What's special about embedded systems?

**observations**
- tighter interface to low-level issues (interrupts, etc.)
- cost & volume are (sometimes) major factors
- difficult to apply "separation of concerns"
- hard to apply "advanced" SE techniques

**research issues**
- environment
  - physical process in the loop
  - often uncertain
  - wide operating space
  - often untrained users
- unattended
  - highly reliable
  - secure
  - self-healing
- resource constrained (time, memory, power)
- hardware-software codesign
- user interface
  - modeling the operator
  - modeling the operator’s model of the system
legacy systems

observations
~90% functions a retained from old product
change inhibited by tight integration of features in ES
"maintenance" over the years confuses the design
side-effects dominate in mature code (whereas bugs
dominated in new code)
LSs were targeted to old HW/SW environments

research issues
need to reverse engineer
specifications/requirements/code
legacy & new code need to co-exist - difficult to make
guarantees about the composition (legacy code is part
of the environment)
requirements

observations
  hard to formalize all aspects of textual requirements
  some requirements can be translated into coding
  standards
  impediments to using more formal tools for
  requirements
    high initial cost
    resistance to driving everything from the
    requirements level (process of re-generating
    everything)
    can't make patches in the field (too expensive)
    forces a level of detail that may be too expensive

research issues
  need tools to assure consistent and complete
  requirements
  need auto-prompting for expanding requirements
  based on formal analysis
tool integration for end-to-end design

observed

need to understand the application-specific processes to integrate things correctly
missing interfaces
  systems engineering (physical system, HW/SW architecture)
  engineering design tools
"back end" gap: need to put results back into format that is compatible with current process

research issues

model/profile target platforms
identify/create aspects that are generic for ES design processes (avoid point solutions at a higher level)
languages (for formal methods)- what to do?
  finally find the universal language
  translate the "proof of concept" into accepted languages (re-write from scratch)
  point compilers into other languages to hide it from the user
migration paths important
this is an issue for commercialization
needed in point solutions to demonstrate value added
documentation

observations
  more handwritten documentation isn't what's needed
  BUT can't build support into ES software as with non-ES code
  multiple versions of documents & code proliferate
  technology for accessing documentation is a solved problem
  auto-upgrade of documentation is not implemented
  auto-generation of source code exists (after the fact)
  don't see particular issues for ES - except that perhaps
  some of the support tools are not there for low-level coding (assembly, drivers, etc.)

research issues
  connections between documentation of engineering & software designs don't exist
testing

*observations*
  current testing needs to go away

*research issues*
  what should testing be in the future and what's going to make that happen?
Closing Observation

accounting impacts what can be done in industry