Evaluating High-Level Program Specifications

Bjoern Hartmann, UC Berkeley EECS
ExCAPE Kickoff Meeting
How do we know if our tools have practical benefits for their users?
1. Build the system.

2. Recruit users.
   Give them programming tasks.

3. Compare their performance to the status quo.
Practical Challenges

• Too late:
  no more time to make changes

• 10x individual differences in programmer productivity
  [Sackman, Erikson, and Grant 1968]
  These differences can outweigh the benefits of new tools in studies!

• The best tools enable users to accomplish new tasks, so there is no easy comparison
Outline

• How to think about evaluation
• Case study: Proton
• Evaluating the ExCAPE challenge problems
Types of Evaluation

• Purpose of Evaluation
  Formative vs. Summative Evaluation

• How to Evaluate?
  Genres of assessment
Purpose of Evaluation

• **Formative:** to inform the design

• **Summative:** to characterize a design, once it is complete
Cheap Formative Techniques

- Prototype: Hardcode / mock up your interface for a few key tasks

- Wizard of Oz: manually simulate the output of your tool
Genres of Evaluation

**Automated** Usability measures computed by software

**Inspection** Based on skills, and experience of evaluators

**Formal** Models and formulas to calculate measures

**Empirical** Usability assessed by testing with real users
Inspection: Heuristic Evaluation

• *Rules of thumb* describing features of usable systems (different sets of rules for programming)

• Small set (3-5) of evaluators (experts) examine interface (or proposal) for compliance with heuristics
## Inspection: Cognitive Dimensions of Notation

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Closeness of mapping</td>
<td>Closeness of representation to domain</td>
</tr>
<tr>
<td>Hidden dependencies</td>
<td>important links between entities are not visible</td>
</tr>
<tr>
<td>Error-proneness</td>
<td>notation invites mistakes</td>
</tr>
<tr>
<td>Secondary notation</td>
<td>extra information in means other than syntax</td>
</tr>
<tr>
<td>Viscosity</td>
<td>resistance to change</td>
</tr>
<tr>
<td>Visibility</td>
<td>ability to view components easily</td>
</tr>
</tbody>
</table>
Empirical: Qualitative Studies

- **Formative: Contextual Inquiry**
  try to understand user’s tasks and conceptual model

- **Summative: Usability Studies**
  look for critical incidents in interface

- Qualitative methods help:
  - Understand what is going on
  - Look for problems
  - Roughly evaluate usability of interface
Empirical: Quantitative Studies

• **Goal**
  - Use to reliably measure *some* aspect of interface
  - Compare two or more designs on a measurable aspect

• **Approaches**
  - Collect and analyze user events that occur in natural use
  - Controlled experiments

• **Examples of measures**
  - Time to complete a task, Average number of errors on a task, Users’ ratings of an interface
Addressing the tool evaluation challenge

- Triangulation of different methods:
  - Demonstrate expressivity (existence proof)
  - Sharpen the focus of quantification: ask more specific questions
- Qualitative observations on open-ended tasks
- Collect Self-report (weak support)
- Within-subject designs (same person uses multiple interfaces)
Case Study
Proton: Multitouch Gestures as Regular Expressions

Kenrick Kin\textsuperscript{1,2} Björn Hartmann\textsuperscript{1} Tony DeRose\textsuperscript{2} Maneesh Agrawala\textsuperscript{1}

\textsuperscript{1}University of California, Berkeley \textsuperscript{2}Pixar Animation Studios
Event-Handling

mouseDown() → mouseMove() → mouseUp()
Event-Handling

- mouseDown()
- mouseMove()
- mouseUp()

