Usability of Programming Languages

MPhil ACS module R201 - Alan Blackwell

Overview

- Practical experimental course
  - lectures are only introductory
- Lecture 1 - theoretical principles
  - classic approaches
  - current trends in leading research.
- Lecture 2 - candidate research methods
  - advantages and disadvantages
- Lecture 3 - specific classes of user
- Lecture 4 - directed by your research interests

Reading List

- Hoc, Green, Samurçay and Gilmore (1990)
  - Psychology of Programming.
- Psychology of Programming Interest Group
  - www.ppig.org
- Cambridge guidance on human participants
- Cairns and Cox (2008)
  - Research Methods for Human-Computer Interaction
- Carroll (2003)
  - HCI Models, Theories and Frameworks: Toward a multidisciplinary science

Lecture 1: Principles of human factors in programming

Cognitive models in HCI

- Engineering model of human 'I/O subsystems' and 'central processor'
  - Derived from human factors/ergonomics
  - Speed and accuracy of movement
  - Include working memory capacity
    - 7 +/- 2 'chunks'
    - Single visual scene
- Programming as 'cognitive ergonomics'?

Cognitive models of programming

- Deciding what to do is harder than doing it
  - HCI models assume a 'correct' sequence of actions
- Classic cognitive models derived from GOFAI
  - problem solving
  - planning
  - knowledge representation
- PoP book
  - ch 1.4 - Human Cognition and Programming
  - ch 3.1 - Expert Programming Knowledge: A Schema-based Approach
  - ch 2.3 - Language Semantics, Mental Models and Analogy
    - cf user interface "metaphor"

Software Development Context

- Cognitive science: individuals in controlled contexts
  - carefully construct experimental tasks to explore schemas, plans, analogy etc
  - correspond to AI constraints of toy problems
- Compare to *wicked problems*
  - goals and criteria under-specified, constraints conflict etc
- Commercial software development is more social
  - understand problem domain, negotiate specification change
- PoP book
  - ch 1.3 - The Tasks of Programming
  - ch 3.3 - Expert Software Design Strategies
  - ch 4.1 - The Psychology of Programming in the Large: Team and Organizational Behaviour
    - cf Information Systems literature

Individual Variation

- Cognitive theories are general theories
  - Consistent aspects of human performance
- But some programmers are far more productive
  - Always more productive in a language they know
  - Performance also correlated with
    - general intelligence
    - self-efficacy
    - diagnostic tests for autism
- "expert" vs "novice"
  - Study knowledge by comparing those with to those without
  - Study naïve users who are not 'crippled' or 'mutilated'
  - Real expert performance may include design research

Major research centres and programmes

Venues

- Psychology of Programming Interest Group (PPIG)
  - annual conference - proceedings available online
  - "Work in Progress" meeting (PPIG-WIP)
- European Association for Cognitive Ergonomics (EACE)
- Empirical Studies of Programmers foundation (ESP)
- IEEE Visual Languages and Human Centric Computing
  - ESP symposia in 2002, 2003
- International conference/workshop on Program Comprehension (ICPC, formerly IWPC)
- ACM CHI
- Evaluation and Assessment in Software Engineering (EASE)

NSF EUSES

- End-Users Shaping Effective Software
  - Margaret Burnett at Oregon State University
  - Brad Myers at Carnegie Mellon University
  - Mary Beth Rosson at Penn State University
  - Susan Wiedenbeck at Drexel University
  - Gregg Rothermel at University of Nebraska
  - Alan Blackwell at Cambridge
- See brand new publication
  - Ko, A.J., Abraham, R., Beckwith, L., Blackwell, A.F., Burnett, M., Erwig, M., Lawrence, J., Lieberman, H., Myers, B., Rosson, M.-B., Rothermel, G., Scaffidi, C., Shaw, M., and Wiedenbeck, S. (2011). The State of the Art in End-User Software Engineering. ACM Computing Surveys 43(3), Article 21.

