Levi Malott, Pasha Palangpour

SMALL SPACECRAFT SOFTWARE MODELING: A PETRI-NET BASED APPROACH
Problem Overview

Identifying software defects early is imperative for delivering products cost effectively, on time, and correctly.
Presentation Overview

• Background
  – Problem Scenario
  – Existing Techniques
  – Our Approach

• Petri Nets
  – Description
  – Examples
  – Advantages

• Conclusions
Problem Scenario

• Software failures are often disastrous
  – Mars Polar Lander, Mars Climate Orbiter, Mars Global Surveyor
• Model-based and formal techniques emerging to combat this problem
  – Involve constructing models that completely describe a software system
• Formal verification of software can be impossible due to state-space explosion.
Existing Techniques

• SPIN
  – No real data types, minimal abstraction techniques

• Simulink
  – Deterministic execution, no deadlock detection, no underlying formal semantics

• UML
  – Static, hard to interpret
Premise of Our Approach

• Develop small spacecraft software model
  – Distributed, concurrent, nondeterministic
• Use the software model for
  – Visualization and interaction
  – Simulation
  – Verification
  – Communication
  – Implementation code blueprints and generation
• Remove software defects at design level
High-level Petri Nets (HLPNs)

• “Formal, graphical, executable technique for the specification and analysis of concurrent, discrete-event dynamic systems.”
• Constructs such as entry/exit points, global buffers/resources, and advanced data types.
  – Closely related to programming languages
• Multiple open source tools
  – CPN Tools \textit{recommended}
  – SNAKEs
  – Snoopy
Example (I)
Advantages

• Quickly develop modular, formally verified small spacecraft software
• User-friendly software visualizations
• Interactive software design
• Remove design errors, simulate execution
• Hierarchical abstractions
Conclusion

• Petri nets reduce the risk of system defects
• Small satellites inherently reduce the risk of state-space explosion.
• Petri nets effectively describe the complexity of spacecraft software.
• Subsystems can be modeled independently, but verified together.
Acknowledgements

Frank J. Redd  Student Scholarship Competition

Dr. Hank Pernicka

Dr. Sriram Chellappan

National Science Foundation

University Nanosat Program

lmnn3@mst.edu