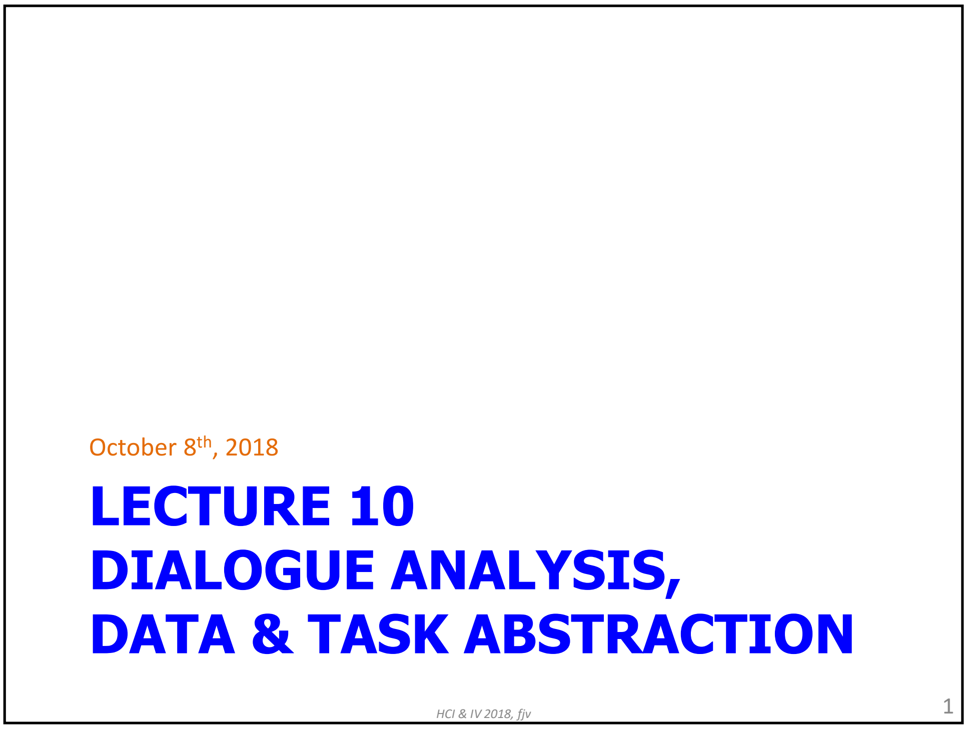
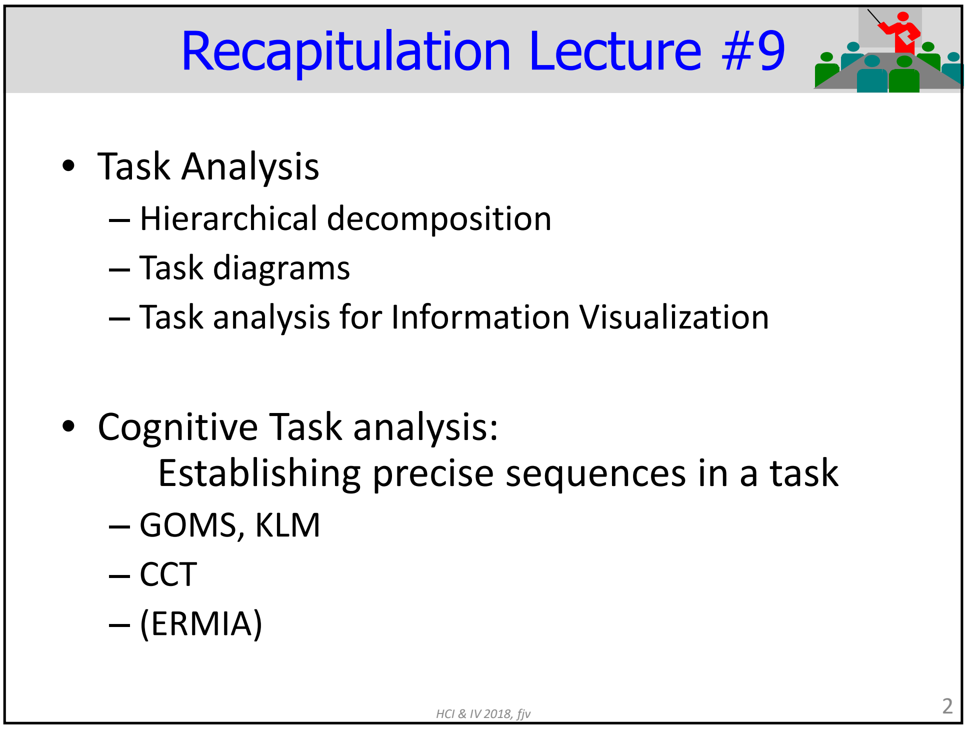


October 8th, 2018

LECTURE 10
DIALOGUE ANALYSIS,
DATA & TASK ABSTRACTION



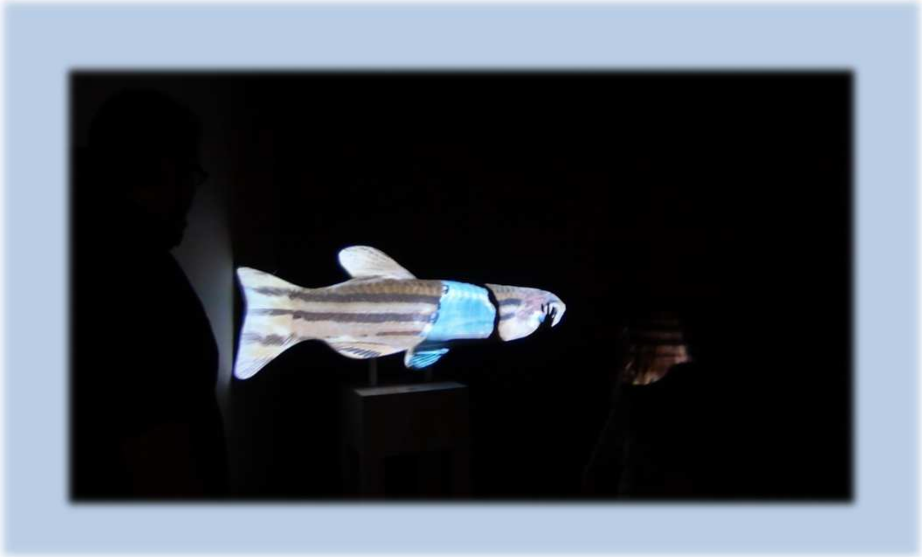
Recapitulation Lecture #9

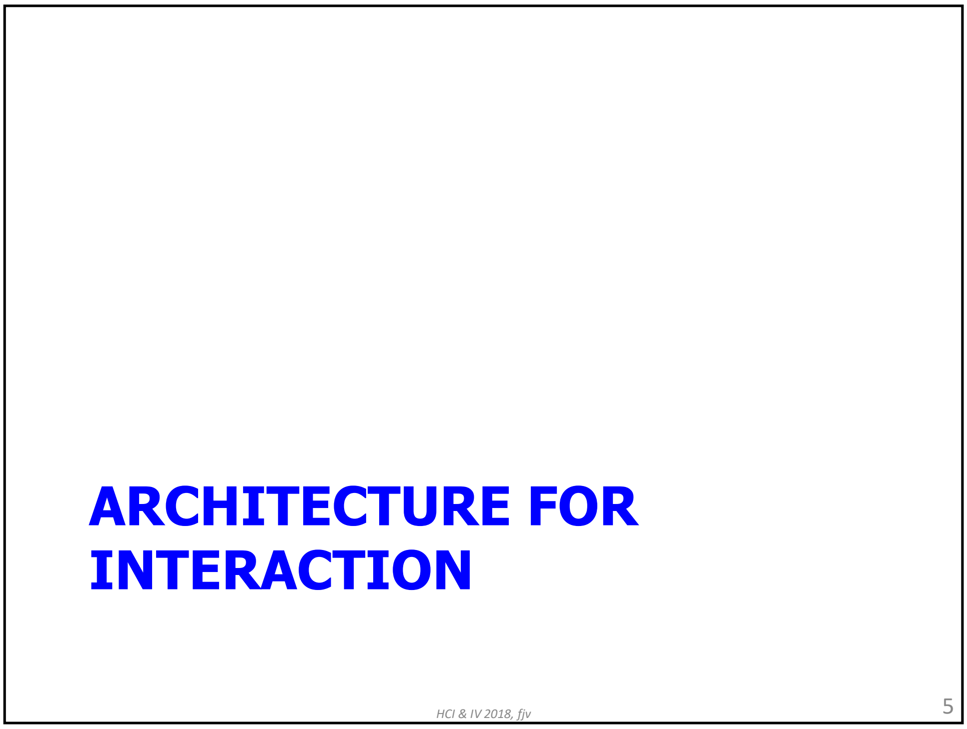
- Task Analysis
  - Hierarchical decomposition
  - Task diagrams
  - Task analysis for Information Visualization
- Cognitive Task analysis:
  Establishing precise sequences in a task
  - GOMS, KLM
  - CCT
  - (ERMIA)

Example from HCI 2013

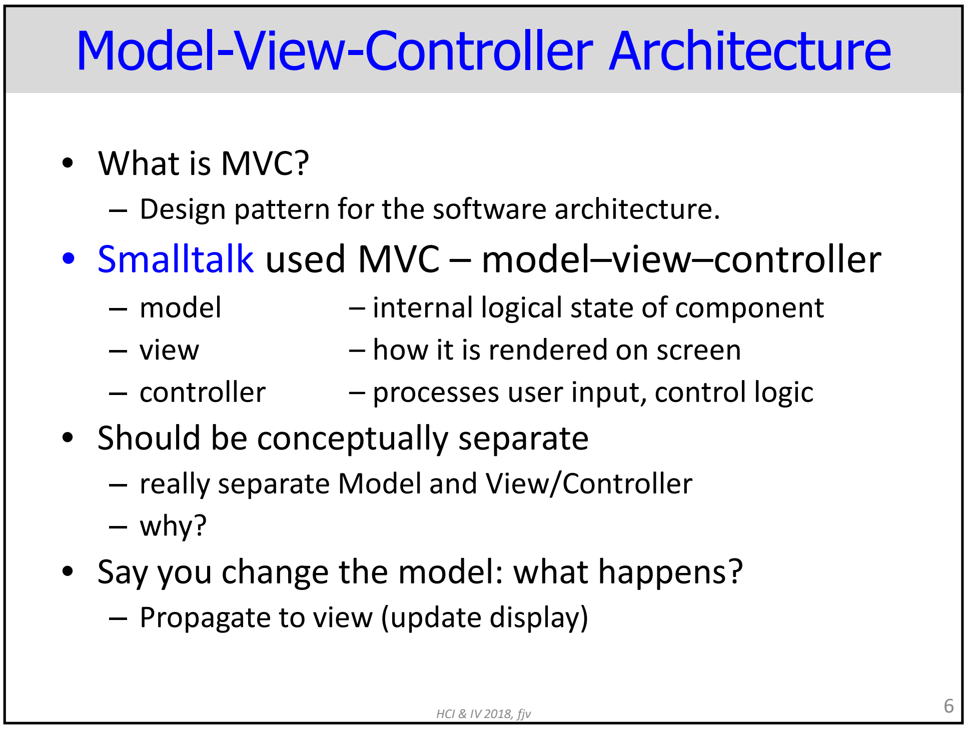
- Project for Augmentation
- Can we make an augmented interactive fish.
- Technical aspects need be solved.
  - Shadow casting
  - Projection
- Users have to sort out how it works.
- Realized with the vvvv environment

