# Human-Computer Interaction

## CPSC 481 - Spring 2019

## Lessons from The Design of Everyday Things
## |||

Adapted from Tony Tang

# Conceptual Models

# Conceptual Models

- *“In interacting with the environment, with others, and with the artefacts of technology, people form internal, mental models of themselves and of the things with which they are interacting.*

- *These models provide **predictive** and **explanatory** power for understanding the interaction.”*

    - Don Norman

# Conceptual Model vs. Conceptual Design

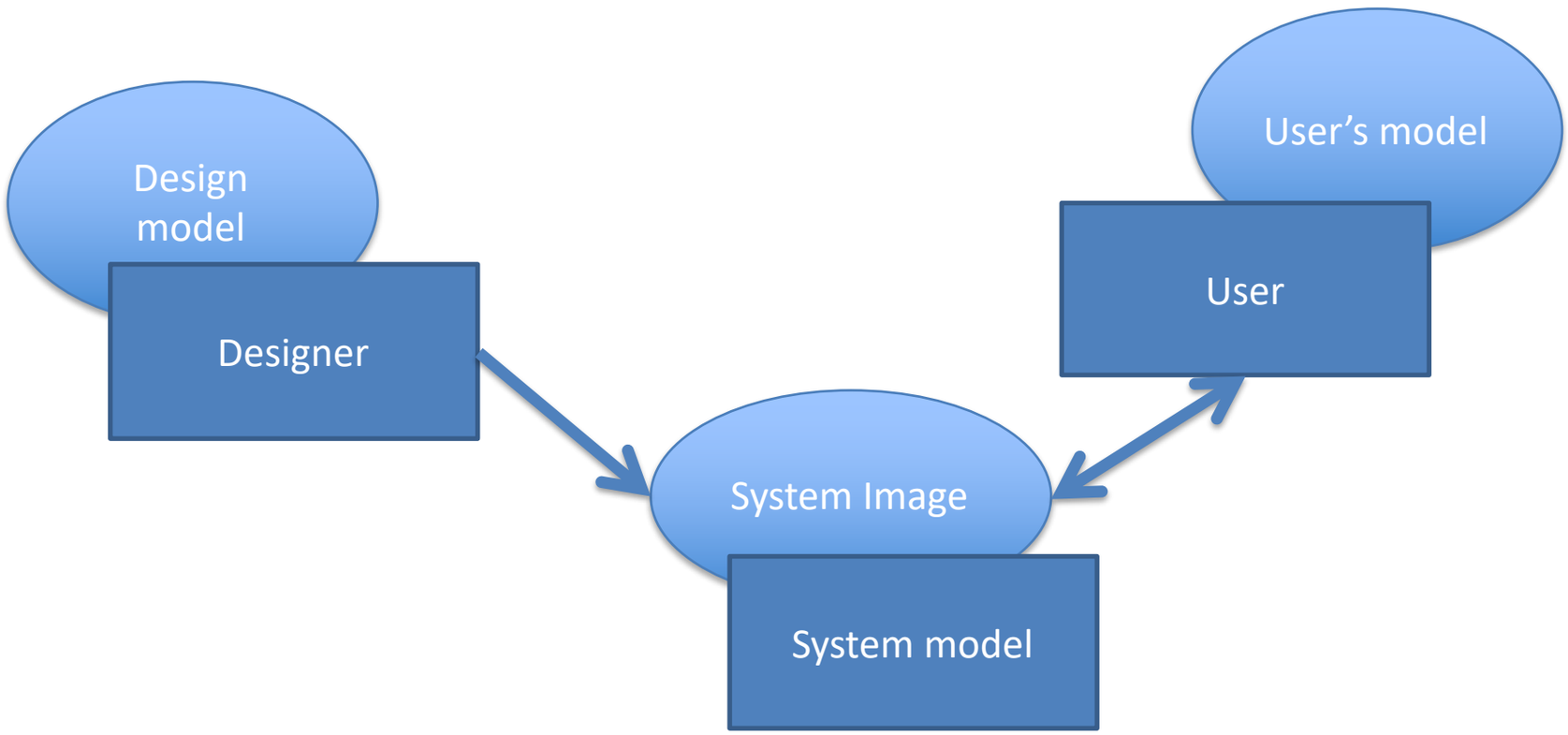
- **Conceptual Model**: something the user has (or forms)
    - Users see and understand the system through mental models
    - Users rely on mental models during usage

# Conceptual Model vs. Conceptual Design

- **Conceptual Model**: something the **user** has (or forms)
  - Users see and understand the system through mental models
  - Users rely on mental models during usage
- **Conceptual Design**: something the **designer** does
  - Defining the *intended* mental model (hiding the technology of the system)
  - Defining a suitable *system image* (applying appropriate design guidelines)

