Lecture 2: Designing for Human-AI Interaction

CS698Y 2025-26 Semester-1

Last class…

- AI is everywhere
- Algorithm-centric → Human-centric
  - Accuracy is not enough!
  - Various examples of accurate AI model use questioned
  - Importance of human aspects
- HAI is different from traditional user interaction design
  - Uncertainty, non-determinism, error, trust, initiative
  - Evolves over time (e.g., with more data, online learning)
  - Explanations and transparency (no one wants arbitrary black box predictions; e.g., course recommendations)

Today:

- Key HAI principles / principles
- How to design according to those principles / guidelines
- Hands-on exercise

Why do we need guidelines?

- Avoid common pitfalls (over trust, confusion, misuse)
- Create consistency in user experience
- Promote fairness, safety, usability
- Bridge the gap between developers and users
- Note: guidelines are “rules of thumb”
  - Practical, comes from experience, but need to be routinely revisited
  - New system kinds => new guidelines
  - Don’t be myopic → look at experiences beyond individual features

Key practical guidelines (from Industry)

- Google’s People AI guidelines (PAIR)→ incl. ethical aspects
- Microsoft’s Human AI experiences (HAX) toolkit
- Google’s Explainability Rubric
- HCI Institute’s toolkit (for ideation)

- A small set of common needs, mistakes, pitfalls
  - Abstracted out to guidelines, patterns, etc.

Steps

- What to build?
- Decide on level of control?
- Ensure a good user experience (HAX guidelines)?
- How to evaluate whether your AI is successful or not?

Step 1: What to build? Do we need AI?

- Two ways features / products are built
  - People-centric: user needs
  - Technology-centric: have the hammer & look for a nail!

- Former is required for long-term success of a product
- So, keep people at its centre!

What is AI good at? (CMU HCII AI Design Toolkit)

- Detection
- Identification
- Estimation
- Forecast / Predict
- Compare
- Discover
- Generate
- Act

Step 1: What to build (contd.)

- How pervasive is the problem?
- Who does it affect?
- How does this problem affect various individuals?
- How does this problem vary across different groups and dimensions of identity?
- Does the problem's impact change over time?

Step 2: Decide on the role of AI

- Automation vs. augmentation
  - Should AI automate things for and act on your behalf?
    - With minimal user invention?
  - Should AI augment human capabilities?
    - Add on to humans, with specific strengths than people (e.g., crunch data).

- Examples:
  - Predict rainfall and declare leave
  - Predict rainfall, and let people decide what they want to do.

Too much automation is problematic

- Errors
- People might over trust
- People get complacent and don’t check routinely

- Don’t automate!
  - Simply framing makes a difference here.
  - Cancer predictor → cancer testing prioritization decision support

Too much automation is problematic

- What would they hand out to an assistant?
  - If not, then don't stick an AI.

- If yes, what skills do they expect the assistant to know?
  - If AI doesn't have such skills, don't replace human with AI / frame careful.
  - How much oversight/supervision/trust → if none, frame carefully.

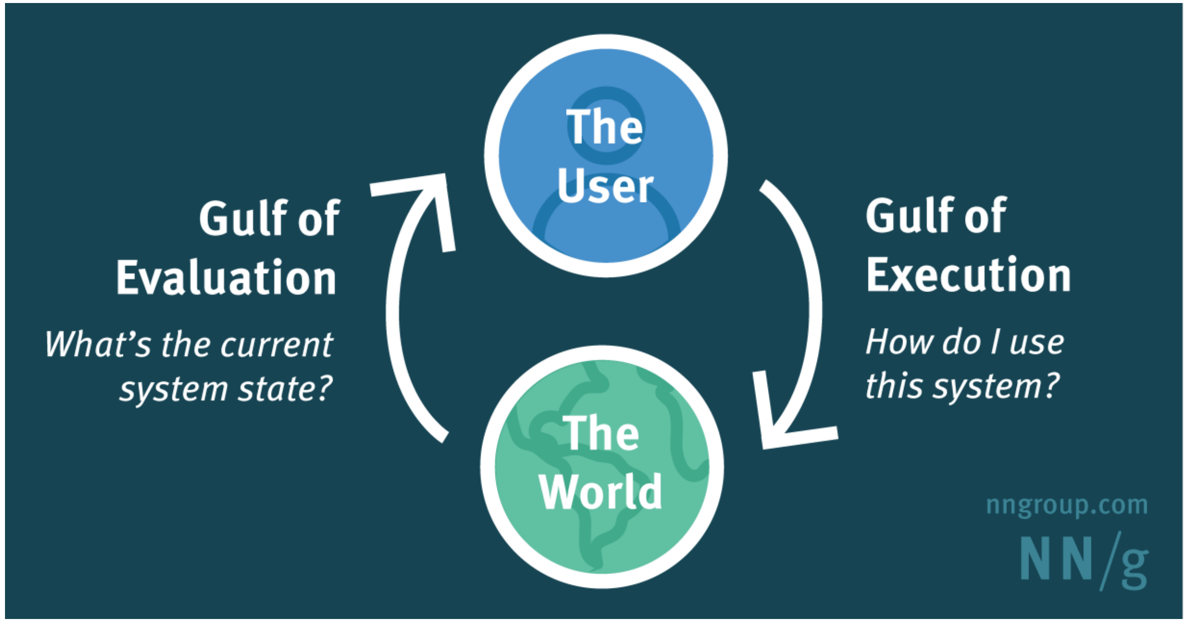
- Repetitive, too large/resource constraints, minimal oversight
  - These are great times to automate!