The Augmented Zebrafish



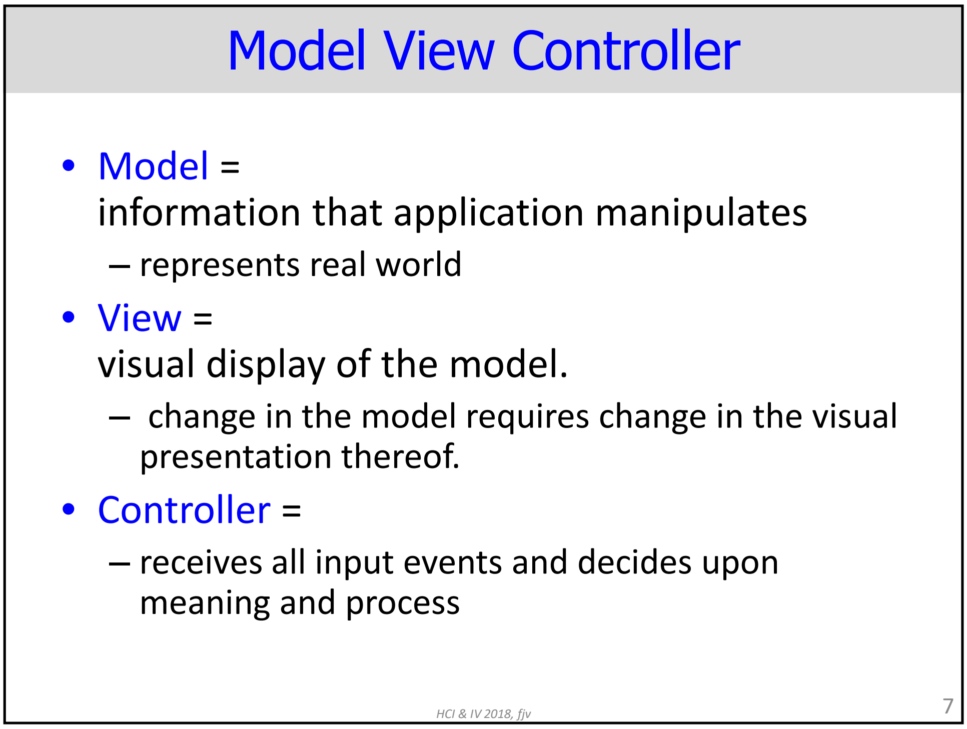


ARCHITECTURE FOR INTERACTION



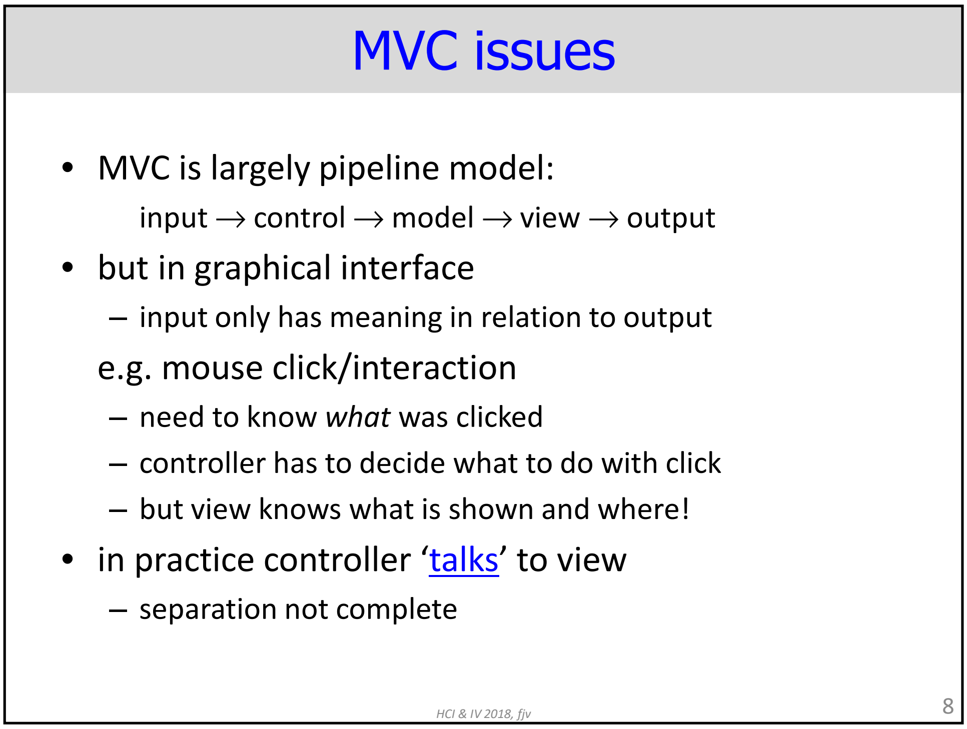
Model-View-Controller Architecture

- What is MVC?
  - Design pattern for the software architecture.
- Smalltalk used MVC – model–view–controller
  - model – internal logical state of component
  - view – how it is rendered on screen
  - controller – processes user input, control logic
- Should be conceptually separate
  - really separate Model and View/Controller
  - why?
- Say you change the model: what happens?
  - Propagate to view (update display)



Model View Controller

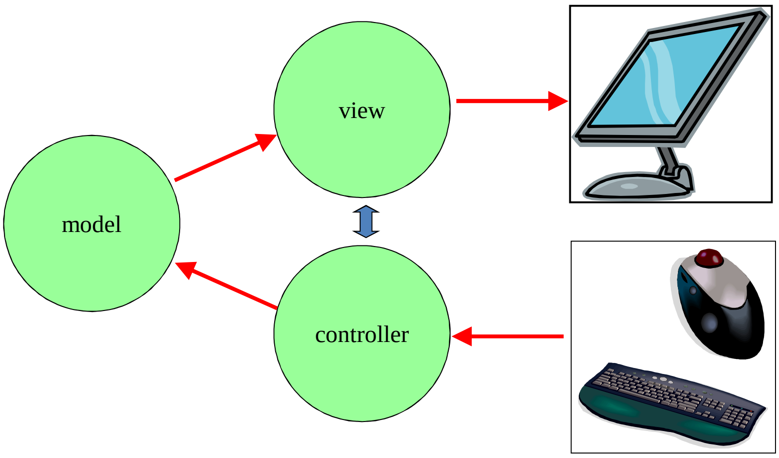
- Model =
  information that application manipulates
  – represents real world
- View =
  visual display of the model.
  – change in the model requires change in the visual presentation thereof.
- Controller =
  – receives all input events and decides upon meaning and process



MVC issues

- MVC is largely pipeline model:
  input → control → model → view → output
- but in graphical interface
  – input only has meaning in relation to output
  e.g. mouse click/interaction
  – need to know *what* was clicked
  – controller has to decide what to do with click
  – but view knows what is shown and where!
- in practice controller ‘talks’ to view
  – separation not complete

MVC: model - view - controller (1)

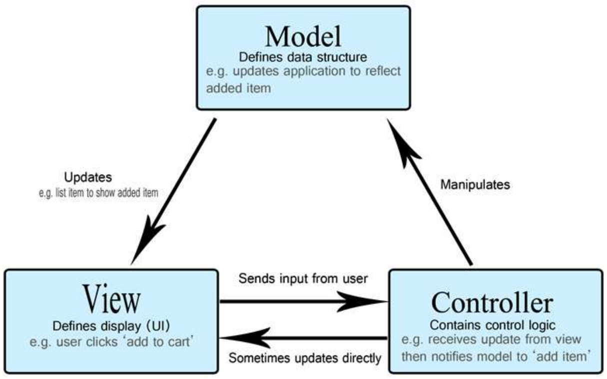


view

model

controller

MVC: model - view - controller (2)



Model
Defines data structure
e.g. updates application to reflect added item

Updates
e.g. list item to show added item

Manipulates

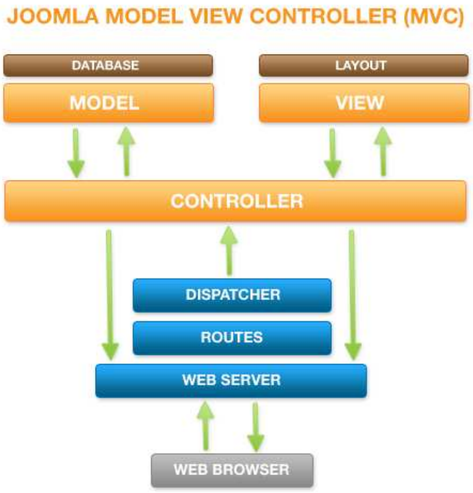
View
Defines display (UI)
e.g. user clicks 'add to cart'

Sends input from user

Sometimes updates directly

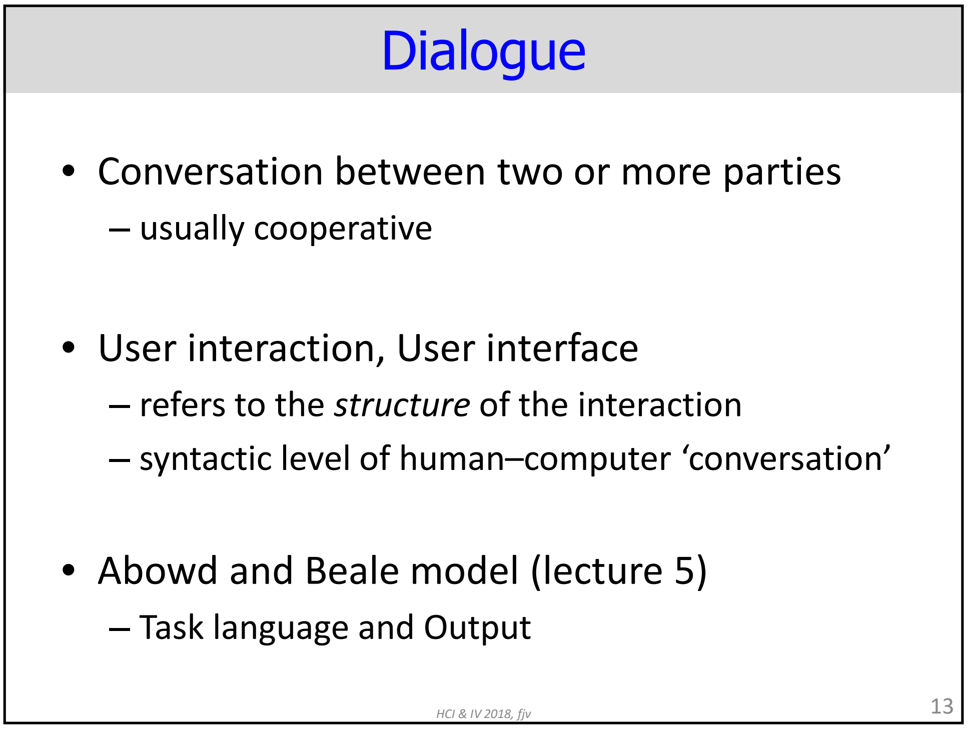
Controller
Contains control logic
e.g. receives update from view then notifies model to 'add item'

