CSC 484 Lecture Notes Week 4, Part 1

Understanding and Conceptualizing Interaction

I. Relevant Reading

-- chapter 2 of the book.

II. **Intro to Ch 2 (Sec 2.1).**

A. On behalf of software engineers, ***I object!***

B. Intro sets up would-be dichotomy.

1. They pose the questions

Intro to Ch 2, cont'd

"In designing a new application, would you start by coding? Or, would you start by talking to users and seeing what else is out there?"

Intro to Ch 2, cont'd

2. Their answer

   "Interaction designers would do the latter."

   begs the question

   "Who wouldn't?"

Intro to Ch 2, cont'd

3. Well-trained SE answers the same as IDer.

4. I.e., start by talking to users and seeing what else is out there.

Intro to Ch 2, cont'd

C. A largely false dichotomy between

1. clueless software engineers, versus

2. inspired interaction designers

Intro to Ch 2, cont'd

D. SEs really aren't that clueles.

1. 30+ years of research.
2. Addresses many issues in this chapter.
3. Has many good ideas.

Intro to Ch 2, cont'd

E. For Agile development, Pg 44 is antithetical:

"... Once ideas are committed to code, they become much harder to throw away".

Intro to Ch 2, cont'd

1. Agile developers say precisely the opposite.

2. Traditional SE has *throw-away* prototypes.

Intro to Ch 2, cont'd

F. Chapter does provide some useful info.

1. ***Hits:***
   a. Provoking thought.
   b. Importance of understanding problems.
   c. Analysis of the *interaction types*.
   d. The interview with Terry Winograd.

Intro to Ch 2, cont'd

i. Winograd plenty talks about

people, *products*, and *examples*.

ii. He never mentions

conceptual models, *metaphors*, *analogies*.

Intro to Ch 2, cont'd

2. ***Misses:***

    a. Misunderstanding of SE.

    b. Maltreatment of conceptual modeling.

III. **Understanding problem space (Sec 2.2).**

A. What you should take away:

1. Importance of having a problem to solve.

2. The notion of identifying and challenging your design assumptions.

Understanding problem space, cont'd

B. The engineer's common refrain --

"What's the problem here?"

Understanding problem space, cont'd

1. Ask whether you're improving existing product, or coming up with brand new idea.

2. Book provides these questions:

Understanding problem space, cont'd

a. What problems are you trying to solve?

b. Why do these problems exist?

c. How is your design going to solve them?

Understanding problem space, cont'd

d. If no specific problems[1], how do your ideas make things better?

---

[1] The engineer would say, *"If you don't have a specific problem, fuhgeddaboudit."*

Understanding problem space, cont'd

C. As part of your work on Assignment 2,

1. Answer each of the questions above.

2. Identify your assumptions, how you're going to validate them.

IV. **Conceptualizing design space (Sec 2.3).**

A. What are we trying to do?

1. solving a particular problem
2. building a product with mass appeal
3. inventing some new form of interaction

Conceptualizing design space, cont'd

B. If *solving a particular problem*, then

conceptual is model based on
understanding concrete user problems.

Conceptualizing design space, cont'd

C. If *building mass-appeal product*, then

conceptual model based on
generalized understanding of potential users

Conceptualizing design space, cont'd

D. If *inventing new form of interaction*, then

conceptual model is
whatever a design team can dream of.

V. Conceptual models, and examples (Sec 2.3.1, 2.3.2).

A. Liddle quote is out of date:

"The most important thing to design is the user's conceptual model. ... That is almost exactly the opposite of how most software is designed."

Conceptual models, examples, cont'd

B. In post-1996 SE, conceptual models abound.

1. In "Vision and Scope" document.

2. In requirements, specs, program design.

Conceptual models, examples, cont'd

C. Major question book does not address:

Is the conceptual model a specific, concrete artifact of the ID process, or is it embodied in other process artifact(s)?

Conceptual models, examples, cont'd

1. If the latter, what does it look like?
2. In what language or notation?
3. Book says "*lingua franca* of design team".

Conceptual models, examples, cont'd

D. "Lingua franca" expressed in
***the storyboards and scenarios*.**

Conceptual models, examples, cont'd

1. Per affordance, concrete UI should ***embody and convey*** conceptual design.

2. To end users, as well as design team.

Conceptual models, examples, cont'd

3. If concrete UI is not so affordant,
    a. it's a bad interface, or
    b. it's a bad concept

Conceptual models, examples, cont'd

4. Either way, interface itself represents the underlying conceptual model.

Conceptual models, examples, cont'd

E. Direct evidence for this in Sec 2.3.2.

1. What is "lingua franca" they choose for "best practice" examples?

2. It's pictures of concrete user interfaces!

Conceptual models, examples, cont'd

F. So -- a well-designed UI ***fully affords***, and hence defines the conceptual design.

Conceptual models, examples, cont'd

1. *Metaphors* and *analogies* should be readily apparent in UI layout.

   a. E.g., electronic spreadsheet looks like paper ledger.

   b. Xerox Star UI looks like a desktop.

Conceptual models, examples, cont'd

i. Screen elements are familiar items

ii. Desk accessories easily recognizable.

