Conceptual Models:

Core to the Design of Interactive Applications

Austin Henderson
UCB/Design MDes 2025
Oct 8, 2025

Conceptual Models:

Core to the Design of Interactive Applications

Reflecting collaboration:

Austin Henderson
Rivendel Consulting

Jeff Johnson
UI WIZARDS, INC.
PRODUCT USABILITY CONSULTING

Agenda – Conceptual Models

- History
- Framework and terminology
- Place in tool life-cycle
- Essential conceptual modeling
- Example: bank account application
- Representing CMs
- Conceptual scenarios
- Enhanced conceptual modeling
- Benefits of having a clear CM
- Selected references
- Q & A

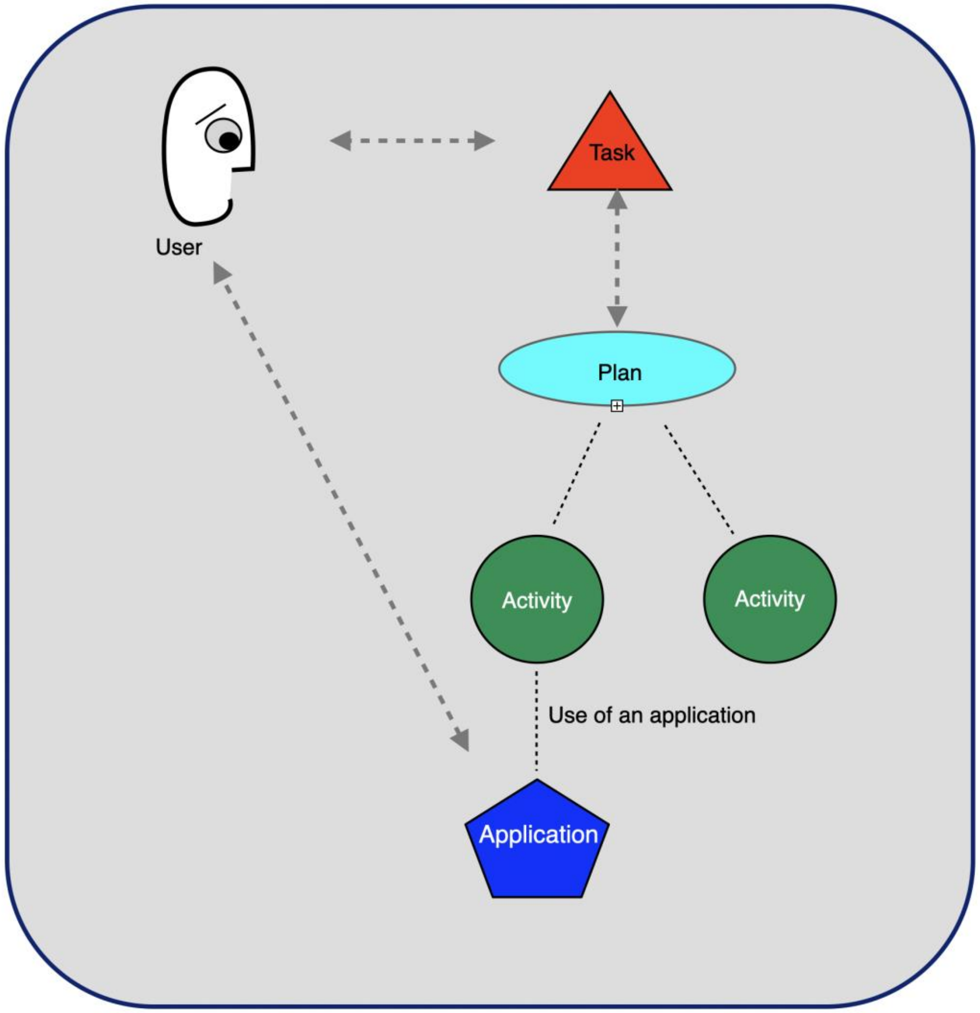
History

- Exposed to CMs @ Xerox (PARC & Star/VP) 1970's
- Article: "Conceptual Models: Begin by Designing What to Design",
  interactions, Jan-Feb, 2002
- Mentioned in Jeff's earlier books:
  - *GUI Bloopers* (2000) & *GUI Bloopers 2.0* (2007)
  - *Designing with the Mind in Mind* (2011)
- Tutorial (CHI): "Designing What to Design:
  a Task-Focused Conceptual Model" 2009…2019
- Book: *Conceptual Models: Core to Good Design*, Morgan & Claypool (2011)
- Book: *Conceptual Models: Core to the Design of Interactive Applications*, Springer/Nature (2024)

Conceptual Models
Jeff Johnson · Austin Henderson
Conceptual Models
Core to the Design of Interactive Applications
Second Edition