# Conceptual Design

- Designer’s role is to provide a meaningful, useful system image so that a user’s model matches the design model



# Some terms…

- **Design model**: what a designer intends to convey
- **System image**:  what the user “sees” - the UI, documentation, labels etc.
- **User’s model**: the user’s mental model developed by the user through interaction with the system
    - i.e. a belief system about the system model

# Mismatches between user's model and system model

- Sometimes a poor model is fostered deliberately
    - Consider a fridge or the heating system in a house vs. heating/cooling system in a car

# Mismatches between user's model and system model

- **Sometimes, it's a representation problem**
  - Document sizes measured in bytes, not pages or words
  - Dates may be in non-standard formats
  - Error messages may use system specific codes
- Other times, it's a **"the designer is not the user"** problem (and the design model uses terms/concepts that are non-existent in the user's model)
  - Remember the main lesson from day 1:
    - YOU ARE NOT THE USER


ASUS RT-AC66U
Logout
Reboot
English
Operation Mode: Wireless router
Firmware Version: 3.0.0.4.374_4561
SSID: proclubboy4 proclubboy4
Quick Internet Setup
General
Network Map
Guest Network
Traffic Manager
Parental control
USB application
AiCloud
Advanced Settings
Wireless
LAN
WAN
IPv6
VPN
Firewall
Administration
System Log
Network Tools
Internet status:
Connected
WAN IP: 192.168.0.10
DDNS: GO
Security level:
WPA2-Personal
Clients: 12
No Device
No Device
System Status
2.4GHz
5GHz
Wireless name(SSID)
proclubboy4
Authentication Method
WPA2-Personal
WPA Encryption
AES
WPA-PSK key
Apply
LAN IP
192.168.1.1
PIN code
91725301
LAN MAC address
74:D0:2B:3F:32:A0
Wireless 2.4GHz MAC address
74:D0:2B:3F:32:A0

# Where do mental models come from?

- **During system usage:**
    - User's own activity leads to a mental model
    - Explanatory theory, developed by the user
    - Often used to predict the future behaviour of the system

# Where do mental models come from?

- **During system usage:**
  - User's own activity leads to a mental model
  - Explanatory theory, developed by the user
    - Often used to predict the future behaviour of the system
- **Observing others use the system:**
  - Casual observation of others working
  - Asking someone else to "do this for me"
  - Formal training

# Where do mental models come from?

- **During system usage:**
  - User's own activity leads to a mental model
  - Explanatory theory, developed by the user
    - Often used to predict the future behaviour of the system
- **Observing others use the system:**
  - Casual observation of others working
  - Asking someone else to "do this for me"
  - Formal training
- **Reading about a system:**
  - documentation, help screens, "for dummies" books

# Where do mental models come from?

- **During system usage:**
  - User's own activity leads to a mental model
  - Explanatory theory, developed by the user
    - Often used to predict the future behaviour of the system
- **Observing others use the system:**
  - Casual observation of others working
  - Asking someone else to "do this for me"
  - Formal training
- **Reading about a system:**
  - documentation, help screens, "for dummies" books
- **Previous understanding:**
  - Experiences with related (or unrelated) systems/ concepts
- **This is <u>done by the user</u> (not the designer)**

# Mental models are “runnable”

- Includes a notion of **causality**:  “doing this will result in this”

- Used for **explanation**
  - To understand why the system responds as it does

- Used for **prediction**
  - To select an appropriate action in a given situation

# Types of Mental Models

- **Structural**:

- **Functional**:

# Types of Mental Models

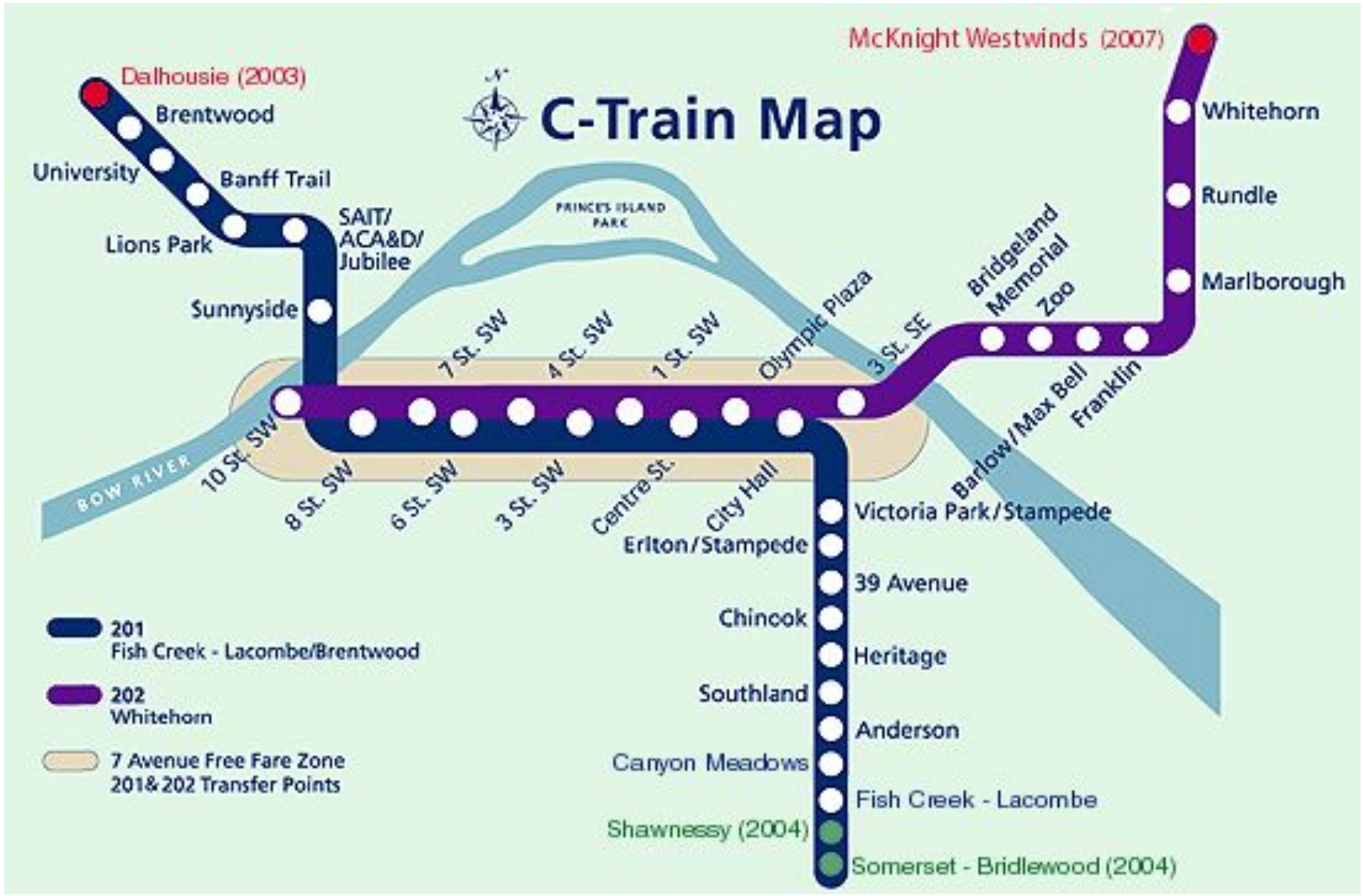
- **Structural**: an image of *what* the system *is*
  - Descriptive of what the system is
  - User may need additional knowledge to actually use it
  - Often more powerful/flexible, though harder to use
  - ***Road map***: may show a particular information, but isn't customized to your particular use of that information
- **Functional**:

# Types of Mental Models

- **Structural**: an image of ***what*** the *is*
    - Descriptive of what the system is
    - User may need additional knowledge to actually use it
    - Often more powerful/flexible, though harder to use
        - ***Road map***: may show a particular information, but it's not customized to your particular use of that information
- **Functional**: action-based; describes ***how*** it is used
    - Prescriptive; specific; step-by-step
    - Does not assume global or system knowledge
    - Easier to use, but not helpful for problem-solving or dealing with the unexpected
    - ***Google directions***: great when everything's there; not so great when the road is closed

# Structural Models -> “what the system is”

- **Maps and schematics**



# Structural Models -> “what the system is”

- **Object-action models**
    - Users think in terms of concrete or abstract objects
    - System supports actions on the objects
- *What are some programs that you think of as having “object that act”, or that we can act on objects?*
- Unix: files are objects, commands act on them etc.
- Powerpoint: text, images etc.