UK/European centres

- PPIG UK
  - Salford (Maria Kutar – chair)
  - York (Thomas Green)
  - Sheffield Hallam (Chris Roast)
  - Open University (Marian Petre and Judith Segal)
  - Sussex (Judith Good)
  - Cambridge (Alan Blackwell)
- Joensuu, Finland (Sajaniemi, Tukiainen, Bednarik)
- Limerick, Ireland (Buckley)
- INRIA Eiffel group, Paris (Détienne, Visser)
- Fraunhofer (Wulf)

Other US centres

- University of Colorado at Boulder
  - Gerhard Fischer & Alex Repenning
- MIT Media Lab
  - Henry Lieberman
- IBM Research TJ Watson
  - Rachel Bellamy
- IBM Research Almaden
  - Allen Cypher
- Microsoft Research Redmond
  Human Interactions in Programming (HIP) group
  - Rob DeLine, Gina Venolia & Andrew Begel

Current areas of theoretical attention

Cognitive Dimensions of Notations

- Programming as interaction with an information structure (Ch 2.2 of PoP book)
- Sample dimension
  - Viscosity: a viscous system is difficult to change
- Resources:
  - Visual language usability paper in JVLC by Green & Petre
  - Tutorial by Green & Blackwell
  - Questionnaire by Blackwell & Green
  - Chapter in Carroll book

CDs Theory

- Any visible notation encodes an information structure.
  - The structure has different parts
  - The parts have various relationships to each other
- Notational Layers
  - one structure is often derived from another with similar parts and relationships
  - e.g. web page, from PHP program, from UML diagram, from whiteboard sketch, from business plan

Notational Activities

- Search:
  - finding information in a familiar structure
- Exploratory understanding:
  - understand a structure you haven't seen
- Incrementation:
  - add new items to existing structure
- Modification:
  - change an existing structure
- Transcription:
  - create a new structure derived from an existing layer
- Exploratory design:
  - create a structure you don't understand yet

Attention Investment

- Cost-benefit equation - compare mental effort:
  - to carry out a programming task
  - against effort saved by the program
- With associated risk/uncertainty:
  - In estimate of effort to finish the program
  - In actual benefit if the program has a bug
  - In chance of damage resulting from a severe bug

Attention Investment Biases

- Some expert programmers:
  - under-estimate costs, and over-estimate benefits
- Novices might be reluctant to engage in programming:
  - If they over-estimate the costs
  - If they over-estimate risk of negative return
  - Tools can provide 'gentle slope' to reduce this bias
  - E.g. surprise – explain - reward

Gender HCI

- Attention investment + self-efficacy theory
- You need confidence to start programming
  - Attention investment means that female students are less inclined to explore programming options.
- You need to do programming to gain confidence
  - Self-efficacy develops through time spent experimenting
- Encourage 'tinkering' to explore behaviour
  - But note that the same kind of tinkering can results in poorer learning for males, who have a tendency to be over-confident, and not to think about what they are doing

Programming by Example

- Based on machine learning techniques
- Infer programs from examples of required output
- Attention Investment benefits:
  - examples can be provided through normal direct manipulation, so reduced perceived cost
  - inferred program is offered to user when already functional, so reduced perceived risk

Natural Programming

- Programmme of Myers' group at Carnegie Mellon
- Study natural/everyday description of algorithms
- Design programming languages compatible with naïve knowledge
  - Pane's HANDS (for children)
  - Miller's LAPIS (for text manipulation)
  - Ko's CITRUS (constraint-based MVC platform)

Variable Roles

- Programme of Sajaniemi's group at Joensuu
- Based on analysis of source code corpuses
- Unlike Myer's focus on naïve knowledge, this focuses on expert knowledge
- Variables are used in only a few ways:
  - fixed, stepper, follower, gatherer etc
- Originally used for educational visualisation, instruction
- May be used for intelligent compilers in future

Agile/Pair Programming

- Study interaction between people doing pair programming
- theoretical focus on sociology rather than psychology
- See Computer-Supported Collaborative Work (CSCW) rather than HCI.