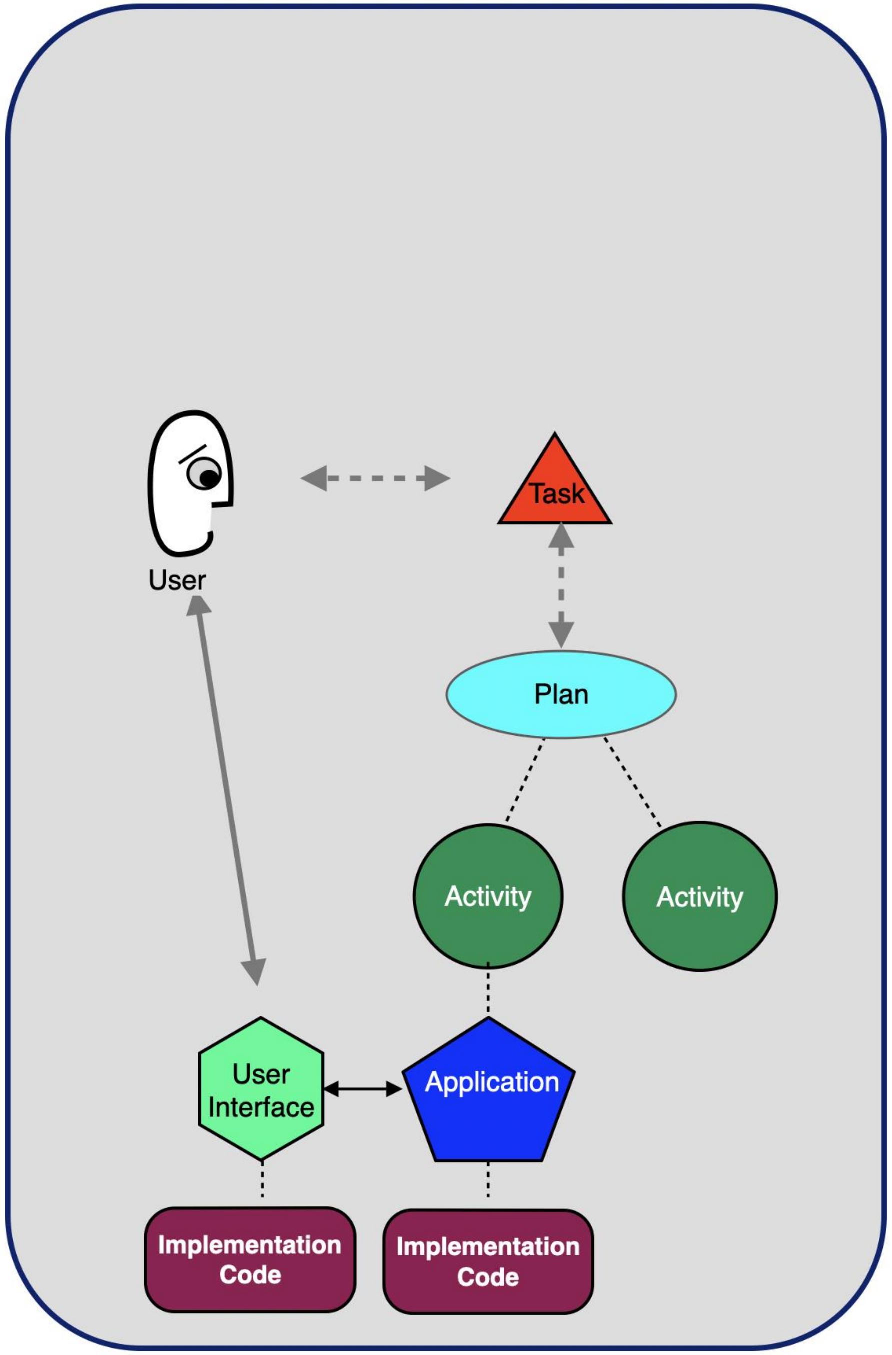
Framework:

- People and Domains



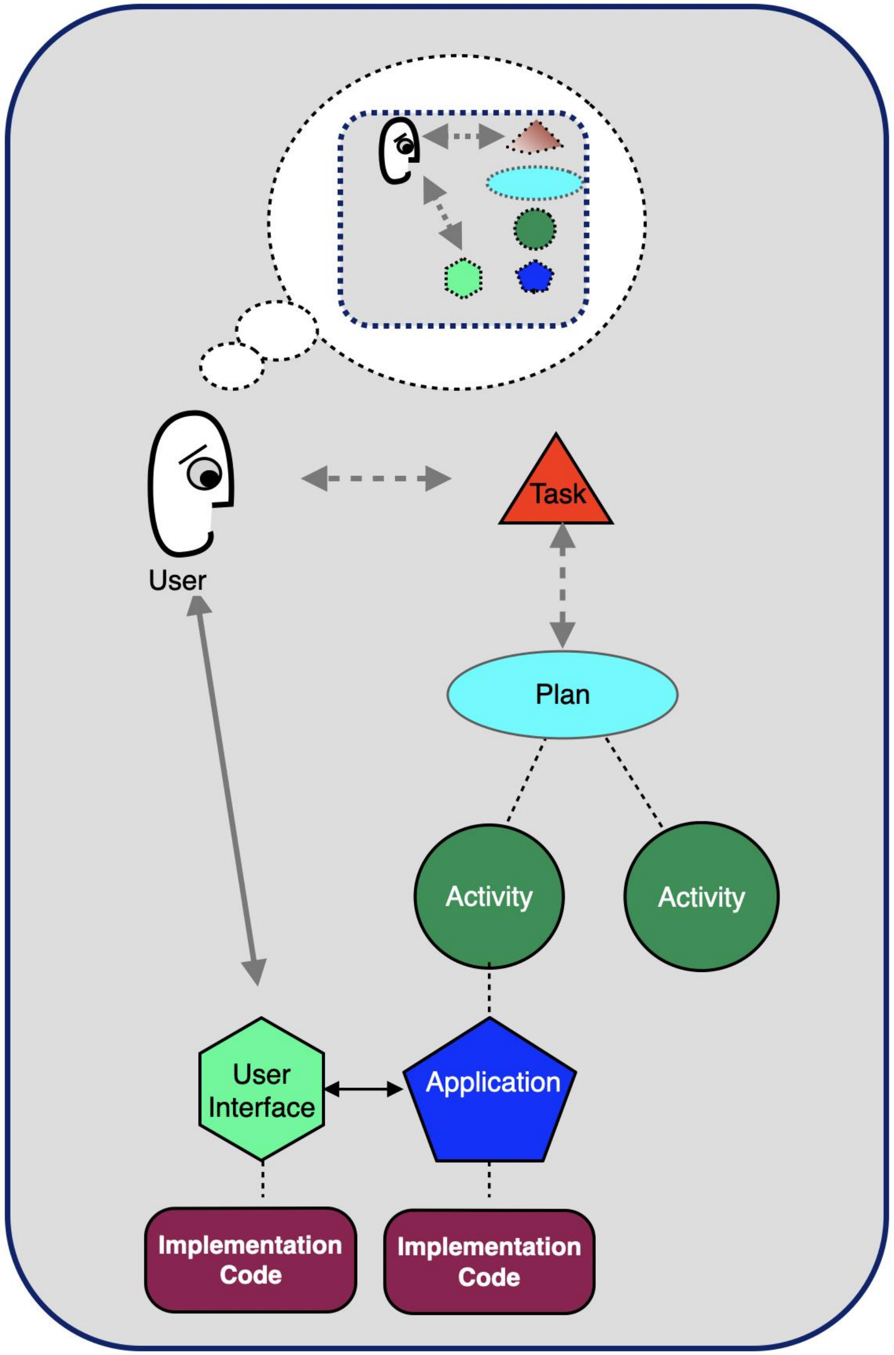
Framework:

- User Interface Implementation



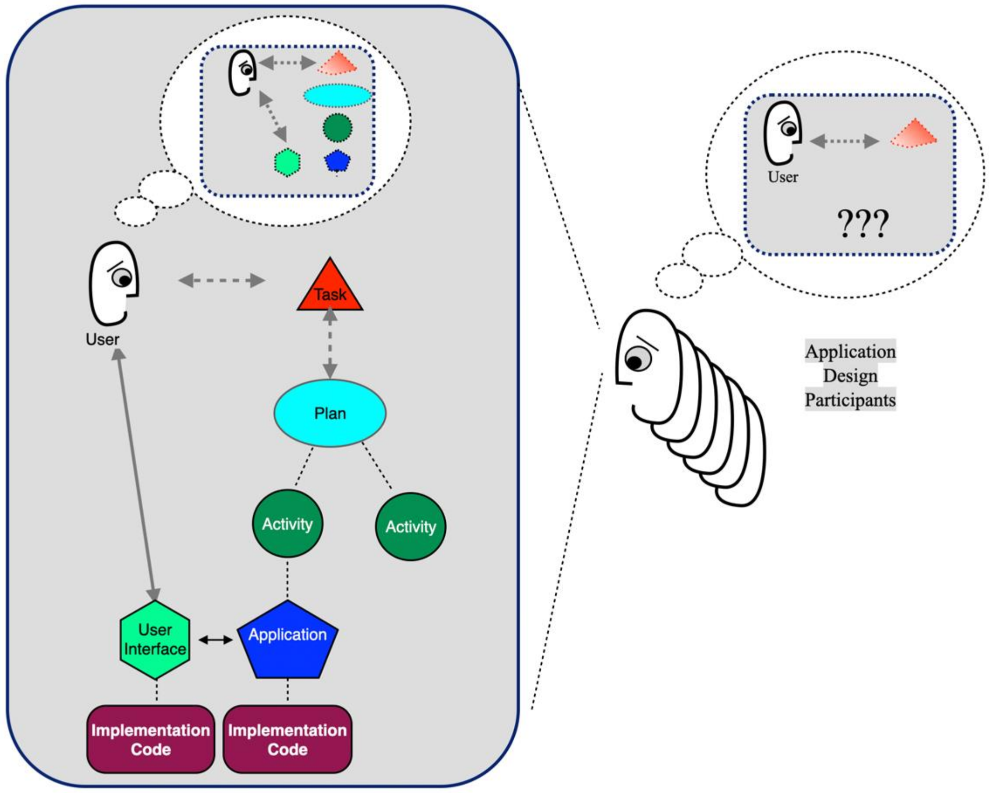
Framework:

- User's mental model



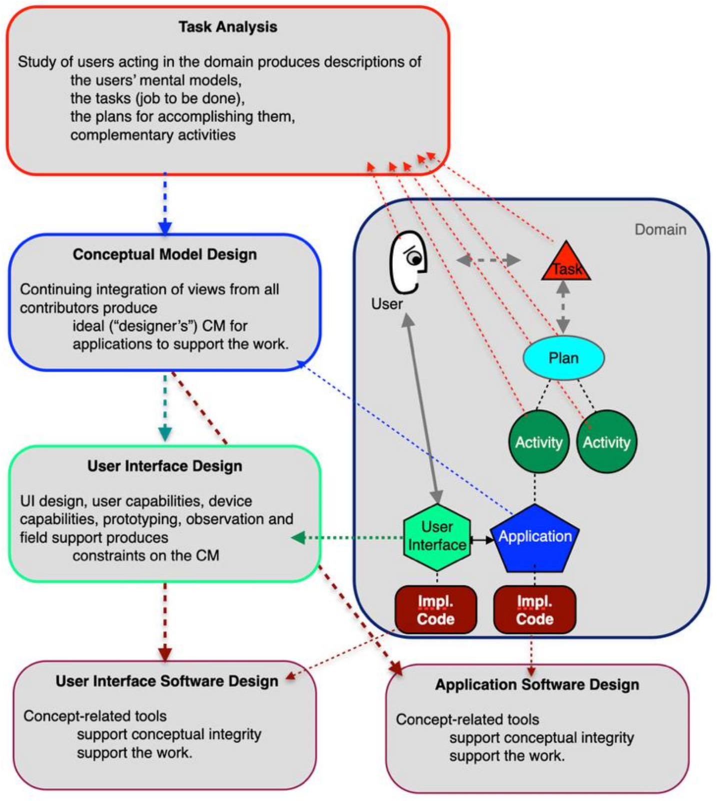
Framework:

- Designer's Mental Model



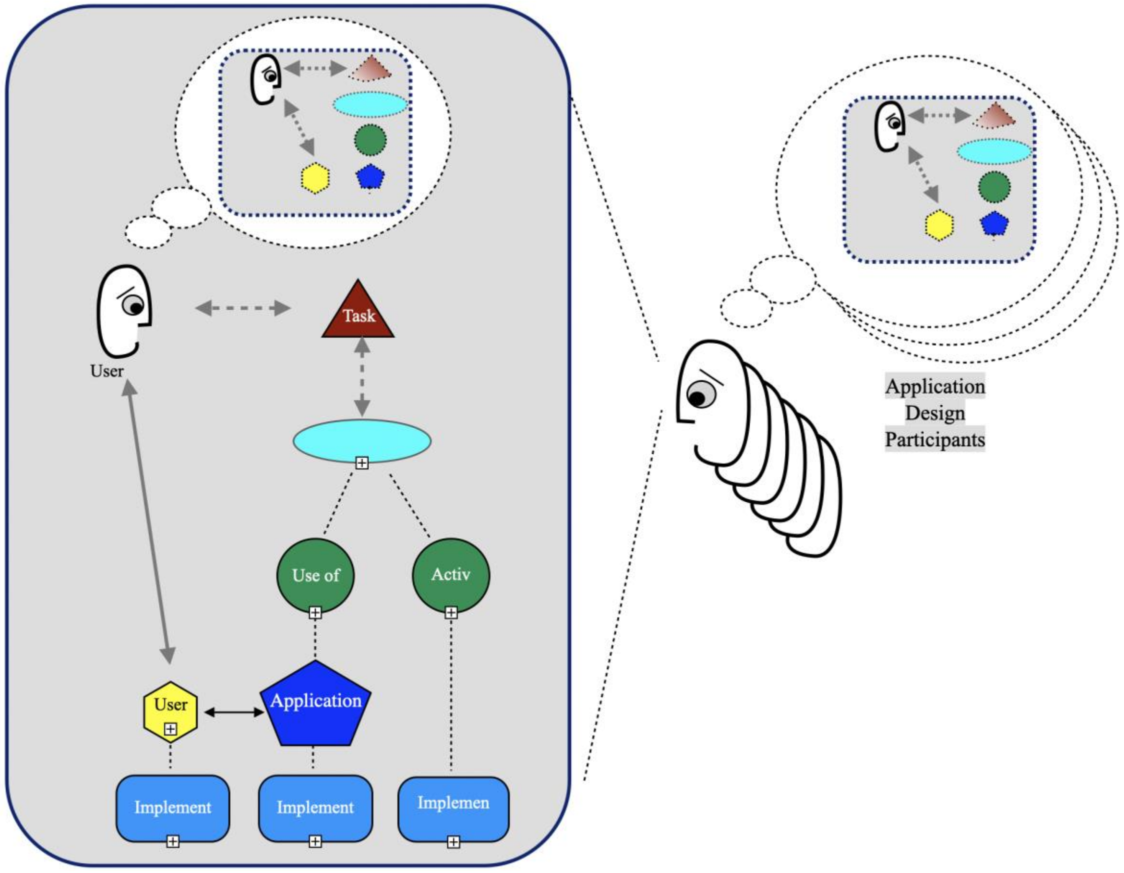
Framework:

- Designs and their creation
  - task analysis
  - CM design
  - Designs of UI(s)
  - software designs


Task Analysis
Study of users acting in the domain produces descriptions of the users' mental models, the tasks (job to be done), the plans for accomplishing them, complementary activities
Conceptual Model Design
Continuing integration of views from all contributors produce ideal ("designer's") CM for applications to support the work.
User Interface Design
UI design, user capabilities, device capabilities, prototyping, observation and field support produces constraints on the CM
User Interface Software Design
Concept-related tools support conceptual integrity support the work.
Application Software Design
Concept-related tools support conceptual integrity support the work.
Domain
User
Task
Plan
Activity
Activity
User Interface
Application
Impl. Code
Impl. Code

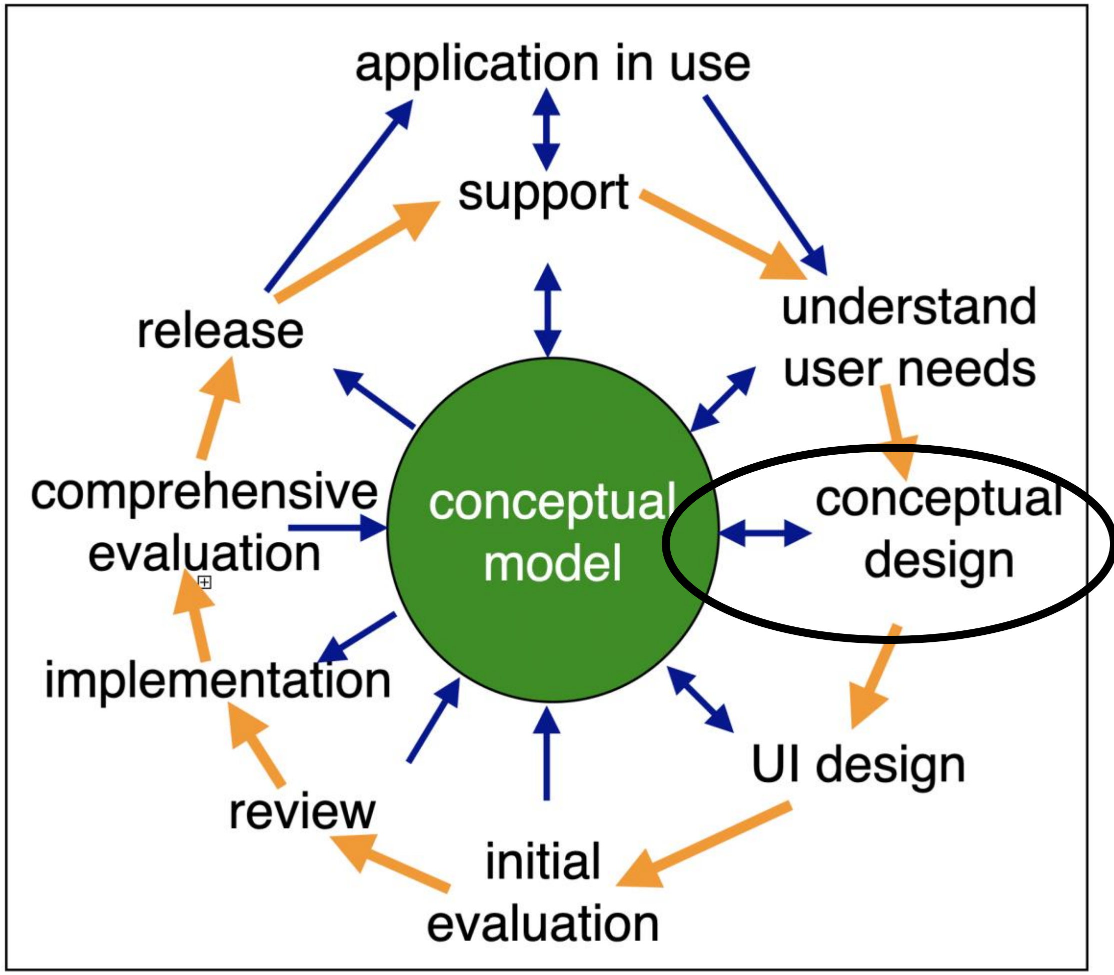
Framework:

- Designer's Model: *target* Users' Mental Models



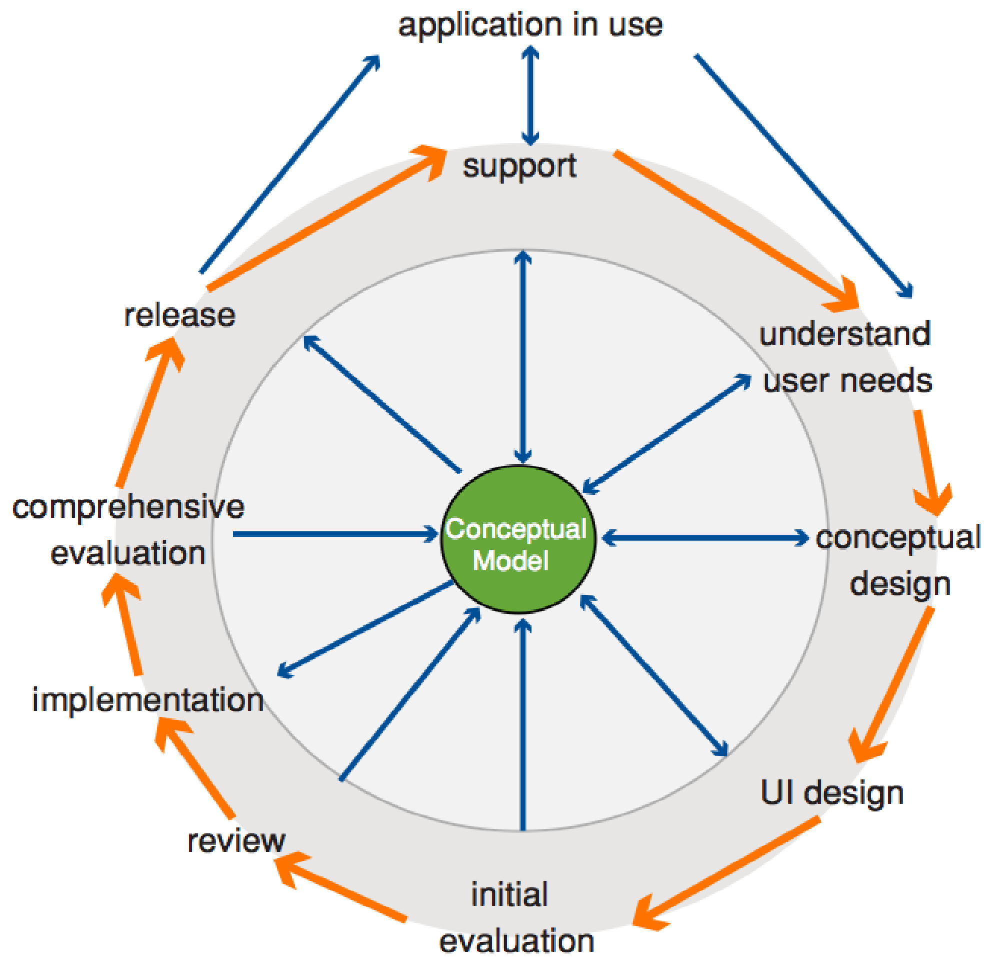
Development life-cycle

- Functionality
- Scenarios
- UI (interaction)
- Implementation
- Documentation
- Support
- Design CM early & often!



Focuses Design and Development Process

- Early step
- Seat at the table
- Coordinate
- Test
- Iterate*



Essential Conceptual Design

- Application has concepts & presentation
- Concepts affect *architecture*
- Consider application's ***concepts*** first; ***presentation*** later
  - First design what the app ***is***
  - before designing how it *looks* or how users *operate* it
- Don't jump right into *sketching* the GUI

Essential Conceptual Design

- Conceptual model:
  - Organization & structure of app’s concepts
  - *Intended* user model
- Goals for conceptual model:
  - Simple: “Less is more!”
  - Focus on task domain

Perform Object/Operation Analysis

- **Objects**: *User-exposed* concepts
- **Attributes** of each object-type
- **Operations** on each object-type
- **Relationships** between objects
  - What is a *subtype* of what?
  - What is a *part* of what?
  - What can *contain* what?

Example: Managing Bank Account

Objects	Attributes	Operations
Customer	name, address, phone number, age	register, unregister, add account, view/edit (attributes)
Account	owner (*customer*), balance, currency, interest rate, date opened, type (*checking, savings, investment, …*), tax status	deposit funds, withdraw funds, open, close, view, reconcile, view/edit (attributes)
Debit Card	owner, accounts (list of Accounts), status, PIN	change PIN, view/edit (attributes)
Transaction	amount, date, description	create, delete, view
Check	number, amount, date, memo, status, from-account	deposit, write, view

Example: Managing Bank Account

Objects:

- **Yes**: customer, account, debit card
- **No**: database, dialog box, mode, string
- **Maybe**: template, command sequence

Example: Managing Bank Account

Attributes:

- **Yes**: name, address, phone number, age, owner (customer), balance, currency, interest rate, date opened, type (checking, savings, investment, …), tax status
- **No**: *transaction* -- memory-size, export encoding

Example: Managing Bank Account

Operations:

- **Yes**: register, unregister, deposit funds
- **No**: click button, load database, edit table row, create record, flush buffer
- **Maybe**: create *template*, save *command sequence*

Example: Managing Bank Account

Relationships:

- **Yes**: customer *owns* account,
  savings account is a *type of* account,
  transaction *changes* account-amount,
  transactions *can occur by* check
- **No**: button click is a type of deposit
- **Maybe**: deposit is a *type of* a transaction

Representing CMs

- Many existing forms; Pick one!
- Suited for context
- Agreed-upon by all
- For clarity
- For efficiency
  - manipulation
  - comparison
  - conveying information

Representation:
Text (story telling)

The clock stores the current time of day, continually updating it to track the passage of time.

It displays the current time constantly.

Users can set the current time.

Users can set an alarm at a specified time, or no alarm.

When an alarm is set and the current time equals the set alarm time, the alarm is triggered.

Representation:
Text (story telling)

The clock stores the **current time** of day,
continually updating it to track the passage of time.

It displays the current time constantly.

Users can set the current time.

Users can set an **alarm** at a **specified time**, or no alarm.

