Software Tools for Building Assured Software Systems

Connie Heitmeyer
Center for High Assurance Computer Systems
Naval Research Laboratory
Washington, DC

TH35: Software-Based Tools for the
Development of Digital Safety Systems
March 15, 2011

NRC's 25th Annual Regulatory Information Conference
North Bethesda, Maryland

OUTLINE

- The current state of software
- Tools for building assured software
- Applying tools to practical systems: Examples
- Providing evidence of software assurance
  - E.g., assurance of safety
- Future directions in software development
- Summary and conclusions

THE CURRENT STATE OF SOFTWARE

SOFTWARE IS EVERYWHERE

Dramatic Increase in Size and Complexity of Software in Industrial Systems

Example: Automobiles

- In 1981, GM passenger cars contained ~50 KLOC
- Today's *average car* contains more than 1 MLOC
- Today's *premium class car* estimated to contain100 MLOC!

Automotive Functions Supported by Today's Software

Air Bag System	Antilock Brakes	Automatic Transmission
Alarm System	Climate Control	Collision Avoidance
Cruise Control	Communication System	Dashboard Instrum.
Electronic Stability Control	Engine Ignition	**Engine Control**
Electronic Seat Control	Entertainment System	Navigation
Power Steering	Tire Pressure Monitoring	Windshield Wiper Control

Dramatic Increase in Software Complexity: Military Systems:

Software in Military Aircraft

Year	Aircraft	% of Pilot Functions
1960	F-4	8%
1982	F-16	45%
2000	F-22	80%
In Testing	F-35	90%

SOFTWARE FAILURES IN DEFENSE SYSTEMS

Details Emerge On Army's Failed NLOS-LS Missile

In testimony before lawmakers yesterday, David Duma…detailed failings of the Army Non-Line of Sight Launch System (NLOS-LS). During most recent tests in February, new navigation software caused six of seven total system aborts.

Defense Tech, April 16, 2010

A U.S. soldier in Afghanistan used a Precision Lightweight GPS Receiver to set coordinates for an air strike. Seeing that the "battery low" warning light was on, he changed the battery, then pressed "Fire." The device was designed, on starting or resuming operation after a battery change, to initialize the coordinate variables to its own location…

The soldier and three comrades were killed in the incident.

" 'Friendly Fire' Deaths Traced to Dead Battery: Taliban Targeted, but US Forces Killed," *Wash. Post*, 22 Mar. 2002

The New York Times

Navy Drone Violated Washington Airspace

By ELISABETH BUMILLER
Published: August 25, 2010

WASHINGTON — The skies over the nation's capital are crowded with presidential aircraft, military flyovers and the Delta shuttle, but this month a strange new bird was briefly among them: a United States Navy drone that wandered into the restricted airspace around Washington before operators could stop it. **The Navy is calling the problem a "software issue" that foiled the drone's operators.**

MANY OTHER EXAMPLES OF SOFTWARE FAILURES*

Aviation: Many recent incidents show risks due to software

- 1997 crash of Korean 747 in Guam: 200 deaths that probably could have been avoided if altitude warning system had been configured correctly
- 2004 air traffic control outage, Palmdale, CA: disrupted 800 flights
  - Prevented any voice communications between controllers and aircraft
  - Aircraft violated minimum separation distance five times; only because of airline collision avoidance systems that no collisions occurred

Medical Devices: Failures in medical devices can be lethal

- From 1990-2000, safety recalls of pacemakers and implantable defibrillators due to software problems affected over 200,000 devices
- FDA study: From 1992-98, 242 of 3,140 device recalls were due to faulty software (80%+ due to defects introd'd in maintenance)

*Jackson+, Software for Dependable Systems: Sufficient Evidence?, NAS, 2007.

NEW AND UPGRADED NPPS CONTAIN MUCH MORE SOFTWARE!

- NPP digital safety systems are experiencing the same growth in software size and complexity
- Under development are new and upgraded NPPs with much more software control and automation
  - Highly complex systems becoming even more complex
  - Many new features being added
- Of major concern: The **safety** of new and upgraded software in NPPs

SAFETY IN NPP SOFTWARE

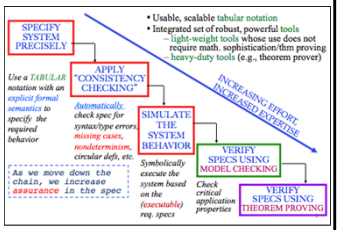
CHALLENGE ➔ How to obtain assurance that complex NPP software satisfies critical safety properties

How can NPP vendors learn from the experiences of others in the software community and not repeat the same mistakes?