Programming Aptitude

- How to identify good programmers?
  - good programmers are commercially valuable
  - Identify talented students
  - Identify students needing additional help
- Seldom any theoretical explanation, just psychometric correlations
  - cognitive style
  - personality measures
  - autism spectrum diagnoses

Organizational Contexts

- Speciality of Microsoft HIP group
- Long-term studies of professional programmers in realistic teams
- Maintaining code bases on an industrial scale
  - E.g. what activities are involved when a new programmmer joins an established team?
- Hard to achieve for academics
  - beyond the resources of academic research budgets
  - relies on access to commercially sensitive information

Syntax and tools

Integrated Development Environments

- The language is not the only usability problem
  - Manage modules & dependencies
  - integrated editors
  - debugging and visualisation tools
- Some research using custom plug-ins for Eclipse
- Burnett's Forms/3 research platform
- Complete novel IDEs for education use
  - BlueJ & Scratch
  - Extensions to CMU Alice
    - Ko's WhyLine
    - Kelleher's storytelling Alice
    - (compare Good's struggles with Neverwinter Nights)

Visual Languages

- The ambition dates back to 60s and 70s
  - Idea of measuring improvement arrived at IEEE VL 1996
  - IEEE VL became IEEE VL/HCC soon afterward
- Pioneering commercial products
  - National Instruments LabVIEW
  - Prograph
- Recent examples
  - Yahoo Pipes
  - Microsoft Kodu
  - Google AppInventor
- Most could benefit from evaluation, or application of Cognitive Dimensions

Spreadsheets

- Widely used, sometimes for surprising purposes
  - A large proportion of commercial spreadsheets contain errors (Panko)
  - Spreadsheet research corpus (Scaffidi)
- Empirical studies and extensions:
  - Excel
  - Burnett's Forms/3, with free-format cells
- Specific usability improvements:
  - testing and debugging facilities such as WYSIWYT (Burnett)
  - type systems and generators (Erwig)
  - Functional programming in Excel (Peyton-Jones, Blackwell, Burnett)

Scripting Languages

- Allow users to customize and extend products, e.g.
  - LISP variants in AutoCAD and EMACS
  - Linden Scripting Language (LSL) in Second Life
  - Apple Automator (and earlier Hypercard)
- Key research concern in end-user programming (later)
  - Note that many evolve into professional languages (Perl, Flash)
  - While others never really considered end-user needs (TCL, JavaScript)
- Can address attention investment by starting with macro recording, then exposing source code for modification
  - Visual Basic in Microsoft Word
  - CoScripter for Firefox (Allen Cypher)

Lecture 2: Research methods in the study of programming.

Ethical Issues in Research

- Review the Cambridge Technology Ethics guide
  - What kind of study are you planning?
  - What potential concerns might there be?
  - What will you do to address them?
- Submit a proposal to the Computer Lab Ethics committee, giving above details.

Controlled Experimental Methods

- ***Participants*** (subjects), potentially in ***groups***
- Experimental ***task***
- Performance ***measures*** (speed & accuracy)
- Trials
- ***Conditions*** / Treatments / Manipulations
  - modify the programming language
  - use different languages
  - Use different features of the programming environment
- ***Effect*** of treatments on sample means
  - Within-subjects (each participant uses all versions)
  - Between-subjects (different groups use different versions)

Controlled Experiments

- Based on a number of observations:
  - How long did Fred take to fix this bug (speed)?
  - Did he get it right (accuracy)?
- But every observation is different.
- So we compare averages:
  - Over a number of trials
  - Over a range of people (participants)
- Results often have a normal distribution
  - Compare difference of means
- Require significance testing
  - What likelihood that result could occur at random?
  - Is difference of means large relative to variance?

Typical experimental tasks

- Production tasks
  - write a program that is correct, and write it quickly
- Comprehension tasks
  - understanding, interpretation or recall
- Search tasks
  - find code responsible for functionality, or bug
- May be possible to use standardised tasks, for comparison to previous PPIG research
  - See Blackwell list
  - But 'toy problems can lack external validity
- Perhaps use the six Cognitive Dimensions activities?