When an alarm is set and the current time equals the set **alarm time**,
the **alarm** is triggered.

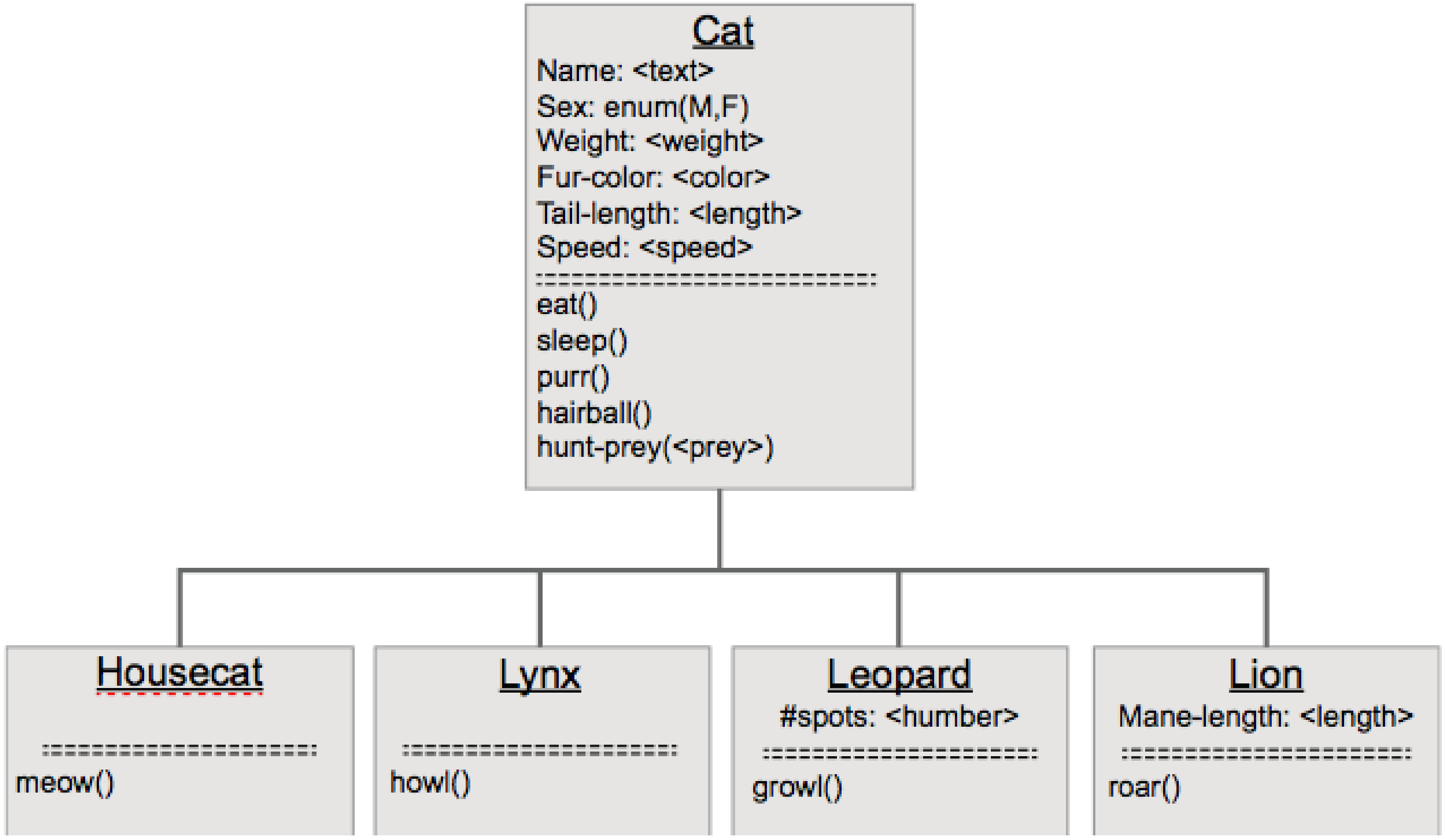
Users can turn off an alarm.