MVC underlies web programming



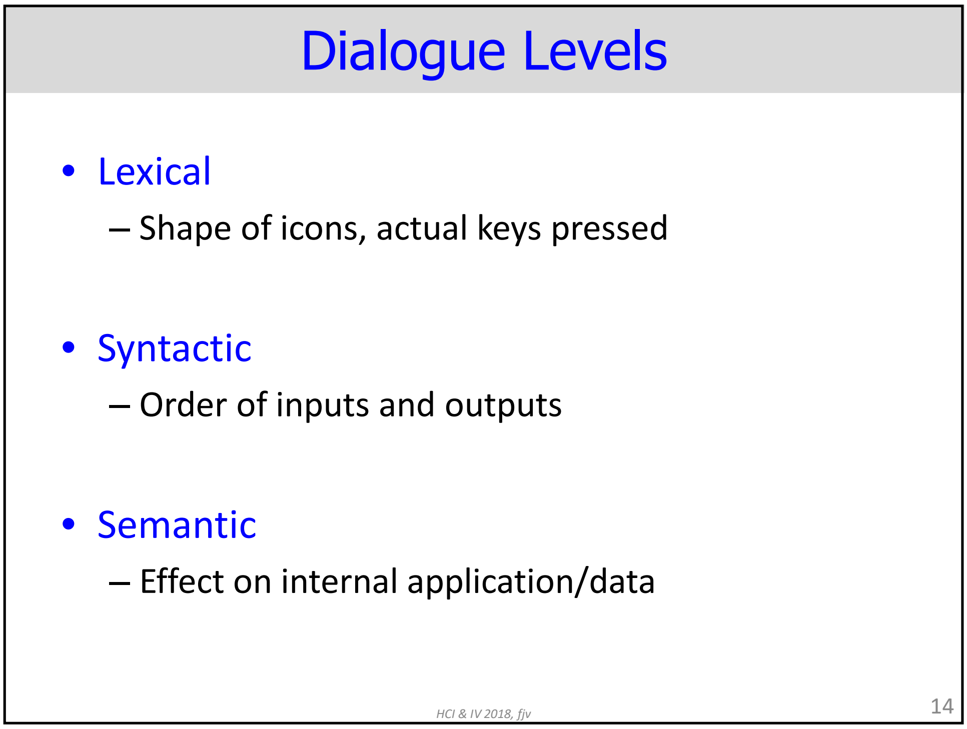
Consistent and Closing Dialogues

DIALOGUE DESIGN



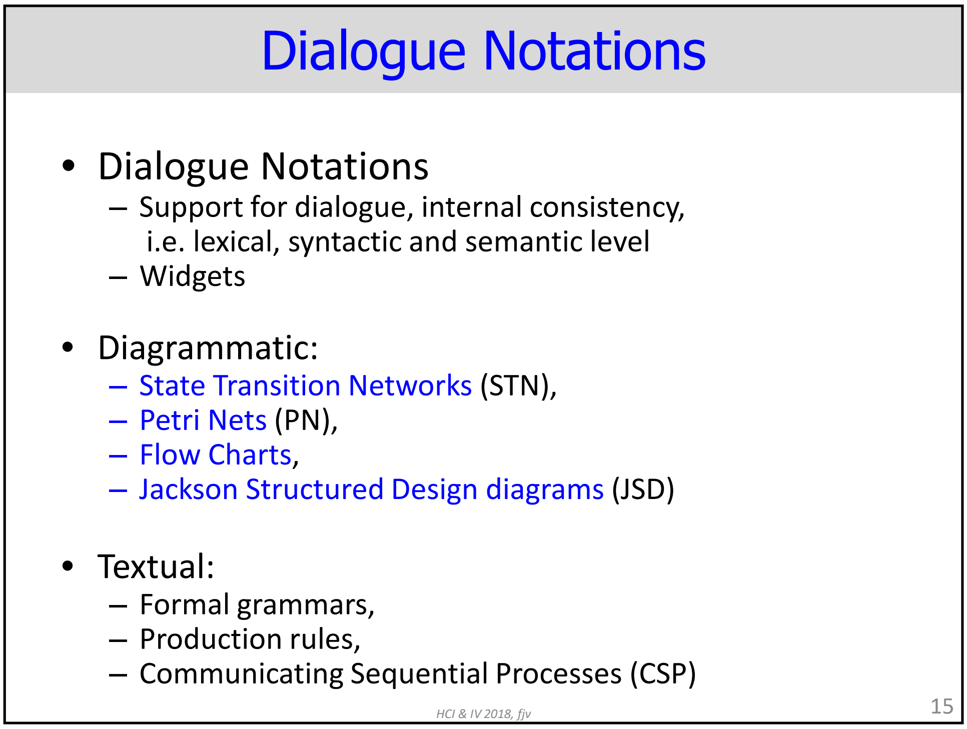
Dialogue

- Conversation between two or more parties
  - usually cooperative
- User interaction, User interface
  - refers to the *structure* of the interaction
  - syntactic level of human–computer ‘conversation’
- Abowd and Beale model (lecture 5)
  - Task language and Output



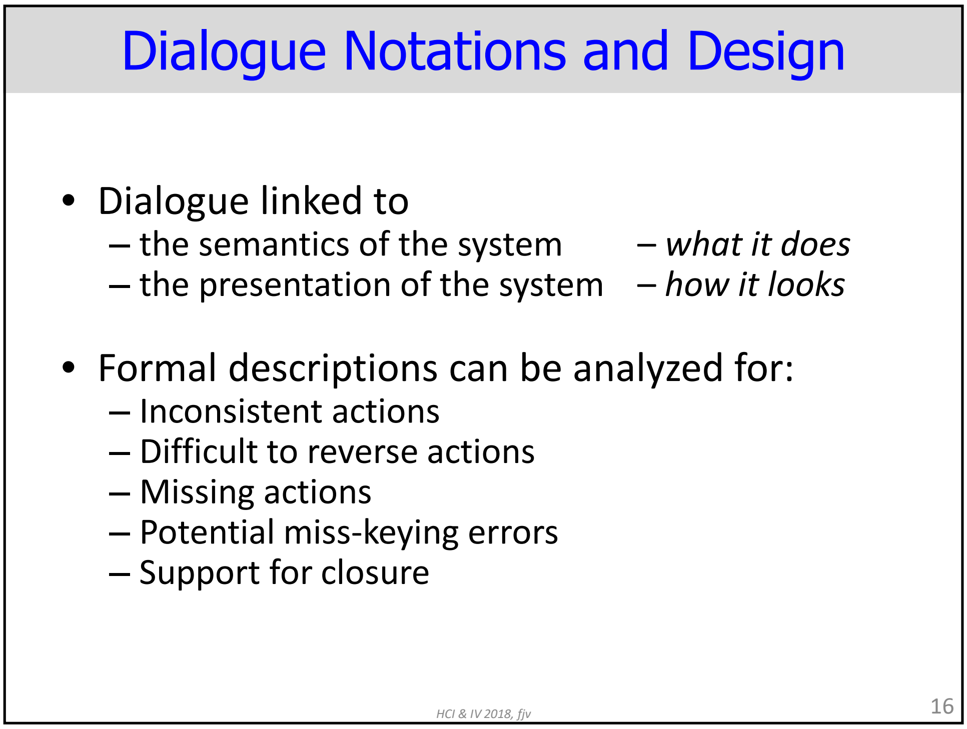
Dialogue Levels

- Lexical
  - Shape of icons, actual keys pressed
- Syntactic
  - Order of inputs and outputs
- Semantic
  - Effect on internal application/data



Dialogue Notations

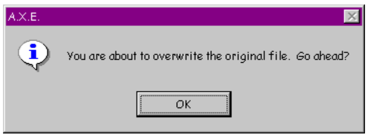
- Dialogue Notations
  - Support for dialogue, internal consistency, i.e. lexical, syntactic and semantic level
  - Widgets
- Diagrammatic:
  - State Transition Networks (STN),
  - Petri Nets (PN),
  - Flow Charts,
  - Jackson Structured Design diagrams (JSD)
- Textual:
  - Formal grammars,
  - Production rules,
  - Communicating Sequential Processes (CSP)



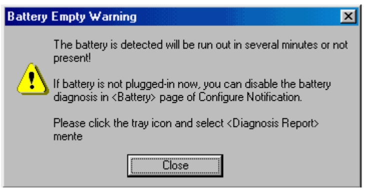
Dialogue Notations and Design

- Dialogue linked to
  - the semantics of the system – *what it does*
  - the presentation of the system – *how it looks*
- Formal descriptions can be analyzed for:
  - Inconsistent actions
  - Difficult to reverse actions
  - Missing actions
  - Potential miss-keying errors
  - Support for closure

Confusing Dialogue



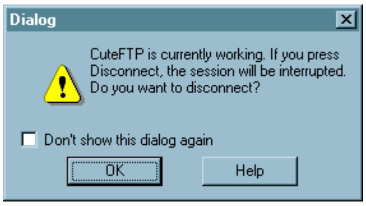
You are about to overwrite the original file. Go ahead?



The battery is detected will be run out in several minutes or not present!

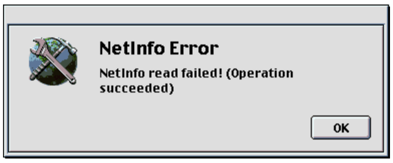
If battery is not plugged-in now, you can disable the battery diagnosis in <Battery> page of Configure Notification.

Please click the tray icon and select <Diagnosis Report> mente



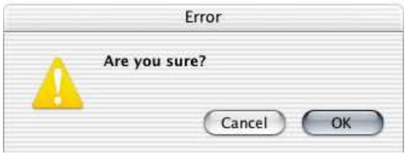
CuteFTP is currently working. If you press Disconnect, the session will be interrupted. Do you want to disconnect?

□ Don't show this dialog again

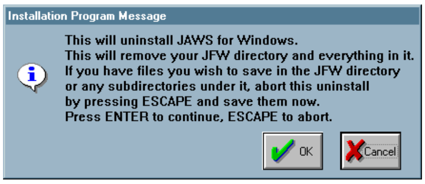


NetInfo Error

NetInfo read failed! (Operation succeeded)



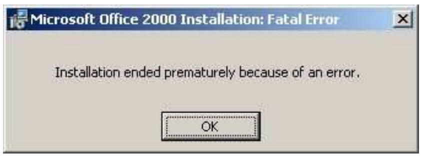
Are you sure?



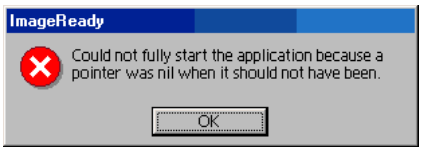
This will uninstall JAWS for Windows.
This will remove your JFW directory and everything in it.
If you have files you wish to save in the JFW directory
or any subdirectories under it, abort this uninstall
by pressing ESCAPE and save them now.
Press ENTER to continue, ESCAPE to abort.

HCI & IV 2018, fjv

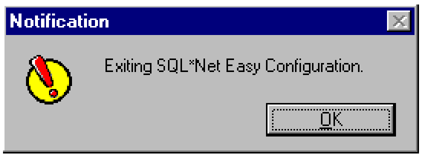
Void Dialogue



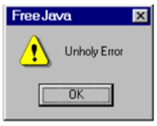
Installation ended prematurely because of an error.



Could not fully start the application because a pointer was nil when it should not have been.



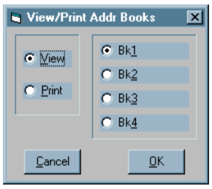
Exiting SQL*Net Easy Configuration.

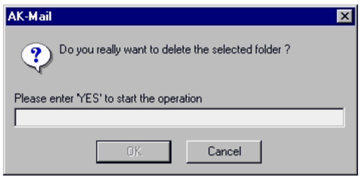


Unholy Error

HCI & IV 2018, fjv

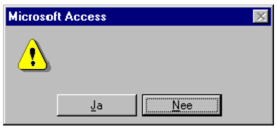
Completely unclear Dialogue

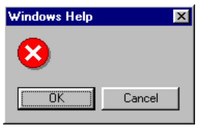


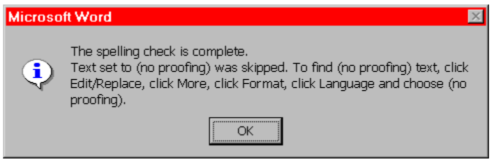


Do you really want to delete the selected folder ?

Please enter 'YES' to start the operation



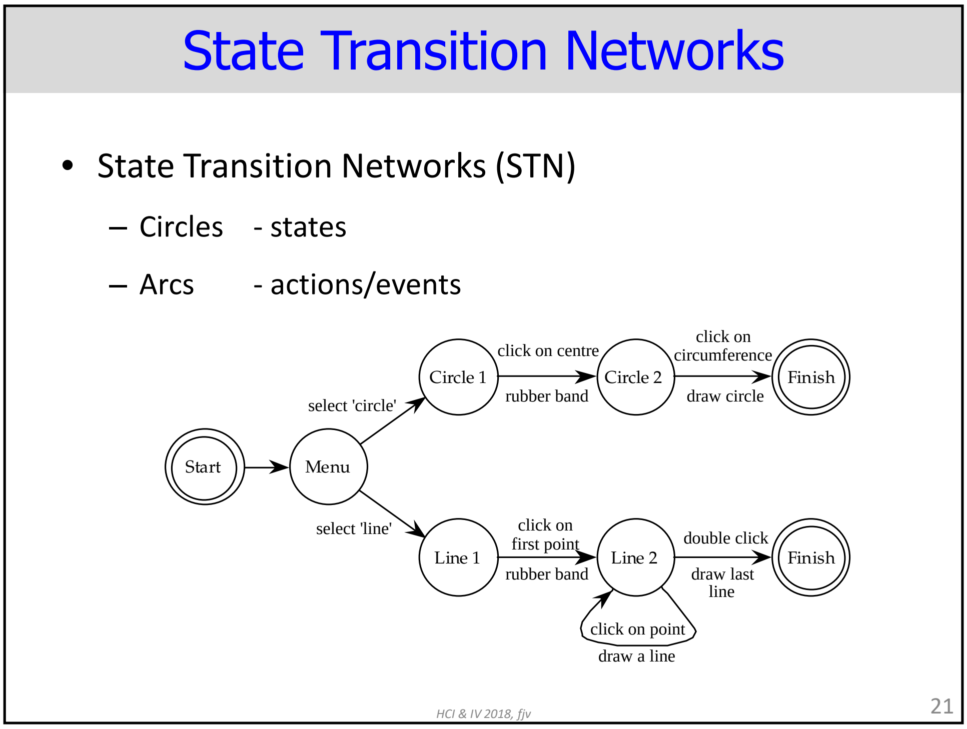




The spelling check is complete.
Text set to (no proofing) was skipped. To find (no proofing) text, click Edit/Replace, click More, click Format, click Language and choose (no proofing).