Experimental Manipulations

- Compare productivity gains (effect size) of version with new feature to one without?
  - Will system work without the new feature?
  - Will the experimental task be meaningful if the feature is disabled?
  - Must new feature be presented second in a within-subjects comparison (order effect)
  - Is your system sufficiently well-designed for external validity of productivity measure?
- Test a fundamental research question?
  - e.g. imperative vs declarative paradigms, textual vs visual syntax
  - Are your two languages properly representative of the paradigms, yet also equivalent in other respects?
  - Are your experimental tasks equally suited to different paradigms?
- Is full implementation necessary?
  - Can you simulate features with Wizard of Oz technique?

Measurement

- Speed (classically 'reaction time')
  - E.g. time to write program
- Accuracy (number of (non)errors).
  - Is program correct?
- Trade-off between speed and accuracy?
  - Or poor performance on both?
  - Check correlation between them
- Task completion:
  - Stop after a fixed amount of time (ideally < 1 hour)
  - Measure proportion of the overall task completed

Self-Report

- Did you find this easy to use? (Likert scale)
  - applied value: appeal to customers
  - theoretical value: estimate 'cognitive load'
- Danger of bias
  - Subjective impressions of performance inaccurate
  - Suffer from experimental demand
    - Participants want to be nice to the experimenter
    - Should disguise which manipulation is the novel one
- May be necessary to capture affect measures:
  - Did you enjoy it, feel creative/ enthusiastic?
- Alternative is to collect 'richer' data …

Think-aloud

- "Tell me everything you are thinking"
  - 'concurrent verbalisation'
- Problems:
  - Hard tasks become even harder while speaking aloud
  - During the most intense (interesting) periods, participants simply stop talking,
- Alternative:
  - make video recording, or eye-tracking trace
  - playback for participant to narrate
  - 'retrospective verbal report'

Qualitative Data

- Protocol analysis methods, e.g.
  - verbal protocol – transcript of recorded verbal data
  - video protocol – recording of actions
- Hypothesis-, or theory-driven
  - Create 'coding frame' for hypothesised categories of behaviour
  - Segment the protocol into episodes, utterances, phrases etc
  - Classify these into relevant categories (with inter-rater reliability)
  - Compare frequency or order statistically
- Grounded theory (ch 7 of HCI Research Methods)
  - Open coding, looking for patterns in the data
  - Stages of thematic grouping and generalization
  - Constant comparison of emerging framework to original data
  - More interpretive, danger of subjective bias

Experiment Design

- Arrangement of participants, groups, tasks, trials, conditions, measures, and hypothesized effects of treatments
- Within-subjects designs are preferred
  - because so much variation between programmers
- This leads to order effects:
  - first condition may seem worse, because of learning effect
  - last condition may suffer from fatigue effect
  - task familiarity – can't use the same task twice
- Precautions:
  - Prior training to reduce learning effects
  - Minimise experimental session length to reduce fatigue effects
  - Use different tasks in each condition, but 'balance' with treatment and order
- These are typically combined in a 'latin square' where each participant gets a different combination

Analysis

- For an easy life, plan your analysis before collecting data!
- Will quantitative data be normally distributed?
  - t-test to compare two groups
  - ANOVA to compare effect of multiple conditions (which include latin square of task and order)
  - Pearson correlation to compare relationship between measures
- Distributions of task times are often skewed:
  - a small number of individuals complete the task quite slowly
  - don't exclude 'outliers' who have difficulty with your system
  - log transform of time is usually found to be normally distributed
- Subjective ratings are seldom normally distributed
  - chi-square test of categories
  - 'non-parametric' comparison of means

Evaluation

- Rather than testing hypothesis, or comparing treatments
  - ask 'is my language usable'?
- More typical of commercial practice, for short-term goals, rather than general understanding
  - Formative evaluation assesses options early in design process
  - Summative evaluation identifies usability problems in a system you have built
  - Repeated for iterative refinement in user-centred design
- Weaker research, because no direct contribution to theory
  - However some mainstream applied research venues are starting to require evidence of any claims made for new tools
  - e.g. ICSE, OOPSLA/SPLASH

Field Study Methods

- Laboratory studies are not adequate for:
  - organizational context of software development
  - interaction within software development teams
  - behaviour of programmers in actual work context
- Typical methods:
  - 'contextual inquiry' interviews
  - 'focus group' discussions
  - 'case studies' of projects or organisations
  - 'ethnographic' field work as participant-observer
- All result in qualitative data, often transcribed, and analysed using grounded theory approaches
- You won't have time!

