Invited talk, Software Engineering Horizons track of ICSE 2012,
the 34th International Conference on Software Engineering,
Zurich, Switzerland, June 2-9, 2012.

Software Engineers are People Too:

Applying Human Centered Approaches to Improve Software Development

Brad A. Myers

Human-Computer Interaction Institute

School of Computer Science

Carnegie Mellon University

http://www.cs.cmu.edu/~bam

bam@cs.cmu.edu

Human Centered Approaches?

- Concerned with everything the user encounters
  - Functionality & Usefulness
  - Content
  - Labels
  - Presentation
  - Layout
  - Navigation
  - Speed of response
  - Emotional Impact
  - Context (social environment in which use happens)
  - Documentation & Help
- Measures:
  - Learnability, Productivity, Errors, ...

What Can Be Addressed?

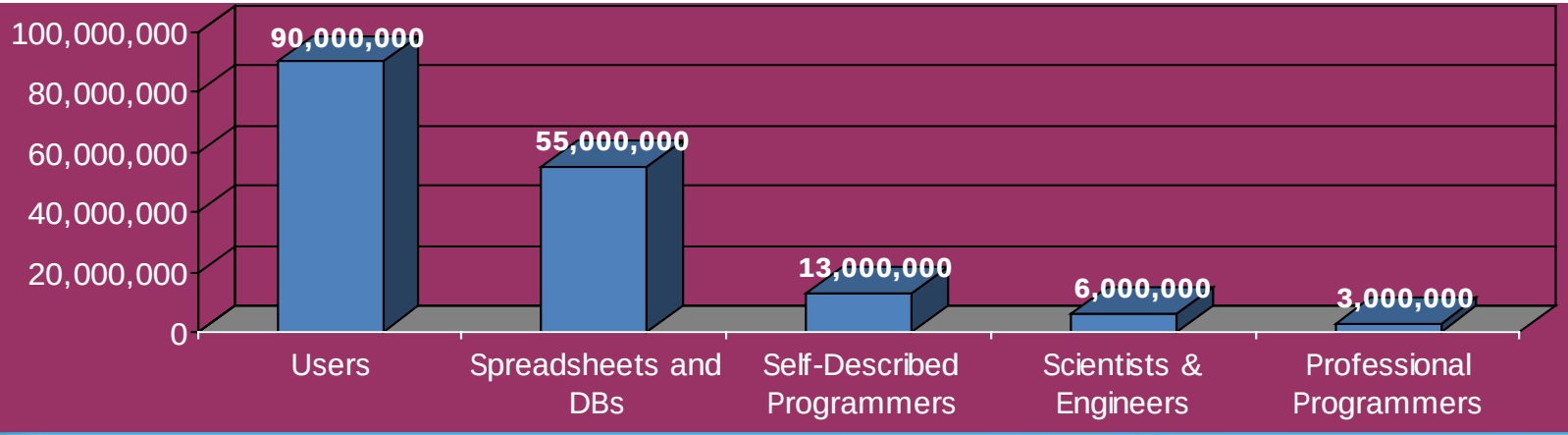
- Everything the developer encounters
- Tools – IDEs & their user interfaces
- Languages themselves
  - Not necessarily just “taste”, “intuition”
  - Error-proneness
- APIs
  - “Interface” between developer and functionality
  - “Languages” by themselves are almost irrelevant these days
- Documentation for all of the above
- Processes & context of development

➢ Consider the whole “system” together

➢ New as well as legacy systems

Who Are Developers?

- Programming tools are not just used by highly-trained professional programmers
- End-User Programmers = People whose primary job is *not* programming
- In 2012 in USA at work: — *[Scaffidi, Shaw and Myers 2005]*
  - 3 million professional programmers
  - 6 million scientists & engineers
  - 13 million will describe themselves as programmers
  - 55 million will use spreadsheets or databases at work
  - 90 million computer users at work in US



"Human Centered Approach" – More Than Lab User Studies

- Design & aesthetics matter & will affect:
  - User's performance
  - Errors
  - Adoption of your tool
- Many different methods for answering many different questions
  - Before design time
  - During design & implementation
  - After implementation

Many HCI Methods

- Contextual Inquiry
- Contextual Analysis
- Paper prototypes
- Think-aloud protocols
- Heuristic Evaluation
- Affinity diagrams
- Personas
- Wizard of Oz
- Task analysis
- A vs. B testing
- Cognitive Walkthrough
- Cognitive Dimensions
- KLM and GOMS (CogTool)
- Video prototyping
- Body storming
- Expert interviews
- Questionnaires
- Surveys
- Interaction Relabeling
- Log analysis
- Focus groups
- Card sorting
- Diary studies
- Improvisation
- Use cases
- Scenarios
- “Speed Dating”
- ...

Dangers of *Not* Applying Human Centered Approaches

- Tools may prove to be not useful
  - Useful = solves an important problem
    - Happens frequently
    - Difficult to solve otherwise
    - Developers believe academic tools solve unimportant problems [How do practitioners perceive Software Engineering Research? http://catenary.wordpress.com/2011/05/19/how-dopractitioners-perceive-software-engineering-research/]

  } HCI questions

  - Tools may not actually solve the problem
    - Example: a study suggested that Tarantula tool identifying potentially faulty statements for debugging was not helpful
      - Changed the task, but telling if the identified statement was *actually* faulty not easier than finding the bug
      - Parnin, C. and Orso, A. 2011. Are Automated Debugging Techniques Actually Helping Developers International Symposium on Software Testing and Analysis (2011), 199–209.

Dangers of *Not* Applying Human Centered Approaches

- Tools may show no measurable impact
  - Desired advantage overwhelmed by problems with other parts
  - Example: Emerson Murphy-Hill found that refactoring tools are under-utilized and programmers do not configure them due to usability issues
    - Emerson Murphy-Hill, Chris Parnin, Andrew P. Black. *How we refactor, and how we know it.* In ICSE '09: Proceedings of the 2009 IEEE 31st International Conference on Software Engineering (2009), pp. 287-297.

Human Centered Approaches are Not Too Difficult for You

- Getting *some* user data better than none
- Observing real usage reveals many opportunities
  - Insights about new issues to address, not necessarily what originally planned
    - Thomas LaToza's Reachability Questions from Architecture study
    - Jeff Stylos's method placement result from study of class size: from 2.4 to 11.2 times faster

