Software, Tools, and Toolkits
Announcements

You just did a quiz. So, that’s cool.
Last time

Intelligence augmentation aims to place AI in context by using it to amplify our own abilities.

Debates rage about the levels of autonomy to grant to AIs: from fully autonomous agents that act on the person’s behalf, to direct manipulation that always leaves the user in full control.

Mixed initiative interaction splits the difference by asking, acting, or doing nothing based on its confidence and assessment of the benefit.

End users and designers seek to draw on these tools.
Today

Threshold and ceiling

Changing problem representations

Learning programming
A Small Matter of Programming

Software engineering is a highly complex task, a microcosm of many challenges in HCI.

Making software engineering more accessible could empower millions to customize applications and write programs.
Programming ain’t easy

Developers struggle to recover others’ implicit knowledge by inspecting code [LaToza, Venolia and DeLine 2006; Ko, DeLine and Venolia 2007; Ko et al. 2006]

Developers rarely hold all information needed for the task, and often must turn to the web [Brandt et al. 2009]

- Just-in-time learning of new skills, clarifying existing skills
- Reminding themselves of details

Barriers span from conceptual (how is this even possible to code?) to pragmatic (how do I express this?) [Ko, Myers, and Aung 2004]
How do we aid programming?
Threshold and Ceiling
What is your programming intervention actually doing?

What is Codex’s design goal? How do we know if it’s succeeding at that design goal?

Are some programming languages “better” than others? How would we know?

Is the VSCode plugin helping? With what?
Threshold/Ceiling Diagram

[Myers, Hudson and Pausch, TOCHI 2000]

Threshold: Difficulty of use

Ceiling: Sophistication of what can be created

Are you trying to lower the threshold, or raise the ceiling?

- C++
- Server-side web
- Client-side web
- Figma
Lowering the threshold

Goal: reduce the effort and cognitive complexity of creating software artifacts
How to lower thresholds

One approach is to reduce the ceiling (expressivity) in exchange for a smaller gulf of execution or evaluation.

Regular expressions are simpler to understand than context-free grammars, but also less expressive.

No-code or low-code front-end web frameworks can be fast to get off the ground but limited in what you can create.

Python manages memory and garbage collection for you, but also trades off some manual optimizability of memory to achieve it.

But, not all lowered thresholds require lower ceilings — we will talk in a moment about how representations shape cognition.
Asking ‘why’ questions of code

[Ko and Myers CHI ’04, ICSE ’09]

Debugging problems often reduce to “why” questions, but these questions are challenging to answer (=high threshold).

Analyze program traces to answer many unanswered “why” and “why not” questions about what just happened.
Data science notebooks

Automatic cleanup of Jupyter notebooks by tracking provenance across cells [Head et al. 2019]
Programming by demonstration (PBD): teach a computer a program by doing it yourself while it watches.

Challenges:

- There is an infinite, and hugely branching, space of programs that might be inferred.
- Inferred macros can be extremely brittle.
PBD on the desktop

Infer a macro by watching the user's behavior
Modern PBD: Excel flash fill

[Gulwani 2011]

Develop a domain-specific language of string transformations, and learn from examples how to decompose it into subproblems.

Machine learning ranks between all possible valid programs.
Raising the ceiling

Goal: increase the complexity of what can be created
How to increase the ceiling

Identify opportunities for **untapped expressivity** in the current language, and position the software to expose that level of expressivity.

This is not about “adding knobs”: it’s about (metaphorically) providing new paint colors in the palette.
Non-programming examples

Engelbart’s chorded keyset
[Engelbart 1968]

Musical instruments: the goal isn’t to reduce the threshold to playing the piano — it’s to enable high musical expressivity
Programmable artist brushes

Jacobs et al. 2018

Attaching computational functions to brushes enables new forms of artistic expression.

Dynamic Brushes is a programming and drawing environment for creating procedural drawing tools.
Recall: knitting design tools

[Albaugh, Hudson, and Yao 2019]

Digital specification enables yarn creations that bend, stretch, and compress in new ways.
Programming as problem representation
Cognitive amplification

By better understanding human cognition, we can design technology that makes us smarter.

Automation can help, but ultimately this power comes from better representation.