Lecture 3: Special classes of programming language use

Educational Languages

- Computer Science Education vs programming for children
  - Papert's Logo
  - Kay's Smalltalk
  - Repenning's AgentSheets
  - Cypher and Smith's StageCast
  - Kahn's ToonTalk
  - Kolling's Greenfoot
  - Carnegie Mellon's Alice
  - MIT's Scratch
- Many use VL techniques, to overcome syntax problems
  - Is it 'cheating' to avoid teaching syntax?
  - Or motivate children by making it easy for them to do things that interest them (videogames or animations)
  - 'learning to program' or 'programming to learn'?
  - 'user-centred' or 'curriculum-centred' design?
- If curriculum, what theoretical principles? Logic? Functional? Objects?

End-User Programming

- In Information Systems 'user' is a (professional) organisation
  - 'end-user' is a person who will actually use the system
  - an 'end-user programmer' both writes the program and uses it.
    - 'end-user development' (EUD)
    - 'end-user customisation' (EUC)
- Interesting research because:
  - An externally valid source of 'novice' programmers
  - Ubiquitous computing increases market for customisation
  - Professional programmers don't complain enough

End-User Programming

- EUP is usually defined to refer to a person who has
  - not trained as a programmer
  - not primarily employed as a programmer
  - does not program for its own sake, but as a means to an end
- Motivation for end-user software engineering (e.g. testing and debugging)
  - programs may be used by other people
  - programs may be business-critical
- Domain-specific languages
  - Programming 'novices' are often domain experts
  - LabView, MATLAB are both DSLs
  - Even some mainstream tools are increasingly domain-specific, e.g. WPF

Creative mashups and composition

- Not like military, industrial, bureaucratic domains
  - those are well structured, with ample resources.
  - leisure, media and the arts imply 'discretionary-use'
- digital media creators collage, sample & mash-up
  - art strategies are next generation agile methods
- Current generation of artist languages
  - Max/MSP (+ Jitter)
  - Processing
  - SuperCollider
- Current research in Cambridge
  - Flow in composition
  - Live coding
  - EUSE for improvisation processes

Domestic automation

- Classic domestic HCI challenges
  - home heating controls
  - VCR programming
  - privacy configuration
- Home networking
  - WiFi, Zigbee. X10
  - AutoHAN
  - software plumber or software DIY?
- Research opportunities
  - Understand domestic economy of digital technology
  - Apply gentle slope and attention investment

Lecture 4: Planning practical empirical studies.

Goal

- Prepare for design of your study
- Previous lectures followed:
  - theories of programming
  - experimental methods
  - specific users and programming technologies
- We use reverse order:
  - specific programming technologies and users
  - experimental methods
  - theories of programming

Candidate programming languages/tools

- your own personal research
  - e.g. MPhil dissertation
- Other research
  - other research in Cambridge
  - recent product releases
  - research prototypes developed elsewhere
- Who is the intended user?
- What will they be trying to achieve?

Representative tasks and measures

- Identify user activities you plan to observe
  - assigned tasks (controlled experiment)
  - or user's goal (observational study)
- Will these explore an interesting research question?
- What measures are relevant to that question?
- Will qualitative data analysis be necessary?
- Will there be a threat to external validity?
  - From task, measure or analysis

Review of study design options

- Do you wish to carry out a comparison, an evaluation, or an open exploratory study?
- If you plan to conduct a controlled experiment, will it be possible to use a within-subjects design?
- What data analysis method will you use?
- What would you need to do in order to complete a pilot study?
- What ethical issues are raised by your planned research?

Theoretical goal

- What do you expect to learn from conducting your study?
- What contribution will it make to the research literature relevant to usability of programming languages?
- Where would you publish the results?

Course structure

- Assignment A, presented at seminars 1 & 2
  - Target language, paradigm, tool or environment
  - Review of relevant literature
  - Study design
  - Outline of analytic methods
- Assignment B, presented at seminars 3 & 4
  - Full experimental report
  - Data analysis and findings
  - Suitable for publication at venue such as PPIG