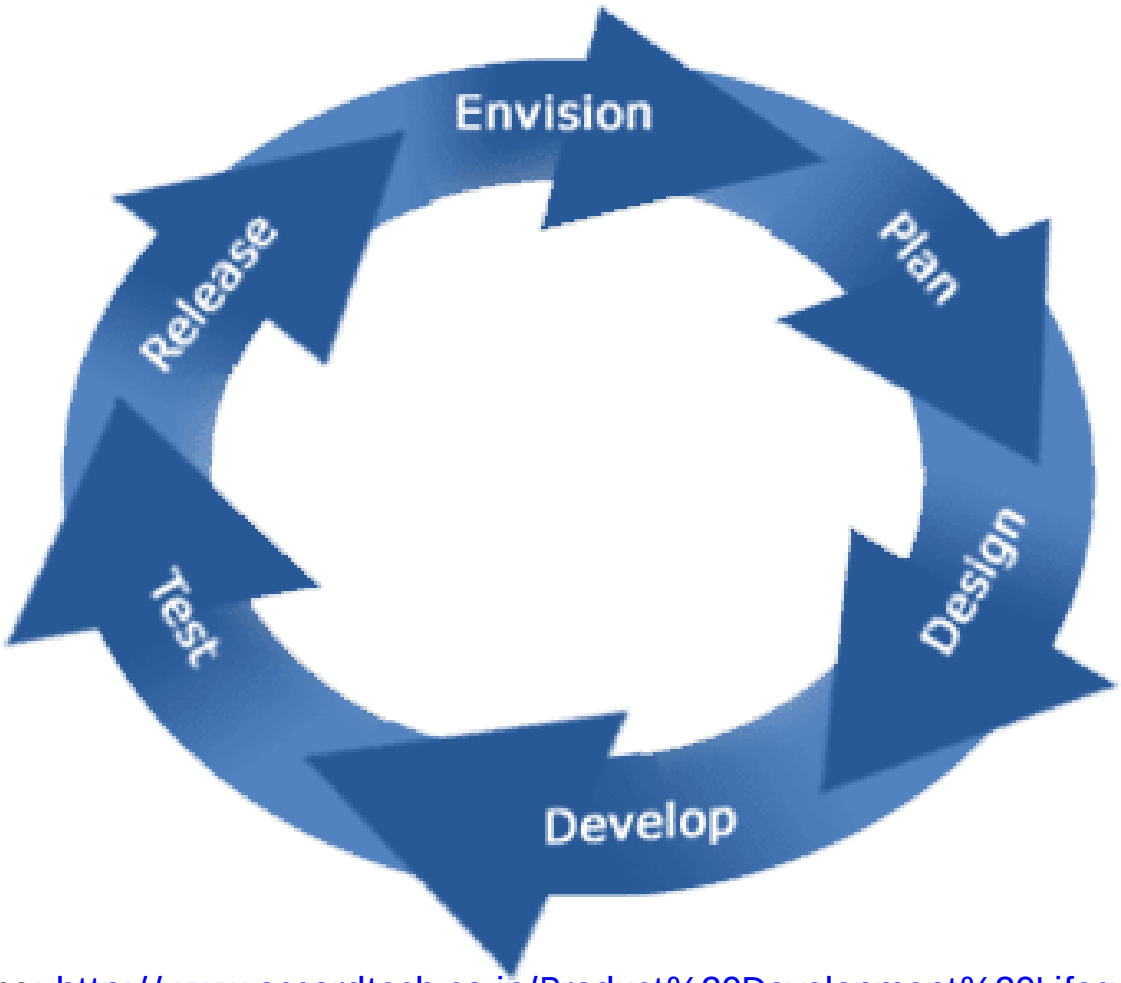
```
server.send ( message )   vs.
mail.send ( server )
```

- Collaborating with Graphic Designers for even a short time can provide significant improvements in aesthetics

Key Decision: What is Your Question?

- What do you need to find out or show?
  - What claim to do you want to make?
- Showing that a tool is *usable* is different from that it is *useful*
- Exploring what people are doing, is different from determining how often an observed behavior happens
- Drives what type of method to use, and tasks to be done with it

Product Lifecycle



Source: http://www.accordtech.co.in/Product%20Development%20Lifecycle.htm

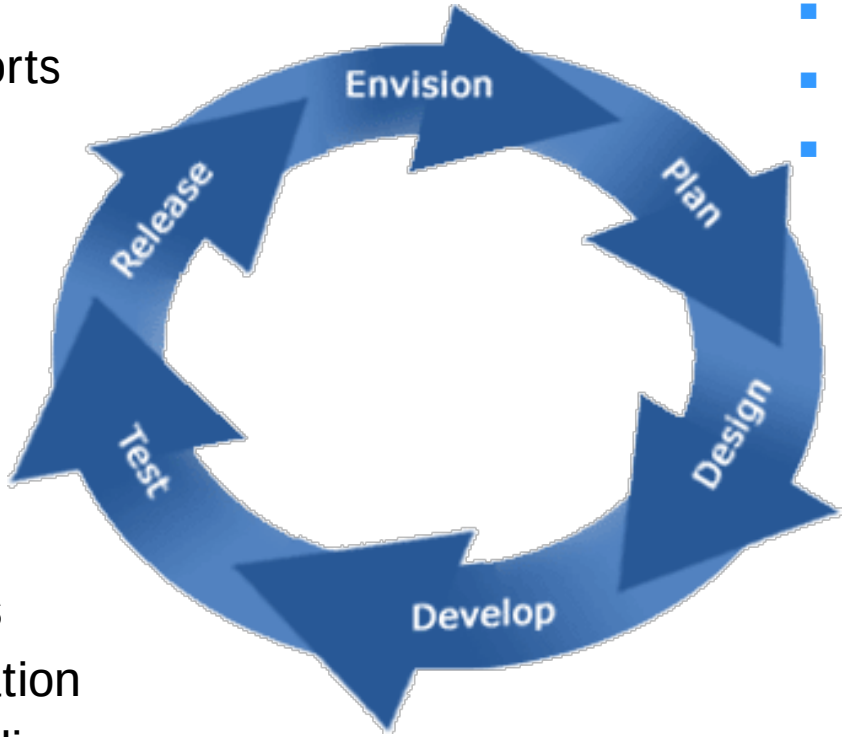
Product Lifecycle

Exploratory Studies

- Contextual Inquiries
- Surveys
- Lab Studies
- Corpus data mining

Field Studies

- Logs & error reports



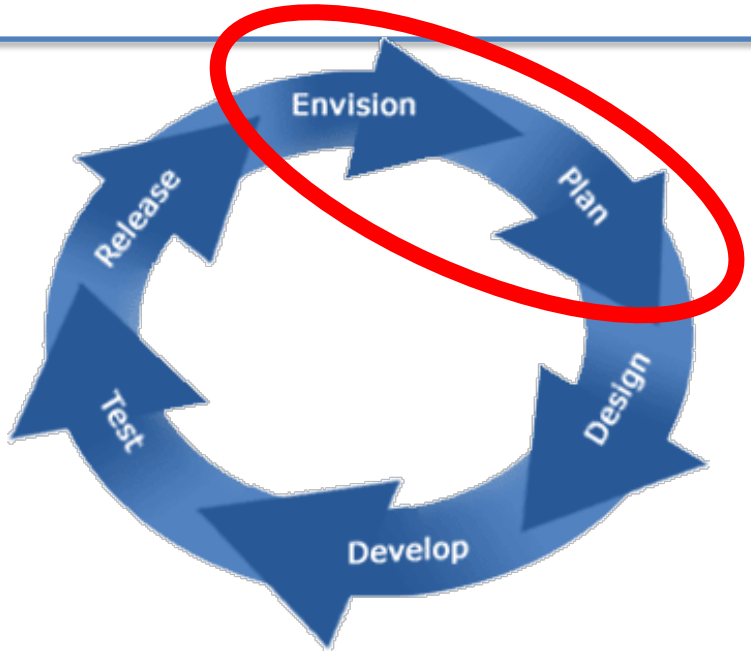
Evaluative Studies

- Expert analyses
- Usability Evaluation
- Formal Lab studies

Design Practices

- "Natural programming"
- Graphic & Interaction Design
- Prototyping

Exploratory Studies



- Identify what is really happening
- Discover important problems
- Quantify need

Contextual Inquiry

- Beyer, H. and Holtzblatt, K., *Contextual Design: Defining Custom-Centered Systems. 1998, San Francisco, CA: Morgan Kaufmann Publishers, Inc.*

