Software for the Real World

Don Winter

Martin Feather, Gabor Karsai, Patrick Lardieri, Cleve Moler, Edward Lee

Workshop on New Visions for Software Design and Productivity
18-19 April, NSF, 2001

Where we are Today

- In software today:
  - Accidental complexity
  - Unpredictability
  - Uncomposability
  - Brittleness
  - Not good at interacting with the real world
- We are stuck in a “lines of code” view of SW
  - Scaling this by putting more people onto it
  - Lose understanding of the system as a whole
  - Separation of specification from construction

The Problem

- In 10 years, the world is going to be a different place… there are going to be computers everywhere.
- If we keep developing software the way we do today, then the world is going to be a very dangerous place…
- or, technology infusion will slow… Your car will not be drive-by-wire, and you’ll still be stuck in traffic.

Modeling “Languages”

- Lacking modeling “languages” for humans to
  - realize complex functionality
  - understand the design
  - formulate the questions
  - predict the behavior

  The issue is not lines of code
  “model” or “design” not “specification”

- Invest in:
  - Modeling “languages” for *systems*
  - finding the useful abstractions
  - computational systems theory
  - composable abstractions
  - expressing time, concurrency, power, etc.

Composing Systems

- Lacking systematic methods for composing systems
  - component frameworks
  - composition semantics
  - on-the-fly composition, admission control
  - legacy component integration

- Invest in:
  - semantic frameworks and theories
  - methods and tools
  - experimental testbeds & challenge problems
  - reference implementations
  - defining architectural frameworks
  - strategies for distribution, partitioning
  - strategies for controlling granularity and modularity

Transformations

- **Lacking theory of transformations between abstractions**
  - relationships between abstractions
  - generators (transformers)
  - multi-view abstractions
  - model abstractors (create reduced-order models)
  - abstractions of physical environments
  - connection with HCSS: verifiable transformations

- **Invest in:**
  - open generator infrastructure (methods, libraries)
  - theories of generators
  - methods for correct by construction transformers
  - parametrized transformers

Legacy

- Lacking methods for dealing with legacy
  - how to incrementally modernize systems
  - lacking methods for integrating new with old
- What to invest in:
  - componentizing legacy code
  - extracting abstractions of legacy systems
  - incremental modernization, reverse engineering, or…
  - make it cheaper to redesign vs. evolve legacy systems