# Structural Models -> “what the system is”

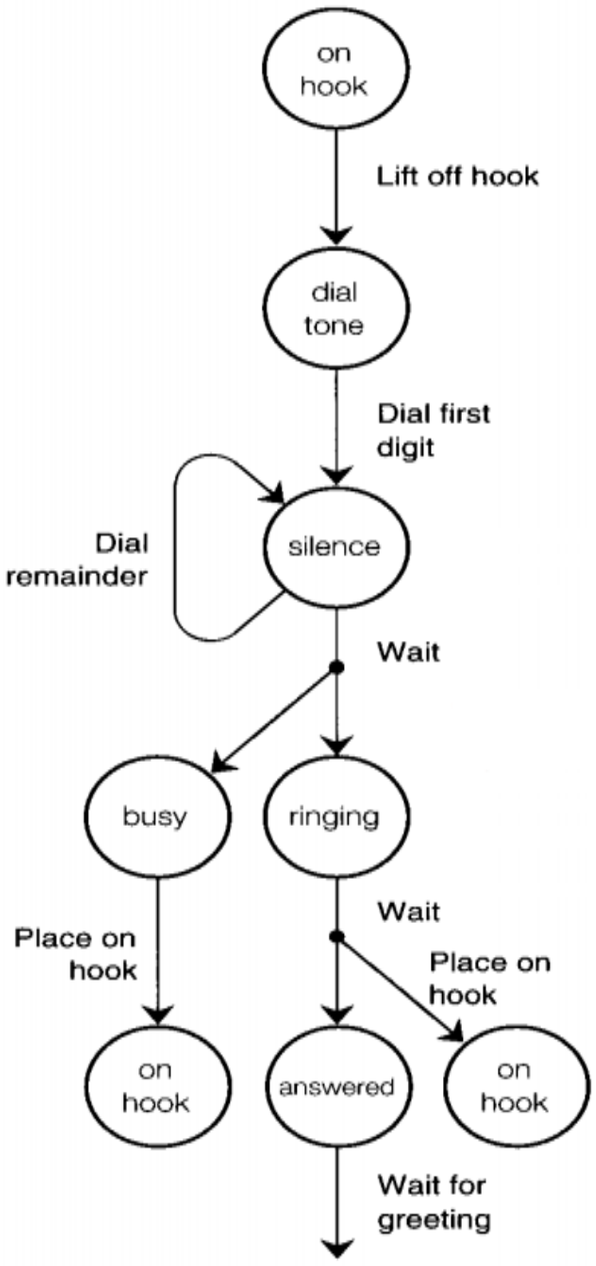
- **Analogies/metaphors**
  - A new system (closely) resembles an old system
  - (usually) intent is to help transfer existing system knowledge
  - e.g. desktop metaphor; spreadsheets

# Functional Models -> “how the system is used”

- Many kinds of user manuals and step-by-step and “how-to” guides
    - U of C’s “job aids”
        - https://www.ucalgary.ca/training/business-processes-job-aids-qrg

# Functional Models -> “how the system is used”

- **State transition model**
  - How most of us think of how phones work (what you see on the right is a **representation** of that model)
  - Changes in state need to be **visible**, otherwise, this kind of model doesn’t form



# Functional Models -> “how the system is used”

- **Mapping models**
  - Users learn a sequence of actions to accomplish tasks
  - Mappings need to be “rote-learned”; often arbitrary
  - Hand-held calculator maps “math” to key presses; keyboard shortcuts

# System Image

- Designers control what the users see (*i.e. the system image*). That's it!
    - Choose a system image to foster a good mental model
- Some interfaces actually display the system!
    - All objects and actions may be visible at all times
    - *e.g.* automobile dashboard (system image) that is literal - this is your car!*
- Some principles for designing this system image:
    - *Currency* (up-to-date-ness) is important
    - *Consistency* (contributes to learnability)
    - *Feedback*

# Where a simple model might be better (to hide system complexity)

- Many systems have messy low-level details
  - Irrelevant to a user's activity
  - Full functionality may not be required
- e.g. MS Word has hundred (thousands?) of commands
  - Most users only use a small subset of these
  - Users can hide complexity through customisation
  - IT admins can provide macro capabilities (bundling low-level commands into a single concept) -> 'submit' program
  - Wizards allow a user to "do what's right", skipping details

# Lessons from the Design of Everyday Things

- We’ve seen that a lot of things are designed poorly, be it computer interface, or physical objects
- Formally, there is a vocabulary around these concepts that we have discussed
    - **Perceived affordances**
    - **Visible contraints**
    - **Causality**
    - **Transfer effects**
    - **Idioms & population stereotypes**
    - **Conceptual models**
    - Individual differences

# Learning Objectives

- You should now be able to:
  - Describe why mismatches between user model and system model can cause problems
  - Identify four ways that people arrive at/develop their mental models
  - Describe two different kinds of mental models (and discuss the differences between them); discuss pros an cons of each
  - Identity principles in the design of effective system images

# Acknowledgements


- Tony Tang
- Lora Oehlberg
- Ehud Sharlin
- Frank Maurer
- Saul Greenberg

# Course information

- Website
  - GitHub Pages https://silvadasilva.github.io/CPSC481-2019S/
- Communications
  - Slack https://cpsc481-2019s.slack.com/
- Readings and Slides
  - Posted online at the main website