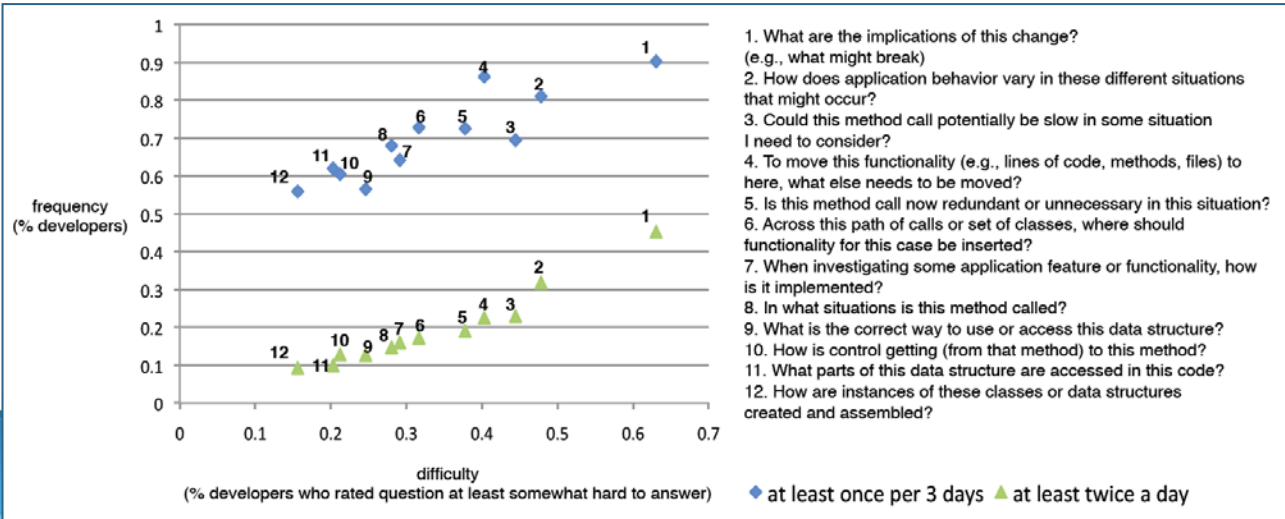
- A kind of “ethnographic” or “participatory design” method
- Watch developers while they are performing their real tasks
- Objective, concrete data about real activities
- May be followed by a survey, to establish generality of the issues

Why Contextual Inquiry?

- Usually reveals many barriers and problems in current practice
- Helps develop *insights*
  - Be open to inspiration
- Not for confirming what you already know
- Qualitative data (not quantitative)
  - CIs are not for gathering statistics, analytics
    - In contrast to surveys & lab studies
- But need to be able to observe real tasks

Example of Contextual Inquiry

- "Developers Ask Reachability Questions"
  - Thomas D. LaToza and Brad Myers, ICSE'2010, Cape Town, South Africa, 2-8 May 2010. pp. 185-194.
  - "Search across feasible paths through a program for target statements matching search criteria"
    - Watched 17 developers investigating unfamiliar code
    - Also survey of 460 developers
    - Over 100 other hard-to-answer questions



Exploratory Lab Studies

- To understand what is happening
- More controlled than field studies
  - Can compare multiple people on same tasks
- Example: studying Eclipse for maintenance tasks
  - Andrew J. Ko, Htet Htet Aung, and Brad A. Myers. "Eliciting Design Requirements for Maintenance-Oriented IDEs: A Detailed Study of Corrective and Perfective Maintenance Tasks". *ICSE'2005*. pp. 126-135. **Winner, Distinguished Paper Award.**
  - Detailed study of fixing bugs and adding features
  - Dataset used for 3 different award-winning papers: interruptions, navigation, code editing behaviors

Interactive Bottleneck	Overall Cost
Navigating to fragment in *same* file (*via scrolling*)	~ 11 minutes
Navigating to fragment in *different* file (*via tabs and explorer*)	~ 7 minutes
Recovering working set after returning to a task	~ 1 minute
Total Costs	~19 minutes

=35%

Exploratory Lab Studies

- Second Example: Barriers in APIs
  - Stylos, Jeff and Clarke, Steve "Usability Implications of Requiring Parameters in Objects' Constructors," in *ICSE'2007. Minneapolis, MN: pp. 529-539.*
  - Ellis, B., Stylos, J., and Myers, B. "The Factory Pattern in API Design: A Usability Evaluation," in *ICSE'2007. Minneapolis, MN: pp. 302-312.*
- Different personas of programmers
  - Opportunistic, Pragmatic, Systematic
- Required parameters not successful

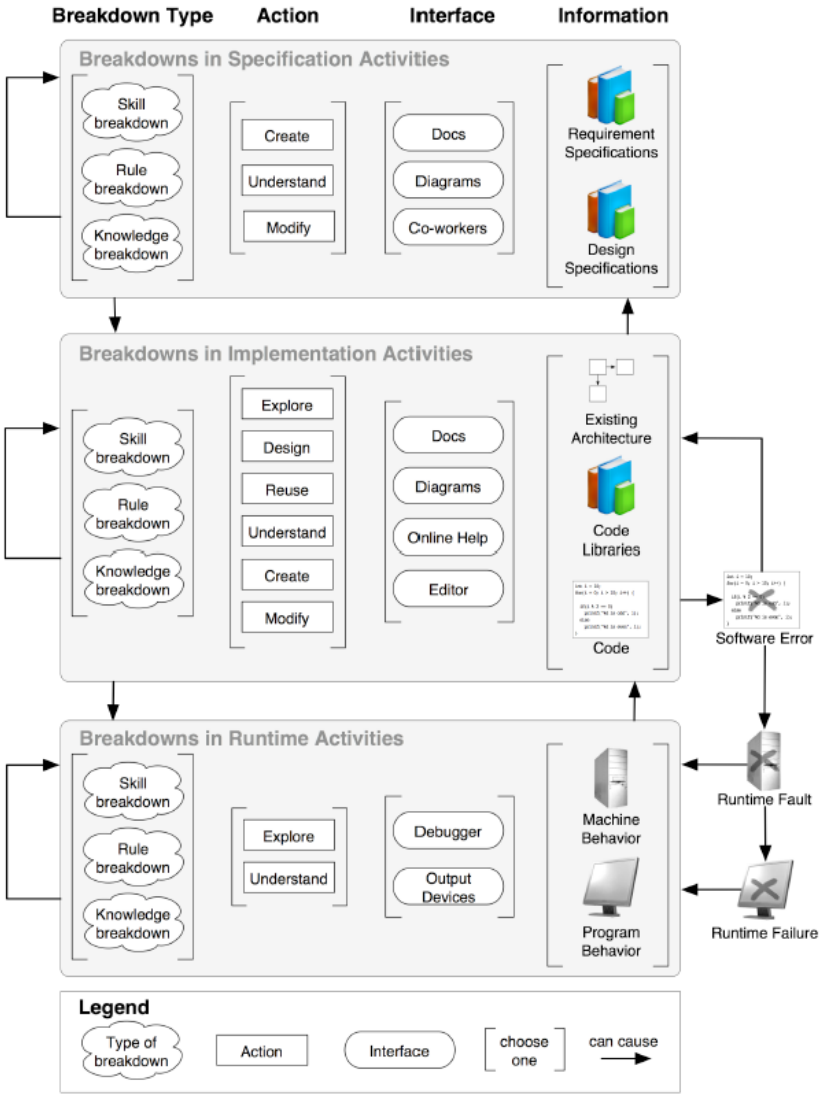
```
var foo = new FooClass(barValue);
var foo = new FooClass();
   foo.val = barValue;
```

- Factory pattern took 2.1 to 5.3 times longer

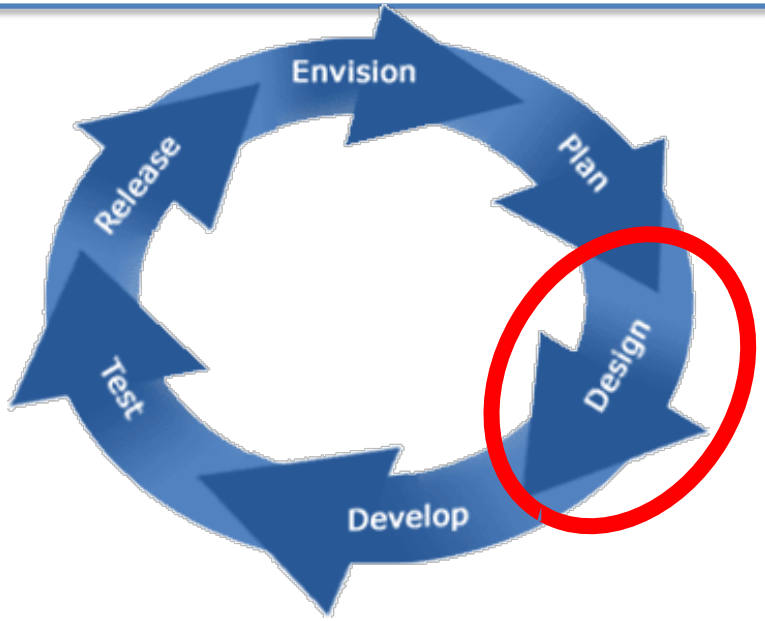
```
AbstractFactory f =  AbstractFactory.getDefault();
Widget w = f.createWidget();
```

Can Create "Models" from Results

- Explanations of observed behaviors
- Idealized, but based on real data
- Example: causes of breakdowns when trying to debug
  - Today, users must guess where bug is or where to look & are often wrong
  - Andrew J. Ko and Brad A. Myers. "Development and Evaluation of a Model of Programming Errors". 2003. *IEEE Symposium on End-User and Domain-Specific Programming (EUP'03)*, part of the *IEEE Symposia on Human-Centric Computing Languages and Environments* (HCC'03). October 28-31, 2003. Auckland, New Zealand. pp. 7-14.