Also consider initiative

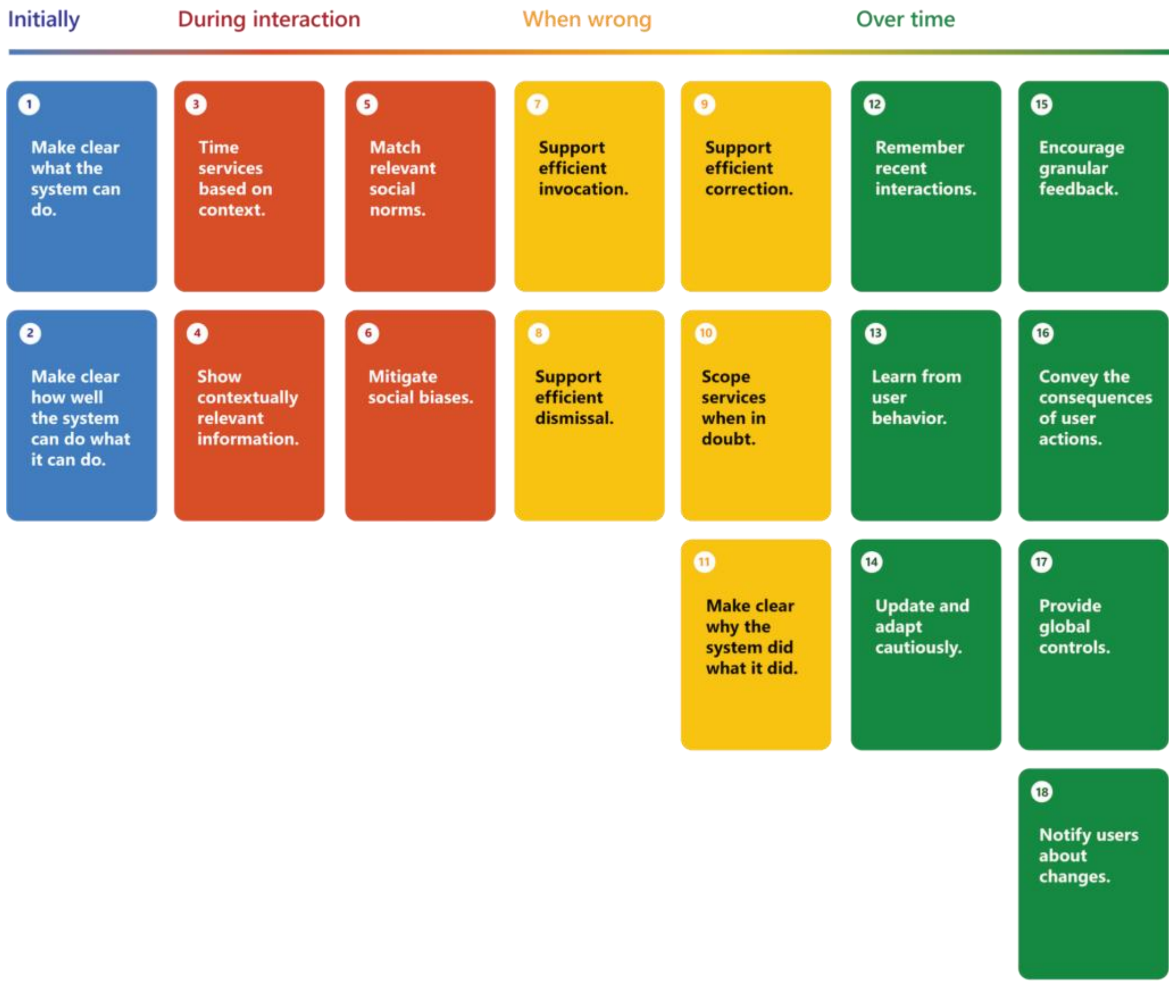
- User takes initiative to invoke AI
  - Spell check turn on!
  - Ask Generative AI
- AI takes initiative, but user can ignore / accept as final action
  - (Mixed initiative): Google doc spell check
- AI takes initiative and acts on behalf of user
  - Typo replacement

- Step 1: What to build?
- Step 2: How to build?
  - Initiative, automation vs. augmentation
- Step 3: How to build well?

Bridging Norman gulfs: Good usability



Microsoft’s HAX guidelines



Initially

1 Make clear what the system can do.

2 Make clear how well the system can do what it can do.

During interaction

3 Time services based on context.

4 Show contextually relevant information.

5 Match relevant social norms.

6 Mitigate social biases.

When wrong

7 Support efficient invocation.

8 Support efficient dismissal.

9 Support efficient correction.

10 Scope services when in doubt.

11 Make clear why the system did what it did.

Over time

12 Remember recent interactions.

13 Learn from user behavior.

14 Update and adapt cautiously.

15 Encourage granular feedback.

16 Convey the consequences of user actions.

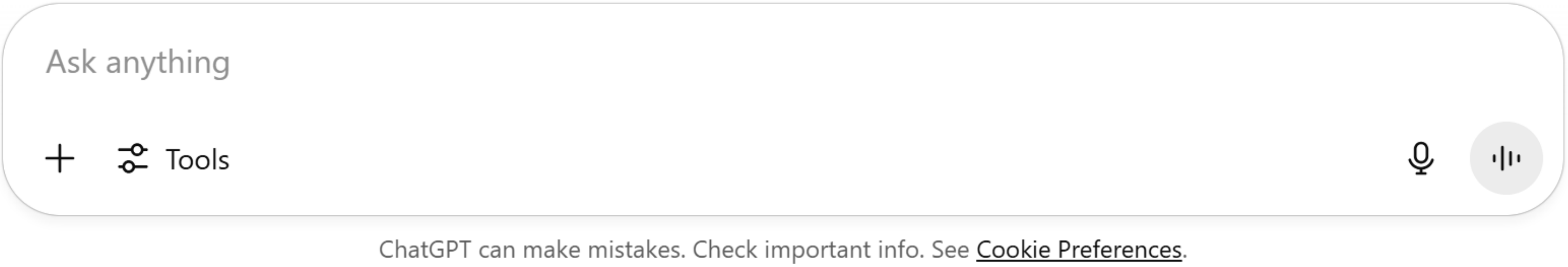
17 Provide global controls.

