microcode Using PVS 1994
  - AAMP-FV Microcode Using PVS 1995
  - JEM Java Virtual Machine Microprocessor Using PVS 1998
  - FCP2002 Microcode Using ACL2 1999
  - AAMP7 Security Separation Kernel Using ACL2 2003
- Formal Validation of Embedded System Requirements 1995 - 2003
  - FGS Mode Logic using SCR's CoRe Method 1995
  - FGS Mode Logic using NRL's SCR Tools 1996
  - FGS Mode Logic Using PVS 1997
  - FGS Mode Logic Using Matrix-X and T-VEC 1998
  - FGS Mode Logic Using RMSL-X, PVS, and NuSMV 2002
  - FOSTRESIAT Logic Using SCADE and Simulink 2004

Methods and Tools for Flight Critical Systems Project

- Five Year Project Started in 2001
- Part of NASA's Aviation Safety Program
- Funded by the NASA Langley Research Center and Rockwell Collins
- Heavy Focus on Requirements Validation
- University of Minnesota is a Subcontractor to Collins
- Modeling Flight Guidance and Flight Management Systems
- Working with Commercial Tool Vendors to Ensure Technology Transfer
Model-Based Development Life Cycle

- Elicitation
- Modeling
- Simulation
- Autocode
- Autotest
- Analysis
- Reuse

Model-Based Development Examples

<table>
<thead>
<tr>
<th>Company</th>
<th>Product</th>
<th>Tools</th>
<th>Specified &amp; Autocoded</th>
<th>Benefits Claimed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Airbus</td>
<td>A340</td>
<td>SCADE</td>
<td>With Code Generator</td>
<td>70% Fly-by-wire Controls</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>70% Automatic Flight Controls</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>50% Display Computer</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>40% Warning &amp; Alarm Computer</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>50% Reduction in Errors</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Reduced Time to Market</td>
</tr>
<tr>
<td>Eurocopter</td>
<td>EC-155/135</td>
<td>Autopilot</td>
<td>With Code Generator</td>
<td>90% of Autopilot</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>50% Reduction in Cycle Time</td>
</tr>
<tr>
<td>Boeing</td>
<td>F/A-18</td>
<td>SCADE</td>
<td>With Code Generator</td>
<td>380,000 SLOC Auto Generated from 1,200 Design Views</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>20X Reduction in Errors</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Reduced Time to Market</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Increased Cost</td>
</tr>
<tr>
<td>GE &amp; Lockheed Martin</td>
<td>FADEC Engine Controls</td>
<td>ADI Beacon</td>
<td>With Code Generator</td>
<td>Not Stated</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>50% Reduction in Cycle Time</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Decreased Cost</td>
</tr>
<tr>
<td>Schneider Electric</td>
<td>Nuclear Power Plant Safety Control</td>
<td>SCADE</td>
<td>With Code Generator</td>
<td>200,000 SLOC Auto Generated from 1,200 Design Views</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>8X Reduction in Errors while Complexity Increased 4x</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Reduction Schedule &amp; Risk</td>
</tr>
<tr>
<td>US Spaceware</td>
<td>DCX Rocket Signaling System</td>
<td>MATLAB MATRIXx</td>
<td></td>
<td>Not Stated</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>50-75% Reduction in Cost</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Reduced Schedule &amp; Risk</td>
</tr>
<tr>
<td>PSA Electrical Management System</td>
<td>SCADE</td>
<td>With Code Generator</td>
<td>50% SLOC Auto Generated</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>60% Reduction in Cycle Time</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5X Reduction in Errors</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Increased Productivity from 25 to 300 SLOC/day</td>
</tr>
<tr>
<td>CSEE Transportation</td>
<td>Subway Signaling System</td>
<td>SCADE</td>
<td>With Code Generator</td>
<td>80,000 C SLOC Auto Generated from 1,200 Design Views</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Improved Productivity from 20 to 300 SLOC/day</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>No Coding Errors</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Received FAA Certification</td>
</tr>
<tr>
<td>Honeywell Commercial Aviation Systems</td>
<td>Primus Epic Flight Control System</td>
<td>MATLAB Simulink</td>
<td></td>
<td>60% Automatic Flight Controls</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5X Increase in Productivity</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>No Coding Errors</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Received FAA Certification</td>
</tr>
</tbody>
</table>

Elicitation of Requirements

- Reuse
- Elicitation
- Modeling
- Simulation
- Autocode
- Autotest
- Analysis
Simulation

Reuse Elicitation
Autotest Modeling
Autocode Simulation
Analysis

Simulation Demonstration

Using Formal Analysis for Early Validation of Requirements

Reuse Elicitation
Autotest Modeling
Autocode Simulation
Analysis
Theorem Provers
Model Checkers
Requirements
Safety Properties
Mode Confusion Properties
What Are Model Checkers?