Representation : Tables

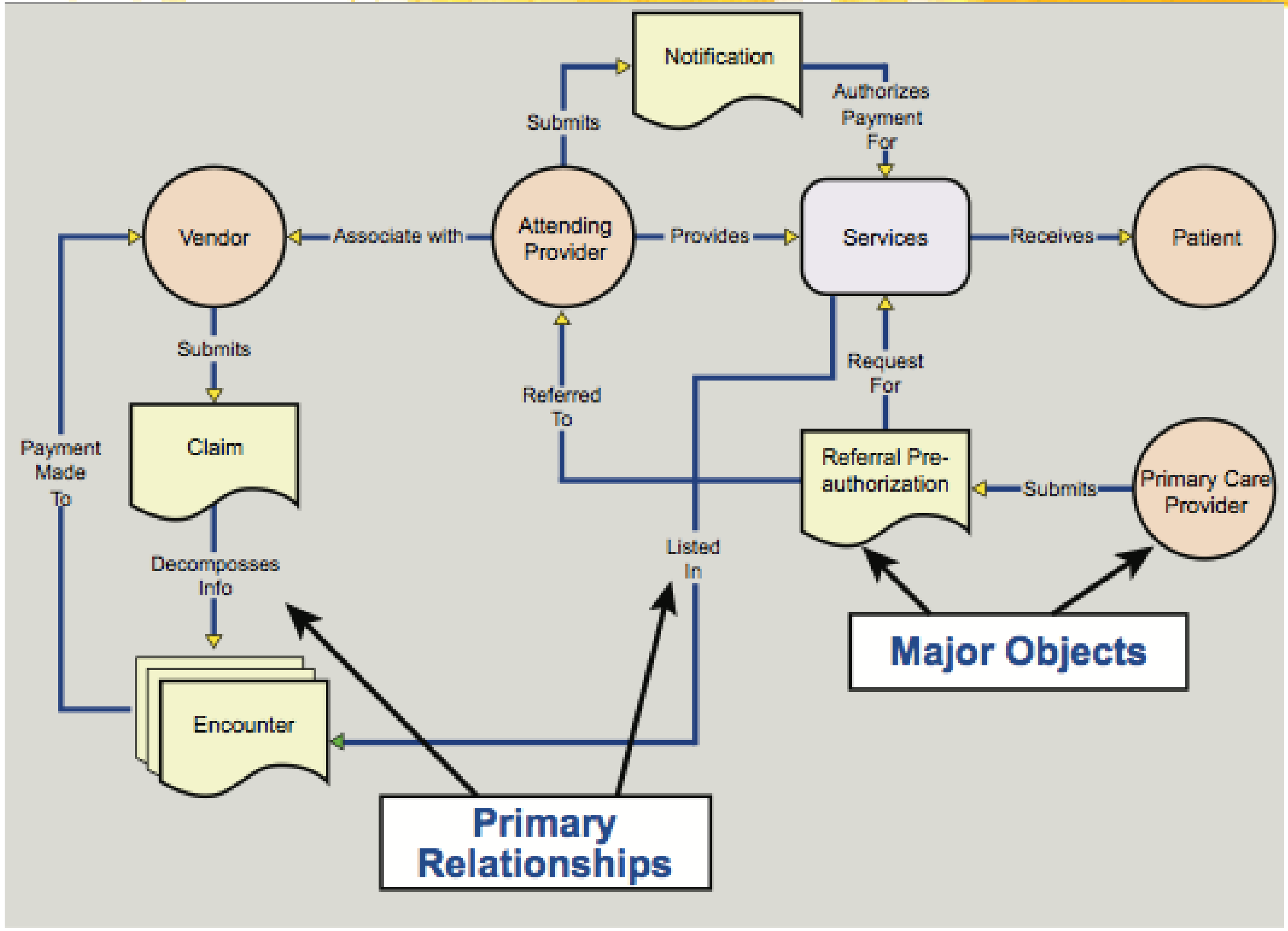
- Calendar CM

Objects	Attributes	Operations
Calendar	owner, current focus	examine, print, create, add event, delete event
Event	name, description, date, time, duration, location, repeat, type (e.g., meeting)	examine, print, edit (attributes)
To-Do Item	name, description, deadline, priority, status	view, print, edit (attributes)
Person	name, job-description, office, phone	send email, view details