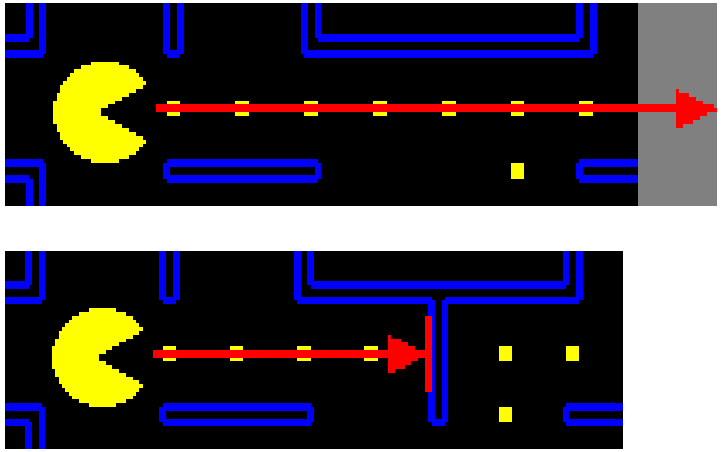
Design Methods



- Now know the problem, what is the solution?
- How do I design it so it is attractive and effective?

"Natural Programming"

- Technique developed by my group to elicit developer's "natural" expressions
  - Mental models of tasks, vocabulary, etc.
- Blank paper tests
- Must prompt for the tasks in a way that doesn't bias the answers
- Examples:
  - PacMan before and after
    - Mostly rule-based (if-then)
  - API designs
    - No-one used factory patterns

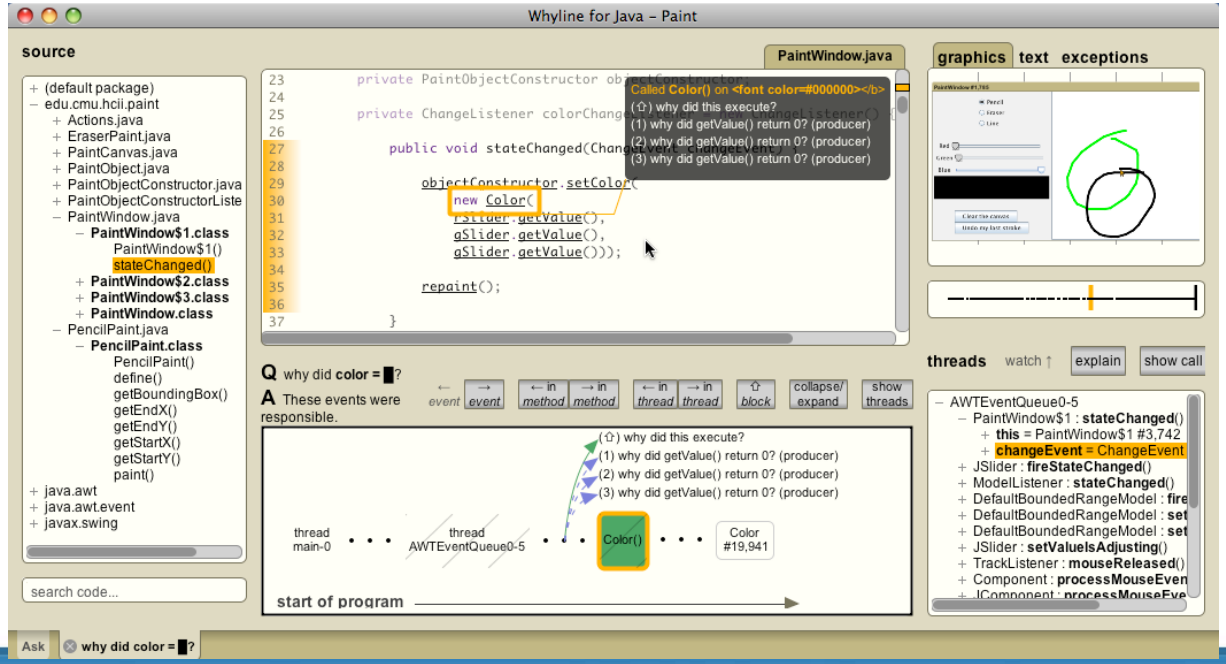


Why Natural Programming?

- When want design to be easily learned by novices
- But biased by what they already know
  - Graphic designers will think PhotoShop is “natural”
  - Programmers will think Java is “natural”

Graphic Design

- Importance of graphic design and interaction design
- Software Engineers (and researchers) are not necessarily the best interaction designers
- Design can have a big impact even with same functionality
- Might involve designers for colors, icons, which controls, layout, ...



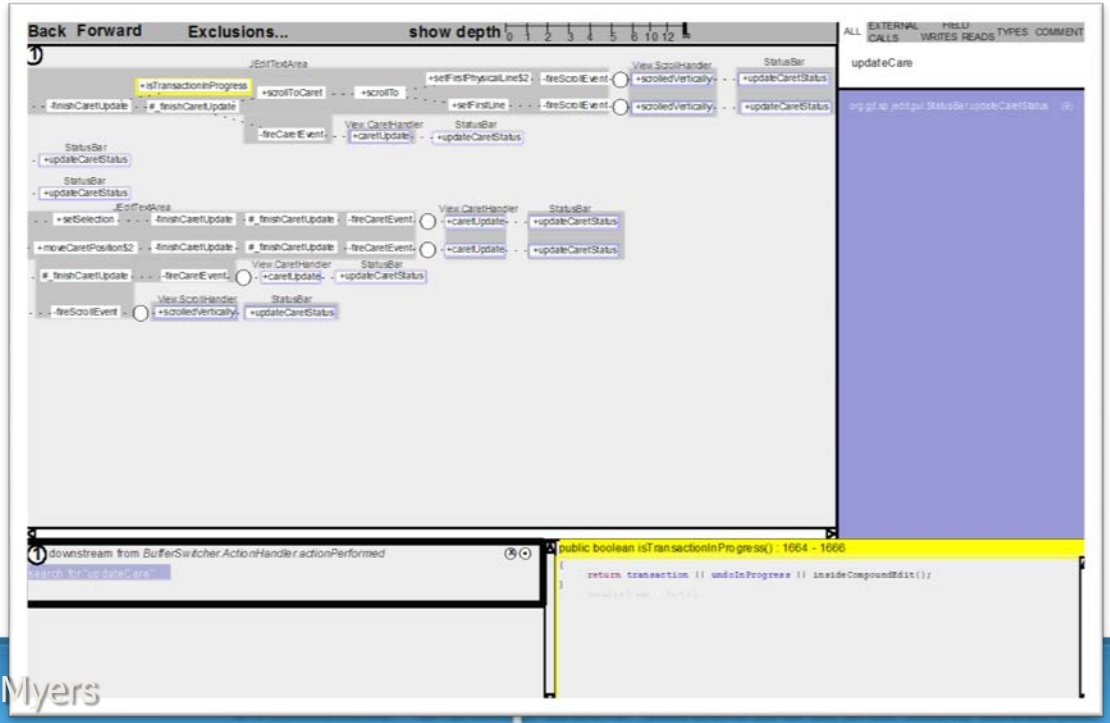
Prototyping

- Try out designs with developers before implementing them
  - **Paper**
    - "Low fidelity prototyping"
    - Often surprisingly effective
    - Experimenter plays the computer
    - Drawn on paper → drawn on computer
  - **Implemented Prototype ("Click through")**
    - Visio, PowerPoint, Web tools (even for non-web UIs)
    - (no database)
  - **Real system**