18 Notify users about changes.

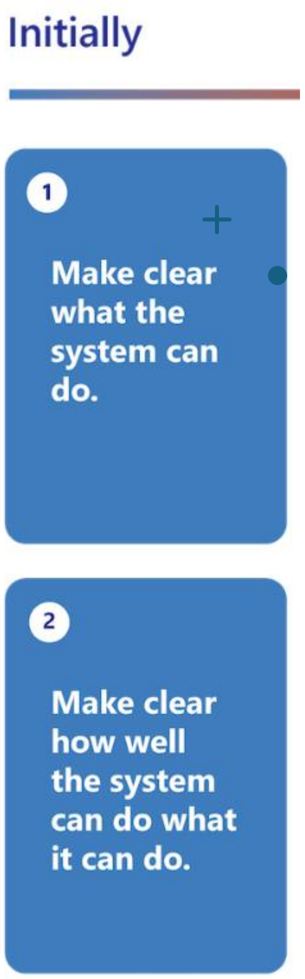
HAX guidelines Set #1: Initial context

- Why is this important?
  - Not clear how to use a system
  - Incorrect use → leading to frustration
  - Under-use of the system

- Make clear how well it can do that
  - Not lead to over trust, or under trust in system.



ChatGPT can make mistakes. Check important info. See Cookie Preferences.



Initially

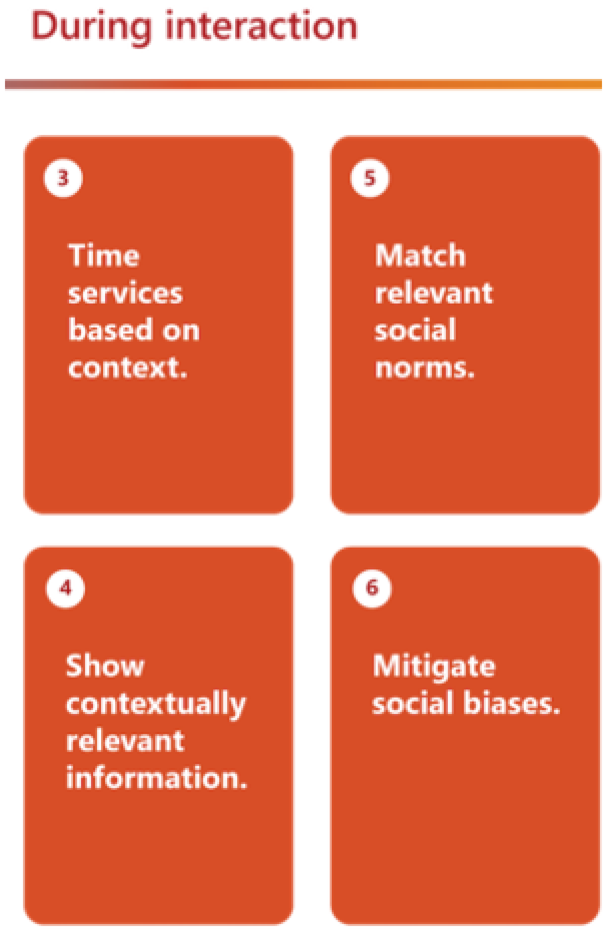
1 Make clear what the system can do.

2 Make clear how well the system can do what it can do.

HAX guidelines Set #2: During Interaction

- Time services based on context
  - When to provide recommendations/suggestions/…
  - Examples:
    - Google Map Navigation: “In 200 m, turn left”
    - Rewrite sentences in Word, slide design in powerpoint.

- Show contextually relevant information
  - Not a list of all possible things
  - “Also see” recommendations on Amazon.



During interaction

3 Time services based on context.

5 Match relevant social norms.

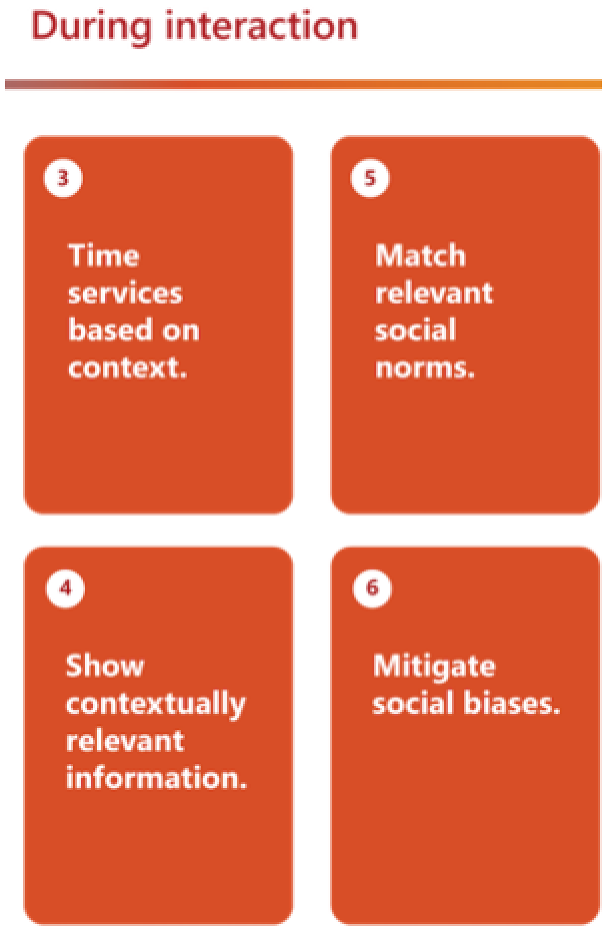
4 Show contextually relevant information.

