End-User Software Engineering: Report from the 2007 Dagstuhl

Margaret Burnett
Theme: End-User Software Engineering

• It all started with End-User Programming (creating new programs).

• But what about the rest of the software lifecycle?

• End-user software engineering includes the entire software lifecycle
  – Not just “create”.
Set-up: EUSE vs. SE

• How is EUSE different than SE?
  – 1. The people don’t have the same values and motivations as professionals.
  – 2. The research has strong foundations in HCI to suit above + lack of training.
  – 3. Has unique languages/environments with constraints not faced in traditional SE (eg, “unsafe” language constructs, immediate visual feedback after every edit, …).
Purposes

• 1. Connections/brainstorming:
  – About specific research ideas. (One-on-one.)

• 2. Learn:
  – Become more familiar with state of current research in EUSE.

• 3. Begin to understand:
  – Analyze/drill down.
  – Synthesize up.

• Here: focusing on 2 and 3.
Learned: Who Does EUP/EUSE When?

• Noticed: Imprecise thinking about EUPs.
  – Big differences -- Segal’s scientists vs. Scaffidi’s InfoWeek readers vs. accountant vs. your dad...
  – ...vs. your mom (gender differences!).
  – Sometimes scenario matters -- is it a role not a person?
    • “Me” doing a spreadsheet motivated/behave differently from “me” doing Lisp programming.
  – Outcome: must be careful to state which subgroup of the EUPs being addressed.
About Requirements and Design

• Requirements, specs:
  – Assertions are possible by EUPs (eg, ranges).

• EUPs’ design notations, processes:
  – Provisional notations, notations from the domain.
  – Usually iterative, sometimes emergent.

• But sometimes EUPs design in collaboration with professional software engineers.
  – Sometimes requires use of semi-formal SE notations.
About Reuse

- Sometimes design starts with reuse.

- Reuse types:
  - By composition, by modification, by just finding and running, by adapting examples, by using templates, by copy/paste, just borrowing ideas, ...

- How to foster “reuse communities”?
  - MathWorks’ MatLab prime example of exploring this question.

- Possible gender differences in willingness to reuse, participate in reuse communities, ...
Are There Errors?

- Gaining confidence about the correctness, suitability, dependability, value, etc. of their programs.
- Spectrum of ways: entirely informal to entirely precise.
- Technologies: program analysis, aids to help user understand and narrow and problem-solve
- Problems: Getting users to understand their assumptions and problems is hard!
Approaches to “Are There Errors”

• Automated analysis is at the root of many of these:
  – Statistical anomaly detection.
  – Dataflow analysis, constraint satisfaction, abstract interpretation, type systems.

• Systematic testing: User/system together work through the program.
  – Eg: WYSIWYT.

• People make mistakes: (say it’s right when it’s wrong).
Getting the Errors Out: Debugging

• Debugging is often the reason people give up.

• Supports for debugging.
  – Eg: Automated suggestions for program changes (GoalDebug).
  – Eg: Supporting “why” questions (WhyLine).
EUSE in the Large

- Interactions with others’ systems, the web, privacy, security decisions
- Conflicting goals: one size fits one? Vs. one size plays nicely with all?
- Current EUP models too naïve, too much noise?
- Goals/challenges: create useful abstractions, more work on credibility relevance
Evaluation

- To understand more about EUSE
- Types:
  - Various sorts of controlled studies, but also collaboration with industries, action research, case studies, ...
- Subject matters:
  - Barriers, hypothesized theories, tool evaluation, social factors, artifacts, algorithms, what they think about, role of context, design space, practices, individual differences...
How to Find Out More

• Ask people who were there
  – <show of hands>

• Proceedings (Real Soon Now):
  – At EUSES Consortium site. http://eusesconsortium.org

• Listen to Susan and Andy
  – Emerging survey paper reflecting these findings and more.