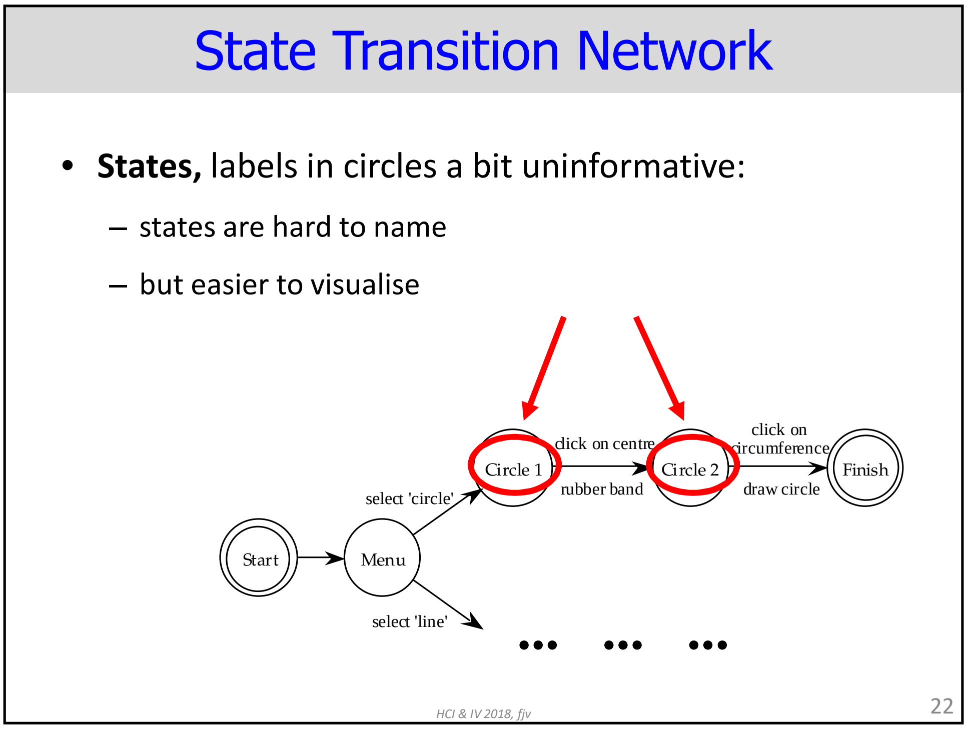
Notations for Dialogue Design

- Dialogue to much intertwined in the program
- For system maintenance, large systems:
  - change platforms (e.g. Windows/Mac)
  - dialogue notations helps to
    - analyse systems
    - separate lexical from semantic (MVC)
  - analyse the dialogue:
    - e.g. can the user always get to see current shopping basket
- In the systems/requirements analysis
  - notations help us understand proposed designs
  - LoFi prototyping in design phase



State Transition Networks

- State Transition Networks (STN)
  - Circles - states
  - Arcs - actions/events

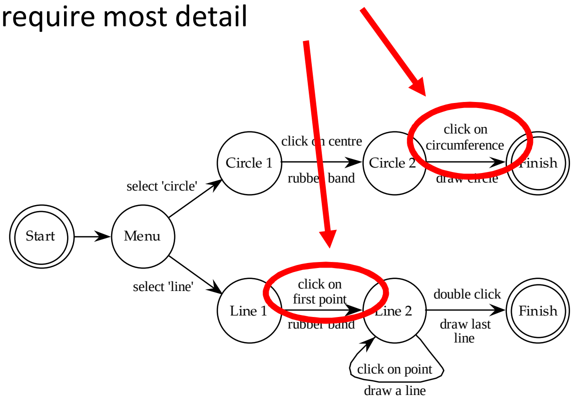


State Transition Network

- **States,** labels in circles a bit uninformative:
  - states are hard to name
  - but easier to visualise

State Transition Network

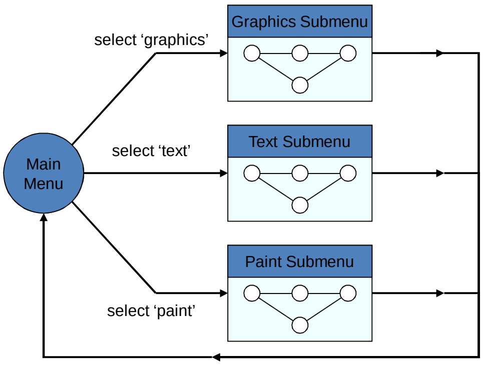
- **Events**, arc labels a bit cramped because:
  - notation is `state heavy‘
  - the events require most detail



HCI & IV 2018, fjv

Hierarchical STNs

- Nested STN organized in a hierarchical manner
- Managing complex dialogues
- Named sub-dialogues



HCI & IV 2018, fjv

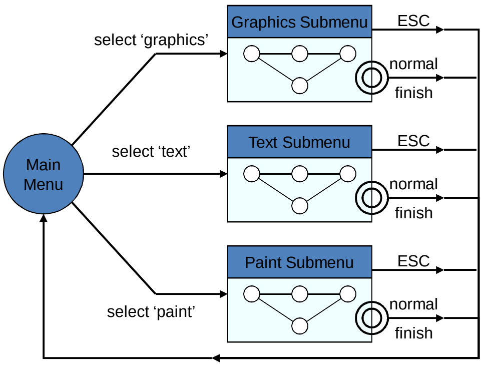
Escapes

- ‘back’ in Program Structure, escape/cancel keys
  - similar behavior everywhere
  - end up with spaghetti of identical behaviors
- Try to avoid this

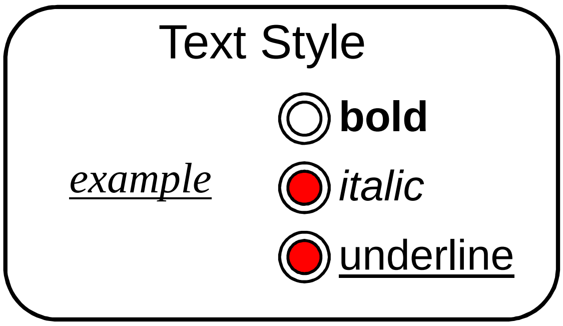
e.g. on high level diagram

‘normal’ exit for each submenu

plus separate escape arc active ‘everywhere’ in submenu



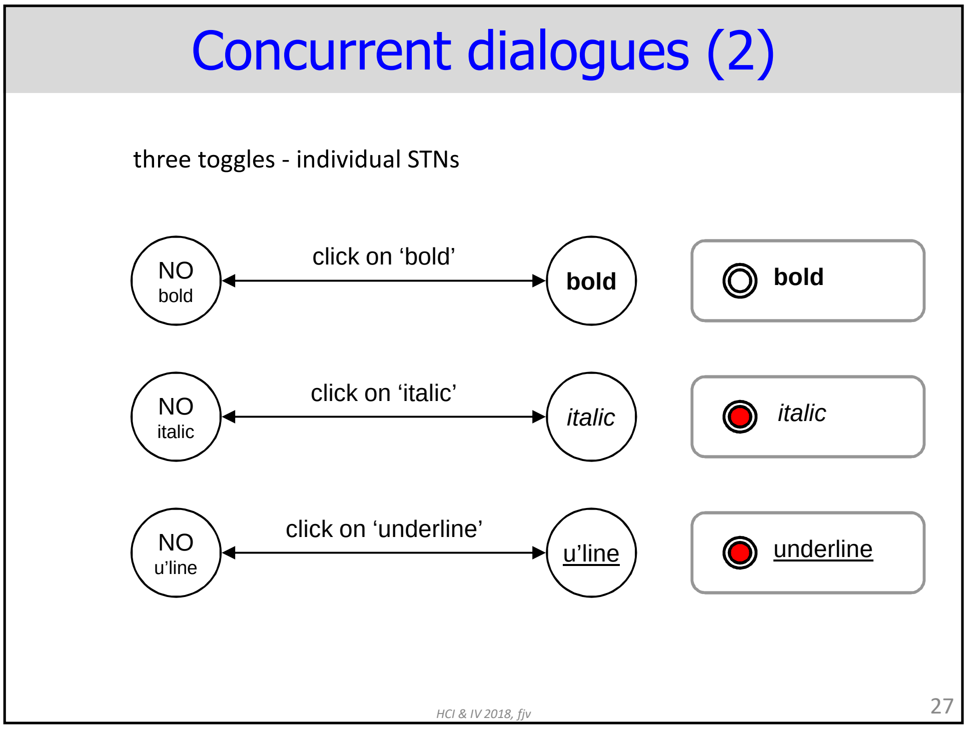
Concurrent Dialogue (1)



Simple dialogue box

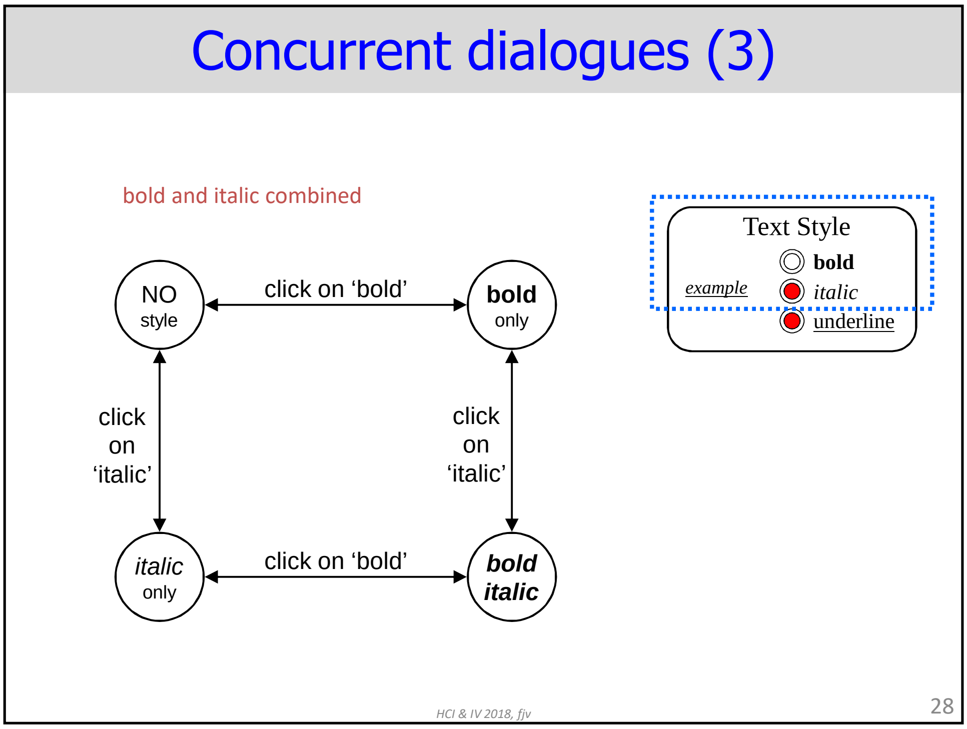
Toggle, either one state or the other

This dialogue, states are not mutual exclusive



Concurrent dialogues (2)

three toggles - individual STNs

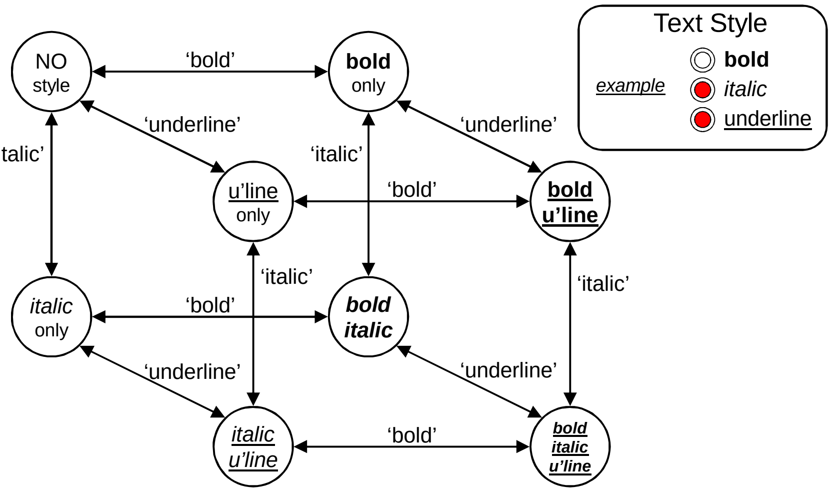


Concurrent dialogues (3)

bold and italic combined

Concurrent dialogues (4)

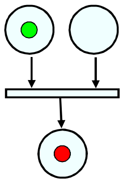
all together - combinatorial explosion



Petri Nets

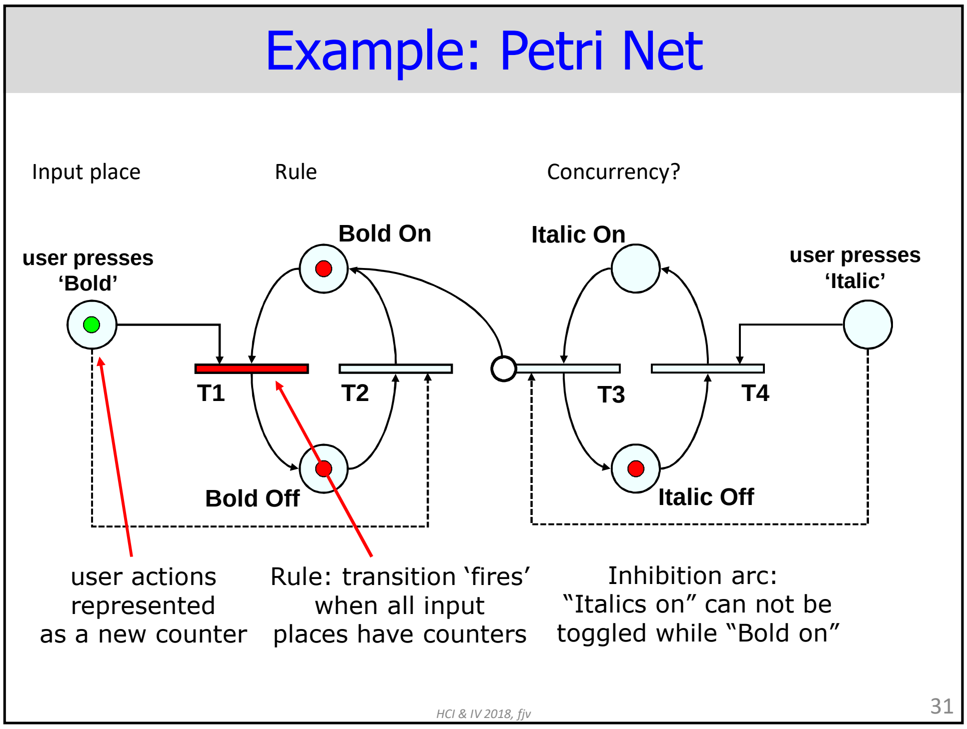
- Notation for reasoning about concurrent activities, computer science (Petri, 1962)