6 Mitigate social biases.

HAX guidelines Set #2: During Interaction

- Match relevant social norms
  - Look for people-people norms
  - Examples:
    - Person navigator automatically says: “in a bit, turn left”
    - India vs. Western Google Map directions
    - “Let me know if you need details” (ChatGPT)
    - Explanations—course recommendations.

- Mitigate social biases
  - Walking speed on Google Maps
  - Gender, conformity, …



During interaction

3 Time services based on context.

5 Match relevant social norms.

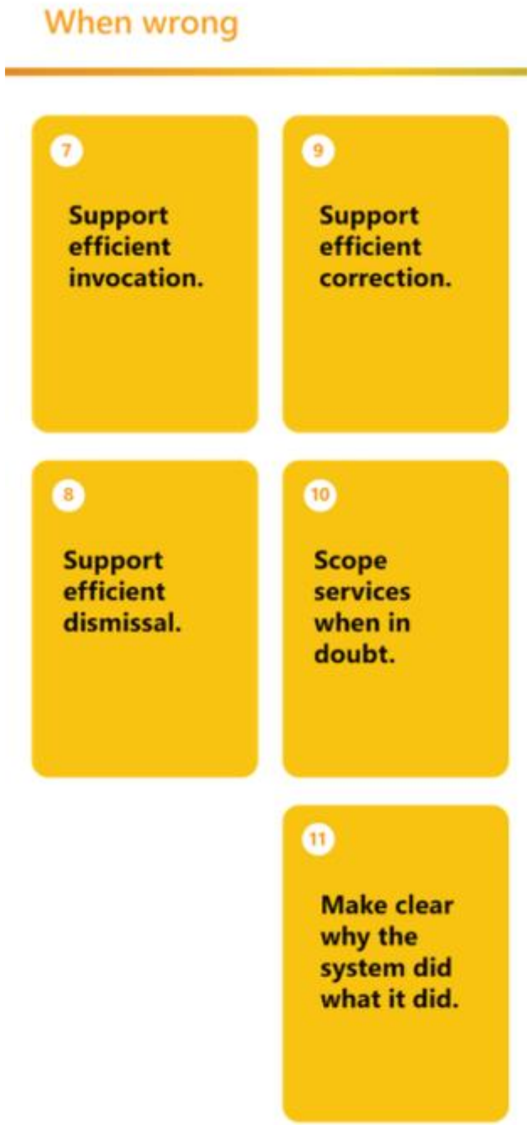
4 Show contextually relevant information.

6 Mitigate social biases.

HAX guidelines set #3: When wrong

- Support efficient invocation / dismissal
  - Designer in Power Point
  - If it doesn’t show up, call “Designer”
  - If it shows up unwarranted, turn off.

- Support efficient correction
  - Auto spelling corrections → simply UNDO



When wrong

7 Support efficient invocation.

9 Support efficient correction.

8 Support efficient dismissal.

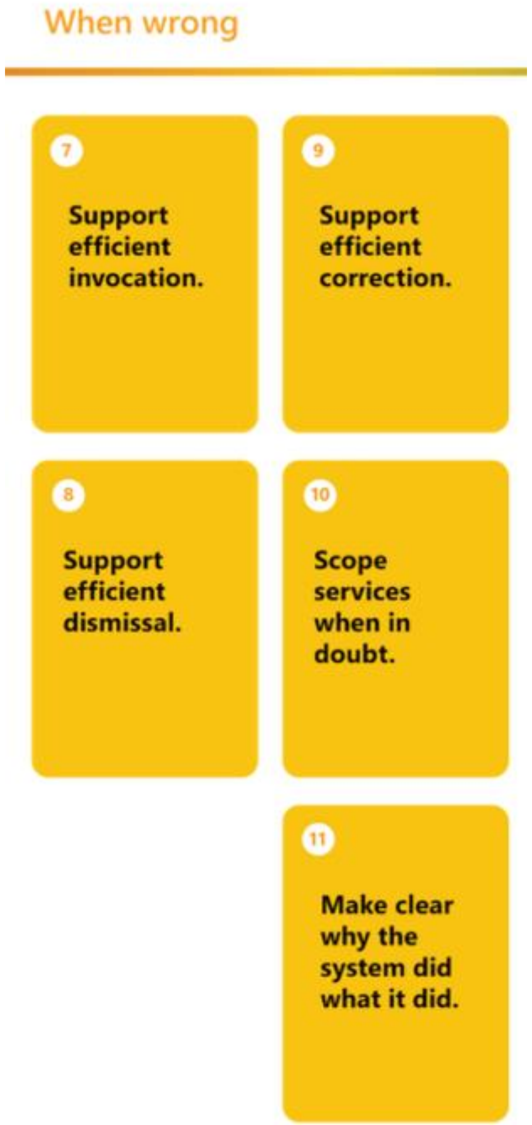
10 Scope services when in doubt.