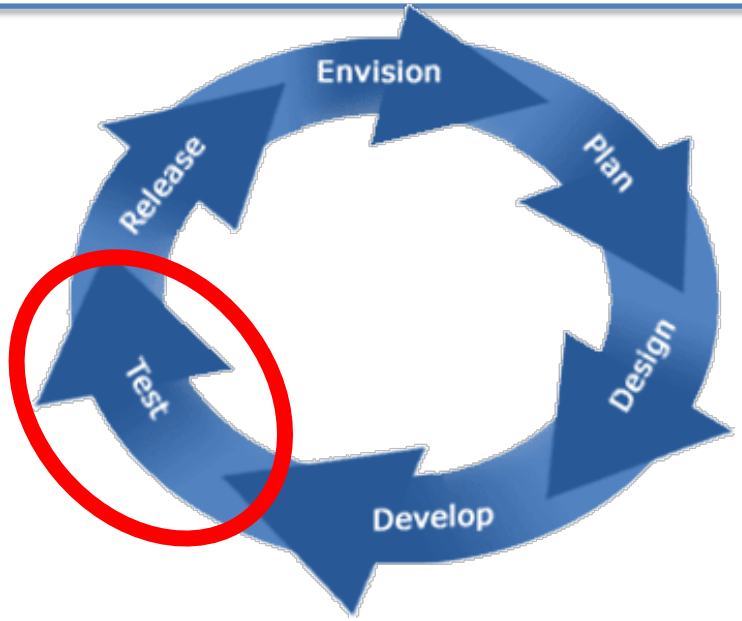
- Need to test these with users!
- Better if sketchier for early design
  - Use paper or "sketchy" tools, not real widgets
  - People focus on wrong issues: colors, alignment, labels
  - Rather than overall structure and fundamental design

Example of Prototyping

- Thomas LaToza designing new visualization tool to try to help answer Reachability Questions
- Prototypes created with Omnigraffle and printed
- Revealed significant usability problems that were fixed before implementation
  - Graphical presentation
  - Controls



Evaluation Methods



- Does my tool work?
- Does it solve the developer's problems?
- "If the user can't use it, it doesn't work!"
  - – Susan Dray

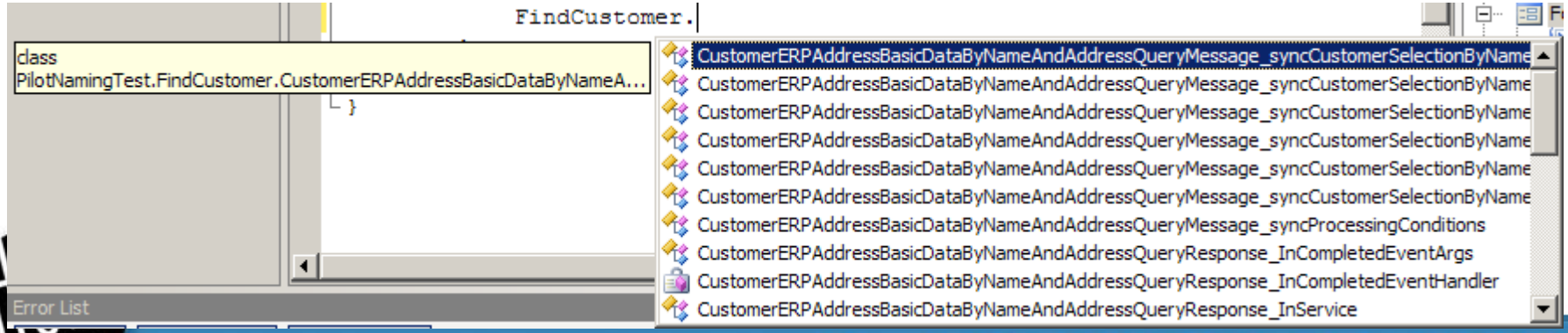
Dray & Associates
Human Centered Innovation

Expert Analyses

- Usability experts evaluating designs to look for problems
  - Heuristic Analysis – [Nielsen] set of guidelines
  - Cognitive Dimensions – [Green] another set
  - Cognitive Walkthroughs – evaluate a task
- Can be inexpensive and quick
- However, experienced evaluators are better
  - 22% vs. 41% vs. 60% of errors found [Nielsen]
- Disadvantage: "just" opinions, open to arguments

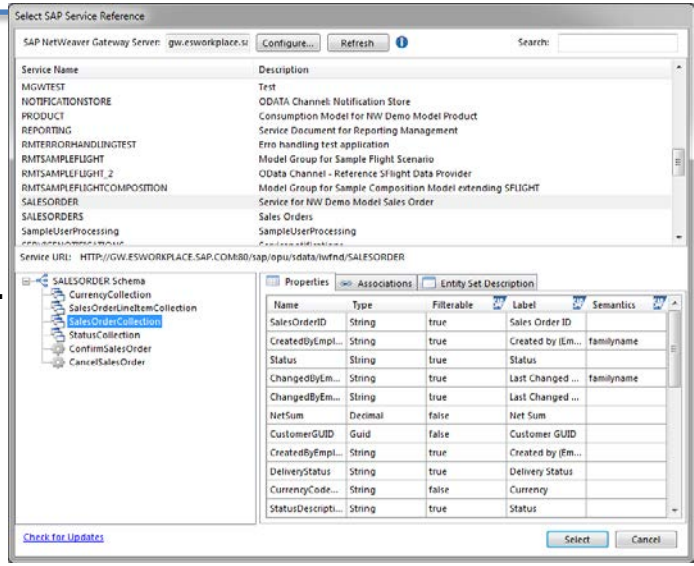
Our Use of Expert Analyses

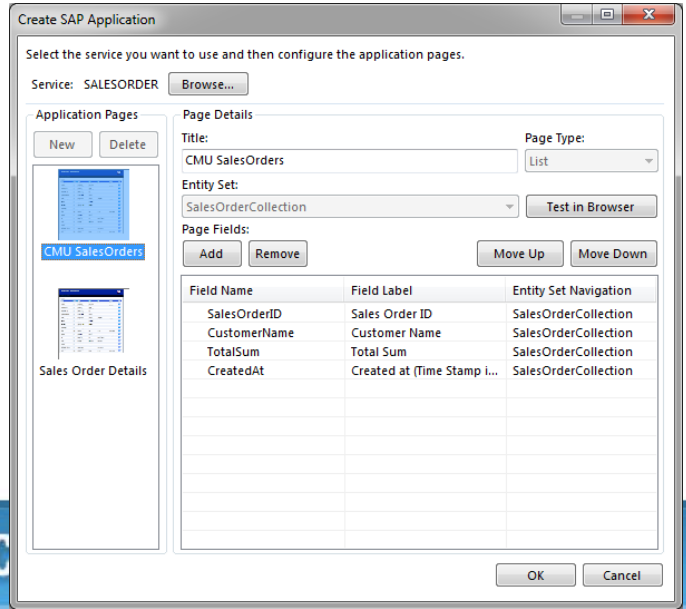
- Collaborating with SAP on their APIs and tools
- We studied SAP's Enterprise Service-Oriented Architecture (eSOA) APIs & Documentation
  - Jack Beaton, Sae Young Jeong, Yingyu Xie, Jeffrey Stylos, Brad A. Myers. "Usability Challenges for Enterprise Service-Oriented Architecture APIs," *2008 IEEE Symposium on Visual Languages and Human-Centric Computing, VL/HCC'08*. Sept 15-18, 2008, Herrsching am Ammersee, Germany. pp. 193-196.
- Naming problems:
  - Too long `MaterialSimpleByIDAndDescriptionQueryMessage_syncMaterialSimpleSelectionByIDAndDescriptionSelectionByMaterialDescription`
  - Not understandable