- Flow graph:
  - Place/Node Local state (cf. STN states)
  - Transitions Local state change (cf. STN arcs)
  - Arcs Connections of states
  - Counters/Tokens Information Unit, Current state
  - Rules Condition for a local state change



- More counters/tokens allowed
  - concurrent dialogue states

- Used for UI specification
  - Interactive Cooperative Objects (ICO)
  - Tool support: PetShop Univ. Toulouse

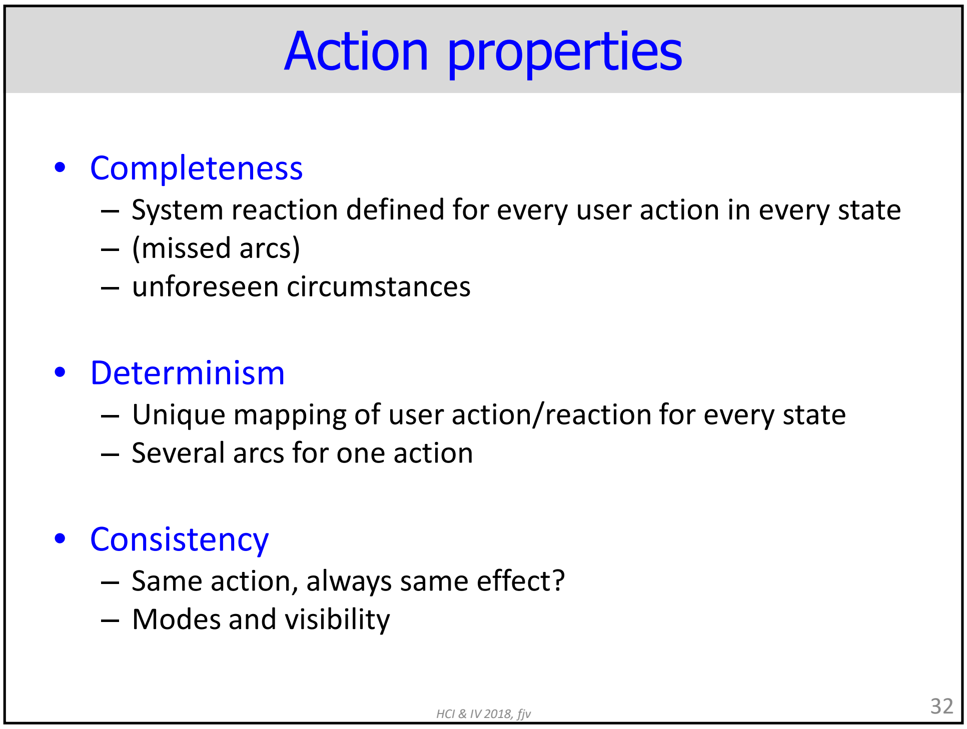


Example: Petri Net

user actions represented as a new counter

Rule: transition 'fires' when all input places have counters

Inhibition arc: "Italics on" can not be toggled while "Bold on"



Action properties

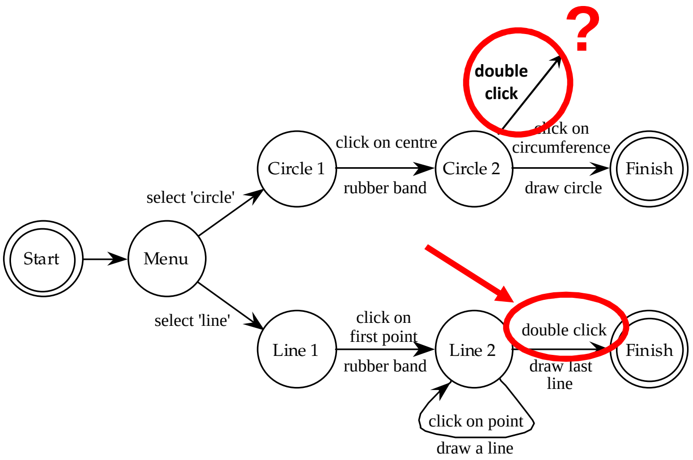
- Completeness
  - System reaction defined for every user action in every state
  - (missed arcs)
  - unforeseen circumstances
- Determinism
  - Unique mapping of user action/reaction for every state
  - Several arcs for one action
- Consistency
  - Same action, always same effect?
  - Modes and visibility

State properties

- Reach-ability
  - can you get anywhere from anywhere?
  - and how easy
- Reversibility
  - can you get to the previous state?
  - but NOT undo
- Dangerous states
  - states you do not want to get to – accidently
- Petri Net: model checking!

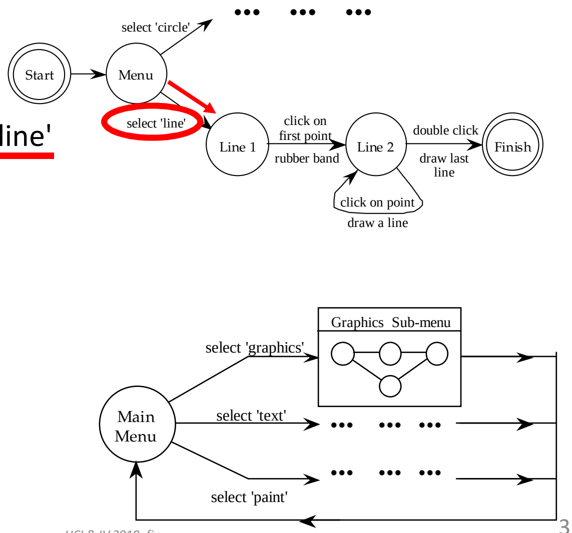
Checking properties (i)

- Completeness (& consistency)
  - double-click in circle states?



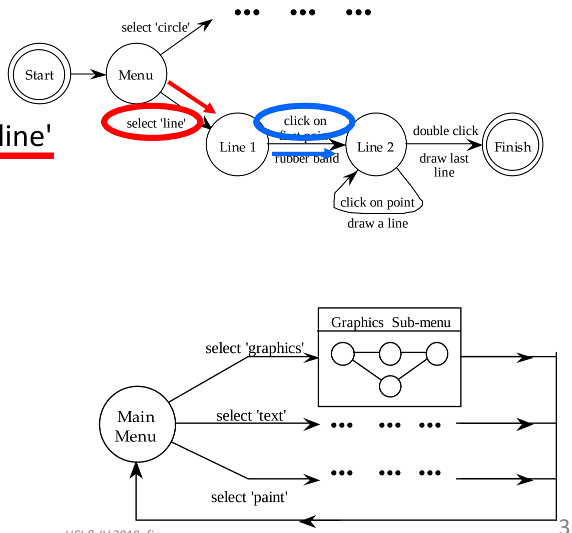
Checking properties (ii)

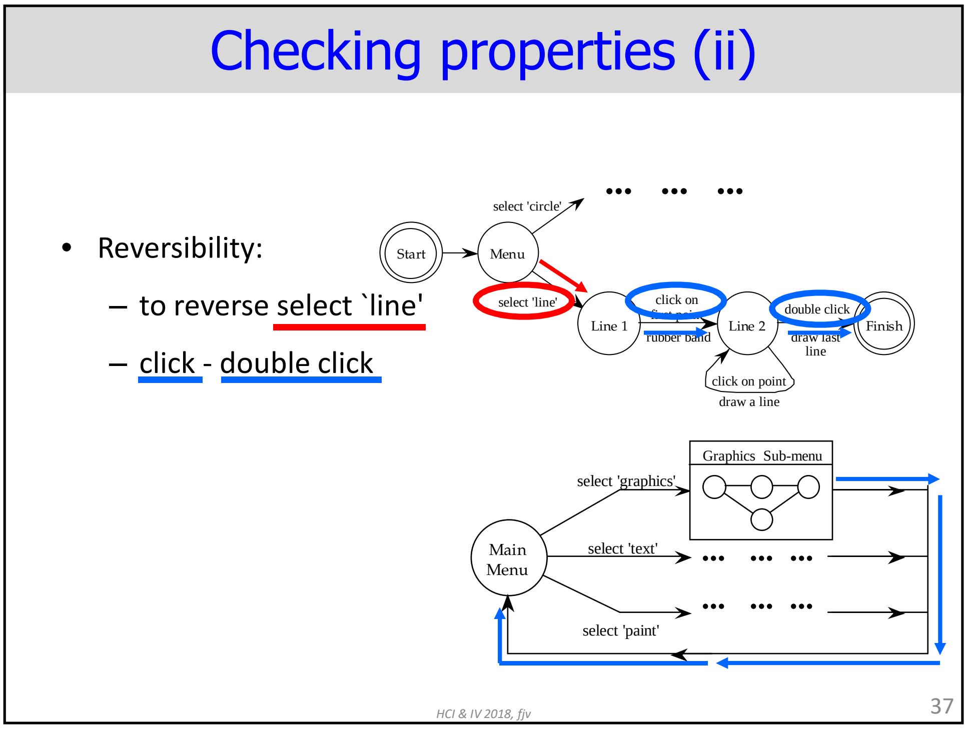
- Reversibility:
  - to reverse select `line'



Checking properties (ii)

- Reversibility:
  - to reverse select `line'
  - click



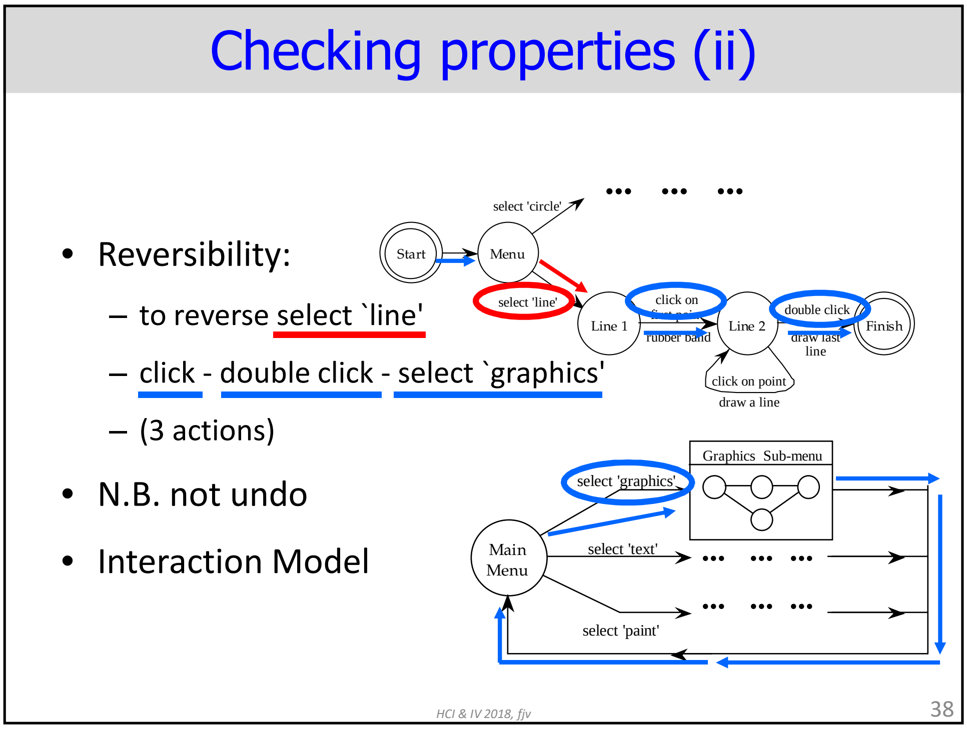


Checking properties (ii)

- Reversibility:
  - to reverse select `line'
  - click - double click

HCI & IV 2018, fjv

37



Checking properties (ii)