11 Make clear why the system did what it did.

HAX guidelines set #3: When wrong

- Scope services when in doubt
  - “Tell me about Mac”.
  - Instead of assuming, ask if it is about:
    Mac prefix in names or Mac computers
  - Service→ scoped down, but graceful and error-proof.

- Make clear why the system did what it did
  - Google Maps → fastest, least traffic, avoids tolls.
  - Sometimes suggests silly routes, without saying why.
  - Not always easy → more on this later!



When wrong

7 Support efficient invocation.

8 Support efficient dismissal.

9 Support efficient correction.

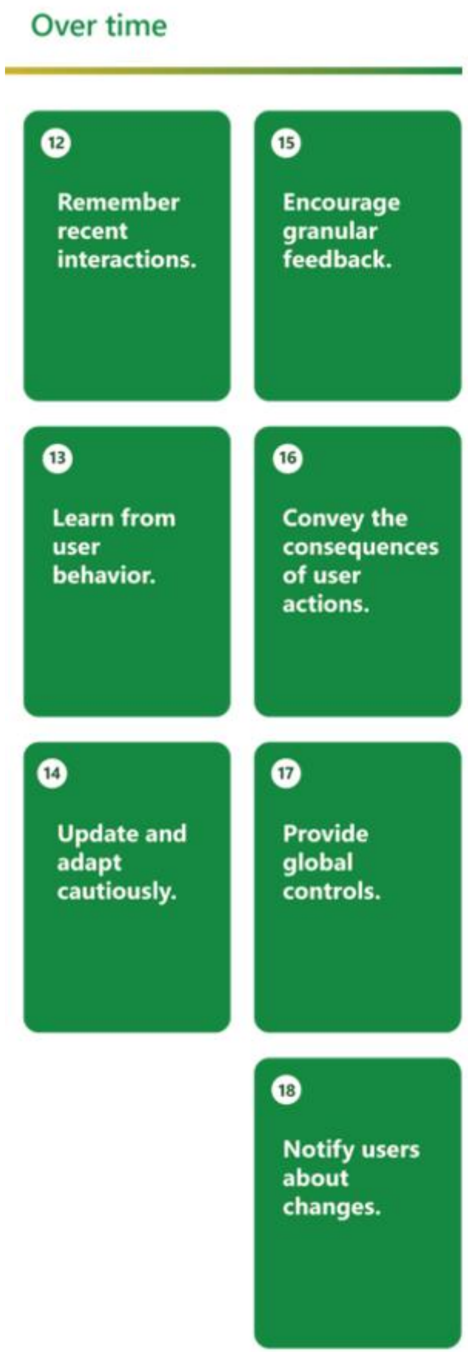
10 Scope services when in doubt.

11 Make clear why the system did what it did.

HAX guidelines set #4: Over time

- Remember recent interactions
  - Common usability need
  - Pick up where you left, recent documents…

- Learn from user behavior
  - Not done very well → privacy laws
  - “You usually open this document around this time”



Over time

12 Remember recent interactions.

13 Learn from user behavior.

14 Update and adapt cautiously.

15 Encourage granular feedback.

16 Convey the consequences of user actions.

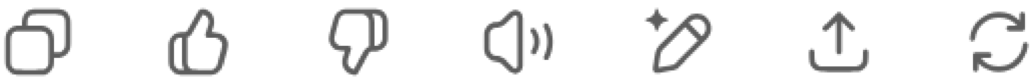
17 Provide global controls.

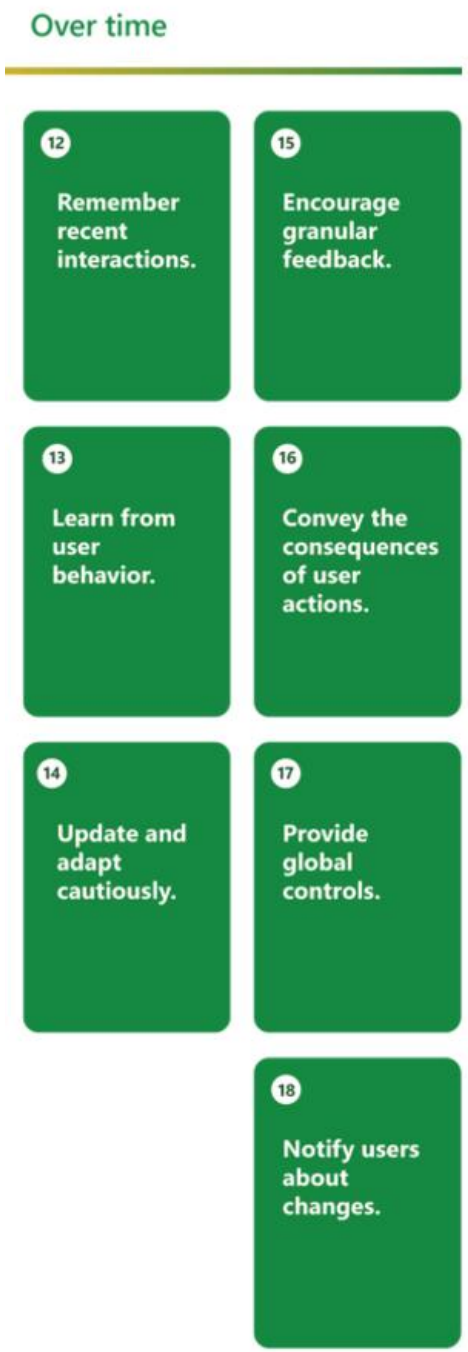
18 Notify users about changes.

