HCES Discussion

Group 5
Tools in Practice

• Programmers are hired because of their domain knowledge, not their programming knowledge;
• Tools need to be usable by / specializable to domain knowledge experts.
Tools in Practice

• If you make tools supported by semantics, don't expect the users to use any part of the tool where the semantics is visible

• One approach to dealing with this it make it so that the application is totally automatic (even less than push-button) and to make management understand that using the semantic aspects of the tools.
Tools in Practice

• Too many tools are too rigid, too bound to a specific application;
• Need to be able to be adapted to each new domain
Refinement

• Need to have languages that can be successively be refined from specifications all the way to code and test suites.
Refinement

• If you had a hierarchy of refinements of requirements through system specifications to design and finally code, where there was maintained a "proof" that each level implies the next up, then, when changes were necessary, we would be able to chase the effects.
  – The refinements and justifications need to be incorporated in the documentation (with autoated support?).
Incremental Development

• Progress will generally be incremental, but we need to take increments in new directions.
• Need to have the ability to explore different design paths and to back out and change a design path previously chosen.
Incremental Development

• Changes in industry must be made incrementally, so even if we have radical new tools/approaches, they must be broken down to be phased in incrementally.
Programmatic Suggestion

• Suggest that there should be a program (or part of a program) to fund industry to talk to academia about the way their process works.
Legacy Artifacts

• The problem with legacy code is trying to understand what effects will occur when you make a change to the code.

• Program slices can help.
Legacy Artifacts

• For legacy code, there are two problems:
  – dealing with existing legacy artifacts, and
  – changing the way current code artifacts are generated and maintained to make them in such a way that they won't be problem legacy code in the future.

• As we generate solutions for problem 2), they may shed light on how to attack problem 1).