- Breakthrough Technology of the 1990's
- Widely Used in Hardware Verification (Intel, Motorola, IBM, ...)
- Conduct an Exhaustive Search of the Global State Space
  - Consider All Combinations of Inputs and States
  - Produces a Counter Example if a Property is Not True
- Easy to Use
  - "Push Button" Formal Methods
  - Very Little Human Effort Unless You Are At the Tool's Limits
- Limitations
  - State Space Explosion ($10^6$ – $10^{30}$ States)
  - Awkward Notation for Specifying Properties (Temporal Logic)

---

Advantage of Model Checking

Testing Checks Only the Values We Select

Even Small Systems Have Trillions (of Trillions) of Possible Tests!

Model Checking Tries Every Possible Input and State!
Model Checking Using RSML-e and NuSMV

Does the system have property X?

Yes!

Automated Check

SMV Properties

Translated All the Shalls into SMV Properties

Only Two Types of Properties Were Needed

I. Safety Constraint Over All States

\[
AG(Is\_This\_Side\_Active \rightarrow (Mode\_Annunciations\_On \leftrightarrow (Onside\_FD\_On | Offside\_FD\_On = TRUE | Is\_AP\_Engaged)))
\]

II. Constraint Over All States and All Next States

\[
AG((\neg Onside\_FD\_On \& \neg Is\_AP\_Engaged) \rightarrow AX(Is\_AP\_Engaged \rightarrow Onside\_FD\_On))
\]
Validate Requirements through Model Checking

- Proved Over 280 Properties in Less Than an Hour
- Found Several Errors
- Some Were Errors in the Model
- Most Were Incorrect Shalls
- Revised the Shalls to Improve the Requirements

What are Theorem Provers?

- Available Since Late 1980’s
  - Widely Used on Security and Safety-Critical Systems
- Use Rules of Inference to Prove New Properties
  - Also Consider All Combinations of Inputs and States
  - Also Equivalent to Testing with an Infinite Set of Test Cases
  - Generate An Unprovable Proof Obligation if a Property is False
- Not Limited by State Space
  - Applicable to Almost Any Formal Specification
- Limitations
  - Require Experience - About Six Months to Become Proficient
  - Constructing Proofs is Labor Intensive

Theorem Proving Using PVS

Does the system have property X?

Model

Engineer

PVS Properties

Guru

Automated Proof

Automatic Translation

PVS Spec.

Properties

PVS Properties
Validate Requirements Using Theorem Proving

- Proved Several Hundred Properties Using PVS
- More Time Consuming than Model-Checking
- Use When Model-Checking Won't Work

Strengths and Weaknesses of Specification Styles

<table>
<thead>
<tr>
<th></th>
<th>Natural Language</th>
<th>Property Based</th>
<th>Constructive Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ambiguity</td>
<td>Likely</td>
<td>Eliminated</td>
<td>Eliminated</td>
</tr>
<tr>
<td>Inconsistency</td>
<td>Likely</td>
<td>Possible</td>
<td>Eliminated</td>
</tr>
<tr>
<td>Incompleteness</td>
<td>Likely</td>
<td>Possible</td>
<td>Eliminated</td>
</tr>
<tr>
<td>Implementation Bias</td>
<td>Possible</td>
<td>Possible</td>
<td>Likely</td>
</tr>
</tbody>
</table>

Early ➔ Life Cycle ➔ Late

Approach to Requirements Validation

- Reuse
- Elicitation
- Informal Properties
- Modeling
- Constructive Model
- Simulation
- Customer Validation
- Analysis
- Formal Validation
**Conclusions**

- **Model-Based Development is the Industrial Use Formal Specification**
  - Providing the Modeling Language Has Well Defined Formal Semantics

- **Convergence of Model-Based Development and Formal Verification**
  - Key is to Get Engineers Producing Specifications that Can be Analyzed

- **Need Several Approaches to Formal Verification**
  - Model-Checking Because it is Simple and Easy to Use
  - Theorem Proving for When Model Checking isn’t Practical

- **Constructive Requirements Models are a Useful**
  - Executable, Consistent, and Complete
  - Autogenerate Code and Test Cases

- **Shalls are Just Informal Property Based Specifications**
  - Easy Way to Elicit an Informal Description of the Requirements
  - Validate Constructive Model by Proving the Shalls!