- touchesDown()
- touchesMove()
- touchesUp()
Current Gesture Implementation

```
_state = GesturePossible;

touchesDown(Array *touches, Event *event)
    if(event->allTouches()->count() == 1)
        if(touches[0]->target() != 'n')
            _state = GestureFailed;
    else if(event->allTouches()->count() == 2)
        if(touches[0]->target() != 'm')
            _state = GestureFailed;
    else
        _state = GestureFailed;

touchesMove(Array *touches, Event *event)
    for(i = 0; i < touches->count(); i++)
        if(touches[i]->touchId() == 0 && touches[i]->target() != 'n')
            _state = GestureFailed;
        else if(touches[i]->touchId() == 1 && touches[i]->target() != 'm')
            _state = GestureFailed;

touchesUp(Array *touches, Event *event)
    if(touches[0]->touchId() == 0)
        if(event->allTouches()->count() == 1 && touches[0]->target() == 'n')
            connectNodes();
            _state = GestureRecognized;
        else
            _state = GestureFailed;
    else
        if(event->allTouches()->count() == 1 || touches[0]->target() != 'm')
            _state = GestureFailed;
```
Managing Large Gesture Sets

[Wobbrock '09]
Managing Large Gesture Sets

[Wobbrock ’09]

Conflict Detection and Resolution
Proton
\[
gesture = D^n_1 M^n_1 \ast D^m_2 (M^n_1 \mid M^m_2) \ast U^m_2 M^n_1 \ast U^n_1
\]
\[
gesture.finalTrigger(connectNodes());
gestureMatcher.add(gesture);
\]
gesture = $D^n$ $M^n_1*D^m_2(M^n_1\mid M^n_2)*U^m_2 M^n_1*U^n_1$

```
gesture.finalTrigger(connectNodes());
gestureMatcher.add(gesture);
```

Declarative Specification
gesture = $D^n_1 M^n_1 \ast D^m_2 (M^n_1 \mid M^m_2) \ast U^m_2 M^n_1 \ast U^n_1$

gesture.finalTrigger(connectNodes());
gestureMatcher.add(gesture);

Declarative Specification

Recognition Code Generation
gesture = $D_1^m M_1^m D_2^m (M_1^n | M_2^n) U_2^m M_1^m U_1^n$

gesture.finalTrigger(connectNodes());
gestureMatcher.add(gesture);

Declarative Specification

Recognition Code Generation

Conflict Detection
D_1^c M_1^c \ast\left(D_2^a (M_1^c | M_2^a) \ast U_2^a M_1^c \ast \left| D_2^a (M_1^c | M_2^a) \ast D_3^a (M_1^c | M_2^a | M_3^a) \ast \ldots \right.\right.

\text{panCamera()}

\left.\left( U_3^a (M_1^c | M_2^a) \ast U_2^a M_1^c \ast \left| U_2^a (M_1^c | M_3^a) \ast U_3^a M_1^c \ast \right) \right) \ast U_1^c

\text{zoomCamera()}
Proton Touch Event
Touch Event Symbol

$E \in \{D, M, U\}$
Touch Event Symbol

E

ObjectType

TouchID

Thursday, June 7, 12
Touch Event Symbol

D

ObjectType

TouchID
Touch Event Symbol

Object Type

Touch ID
Touch Event Symbol

E  ObjectType

TouchID
Touch Event Symbol

E

ObjectType
Touch Event Symbol

E^{ObjectType}_{2}
Touch Event Symbol

$E^3_{ObjectType}$
Touch Event Symbol

E

ObjectType

TouchID

Object Type

Thursday, June 7, 12
Touch Event Symbol

E

s = star

TouchID
Touch Event Symbol

E_b = background

TouchID
Touch Event Symbol

$D^s_1$

Touch Down with 1st Touch on Star
Touch Event Symbol

$M^b_2$

Touch Move with 2nd Touch on Background
Translation Gesture

\[ D^s_i M^s_i * U^s_i \]

\[ \text{translate()} \]

s

b
Translation Gesture

\[ D_i^s M_i^s U_i^s \]

\text{translate()}

Thursday, June 7, 12
Translation Gesture

$D_i^s M_i^s U_i^s$

$\text{translate()}$

Thursday, June 7, 12
Translation Gesture

\[ D_i^s M_i^s U_i^s \]