HAX guidelines set #4: Over time

- Update and adapt cautiously
  - Why?
  - Be transparent, notify old vs. new responses
  - Allow to go back to old version
  - Warn if an old version is going to be retired

- Encourage granular feedback





Over time

12 Remember recent interactions.

13 Learn from user behavior.

14 Update and adapt cautiously.

15 Encourage granular feedback.

16 Convey the consequences of user actions.

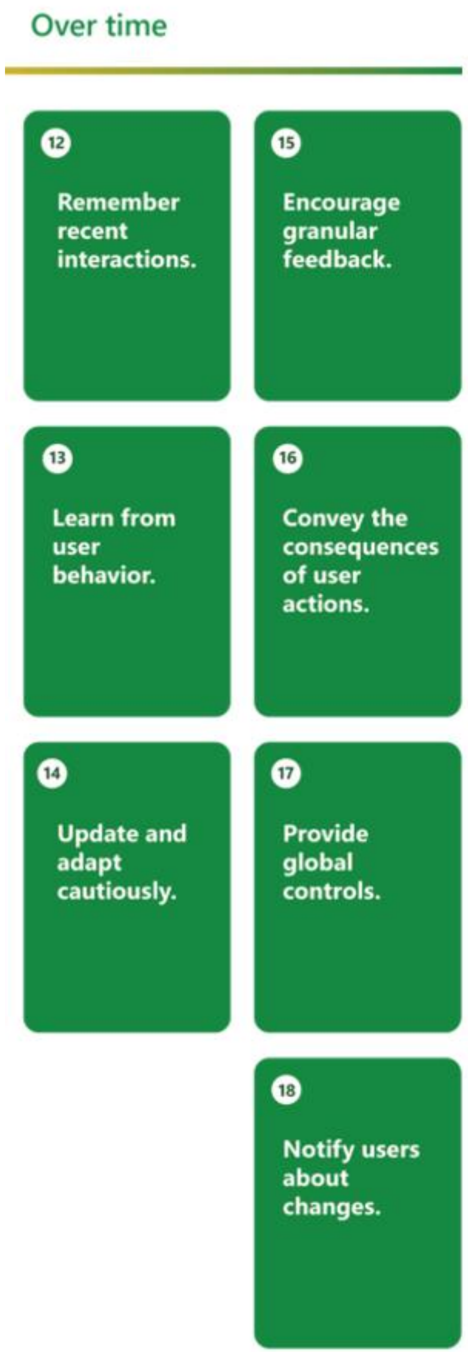
17 Provide global controls.

18 Notify users about changes.

HAX guidelines set #4: Over time

- Convey the consequences of user actions
  - Delete → say if it can be recovered or not
  - Dismiss → will not show up, until you hail from settings.
  - Internet will not be used

- Provide global controls
  - Turn on/off
  - What data to consider or not
  - Automatic vs. only when hailed …



Over time

12 Remember recent interactions.

13 Learn from user behavior.

14 Update and adapt cautiously.

15 Encourage granular feedback.

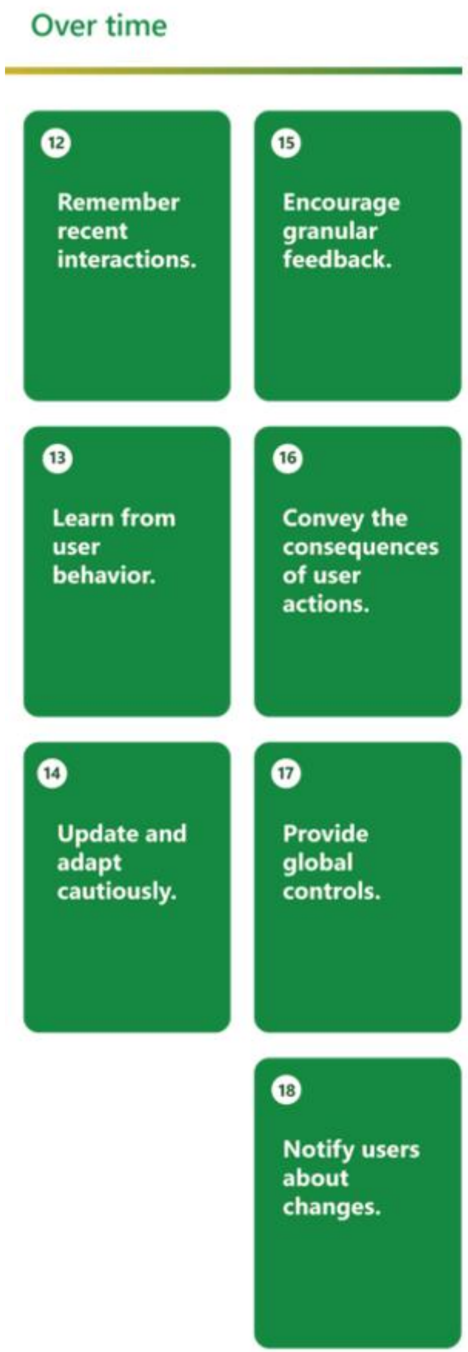
16 Convey the consequences of user actions.

17 Provide global controls.

18 Notify users about changes.

HAX guidelines set #4: Over time

- Notify users about changes
  - Again, common usability guideline
  - Important, because changes are not visible
  - Clarify what changes, allow going back, …



Over time

12 Remember recent interactions.

13 Learn from user behavior.

14 Update and adapt cautiously.

15 Encourage granular feedback.

16 Convey the consequences of user actions.

17 Provide global controls.

18 Notify users about changes.

Initially

During interaction

When wrong

Over time

Warning!

Do not think of individual features; but of entire experience of the user using the tool.

How do we use these guidelines?