Conceptual models, examples, cont'd

2. Conceptual model *lexicon* comprised of words used skillfully in UI display.

Conceptual models, examples, cont'd

G. Other aspects of conceptual model conveyed effectively by operational prototype.

1. Xerox Star drag-and-drop concept.

2. Direct manipulation behavior in Star UI.

Conceptual models, examples, cont'd

H. Preceding discussion does *NOT* mean that early versions of UI look like final product.

1. Designers not tied to specific UI "widgets".

2. Don't get stuck in conventional UIs.

Conceptual models, examples, cont'd

3. Conceptual model progresses from

storyboard sketches, to

illustrated scenarios, to

interface prototypes.

Conceptual models, examples, cont'd

I. A thought experiment --

How does Steven Spielberg present his conceptual model for a movie?

J. What about Frank Lloyd Wright's conceptual model of a building?

VI. **Comments on Johnson and Henderson.**

A. IMO, they profoundly misunderstand conceptual modeling

1. the contributions of SE and AI

2. how concrete examples *define and convey* via affordance

Johnson and Henderson, cont'd

B. Regarding first misunderstanding,

"... our experience with our clients indicates that conceptual models of this sort are almost completely unknown outside of the HCI ..."

Johnson and Henderson, cont'd

1. Their experience is bogus.

2. Models "of this sort" have been subject of SE research well over 30 years.

Johnson and Henderson, cont'd

3. SE notations may be stodgy but J&H offer no constructive alternative.

4. Software engineers have plenty to offer.

Johnson and Henderson, cont'd

C. On the second misunderstanding,

1. A key "concept" they site is whether to represent data as a flat list or hierarchy.

2. Should be immediately obvious looking UI.

Johnson and Henderson, cont'd

D. J&H say

"How the system *presents itself* to users"

is does not convey a conceptual design.

1. But this leaves out the users.

2. If storyboard does't convey conceptual design, then say what specifically does.

3. They don't.

VII. **Interface metaphors and analogies.**

A. They’re fine, but,

1. Choose ones that are understandable and compelling to users.

2. And don’t over do it.

Interface metaphors and analogies, cont'd

B. In the book, the part on "opposition to using metaphors" is longer than the other parts

They got that right.

VIII. **Interaction types (Sec 2.3.4).**

A. Some useful info.

B. They present four specific types:

Interaction types, cont'd

1. ***Instructing*** **--** users instruct, i.e, command the system to do things.

2. **Conversing --** users have a two-way dialog with the system.

Interaction types, cont'd

3. ***Manipulating*** **--** users open, close, move, edit data provided by the system.

4. ***Exploring*** **--** users move through a large space or virtual environment.

Interaction types, cont'd

C. Types are definitely not mutually exclusive.

1. Can often provide more than one.

2. Allow users to seamlessly progress.

IX. Instructing Interfaces (Pages 65-67)

A. Iconic UIs "easier" than command-language.

B. Good reasons to use a command-language:

1. Number of instructions too large for icons.

2. The interface needs to be scriptable.

Instructing Interfaces, cont'd

C. Questionable reasons for command lang:

1. Easier to implement, e.g.,

   a. a small vending machine keypad,

   b. a program with a simple text UI vs GUI

Questionable command language UIs, cont'd

2. It's easer to maintain, e.g.,

   a. easier to re-map vending machine codes

   b. easier to maintain non-GUI program, when deployed on multiple platforms.

Questionable command language UIs, cont'd

D. Here "questionable" means UI is not optimal, but there may be other trade-offs involved.

X. Conversing Interface (Pages 67-70).

A. Involve a two-way conversation,
assume therefore that system is "smart".

Conversing Interface, cont'd

B. Good reasons to use:

1. User has little or no knowledge of available commands.

2. System has enough data and intelligence.

3. Intelligent agents are both good and bad examples.

Conversing Interfaces, cont'd

C. Bad reasons to use:

1. It's cheap, e.g., cheaper than a person or AI system to give an answer.

2. It looks cute and *seems* intelligent.

XI. Manipulating Interfaces (Pages 70-75).

A. Involve manipulating "real-world" representations of objects and operations. E.g.,

1. dragging a file icon into a trash can

2. commanding a robotic device with joystick

Manipulating Interfaces, cont'd

B. *Direct manipulation* refers to well-known form of window-based computer UIs.

Manipulating Interfaces, cont'd

C. Good reasons to use a direct manipulation

1. Action can more efficiently or effectively be performed, e.g., drawing.

2. Can be easier to learn.

Manipulating Interfaces, cont'd

D. Reasons not to use a direct manipulation UI.

1. It takes (substantially) longer than a simple command, e.g., "change-all" command.

2. A sufficiently expert user community can be more productive with command-language.

XII. **Exploring Interfaces (Pages 75-83).**

A. Virtual environments, or
physical context-aware environments.

B. Relatively new forms of interaction.

Exploring Interfaces, cont'd

C. Have been effective in

- games
- architectural exploration
- larger-scale geographic exploration

D. Yet to take off in other areas,
e.g., "smart homes".

XIII. **Theories, models, frameworks (Sec 2.4).**

A. A ***theory*** is

- high level explanation of human-computer interaction,
- based on theories of human behavior and cognition.

Theories, models, frameworks, cont'd

B. A ***model*** is

- abstraction of human-computer interaction,
- typically of a specific aspect,
- designed to provide basis for design and evaluation.

Theories, models, frameworks, cont'd

C. A ***framework*** is

- prescriptive set of principles and organizational guidelines
- designed to provide broader view of how to approach design and evaluation.

Theories, models, frameworks, cont'd

D. Subsequent book chapters cover these subjects in further detail.