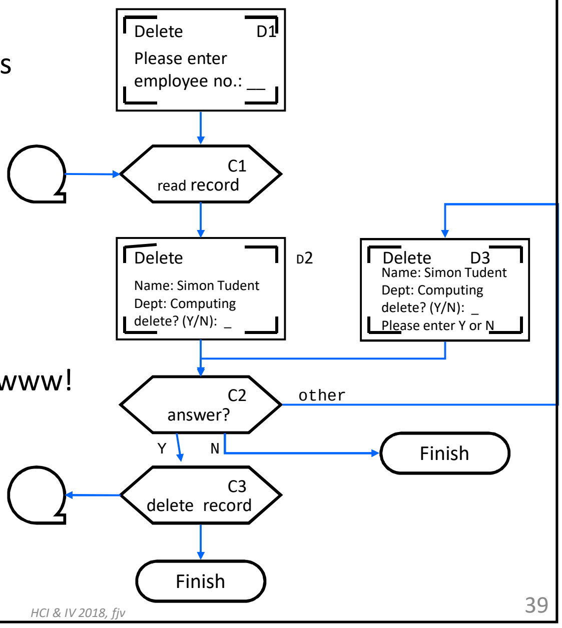
- Reversibility:
  - to reverse select `line'
  - click - double click - select `graphics'
  - (3 actions)
- N.B. not undo
- Interaction Model

HCI & IV 2018, fjv

38

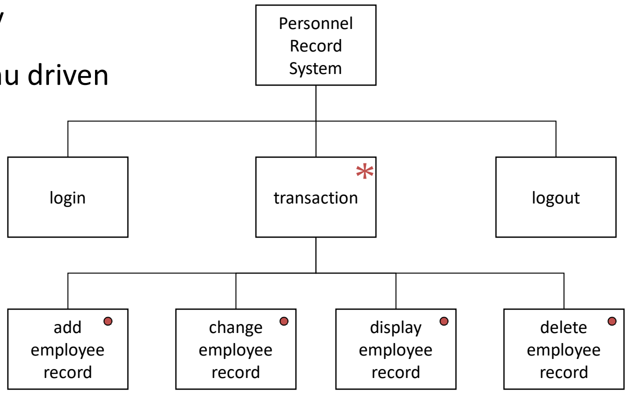
Flowcharts

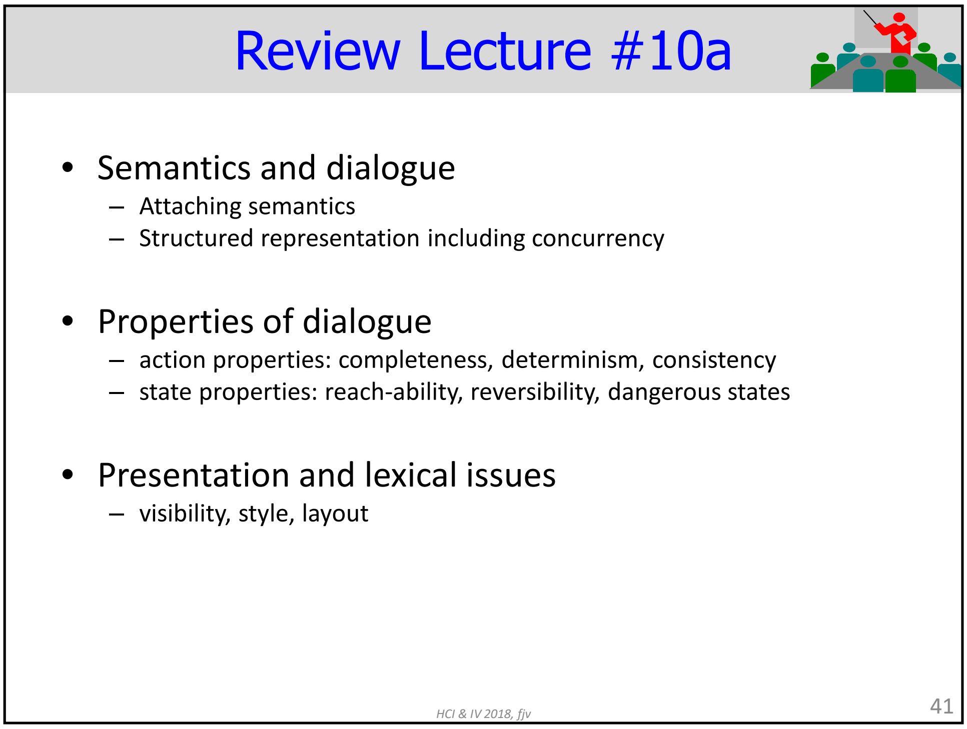
- Familiar to programmers
- Boxes
  - process/event
  - not state
- Use for dialogue
  - not for algorithm
- e.g. choose widgets on www!



Jackson Structured Design diagrams

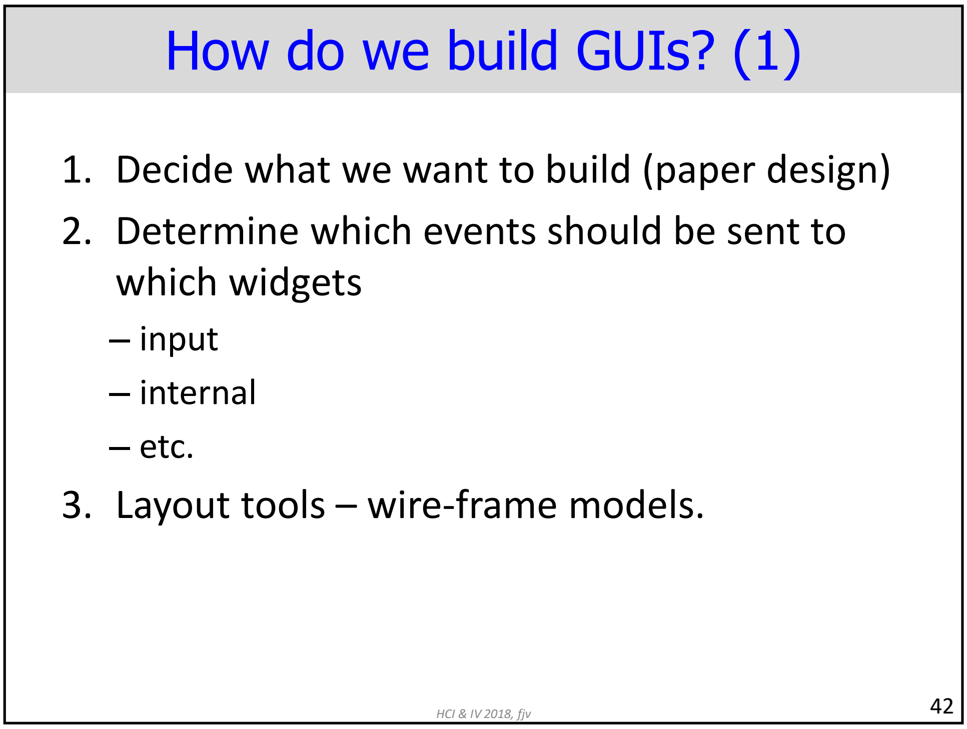
- JSD for tree structured dialogues
  - Less expressive
  - Greater clarity
  - Typical in menu driven software
  - Iteration
  - Optional





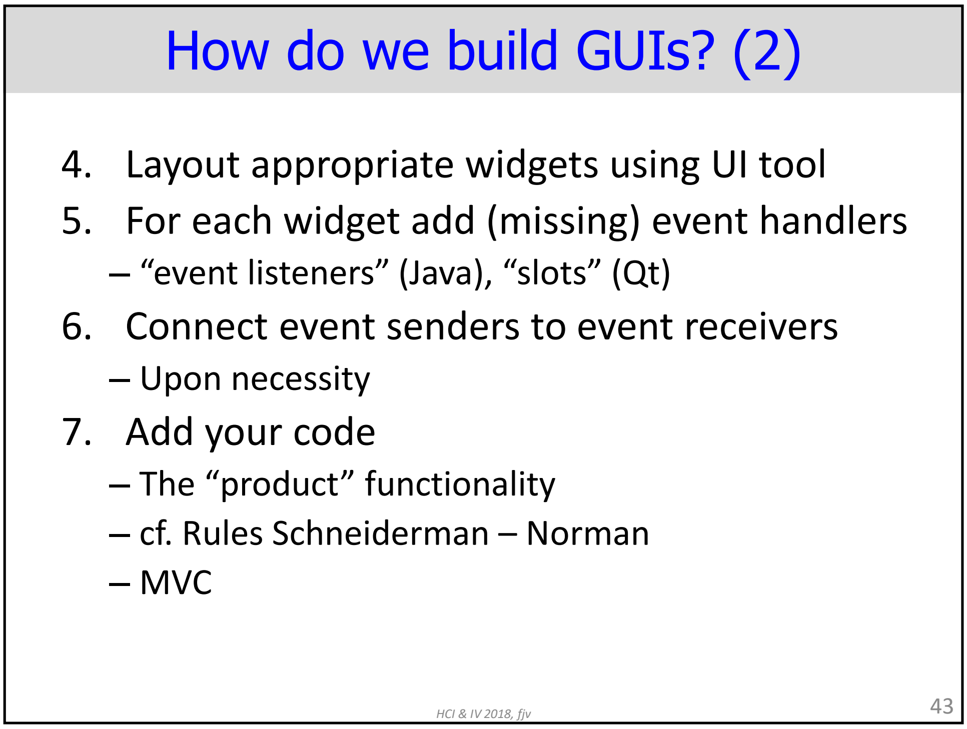
Review Lecture #10a

- Semantics and dialogue
  - Attaching semantics
  - Structured representation including concurrency
- Properties of dialogue
  - action properties: completeness, determinism, consistency
  - state properties: reach-ability, reversibility, dangerous states
- Presentation and lexical issues
  - visibility, style, layout



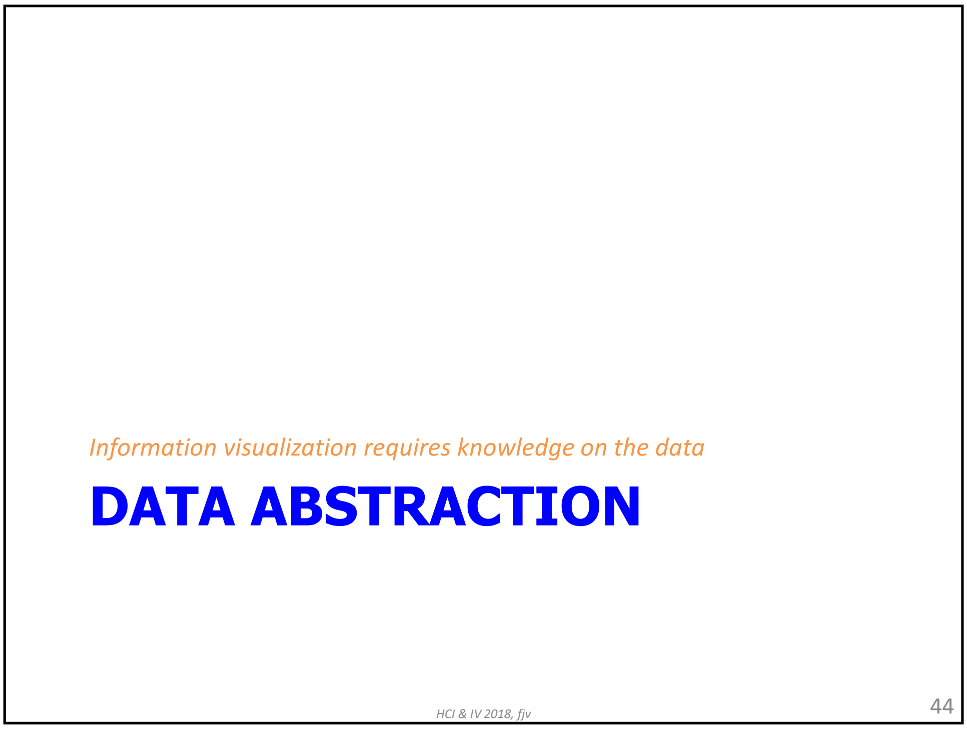
How do we build GUIs? (1)

1. Decide what we want to build (paper design)
2. Determine which events should be sent to which widgets
   - input
   - internal
   - etc.
3. Layout tools – wire-frame models.



