Specification: The Biggest Bottleneck in Aerospace V&V and Autonomy

Kristin Yvonne Rozier
University of Cincinnati

May 6, 2016
Design-Time Verification!

- Expected **design-time component**
- Recommended in **DO-178B/C, D0-254 standards** for
- Successfully applied in many aerospace contexts...
Runtime Verification and System Health Management!

- Required for Autonomy
- New: Intelligent Interfaces
- Hot topic: UTM, Mars, NextGen, ...
A Goal Aerospace System Design Process

- System Design
- Model Check
- SPEC DEBUGGING
- Model Verification Specification
- USE SPECIFICATIONS FOR RUNTIME MONITORING
- YES
- NO
- ERROR

- Build Prototype
- Testing and Simulation
- ERROR
- NO

- SPEC DEBUGGING
- M = Formal System Model

- Specification: The Biggest Bottleneck in V&V and Autonomy

Laboratory for Temporal Logic
Kristin Yvonne Rozier

Successes
Challenges
Future Challenges

Error... Garbage in, garbage out!
A Goal Aerospace System Design Process

---

**Successes**

**Challenges**

**Future Challenges**

---

**System Design**

**Build**

**Prototype Simulation**

**Testing and**

---

**ERROR**

---

**DEBUGGING**

**SPEC**

---

**M = Formal System Model**

---

**Model Validation via Model Checking**

---

**Model Validation Specification**

---

**REVISE**

---

**Specification**: The Biggest Bottleneck in V&V and Autonomy

---

**Laboratory for Temporal Logic**

Kristin Yvonne Rozier

---

**... Garbage in, garbage out!**
The Bottom Line

Bottom Line: INPUTS to formal analysis are the BIGGEST challenge
Synthesis!

Model checking: check $M \models \phi$

Problems:

- Designing $M$ is hard and expensive
- Re-designing $M$ when $M \not\models \phi$ is hard and expensive
- Synthesis: start from $\phi$, design $M$ such that $M \models \phi$
  - Simplifies verification
  - No re-design
  - For algorithmic derivations: no design!

What about $\phi$?

We need LTL Genesis!

---

Some critical systems are designed **without** formal requirements.

Some design processes **don’t formally define** requirements until the testing phase.

Early specifications often **mix many different objectives**:
- levels of detail/abstraction
- how the system is defined vs how the system should behave
- legal-speak on why the system satisfies rules
- desires/opinions of designers
Specification Extraction Strategies

- **Human Authorship:**
  - Train system designers to write formal specifications first
  - Pair designers with formal methods team to write specifications

- **Natural Language Processing:** extract formal specifications from English Operational Concepts
  - Highly input-dependent: assumptions, implied/arbitrary functions
  - Hard to measure correctness, completeness

- **Specification Mining:** extract behaviors from existing systems

- **Static Analysis:** map all paths of a program
  - Hard to differentiate normal usage from exceptions

- **Learning/Dynamic Invariants:** analyze actual executions; observe use-cases

- **Specification Wizard:** Semi-automated exploration of system facets, guided by human input

---

Specifications for Free?³

- Combine specification mining, test-case generation, static analysis, and dynamic invariants to extract specifications automatically!
- Can use specifications mined from code
  - Specification validation == software defect detection
  - Promising for software runtime verification
  - Still need code...
    - What about early design time?
    - What about cyber-physical system specifications?
- Can use specifications extracted from last version for new designs
  - Challenges with specialization/levels of abstraction/relevance
- Other challenges:
  - Scalability
  - Efficiency
  - Expressiveness

How should we measure specification quality?

- How can we know when we’re done?
- How good are the specifications?
- How can we measure the completeness, correctness, coverage, or general quality of a set of specifications?
Sanity Checks

- Satisfiability
- Vacuity
- Realizability
- Coverage
How do we best use specifications?

- Design lifecycles for different cyber-physical systems?
- How to indoctrinate formal specification into diverse teams of system designers?
- Barriers to adoption:
  - time to write/validate
  - learning curves
  - culture
- Need an end-to-end process for specification extraction, usage

What should our roadmap look like for a future full of well-specified (formally analyzable) critical systems?
Specification Usage

**Specification Use Strategies**

- **Property-Based Design**: from specifications to systems

- **Synthesis**: generate $M$ such that $M \models \phi$
  - For cyber-physical systems?

- **Specification-Based Testing**: use specifications in test-case generation

- **From Design- to Run-Time**: carry specifications through the design cycle
  - Specification design lifecycle?

- **Maintenance**: using specifications in system up-keep
  - Maintenance of specifications?
Specification Challenges: to Infinity and Beyond!\textsuperscript{4}

- Where are we now?
  - Continuously re-assess . . .
- Where will we get specifications from?
- How should we measure specification quality?
- How do we best use specifications?

Specification Challenges: to Infinity and Beyond!\(^4\)

- Where are we now?
  - Continuously re-assess . . .
- Where will we get specifications from?
- How should we measure specification quality?
- How do we best use specifications?

... in the context of cyber-physical systems?