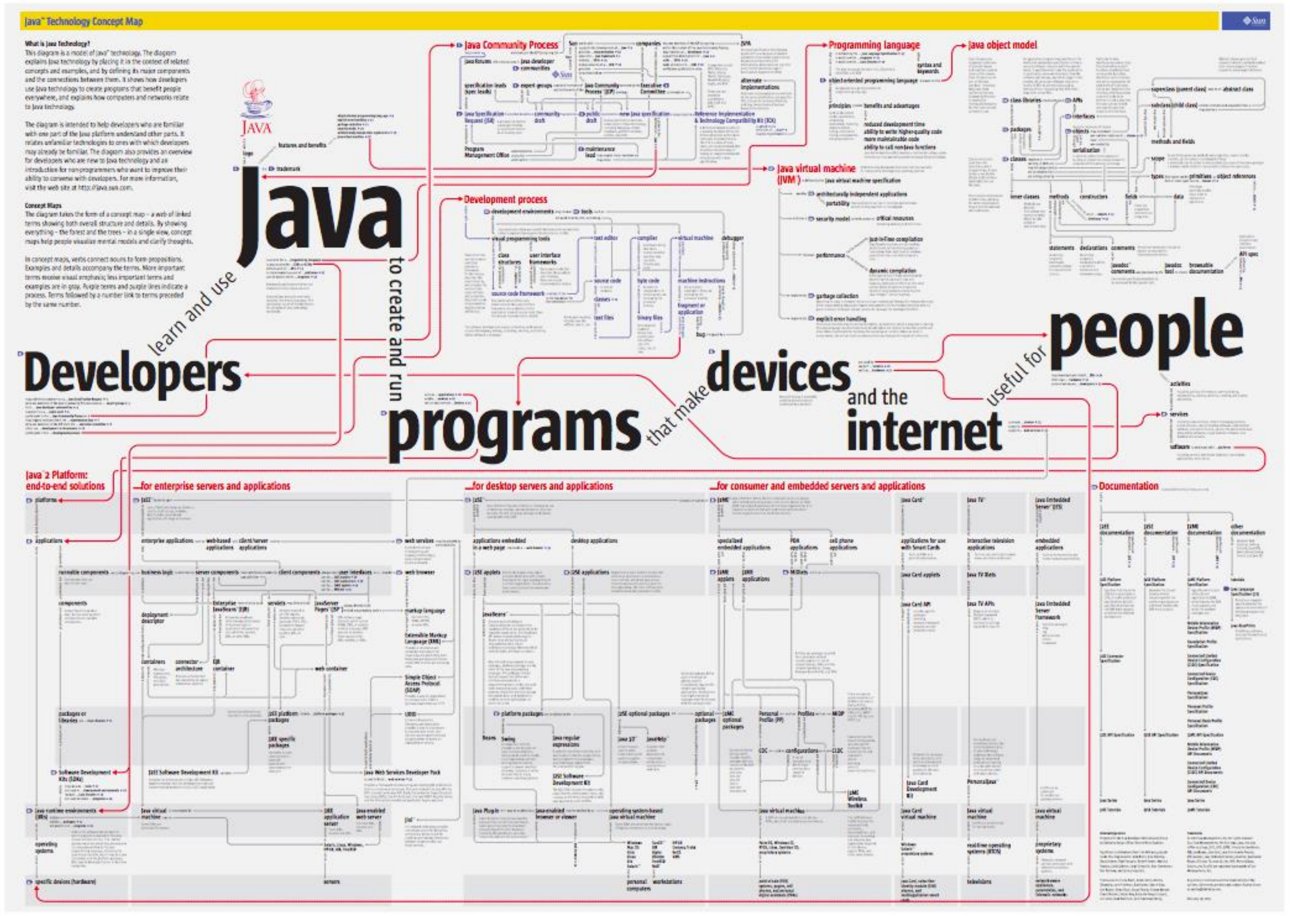
Representation : UML



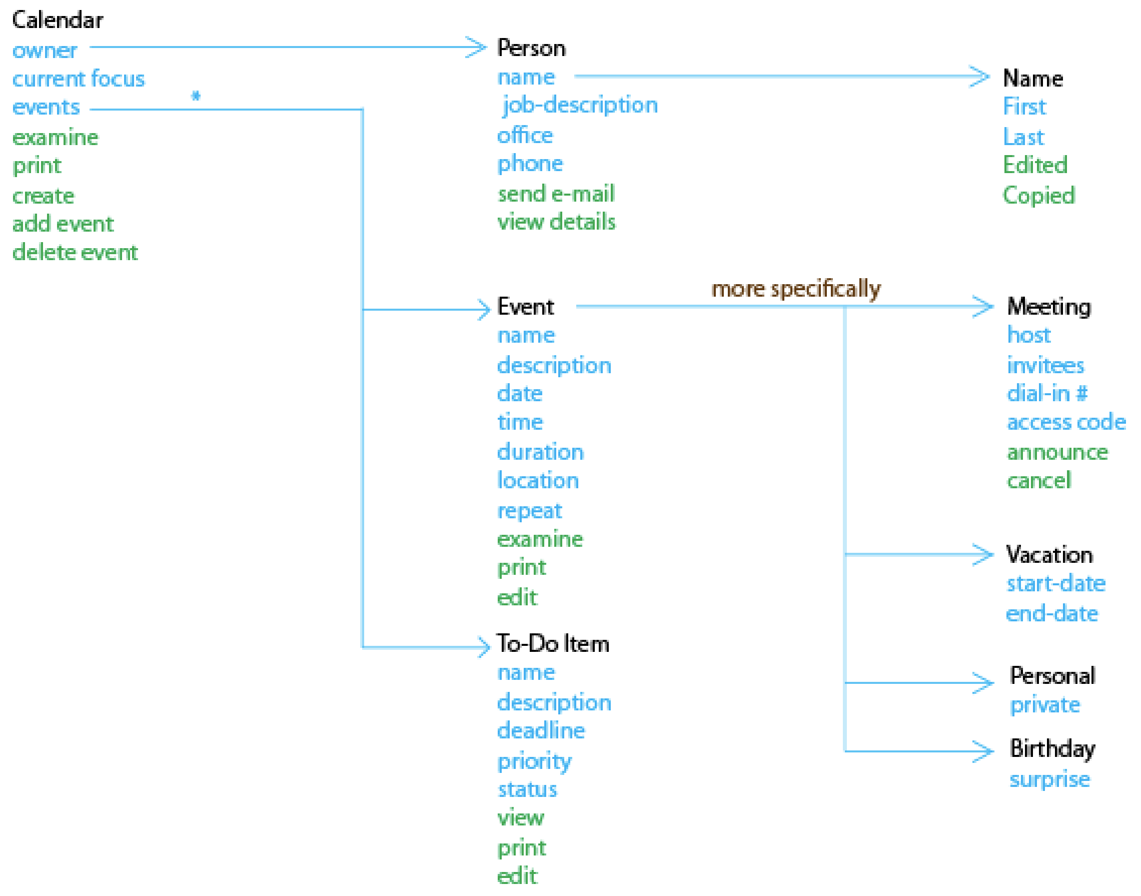
Representation:
Entity Relationship Diagrams



Representation:
Concept Maps


Java™ Technology Concept Map
Java
learn and use
Developers
to create and run
programs
that make
devices
and the
internet
useful for
people
Java Community Process
Programming language
Java object model
Java virtual machine (JVM)
Development process
Java 2 Platform: end-to-end solutions
for enterprise servers and applications
for desktop servers and applications
for consumer and embedded servers and applications
Documentation

Representation: Concept Map



Conceptual Scenarios

- Stories of users doing tasks
  - Aka: essential use-cases
- Expressed at *conceptual* level
  - Using terms from vocabulary
- Not keystroke or GUI-control level
  - Too early for that

Conceptual Scenarios

John just returned from a two week vacation in San Francisco, and wants to show his photos to his friends. He's already deleted the truly bad ones from his camera, but now he needs to download the remaining ones so he can see them in larger format and edit them down to a slideshow of only the best shots. He starts up the photo management program, connects to where the photos are stored, and imports the pictures. Next, he browses through the photos. Some that looked OK on his camera are actually out of focus, so he deletes them. A few photos of people have the red-eye problem, so he fixes that. He creates an album and names it "SF 2013", and puts the top 50 photos into it. He moves some photos around in the show so the slideshow flows better, then sets it to fade between photos. He checks the slideshow to make sure it flows well, then quits the app.

- **No presentation or keystroke-level details!**
- **Can separate into use-cases.**

Enhanced conceptual modeling

- Using companion models
  - Progressive disclosure
  - Component models
  - Surrounding models
- Modeling interactions
  - Managing errors
  - Anticipating trouble
- Evolving the application
  - Managed growth
  - Anticipated growth
  - Unanticipated growth

Benefits of having a clear Conceptual Model

- Declares what is (& isn't) exposed to users
- Basis for high-level task scenarios
- Basis for product vocabulary
- Clear target for designing the rest of the UI
  - See object relationships => allows simplification
- Jump-starts and focuses implementation
  - Basis for & distinction from internal object-model
- Facilitates documentation, training, support
- *Saves time and money!*

Selected References

- Johnson, J. and Henderson, D.A. (2024), *Conceptual Models: Core to the Design of Interactive Applications*, Spring Nature.
- Johnson, J. and Henderson, D.A. (2002), "Conceptual Models: Begin by Designing What to Design", *Interactions*, Jan-Feb 2002, pp. 25-32.
- Moran, T.P. (1983) "Getting into a system: External-internal task mapping analysis", In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems (CHI '83)*, Ann Janda (Ed.). ACM, New York, NY, USA, pp. 45-49.
- Norman, D.A., and Draper, S.W. (1986). *User Centered System Design: New Perspectives on Human-Computer Interaction*, Hillsdale, New Jersey: CRC.
- Suchman, L.A. (2007), *Human-Machine Reconfigurations: Plans and Situated Actions*, Cambridge University Press.
- Young, R.M. (1981) "The Machine Inside the Machine: Users' Models of Pocket Calculators." *International Journal of Man-Machine Studies* (1), pp. 51-85.

Questions?