The Future of UTP

Jim Woodcock (York)

UTP 2008 Dublin

Contents

The future of UTP

- UTP community very small
- diverse interests
  - miracles, multirelations, general correctness, parallel assignment, temporal logic model checking, mechanisation, transaction calculus, conformance testing, event-driven systems, components, interrupts, synchronicity, refinement calculus, hardware semantics, mobility, laziness
- **three questions:**

1. **why do we bother?** — **vision**
2. **where are we going?** — **roadmap**
3. **what will our overall contribution be?** — **collaboration**

How do we build the community?

- **education**
  - why don't you teach UTP?
  - open course materials
  - range of differnt courses: breadth & depth
  - annual summer school
- **research**
  - **how does my work fit in?**
  - we need a model for
    - collaboration
    - constructive competition
    - incremental research

The Grand Challenge on Verified Software

- **deliverables**
  1. **a comprehensive theory of programming**
     - all features needed to build practical and reliable programs
  2. **a coherent toolset**
     - automating the theory and scaling up to large codes
  3. **a collection of verified programs**
     - replacing existing unverified ones
     - continuing to evolve as verified code
     - a repository

qpq.csl.sri.com/vsr/private/theory/theoryreport.pdf

Research directions in UTP

1. theory of pointer structures [Chen, Qin]
2. theory of object orientation
3. theory of infinite data structures
4. mechanised theory and theorem prover [Chen]
5. theory of information confinement [Fidge]
6. cross-platform and cross-language compatibility
7. theory of synchronous systems [Butterfield]
8. theory of probabilistic programming [Butterfield, Chen]
9. theory of rigorous software testing [Aichernig]
10. theory of component-based software systems [Liu]
11. unification of more programming paradigms
12. unification with industrial design techniques

A proposal

- **UTP e-book: a repository for theory, tools, and examples**
  - build on Cavalcanti, Oliveira, & Zeyda's ProofPowerZ work
  - 500+ theorems, 80,000 lines of proof scripts
  - mechanise the rest of Hoare & He
  - **community activity:** agree, plan, and then add new chapters
  - **mechanise everything, make progress**
  - requires cooperation and collaboration
- **develop the UTP Symposium**
  - **focused call-for-papers, archival special journal issues**
  - **full papers**
    - mechanical proof
  - **work in progress**
    - hand-written proof

Why we shouldn't do this

- I'm not interested in collaboration, I just want to do my thing
  - carry on, you don't have to be part of the community
- I can't work with X
  - get over it
  - we must stop denigrating the achievements of our predecessors and rivals—a very deleterious habit in Computer Science
- it's far too ambitious
  - that's precisely why we should do it

Index