Our Use of Expert Analyses

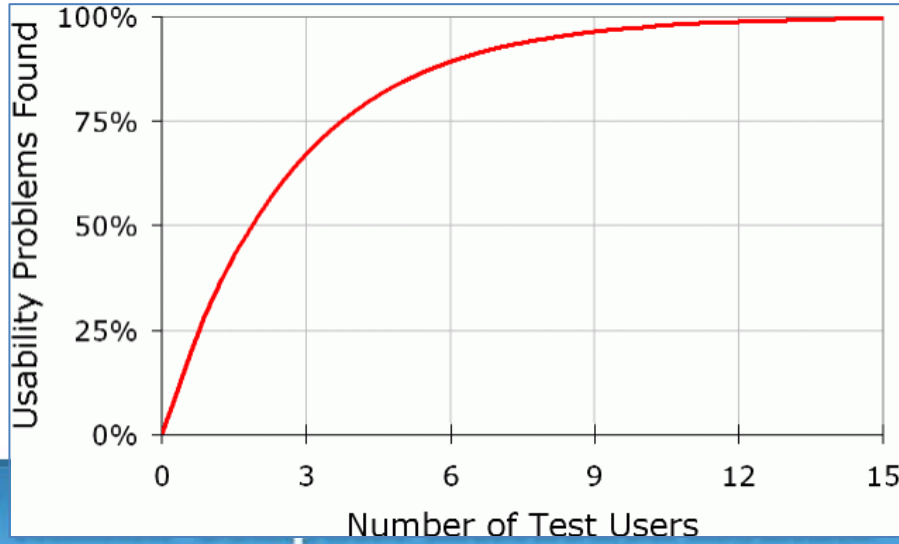
- Andrew Faulring, Brad Myers,Yaad Oren, Keren Rotenberg. "A Case Study of Using HCI Methods to Improve Tools for Programmers," *Cooperative and Human Aspects of Software Engineering (CHASE)*, An ICSE 2012 Workshop. Zurich, Switzerland, June 2, 2012.
- We evaluated the SAP NetWeaver Gateway developer tool for Visual Studio
- Identified many usability issues
  - We used Heuristic Analysis & Cognitive Walkthrough
  - Issues were fixed as part of their agile development process





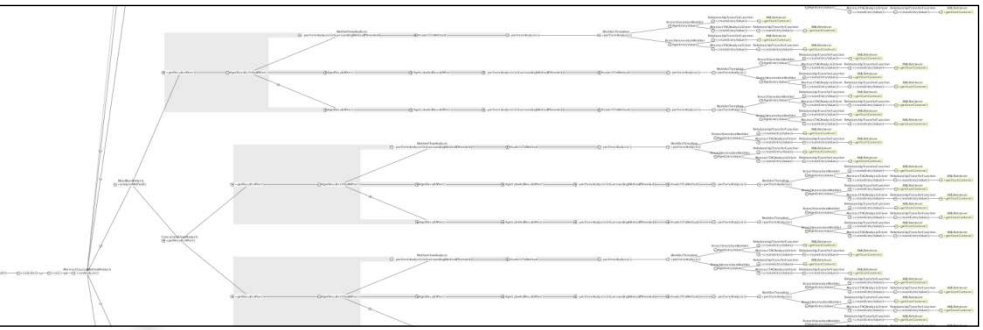
Usability Evaluations

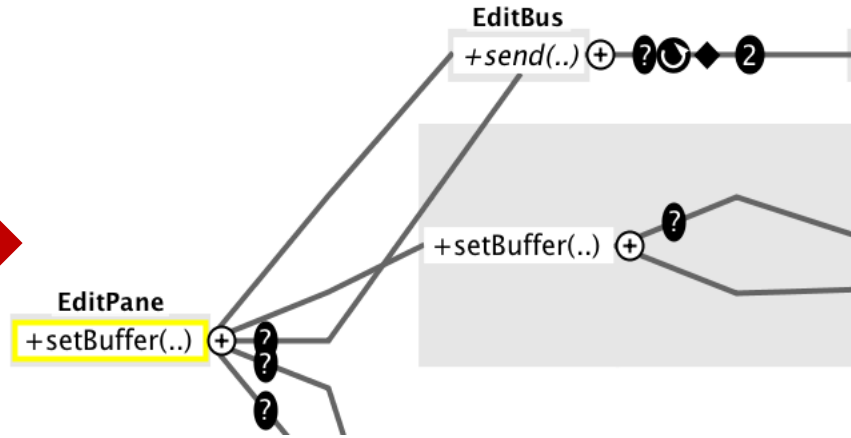
- Different from formal A vs. B "user studies"
  - Understand usability issues
  - Should be done early and often
    - Doesn't have to be "finished" to let people try it
- "Think aloud" protocols
  - "Single most valuable usability engineering method" -- [Nielsen]
  - Users verbalize what they are thinking
    - Motivations, why doing things, what confused about
  - Don't need many users



Example of Our Use

- Thomas LaToza's REACHER tool for Reachability Questions went through multiple iterations
  - Revised based on paper prototype (discussed already)
  - Revised based on 1st evaluation of full system
    - E.g., replaced duplicates of calls to methods with pointers
    - Changed to preserve order of outgoing edges
    - Redesign of icons, interactions





Why Usability Analysis

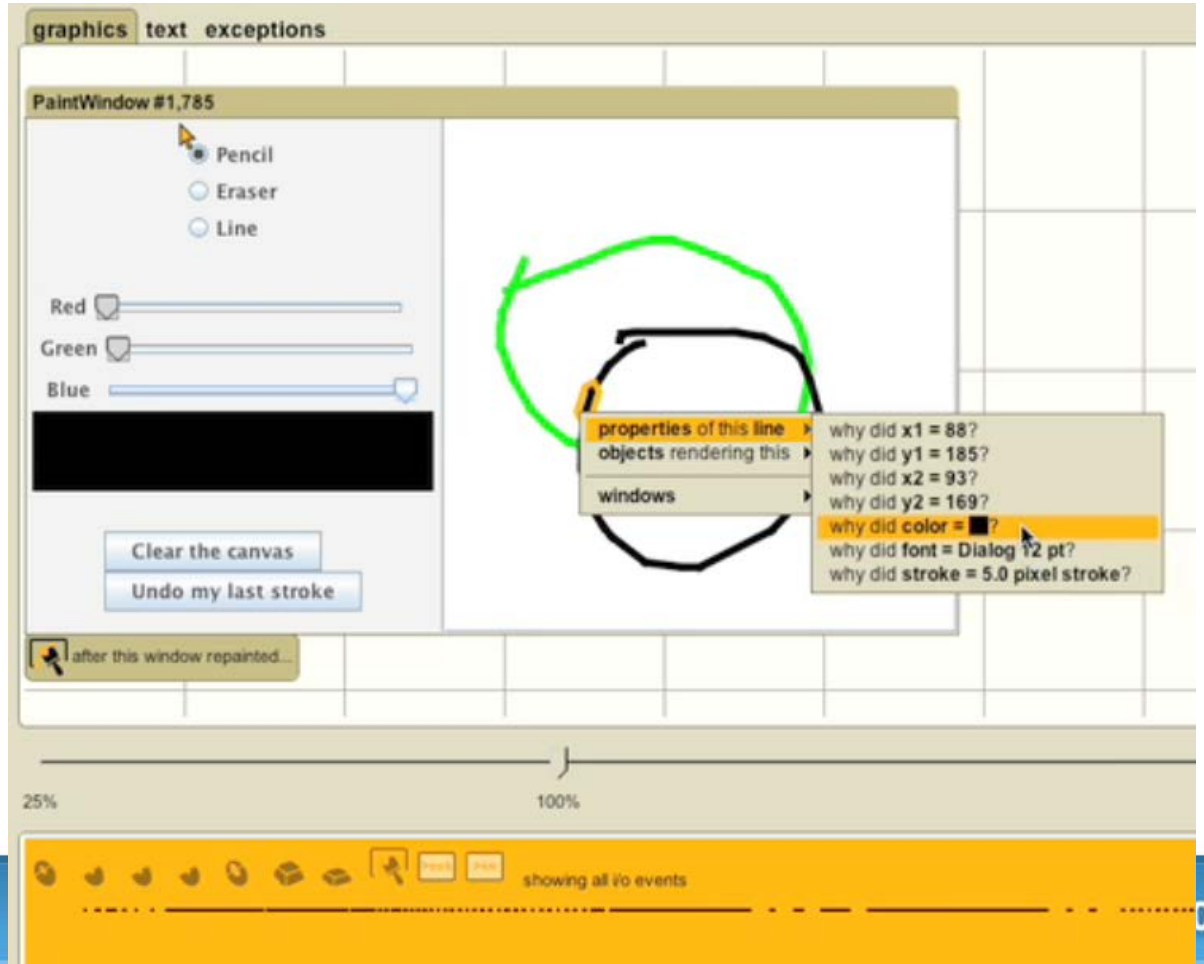
- Improve the user interface prior to:
  - Deployment
  - A vs. B testing (as a “pilot” test)
- Demonstrate that users *can* use the system
  - Show that novel features of the UI are understandable

