Foundations for Ilities, Tradeoffs, & Affordability RT-113, CSER 2015

Kevin Sullivan
University of Virginia
(sullivan@virginia.edu)

Introductions

- Computer scientist
- Software-intensive cyber-physical-human systems
- Acknowledgements:
  - SERC, RT-113
  - NSF CMMI / Systems Science (SYS) #1400294
  - NSA Science of Security Lablet
- Co-authored 2014 SERC workshop report on the relationship between systems & software engineering
- RT-113, “Foundations”

Overview

- What are we trying to do?
- What is the state of the art and its shortcomings?
- What is our approach and why do we think it will work?
- If we succeed, who will (or should) care?

What are we trying to do?

Improve SE by developing & validating foundational scientific theory of ilities, tradeoffs, stakeholder value

- Not just formally mathematical but computable
- Both general and specializable (parameterized)
- Having both explanatory and predictive power
- Validated through use, feedback, evolution
- Two levels: tradeoff framework, design spaces

Key concepts

- stakeholders
- ilities (non-functional system properties)
- design space specifications
- design representations
- design synthesis functions (specification -> design reps)
- ility measurement functions (design reps -> ility values)
- value functions (design reps -> stakeholder value)

State of the art and its shortcomings

- Ilities
  - vague & inconsistent natural language definitions
  - lack general, validated, ility measurement functions (and ways of specializing them to given design reps)

- Design spaces & stakeholder value
  - lack general specification notations & methods
  - and frameworks for synthesizing, analyzing spaces

Our approach

- Formal, abstract, computational specification
  - constructive logic (e.g., Coq)
  - relational logic (e.g., Alloy)
  - process algebras (e.g., TLA+)
- Automated software synthesis from specifications
  - analysis frameworks from Coq specifications
  - spaces of design representations from Alloy specs
  - ility measurement functions from Coq specifications
- Web services for use, interoperability, & validation

Why we believe it will work

- Ility models
  - Ross et al. / MIT model of change-related ilities
  - Coq-based formalization of informal model
  - REST web service synthesized from Coq spec
  - Facilitated use, critique, evolution of MIT concept
- Design space specification, synthesis, and analysis
  - Relational logic encoding of UML class models, SQL storage designs
  - Use of relational logic model finders for design space synthesis
- Theory of tradespace analysis
  - Parameterized Coq specification
  - Synthesized map-reduce-based framework
  - Design space spec -> synthesizer -> measurements -> tradespace

Insights

- Formal languages enable expressions of concepts critical to systems engineering with high precision and generality, in “computable” forms
- Scientific foundations for ITaP should ultimately best expressed in this way
- Communicating the science for practical evaluation, critique, evolution, and validation requires packaging it in computable frameworks and services
- Software synthesis supports co-evolution of science and implementations
- A useful theory of design space structure, key to reasoning about such properties as evolvability and resilience, clearly requires use of abstract, formal specification notations and synthesis techniques
- The proof engineering capabilities of Coq are rapidly gaining visibility for their ability to enable rigorous reasoning about system properties, notably including security

Who cares (or should care)?

- Practitioners, Researchers, Agencies, Industry, Users
- Provides a path for systems engineering to advance state of the art by exploiting enormous advances made in CS and software engineering over past few decades
- Provides a way out of interminable definition wars by replacing vague definitions of ility terms with precisely defined families of ility measurement functions over families of system models
- Provides engineering foundations for new interoperable frameworks and tools for practical ility/tradeoff analysis

Conclusion

- The problems are important and historically intractable
- Current informal approaches just aren’t good enough
- A new approach to systems engineering foundations exploiting sophisticated methods of formal specification and software synthesis shows real promise
- As systems properties come to derive ever more directly from software, formal methods of software specification, synthesis, and analysis, including proof engineering (using tools such as Coq), will become vital to sys eng.