HCI Theories

- Given enough study and data, HCI researchers seek to form tested (or testable), reliable, and general *theories* about HCI
- Unlike guidelines and principles, which are primarily *instructive* — they tell you what to do or how to do it — theories perform other functions:
  - Describe, explain, or predict
  - Model (the verb, not the noun) or classify
  - Involve different levels or aspects, e.g. motor-task vs. perceptual vs. cognitive activity

Theories in a "Young" Field

HCI is a young field, and thus its theoretical foundations do not exhibit the two characteristics of more mature disciplines:

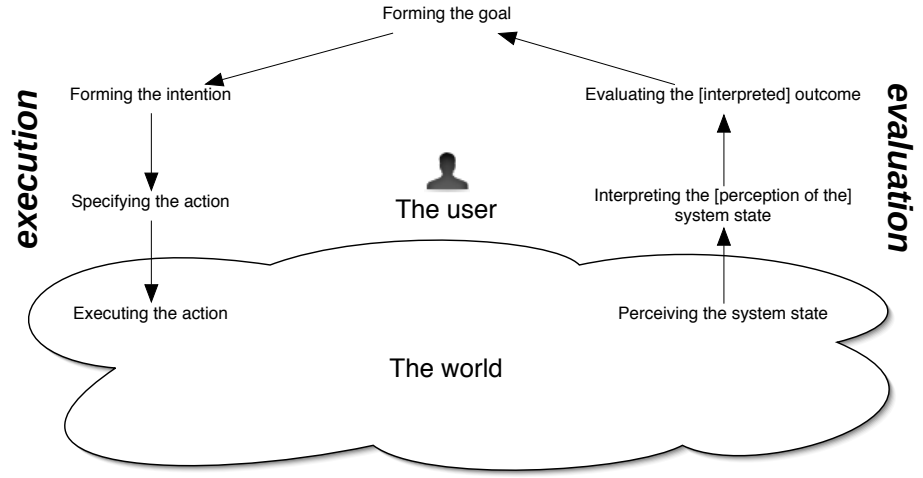
1. *Theories are central to research and practice* — today in HCI, research and practice are frequently still based on intuition and trial-and-error
2. *Theories lead rather than lag behind application* — the latest in HCI are frequently found in commercial products and not in universities or labs

Levels of Analysis Theories

- Early HCI theories take a "level" approach similar to software architecture, programming languages, or networking; for example:
  1. Conceptual level
  2. Semantic level
  3. Syntactic level
  4. Lexical level
- Syntax and lexicons have decreased (or been transformed?) in today's GUIs, thus requiring either new levels or new ways of analysis

Stages-of-Action Theories

Stages-of-action theories seek to predict or model a state machine of sorts for user activity (Norman 1988):



From Theory to Principles

- Norman's theory provides the basis for his own set of design principles
- Failures in the stages are characterized either as *gulfs of execution* or *gulfs of evaluation*
- Good design bridges these gulfs; thus the principles of:
  1. Visibility
  2. A good conceptual model
  3. Good mappings
  4. Feedback

Other Stages-of-Action Analyses

- Action as exploration — what if the user doesn't have a particular goal yet, but just wants to "see what's out there?" (Polson and Lewis, 1990)
- Stages of information seeking (Marchionini 1995):

Recognize and accept information problem

Define and understand problem

Choose search system

Formulate query

Execute search

Examine results

Make relevance judgments

Extract information

Reflect/iterate/stop

GOMS/Keystroke-Level Theories

Successor to levels-of-analysis theory, introduced by CMU theorists (Card, Moran, and Newell 1980; 1983)

1. Goals: what does the user wish to accomplish?
2. Operators: elementary "acts" that the user can perform
3. Methods: specific sequences of operators that accomplish the goals
4. Selection Rules: criteria for choosing among methods that accomplish the same goal

- Card, Moran, and Newell also proposed the keystroke-level model, which maps operators (keystrokes, pointing, thinking, waiting, etc.) into idealized times and sums them up to predict error-free expert performance…but note how this does not cover learning, memorability, nor satisfaction
- Many variations and extensions to the original GOMS (now called CMN-GOMS) over the years
  - Kieras 1988, Anderson and Lebiere 1998, Pew and Gluck 2004
  - Summaries/reviews by John and Kieras 1996; Baumeister, John, and Byrne 2000