\text{translate()}

Thursday, June 7, 12
Translation Gesture

\[ D_i^s M_i^s U_i^s \]

\[ \text{translate()} \]

\[ b \]
Rotation Gesture

\[ D_s^1 M_s^1 D_b^2 (M_s^1 | M_b^2) (U_s^1 M_b^2 U_b^1 | U_b^1 M_s^1 U_s^1) \]

rotate()
Rotation Gesture

\[ D_1^s M_1^s \ast D_2^b (M_1^s \mid M_2^b) \ast (U_1^s M_2^b \ast U_2^b \mid U_2^b M_1^s \ast U_1^s) \]

\[ \text{rotate()} \]

Thursday, June 7, 12
Proton Architecture

- Hardware
- Stream Generator
  - Hit-Tester
- Gesture Matcher
  - Gestures
- Gesture Picker
  - Confidence Calculators

Flow:
- Raw input
- Touch event stream
- Matched gestures
- Execute gesture callback
Gesture Tablature
Gesture Tablature

s

s
Gesture Tablature

touch down
Gesture Tablature

touch down  touch move
Gesture Tablature

touch down  touch move  touch up
Gesture Tablature
Gesture Tablature

Thursday, June 7, 12
Gesture Tablature

s

b

s

b

Thursday, June 7, 12
Gesture Tablature
Gesture Tablature

rotate()

s
---
b
---
s
b
---
Gesture Tablature

\[ D^s_1 M^s_1 \ast D^b_2 (M^s_2 \mid M^b_2) \ast (U^s_1 M^b_2 \ast U^b_2 \mid U^b_2 M^s_1 \ast U^s_1) \]
Tablature Editor

Proton: Gesture Tablature Editor

GESTURE NAME
Rotate

New Track
Delete Track

() * |

s

a

s

a

rotate()
Existence Proof: Applications
2D Shape Manipulation
Sketching
User Study:
Build applications with/without Proton
User Study:
Build applications with/without Proton
User Study: Readability of Notation
Method

• Question: How quickly and accurately can programmers comprehend gestures expressed as: [iOS code, regular expressions, tablature]?

• Recruited n=12 experienced programmers

• Task: given a definition, identify a video of a gesture matching that definition
_state = GesturePossible;

touchesBegan(Array *touches)
    if(touches->allTouches()->count() > 2)
        _state = GestureFailed;
    if(touches[0]->target() != 'a')
        _state = GestureFailed;

touchesMoved(Array *touches)
    for(i = 0; i < touches->count(); i++)
        if(touches[i]->target() != 'a')
            _state = GestureFailed;

touchesEnded(Array *touches)
    if(touches[0]->target() != 'a')
        _state = GestureFailed;
    return;
    if(touches->allTouches()->count() == 2)
        if(touches[0]->touchId() == 0)
            execute();
        else
            _state = GestureFailed;
    else
        _state = GestureRecognized;
\[ D_0^a M_0^a \ast (D_1^b (M_0^a \mid M_1^b) \ast U_1^b M_0^a) \ast U_0^a \]
Counterbalancing

Participant 1

Participant 2

Participant 3

...
Results

Time (s) to Understand Gesture

<table>
<thead>
<tr>
<th>Platform</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tablature</td>
<td>1x</td>
</tr>
<tr>
<td>Expression</td>
<td>4.7x</td>
</tr>
<tr>
<td>iOS</td>
<td>140</td>
</tr>
</tbody>
</table>
Next Study:
Writability of Notation
Other Perspectives

• **Debugging:** Present gestures with bugs, ask developers to fix them

• **Maintenance, extension:** Give an existing program, ask developers to extend it

• **Creation:** Assign open-ended programming tasks, record cognitive process through thinkaloud protocol
Scaling to ExCAPE

- 8 institution, 18 PIs, 4 challenge areas: a LOT of diversity in applications & users.
- For each challenge area or team, identify a graduate student user evaluation lead.
- Think about what you can do now, this year, to reduce uncertainty about usability and utility for users (prototype, approximate, simulate, fake)