Words of wisdom

Bob Cook, Ansgar Fehnker,
Zonghua Gu, Dong Jia, Insup Lee,
Bill Milam, Joel Ouaknine,

Observations

- Embedded systems are different than …
- Domain is difficult (e.g., physical aspects such as friction, thermodynamic, not capture by UML)
- Need to come up with languages that are relevant/close to domain of applications
- Cross-fertilization between computer scientists and domain experts.

Integrated environment

- End-to-end information capture and traceability: Linkage traceability among requirements, specification, implementation
- Requirement for higher level of integration between software development environment (physical environment) and embedded systems (e.g., debugger, testbed)
- Need model interchange standards based on formal semantics

Requirements capture

- New requirement driven by regulatory requirements, customer expectations
- Need right/natural languages for initial capturing of requirements:
  - Pattern based language, Restricted NL, NL
- Analyzable requirements
  - Checking inconsistency, holes across all the requirements (i.e., inter-requirements consistency checking)

Interoperability between tools

- Single standardized language possible?
  - Incentive not to be locked into a tool vendor.
  - Unified model format?
    - HSIF (hybrid system interchange format)
    - Hybrid systems research still immature, makes it harder.
  - Hard problem, semantics different between tools.
    - Formal semantics
- Tools must fit into existing processes, incremental change to processes.
  - Cannot start a new division for a new product line.
  - Don't trust code generators, rely on testing, hand coding.
  - Model-level testing vs. code-level testing (complementary, relationships?)
  - Hierarchical design, high-level description of a piece of legacy code.

Metrics? Experimental Evidence

- Reduced TTM?
- Do additional tests have an impact on quality?
  - Impact on errors per KLOC, TGW (things going wrong)?
- Cost?
- Industrial partnerships essential.
- May be subjective, depends on person using the tool.

Documentation

- Docs should be 1st class citizen! Not a chore.
  - Start from homework assignments in college. Resolve ambiguities
- Not addressed by researchers.
- Minimize number of artifacts that need to be maintained, including documentation.
  - Automatic generation of documentation.
  - Literate programming? Doc and code mixed together. Only Don Knuth knows how to do it.
- New paradigm for documentation for mixing code and requirements together.
- Clear measurable benefit for good docs.
- Lack of repository of reusable software components
  - A repository would drive documentation standards.
- Not just docs for software, also engineering, regulatory…