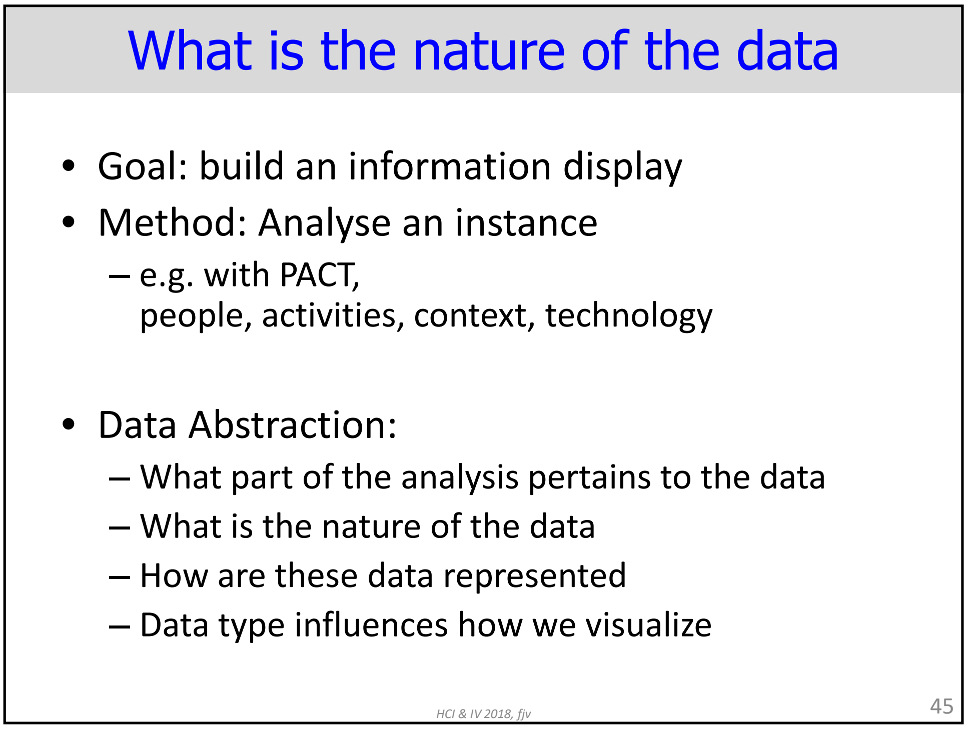
How do we build GUIs? (2)

4. Layout appropriate widgets using UI tool
5. For each widget add (missing) event handlers
   - “event listeners” (Java), “slots” (Qt)
6. Connect event senders to event receivers
   - Upon necessity
7. Add your code
   - The “product” functionality
   - cf. Rules Schneiderman – Norman
   - MVC



Information visualization requires knowledge on the data

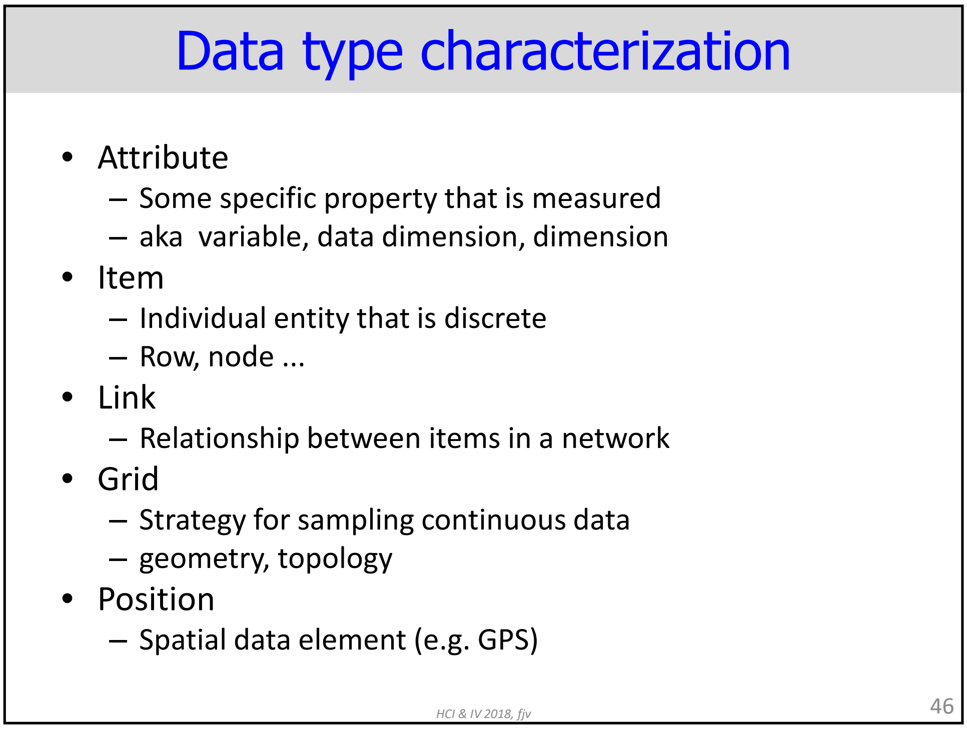
DATA ABSTRACTION



What is the nature of the data

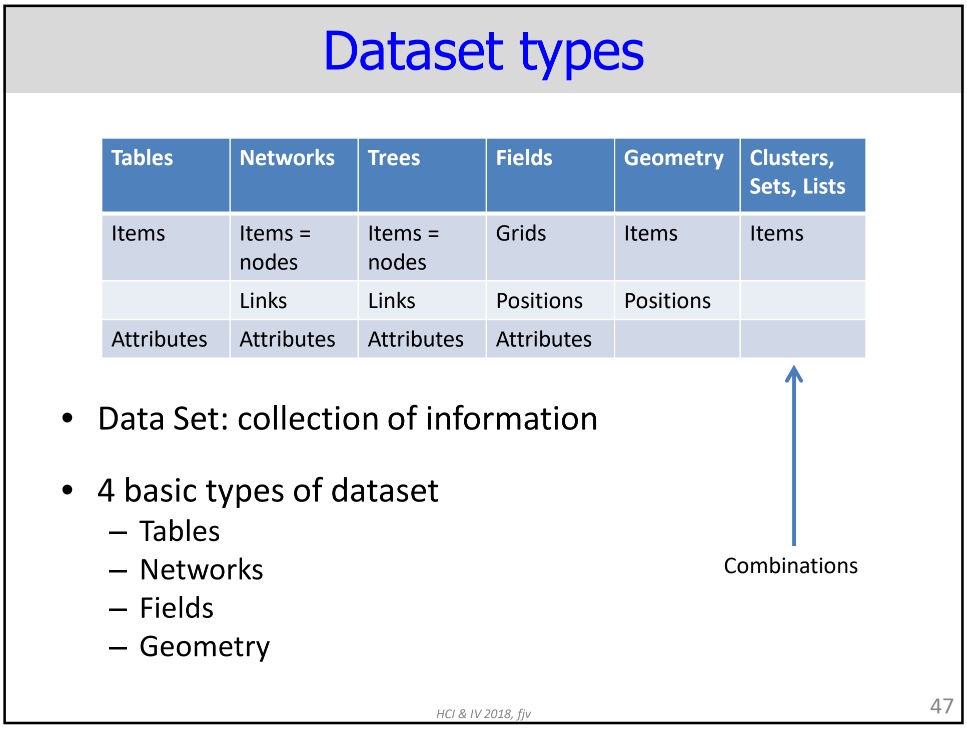
- Goal: build an information display
- Method: Analyse an instance
  - e.g. with PACT,
    people, activities, context, technology

- Data Abstraction:
  - What part of the analysis pertains to the data
  - What is the nature of the data
  - How are these data represented
  - Data type influences how we visualize



Data type characterization

- Attribute
  - Some specific property that is measured
  - aka variable, data dimension, dimension
- Item
  - Individual entity that is discrete
  - Row, node ...
- Link
  - Relationship between items in a network
- Grid
  - Strategy for sampling continuous data
  - geometry, topology
- Position
  - Spatial data element (e.g. GPS)



Dataset types

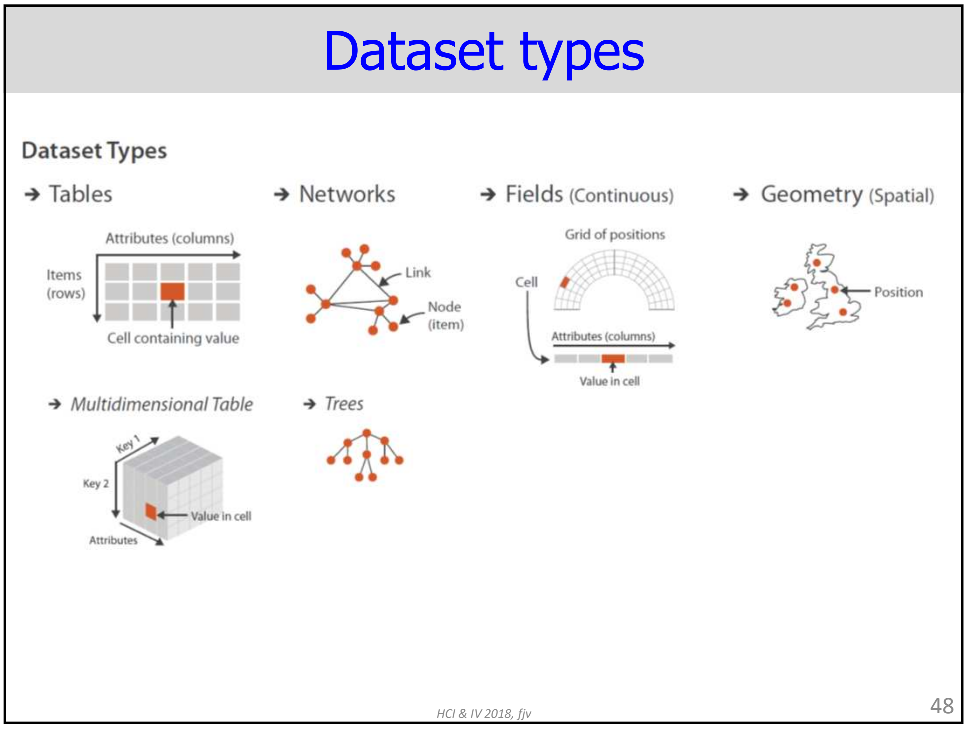
Tables	Networks	Trees	Fields	Geometry	Clusters, Sets, Lists
Items	Items = nodes	Items = nodes	Grids	Items	Items
	Links	Links	Positions	Positions	
Attributes	Attributes	Attributes	Attributes		

- Data Set: collection of information
- 4 basic types of dataset
  - Tables
  - Networks
  - Fields
  - Geometry