“The powers of cognition come from abstraction and representation: the ability to represent perceptions, experiences, and thoughts in some medium other than that in which they have occurred, abstracted away from irrelevant details.” [Norman 1994]
Take turns picking numbers in 1,2,3,4,5,6,7,8,9 without replacement. Win if any three of your numbers add up to 15. It's OK if you have extra numbers in your hand, as long as three of them add up to exactly 15.
Ready, set, go!

I will show the series of moves from players A and B so far. Raise your hand when you know what B's best next move should be.

A takes 4.
B takes 9.
A takes 2.
B takes 8.
A takes 5.

What should B do?
Re-encoding number scrabble

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>9</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>A</td>
<td>B</td>
<td>A</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td></td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Domain-specific languages

DSLs, or domain-specific languages, are programming languages that are tailored to a specific domain

- SQL (databases)
- d3 / Vega Lite (visualization)
- pytorch (machine learning)

Successful DSLs reshape the cognitive representation of the task, reducing the gulfs of execution and evaluation and empowering development in their application domain
Representations for vis
[Bertin 1983; Mackinlay 1986; Satyanarayana 2016]

How do we tell a machine to create this? Paint pixels?

It’s extremely challenging until we adopt a representation that visualizations are encodings of data types into marks.

```javascript
vl.markPoint()
 .data(data2000)
 .encode(
   vl.x().fieldQ('fertility'),
   vl.y().fieldQ('life_expect'),
   vl.size().fieldQ('pop').scale({range: [0, 1000]}),
   vl.color().fieldN('cluster'))
 .render()
```

[Heer on Observable]
I have too much data to fit in my computer. How do I count the number of times the word “HCI” appears on the web?

Representation: Map-Reduce [Dean and Ghemawat 2008]

First, run a Map phase that runs a simple function over each webpage. That function outputs the number of HCIs, and can be run completely in parallel across every page on the web.

Second, run a Reduce phase that collects the outputs from the Map phase and aggregates them: here, via a sum.
Data science representations

How do I get the average of two variables for two plants?

Base R:
```r
as.data.frame(t(sapply(X = split(x = CO2[which(CO2$Plant %in% c("Poppy", "Daffodil")),
which(colnames(CO2) %in% c("stemdiameter", "leafdiameter"))],
   f = CO2$Plant[which((CO2$Plant %in% c("Poppy", "Daffodil")))],
   drop = TRUE), FUN = function(x) {apply(x, 2, mean)})))
```

tidyverse:
```r
CO2 %>%
  filter(Plant %in% c("Poppy", "Daffodil")) %>% # Get the two plants we care about
  select(Plant, stemdiameter, leafdiameter) %>% # Focus on the variable we want
  group_by(Plant) %>% # Separate/group the analysis to focus on individual plant
  summarise_all(mean) # Get the means for area and uptake
```

[Wetlands and Landscapes]
Learning programming
Constructionist learning: learning happens most effectively when people are making tangible objects.

Lego Mindstorms followed this mold and was named after it.
Scratch

[Resnick et al. 2009]

Logo's inheritance: block-based programming of simple animations and games as a gateway to programming for children
Online python tutor

[Guo 2013]

Embeddable Python data structure visualization

Over 200,000 users and a dozen universities using it

```python
1  def listSum(numbers):
2       if not numbers:
3           return 0
4       else:
5           (f, rest) = numbers
6           return f + listSum(rest)
7
8       myList = (1, (2, (3, None)))
9       total = listSum(myList)
```

Edit code

Frames

- Global variables
  - listSum
  - myList

- Objects
  - function listSum(numbers)

- tuple
  - 0 1
  - 0 2
  - 0 3

- tuple
  - 1

- tuple
  - 2

- tuple
  - 3
  - None
Watch many learners code and debug in real time.
Clustering student programs

[Glassman and Miller 2015]
Successful programming tools shift our cognitive problem representations to make the task more readily solvable.

Programming tools often either aim to reduce the threshold or increase the ceiling — how depends on which one we’re pursuing.

Successful programming tools shift our cognitive problem representations to make the task more readily solvable.

Tools for learning programming help externalize our cognition to better understand what code is doing (or ought to be doing).
References


References