Consistency-Through-Grammar Theories

- Premise: user interface consistency can be attained by applying formal grammar concepts
- Appealing for the strong theoretical foundation behind formal grammars
- First proposed by Reisner (1981) — she defined two *action grammars* for a GUI, and showed that the simpler grammar was easier to learn
- Payne and Green (1986) extended the grammar to multiple levels (*task-action grammars* or TAGs), adding notions of completeness and grammar checking

	Direction	Unit
move-cursor-one-char-forward	forward	char
move-cursor-one-char-backward	backward	char
move-cursor-one-word-forward	forward	word
move-cursor-one-word-backward	backward	word

task [Direction, Unit] → symbol [Direction] + modifier [Unit]
symbol [Direction = forward] → *right arrow key*
symbol [Direction = backward] → *left arrow key*
modifier [Unit = char] → *none*
modifier [Unit = word] → *ctrl*

	Action
move-cursor-one-char-forward	*right arrow key*
move-cursor-one-char-backward	*left arrow key*
move-cursor-one-word-forward	*ctrl + right arrow key*
move-cursor-one-word-backward	*ctrl + left arrow key*

Widget-Level Theories

- As GUI frameworks have evolved, decomposition into raw mouse clicks and keystrokes tends to become too tedious — why not go up a level instead, to the "widgets" that comprise the user interface?
- Sears (1992) mapped tasks to components such as scrolling lists, buttons, etc. then associated metrics with these components; in addition, spatial relationships among widgets (adjacency, sequence) were added to the overall measurement/prediction
- Groupings of widgets and common composite tasks have led to *HCI patterns* — akin to object-oriented design patterns — serving as another layer for prediction and analysis (Van Duyne, Landay, and Hong 2002)

Context-of-Use Theories

- Premise: users ≠ lab rats
- Knowledge does not just exist within the user and the system, but also in the environment (Suchman 1987) — for example, many students would rather work in study groups than alone in their rooms
- Bruce Tognazzini's GE dishwasher anecdote
- Increasing in significance as computing devices become more embedded, pervasive, and ubiquitous (yes, this is a budding term — *ubiquitous computing*)

The Object-Action Interface (OAI) Model

Decomposes the "universe" into *objects* and *actions*, situated either in the *task* aspect or *interface* aspect

- *Task objects* = the real world
- *Task actions* = user intentions, manipulations, and activities on task objects
- *Interface objects* = the "system image," often using a *metaphor* that tries to correspond to the real world
- *Interface actions* = user plans on how to interact with interface objects in order to accomplish the desired task actions

- Users acquire task knowledge (both objects and actions) through experiences in the real world (education, on-the-job training, daily life)
  - Developers who wish to create applications for a task should thus also acquire this knowledge at a real-world level
- Interaction design thus becomes a *mapping* activity — developers try to create an interface that corresponds to the task objects and actions
  - Users who are trying to learn an application thus try to make the connection between interface objects and actions and the task objects and actions that they already know

OAI in Practice

- Developers/designers should become as proficient (or more so) as their users in the task
- Use hierarchies to decompose objects and actions from high-level or composite concepts to their primitive or atomic elements
- The universe of possible interface objects and actions has evolved into two broad categories:
  - "Established" interface objects and actions, as promulgated by widely used GUIs and metaphors
  - "Domain-specific" interface objects and actions — more specialized, but map better to task objects and actions

- "Computer literacy" in a GUI world can now be defined with a little more precision — it is a user's familiarity with a range of interface objects and actions
  - Buttons, check boxes, radio buttons, scrolling lists, drop-down menus, window controls, etc.
  - Single-clicking, double-clicking, dragging, selecting
- The remaining burden is on the developer/designer to ensure that customized interface objects and actions — e.g. specialized to a particular task domain — provide sensible mappings to the real world
- The evolution of syntax: whereas user interfaces used to be one-dimensional — "type this" — the range of actions today is much broader and more context sensitive — "click this, then drag it there"