Formal A vs. B "User Studies"

- Formal *A vs. B* lab user studies are "gold standard" for academic papers – to show something is better
- But many issues in the study design
  - "Confounding" factors which were not controlled and are not relevant to study, but affect results
  - Tasks or instructions are mis-understood
  - Use prototypes & pilot studies to find these
- Statistical significance doesn't mean real savings
- Be sure to collect qualitative data too
  - Strategies people are using
  - Why users did it that way
  - Especially when unexpected results

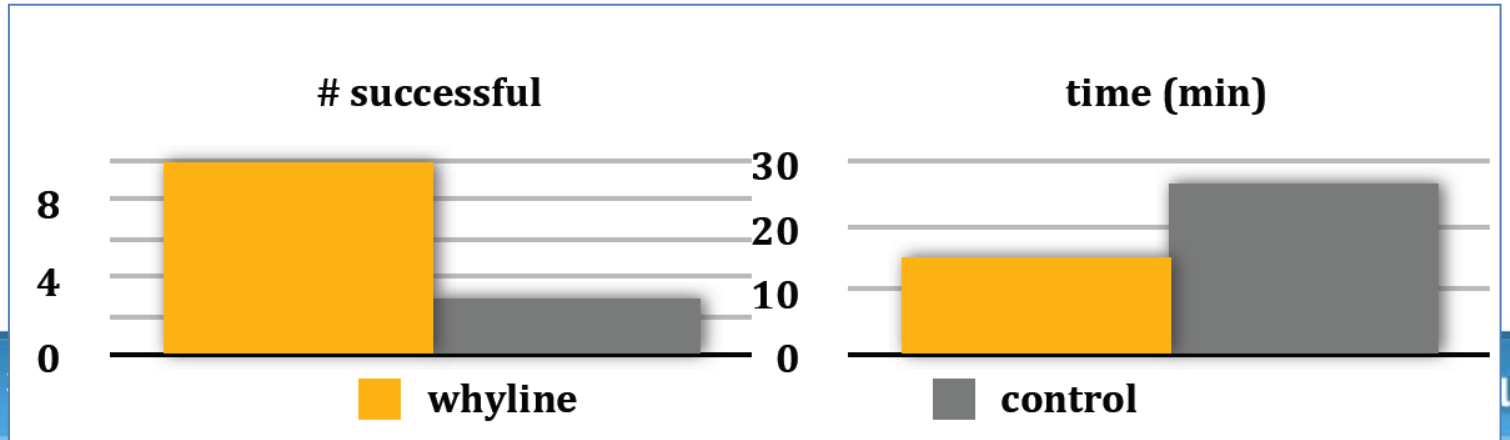
Example of *A vs. B* Study: Whyline

- PhD work of Andy Ko
- Allow users to directly ask “Why” and “Why not”



Whyline User Studies

- Initial study:
  - Whyline with novices outperformed experts with Eclipse
  - Factor of 2.5 times faster
- Formal study:
  - Compared to Whyline with key features removed (rather than Eclipse)
  - Tasks: 2 real bug reports from real open source system (ArgoUML)
  - Whyline was over 3 times as successful, in ½ of the time

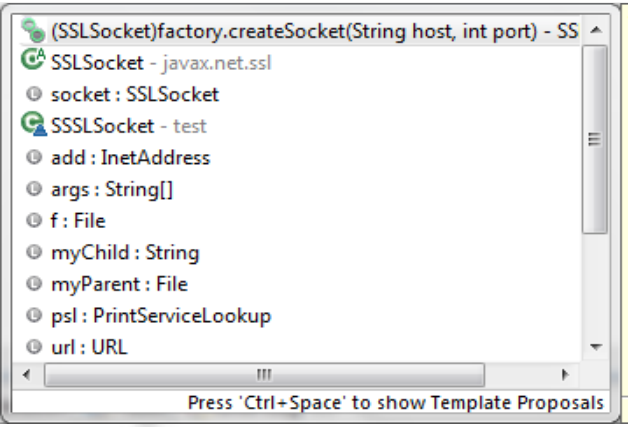


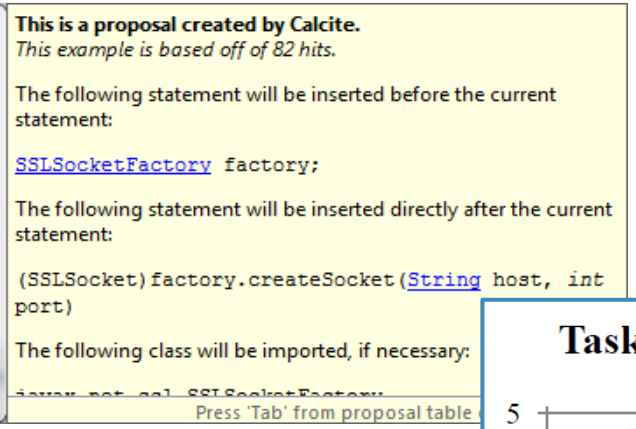
Another Lab Study: Calcite

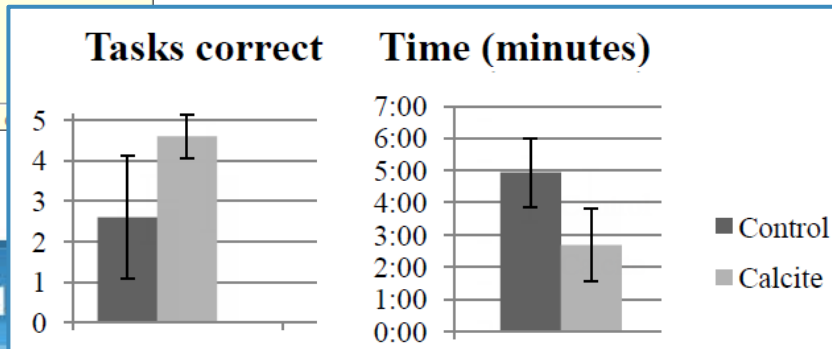
- Calcite: Construction And Language Completion Integrated Throughout