TOOLS FOR BUILDING ASSURED SOFTWARE SYSTEMS

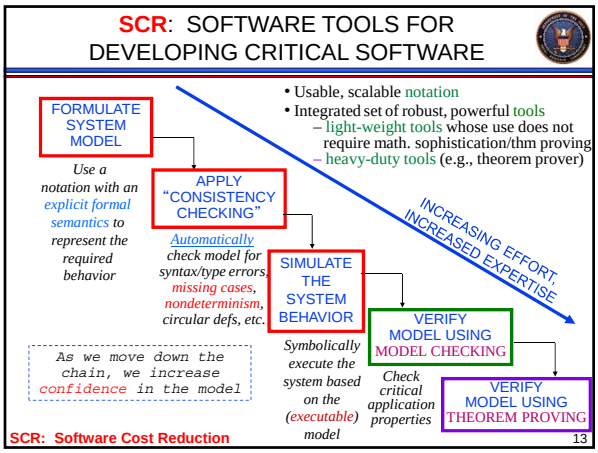
System Modeling & Analysis: APPROACH

- We have developed a mathematically based method called **SCR** for modeling and analysis of critical systems (Heitmeyer+, 1996), (Heitmeyer+, 2005).
- The method provides software developers with tools for
  - **building** a system model
  - **simulating** the system behavior by executing the model
  - **analyzing** the model for properties of interest



SCR: Software Cost Reduction

SCR: SOFTWARE TOOLS FOR DEVELOPING CRITICAL SOFTWARE



SCR: Software Cost Reduction

13

APPLYING TOOLS TO PRACTICAL SYSTEMS: EXAMPLES

USING TOOLS TO MODEL AND ANALYZE PRACTICAL SYSTEMS

EXAMPLES

Analysis of

1. completeness and determinism in software models for two avionics systems
2. safety properties in software for a Navy system

3/20/2012

15

MODELING + AUTOMATED CHECKING OF A NAVY SOFTWARE SPEC

Consistency checking of the A-7 requirements document

- Document contains a complete spec of the requirements of the **A-7 aircraft's Operational Flight Program**
  - Checked manually for errors by two independent review teams
- Results of applying our consistency checker
  - Check of 36 function definitions
    - Results: 17 missing cases detected
  - Checked a total of 4319 logical expressions
    - Results: 57 instances of non-determinism detected

Example: Input that could trigger transition from Inertial mode to either Doppler_Inertial or Air_Alignment mode

```
Doppler_up' WHEN [NOT CA_stage_complete AND
latitude > 70 deg. AND NOT present_position_entered
AND NOT latitude > 80 deg. AND IMSMODE=Gndal]
```

Consistency checking finds MANY errors that human inspections miss and does so very quickly (seconds to minutes)

3/20/2012 16

MODELING + AUTOMATED CHECKING OF A COMMERCIAL SOFTWARE SPEC

Rockwell Aviation's Flight Guidance System (FGS)

- Experimental application of SCR tools by Rockwell
- Despite extensive reviews by Rockwell, the tools found many errors in the FGS spec
  - 28 errors detected, "many of them significant"
  - one third each: 1) constructing the spec, 2) applying the consistency checker, and 3) simulating the system behavior

Example: Disjointness error leading to two possible flight modes

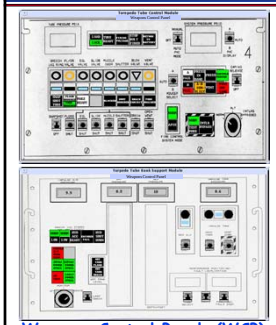
Example: Missing cases (Lateral Armed Annunciation field undefined in certain cases)

"...preliminary execution of the specification and completeness and consistency checking [with the SCR tools] has found several errors in a specification that represented our best effort at producing a correct specification manually."

Steve Miller
Rockwell-Collins Aviation

17

MODELING + AUTOMATED CHECKING FOR SAFETY PROPERTIES



Weapons Control Panel (WCP) for a Navy system

WEAPONS CONTROL PANEL

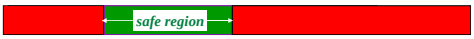
- Used to monitor the status & prepare the launch of weapons
- Sizable, complex program (~30 KLOC)
- System inputs
  - switches and dials
  - numeric quantities (read by sensors)
- System outputs
  - lights
  - doors and valves (set by actuators)
- Contractor spec of WCP software requirements contains 250+ variables
- Required to satisfy **six safety properties**

3/20/2012 18

CHECKING THE WCP MODEL FOR SAFETY PROPERTIES: RESULTS

One of the six safety properties

Opening the Torpedo Tube Vent Valve shall be prevented unless the Missile-to-Torpedo-Tube differential pressure is within safe limits



Analysis of the WCP model with our tools showed that the model violated all six safety properties!