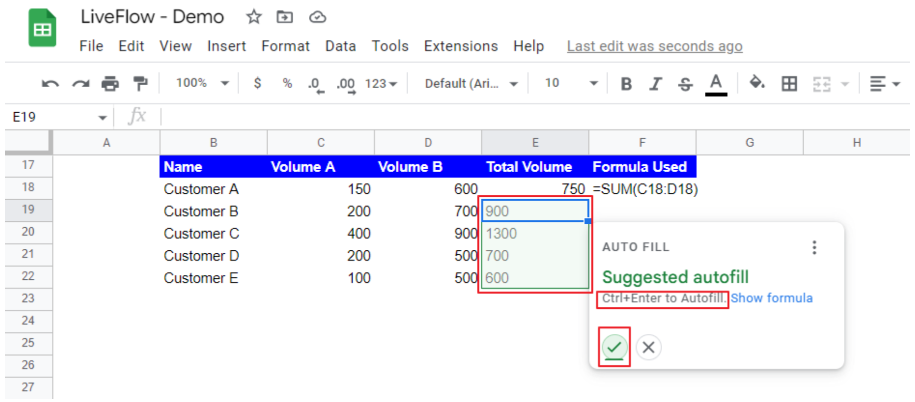
- Evaluate an existing tool / AI feature
- When designing a new AI feature
  - Easier to do it before than change it later
  - Sometimes, involves changing the model itself!

Hands-on activity…

- … But first, Acadly attendance!

Hands-on activity

- Work in pairs
- Pick one of two features:
  - Google sheets → formula autofill
  - Google Chrome → Open PDF document → summary generated on left
- See how well the feature of your choice does on these 18 guidelines


LiveFlow - Demo
File Edit View Insert Format Data Tools Extensions Help
Last edit was seconds ago
100%
Default (Ari...
10
E19

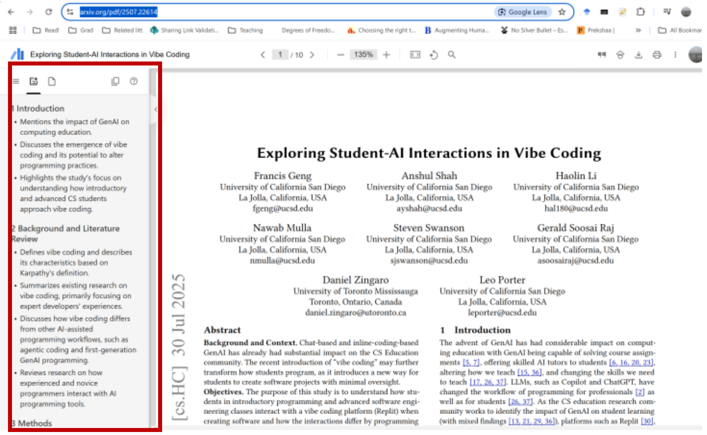
	A	B	C	D	E	F	G	H
17		Name	Volume A	Volume B	Total Volume	Formula Used		
18		Customer A	150	600	750	=SUM(C18:D18)		
19		Customer B	200	700	900			
20		Customer C	400	900	1300			
21		Customer D	200	500	700			
22		Customer E	100	500	600			
23								
24								
25								
26								
27								

AUTO FILL

Suggested autofill

Ctrl+Enter to Autofill. Show formula

✓ ✕



Exploring Student-AI Interactions in Vibe Coding

1 Introduction

- Mentions the impact of GenAI on computing education.
- Discusses the emergence of vibe coding and its potential to alter programming practices.
- Highlights the study's focus on understanding how introductory and advanced CS students approach vibe coding.

2 Background and Literature Review

- Defines vibe coding and describes its characteristics based on Karpathy's definition.
- Summarizes existing research on vibe coding, primarily focusing on expert developers' experiences.
- Discusses how vibe coding differs from other AI-assisted programming workflows, such as agentic coding and first-generation GenAI programming.
- Reviews research on how experienced and novice programmers interact with AI programming tools.

3 Methods

Exploring Student-AI Interactions in Vibe Coding

Francis Geng
University of California San Diego
La Jolla, California, USA
fgeng@ucsd.edu

Anshul Shah
University of California San Diego
La Jolla, California, USA
ayshah@ucsd.edu

Haolin Li
University of California San Diego
La Jolla, California, USA
hal180@ucsd.edu

Nawab Mulla
University of California San Diego
La Jolla, California, USA
nmulla@ucsd.edu

Steven Swanson
University of California San Diego
La Jolla, California, USA
sjswanson@ucsd.edu

Gerald Soosai Raj
University of California San Diego
La Jolla, California, USA
asoosairaj@ucsd.edu

Daniel Zingaro
University of Toronto Mississauga
Toronto, Ontario, Canada
daniel.zingaro@utoronto.ca

Leo Porter
University of California San Diego
La Jolla, California, USA
leporter@ucsd.edu