  http://www.cs.cmu.edu/~calcite

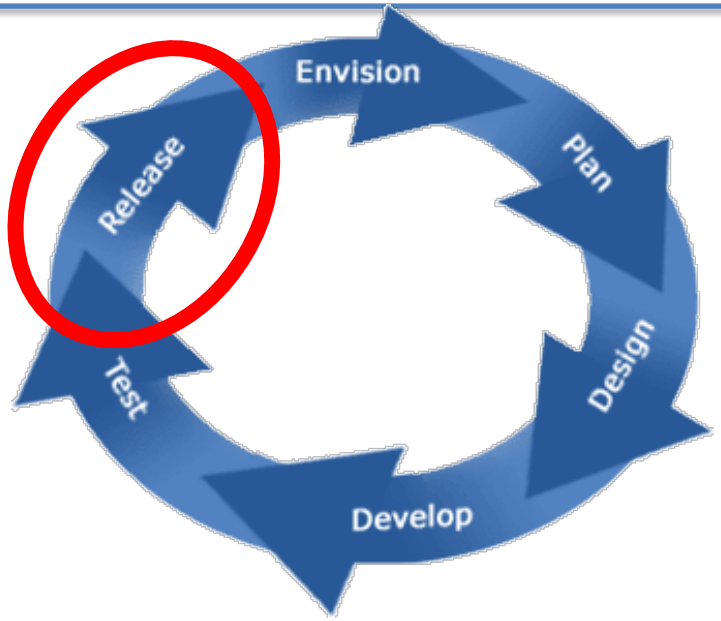
- Augmented code completion in Eclipse
  - How to create objects of specific classes: `SSLSocket s = ???`







Field Studies of System in Use



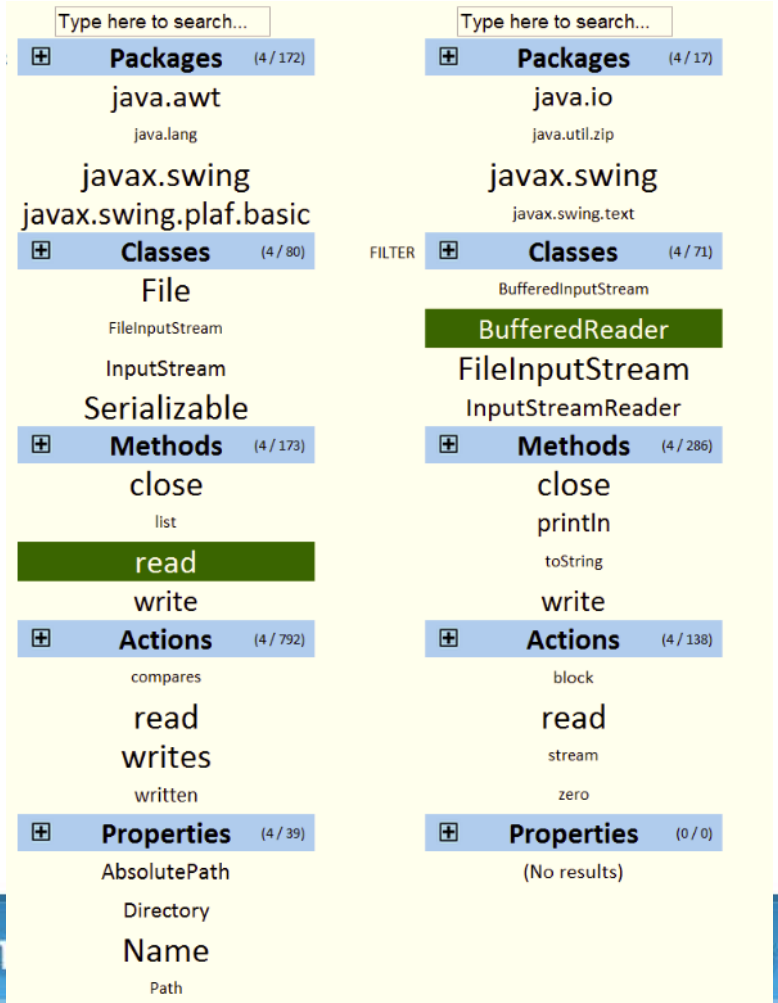
- Find out what happens when the tool is really used
- Requires significant effort to make the tool sufficiently solid

Logging Actual Use

- Easier if instrument your tools
- Objective use data better than users' recollections and opinions
- Many levels of data can be collected
  - Privacy issues
- Example: Flourite logger for Eclipse
  - Records all edits and events, including scrolling operations & source code,
  - Necessary to identify patterns of backtracking

Example of Field Analysis

- Apatite: Associative Perusing of APIs That Identifies Targets Easily http://www.cs.cmu.edu/~apatite
- Novel documentation tool that works *by association*
  - E.g., methods often used together
- Can start with verbs (actions) and find what classes implement them
- Couldn't figure out a comparison tool or tasks for a lab study
- Deployed on the web
- Mostly used for fast lookup from partial names



Why Field Studies?

- Understand which features are used and how
  - Not necessarily *why*
  - Can sometimes follow up with questionnaires, interviews of actual users
  - Developers often are surprised at how system is used
- Demonstrate that people choose to use the system when optional
- Easy to instrument web systems, some on-line tools

Summary: Our Group

- We have followed this methodology
  - 30 studies; 17 systems in 16 years
- Doing evaluative studies provides new insights that can inspire significantly new designs for languages and tools for software engineers
- Design methods result in better tools
- New designs can be evaluated

More on This Topic

- CHASE and USER workshops at ICSE
- Thomas D. LaToza and Brad A. Myers, "Designing Useful Tools for Developers", *PLATEAU 2011: Evaluation and Usability of Programming Languages and Tools*, workshop at the Onward! 2011 and Splash 2011 conferences, Portland, Oregon, October 24, 2011. On-line pdf or local pdf.
- Thomas D. LaToza, Brad A. Myers. "On the Importance of Understanding the Strategies that Developers Use", *Cooperative and Human Aspects of Software Engineering (CHASE'10)*, An ICSE 2010 Workshop. May 2, 2010. Cape Town, South Africa. pp. 72-75. pdf
- Reading list for "**Human Aspects of Software Development (HASD)**" by Thomas LaToza and Brad Myers http://www.cs.cmu.edu/~bam/uicourse/2011hasd/

Thanks to:

- Funding:
  - NSF under IIS-1116724, IIS-0329090, CCF-0811610, IIS-0757511 (Creative-IT), NSF ITR CCR-0324770 as part of the EUSES Consortium
  - SAP
  - Adobe
  - IBM
  - Microsoft Research RISE

- >30 students:
  - Htet Htet Aung
  - Jack Beaton
  - Ruben Carbonell
  - John R. Chang
  - Kerry S. Chang
  - Polo Chau
  - Luis J. Cota
  - Michael Coblenz
  - Dan Eisenberg
  - Brian Ellis
  - Andrew Faulring
  - Aristiwidya B. (Ika) Hardjanto
  - Erik Harpstead
  - Sae Young (Sophie) Jeong
  - Andy Ko
  - Thomas LaToza
  - Joonhwan Lee
  - Leah Miller
  - Mathew Mooty
  - Gregory Mueller
  - Yoko Nakano
  - Stephen Oney
  - John Pane
  - Sunyoung Park
  - Chotirat (Ann) Ratanamahatana
  - Christopher Scaffidi
  - Jeff Stylos
  - David A. Weitzman
  - Yingyu (Clare) Xie
  - Zizhuang (Zizzy) Yang
  - YoungSeok Yoon

Software Engineers are People Too:

Applying Human Centered Approaches to Improve Software Development

Brad A. Myers

Human-Computer Interaction Institute

School of Computer Science

Carnegie Mellon University

http://www.cs.cmu.edu/~bam

bam@cs.cmu.edu