- Finding these violations manually would have been extremely difficult since the model is so large
- For large models, need software tools

CHECKING THE WCP MODEL FOR SAFETY: REQUIRED EFFORT

TASK	PERSON WEEKS
Translate contractor spec into SCR	0.8
Use tools to detect syntax and type errors, missing cases, etc.	0.2
Correct errors	0.3
Detect safety violations	0.8
TOTAL	2.1

Small effort required was surprising given that

- The contractor model was large and complex
- The contractor had no prior knowledge of the SCR notation and method

PROVIDING EVIDENCE OF SOFTWARE ASSURANCE

ROLE OF TOOLS IN BUILDING ASSURED SOFTWARE

- A model provides a basis for tool-based analysis
  - Consistency checkers detect missing cases/non-determinism
  - Simulators can help validate the model
  - Verifiers can prove properties of the model or show that the model does not satisfy desired properties
- Using tools exposes errors that humans miss, e.g.,
  - Missing cases
  - Unwanted non-determinism
  - Violations of critical properties, such as safety properties
- Unlike humans, tools can analyze large, complex models
- Using a tool to construct a model can expose errors
  - Process of creating a model often exposes errors
    - Example: Errors detected in formulating models of NASA software

WHILE TOOLS ARE USEFUL, THEY ARE NOT ENOUGH

- A natural language version of the model and properties can be extremely valuable
  - For validating the model
  - For validating the assumptions
- Discussion of the model and properties with stakeholders can lead to important modifications
  - Remove/Fix incorrect assumptions
  - Correct the model
  - Correct the statement of properties
  - Suggest new (safety) properties

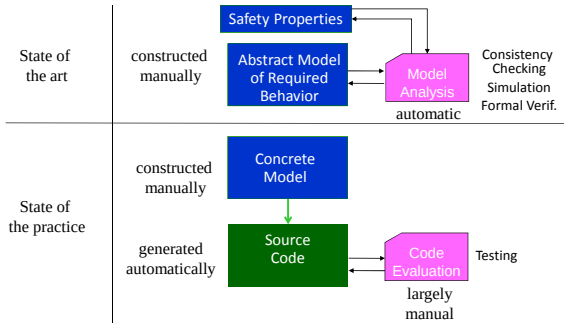
FUTURE DIRECTIONS IN SOFTWARE DEVELOPMENT

TRENDS IN SOFTWARE DEVELOPMENT: MODEL-BASED APPROACHES

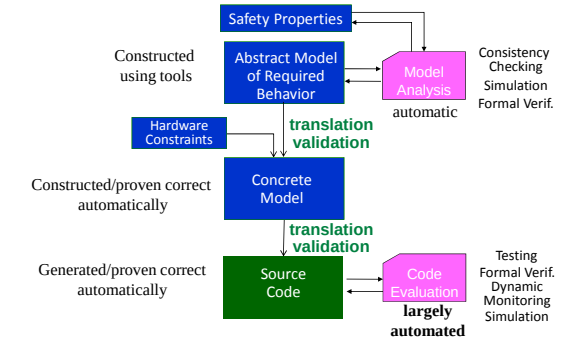
- Use of model-based software development growing
  - Military systems
  - Automobile Industry
- Goals: To provide
  - A precise statement of the required software behavior
    - A system model
    - The set of properties, e.g., safety, the system must satisfy
  - Software assurance
    - Software testing
    - System simulation
    - Formal proofs
    - Demonstration that behavior of software code corresponds to behavior specified by model

Different forms of evidence

MODEL-BASED SOFTWARE DEVELOPMENT: CURRENT STATUS



MODEL-BASED SOFTWARE DEVELOPMENT: FUTURE VISION



SUMMARY AND CONCLUSIONS

CONCLUDING REMARKS

- Tools can be extremely useful in developing, evaluating, and changing software for NPPs
  - Find errors human inspections miss (e.g., missing cases)
  - Help user validate a model
  - Detect safety violations
  - Support formal verification of safety properties
- Major contribution of tools: Liberate people to do the hard intellectual work required to build high quality models + software
  - The "combination of human analysis and tool-based analysis is more powerful than either alone…" *John Rushby, SRI*
- Tools are not enough
  - Precise natural language versions of model are invaluable
  - Improved languages and models are needed

NEED FOR NEW RESEARCH

- Model-based approaches are increasingly being used to develop software
  - Military systems
  - Automobile Industry
- New research and technology is needed
  - Tools to aid users in eliciting system and software requirements
  - Tools to help users build models
  - Tools for proving that software code satisfies a model or a set of specified properties