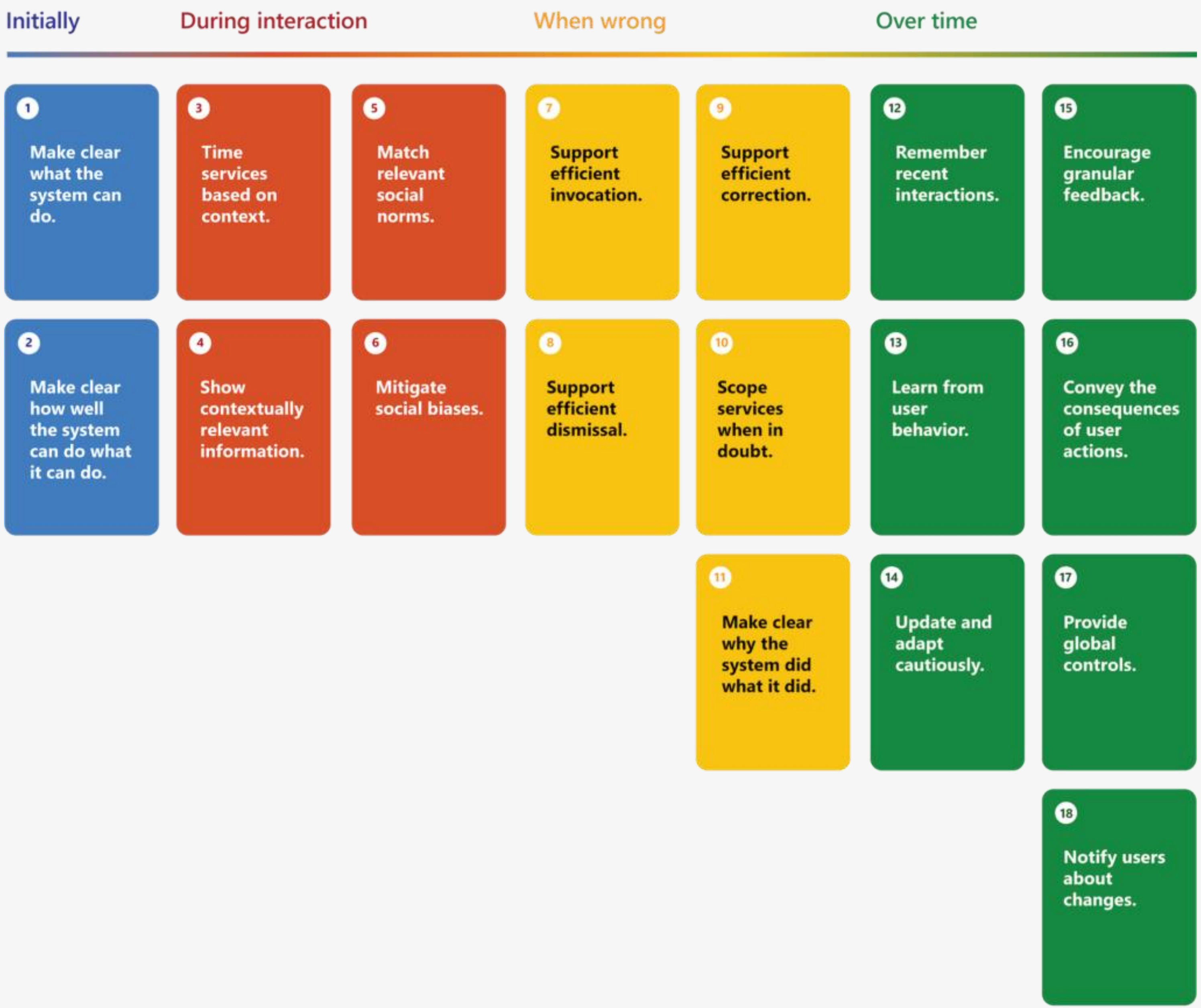
[cs.HC] 30 Jul 2025

Abstract

Background and Context. Chat-based and inline-coding-based GenAI has already had substantial impact on the CS Education community. The recent introduction of "vibe coding" may further transform how students program, as it introduces a new way for students to create software projects with minimal oversight.
Objectives. The purpose of this study is to understand how students in introductory programming and advanced software engineering classes interact with a vibe coding platform (Replit) when creating software and how the interactions differ by programming

1 Introduction

The advent of GenAI has had considerable impact on computing education with GenAI being capable of solving course assignments [5, 7], offering skilled AI tutors to students [6, 16, 20, 23], altering how we teach [15, 36], and changing the skills we need to teach [17, 26, 37]. LLMs, such as Copilot and ChatGPT, have changed the workflow of programming for professionals [2] as well as for students [26, 37]. As the CS education research community works to identify the impact of GenAI on student learning (with mixed findings [13, 21, 29, 36]), platforms such as Replit [30],



Initially

1 Make clear what the system can do.

2 Make clear how well the system can do what it can do.

During interaction

3 Time services based on context.

4 Show contextually relevant information.

5 Match relevant social norms.

6 Mitigate social biases.

When wrong

7 Support efficient invocation.

8 Support efficient dismissal.

9 Support efficient correction.

10 Scope services when in doubt.

11 Make clear why the system did what it did.

Over time

12 Remember recent interactions.

13 Learn from user behavior.

14 Update and adapt cautiously.

15 Encourage granular feedback.

16 Convey the consequences of user actions.

17 Provide global controls.

18 Notify users about changes.

- How do these apply to Google Sheets formula autofill / PDF document summary in Chrome?
- For each guideline:
  - Say if the guideline applies
  - If yes:
    - Does it follow the guideline? How?
    - If not, what can be done?
- Write roll number & names
- Turn in end of class

Summary…

- Last class → importance of human-AI interaction
- Today → designing human-AI interfaces
  - Guidelines from Microsoft
  - 18, broken down to:
    - Initial
    - During interaction
    - During errors
    - Long term use
  - Tool for evaluating an interface / designing an interface

Readings

- Google PAIR Guidebook
  - Read chapter 1: User needs + defining success
  - Read chapter 2: Mental models + expectations

- Guidelines for Human-AI Interaction, Amershi et al. 2019
  - https://www.microsoft.com/en-us/research/wp-content/uploads/2019/01/Guidelines-for-Human-AI-Interaction-camera-ready.pdf

Next class…