The Future of Software Certification - a Roadmap

Ewen Denney
Robust Software Engineering
NASA Ames Research Center
California, USA

Autocode assurance issues

- Commercial code generators historically buggy
  - despite extensive heritage, bugs still remain
  - bugs often impossible to detect at model level or via simulation
- Commercial code generators are black boxes
- Autocode difficult to understand and review
- Diverse sources of domain knowledge
  - mathematical, algorithmic
  - physical, engineering
- Models not good for expressing requirements

Autocode review documents

- Verification says *that* the code is safe
- Certification says *why* the code is safe
- *Review document* explains how code complies with requirements:
  - Chain of reasoning from assumptions to requirements
- Traces between code, documentation and V&V artifacts
- Based on *proof:*
  - for all possible inputs, if the safety assumptions hold
  - then for all possible execution paths,
  - the safety requirements hold.

Example: Coordinate systems

- Level 2 Coordinate Systems (CxP 70138):

  "*All pertinent geometric technical data … shall be in the coordinate systems described in this document.*"

- Problem:
  - Not directly represented in model or code
  - Transformations involve mathematical computations

AutoCert Demo

Summary

- AutoCert encodes and checks mathematical reqs
- Low to no false positives/negatives
- Make assumptions, data, equations explicit
- Traces code and model to verification artifacts
- Turns requirements into source code annotations
- Provides “oversight” of autocoder: IV&V
- Qualifiable: small kernel of trusted components
- Tight integration with Matlab tool suite
  - Minimal impact to existing process

Future work

- Greater domain coverage
  - More Simulink blocks/EML functions
  - Control law analysis
- More extensive documentation
  - Trace to external requirements
  - Safety cases
- Test case generation
- NExIOM integration

Other properties

- Execution safety
  - array bounds, variable initialization before use
- Representation conventions
  - consistent use of physical units
  - Euler angles: YPR vs RPY
  - quaternion handedness
  - time formats
- Dead code analysis

Traceability

- Traceability:

  "the ability to link requirements back to rationales and forward to corresponding design artifacts, code, and verification artifacts"

- "why is this line of code safe?"

  code → verification conditions → assumptions

- "how is this requirement satisfied?"

  property → verification conditions → code