Combinations

HCI & IV 2018, fjv

47

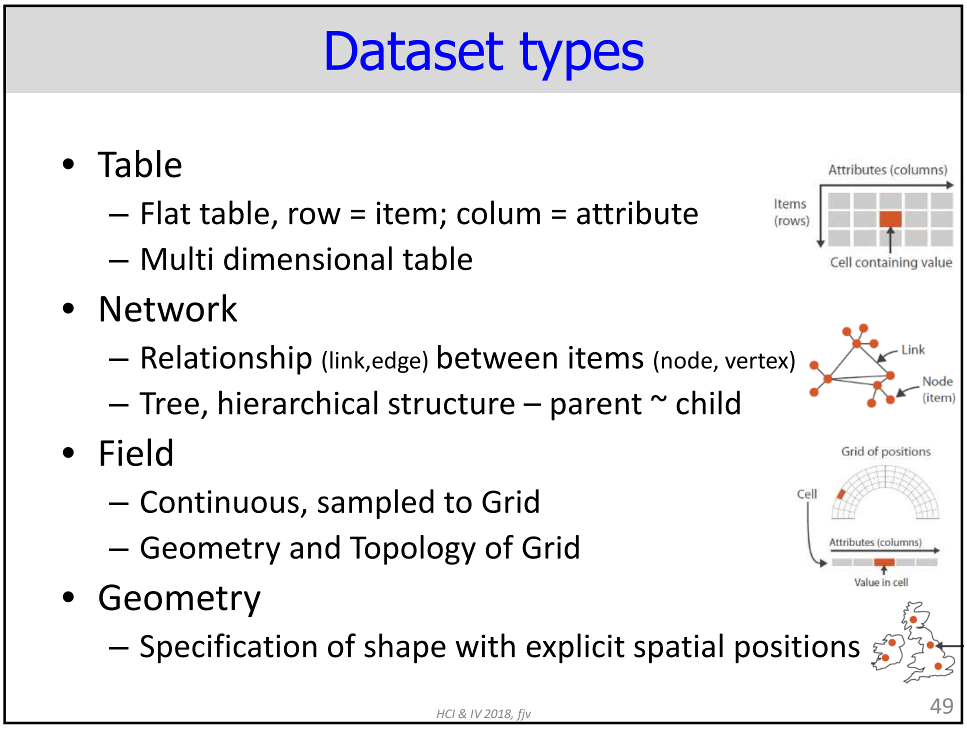


Dataset types

Dataset Types

HCI & IV 2018, fjv

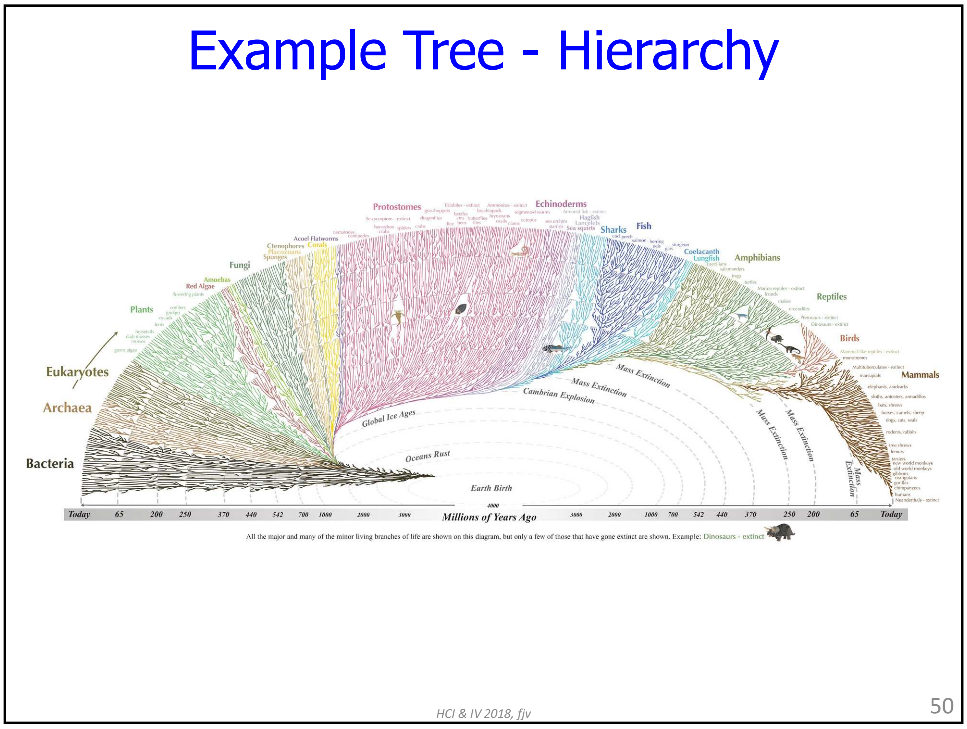
48



Dataset types

- Table
  - Flat table, row = item; colum = attribute
  - Multi dimensional table
- Network
  - Relationship (link,edge) between items (node, vertex)
  - Tree, hierarchical structure – parent ~ child
- Field
  - Continuous, sampled to Grid
  - Geometry and Topology of Grid
- Geometry
  - Specification of shape with explicit spatial positions

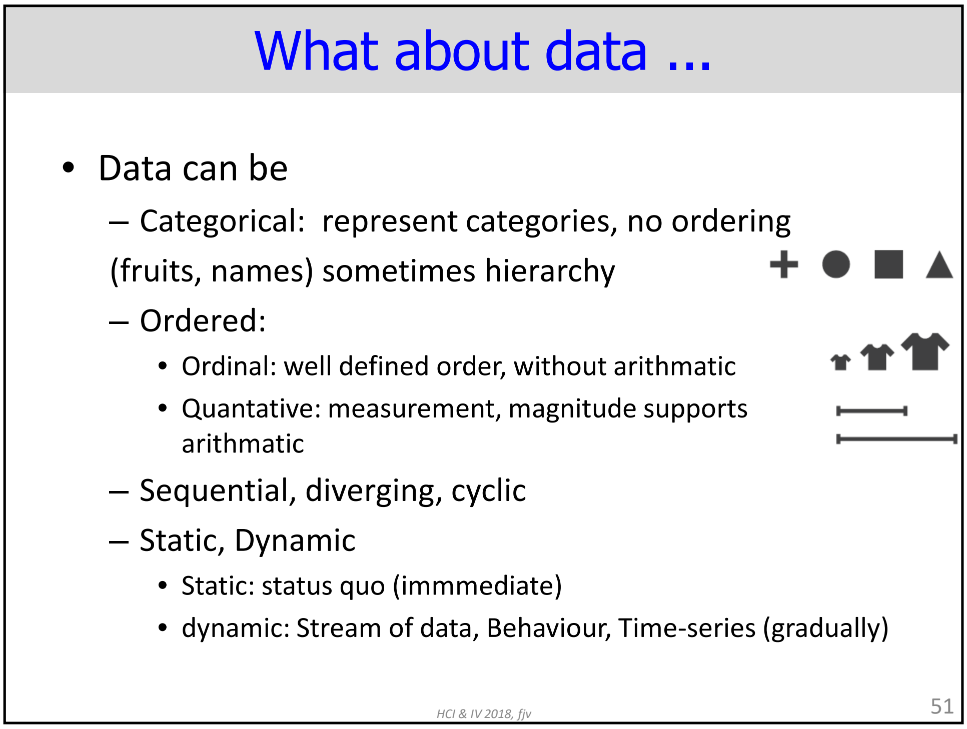
HCI & IV 2018, fjv



Example Tree - Hierarchy

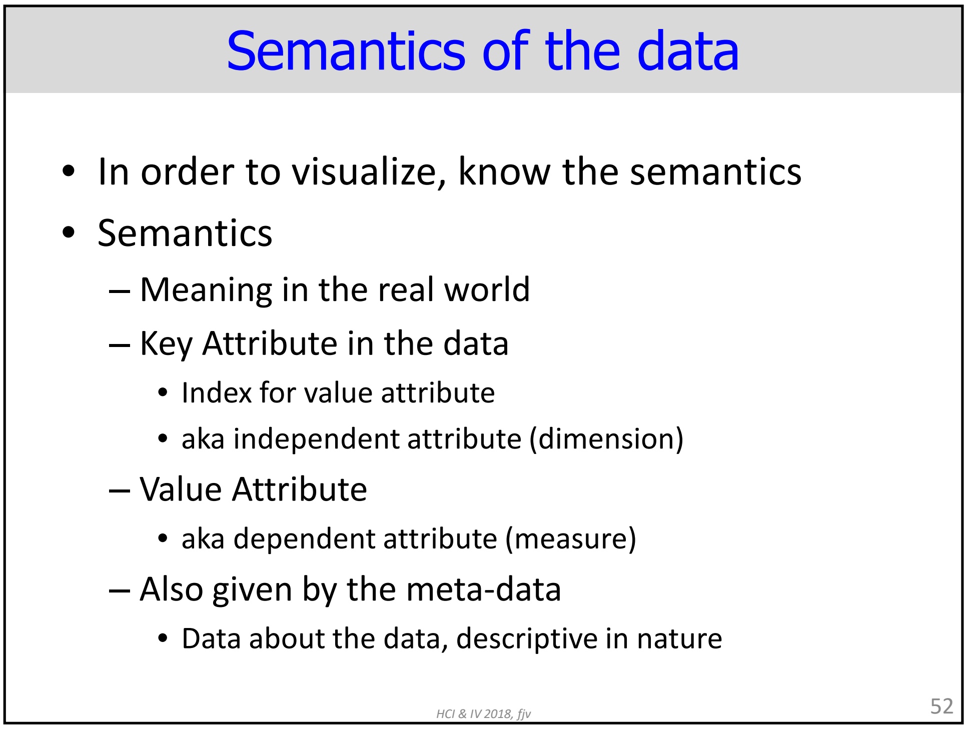
All the major and many of the minor living branches of life are shown on this diagram, but only a few of those that have gone extinct are shown. Example: Dinosaurs - extinct

HCI & IV 2018, fjv



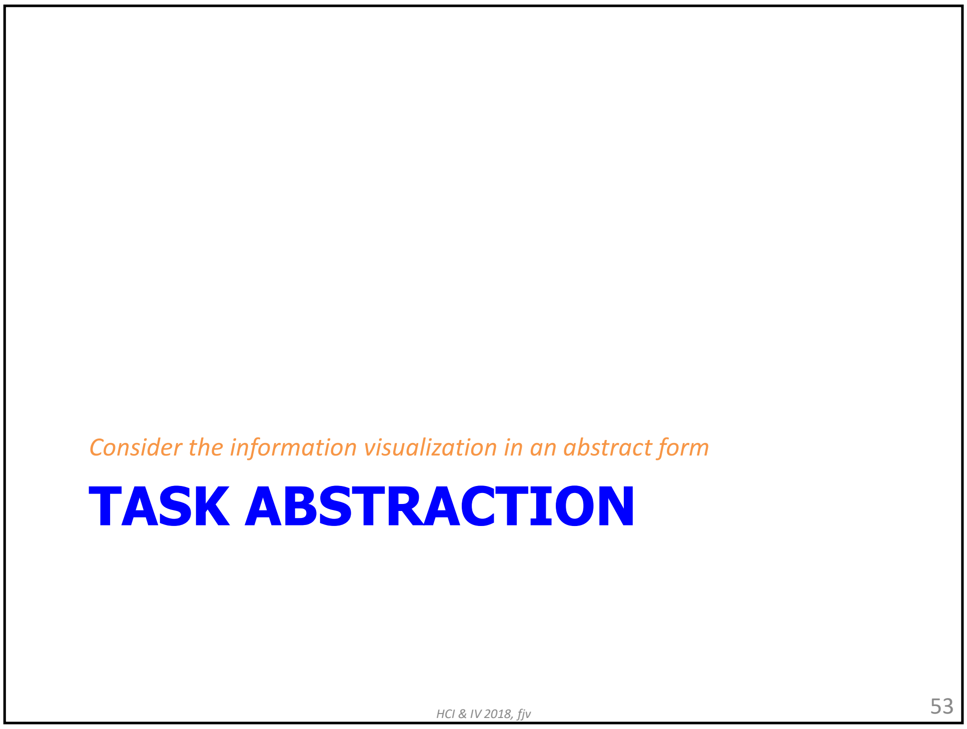
What about data …

- Data can be
  - Categorical: represent categories, no ordering (fruits, names) sometimes hierarchy + ● ■ ▲
  - Ordered:
    - Ordinal: well defined order, without arithmatic
    - Quantative: measurement, magnitude supports arithmatic
  - Sequential, diverging, cyclic
  - Static, Dynamic
    - Static: status quo (immmediate)
    - dynamic: Stream of data, Behaviour, Time-series (gradually)



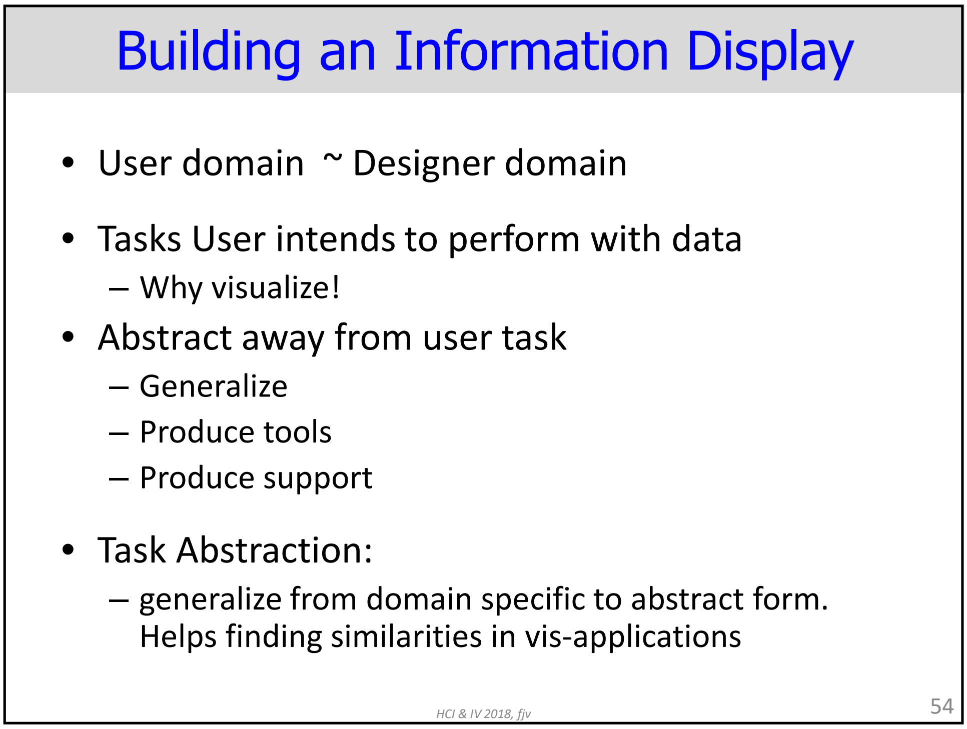
Semantics of the data

- In order to visualize, know the semantics
- Semantics
  - Meaning in the real world
  - Key Attribute in the data
    - Index for value attribute
    - aka independent attribute (dimension)
  - Value Attribute
    - aka dependent attribute (measure)
  - Also given by the meta-data
    - Data about the data, descriptive in nature



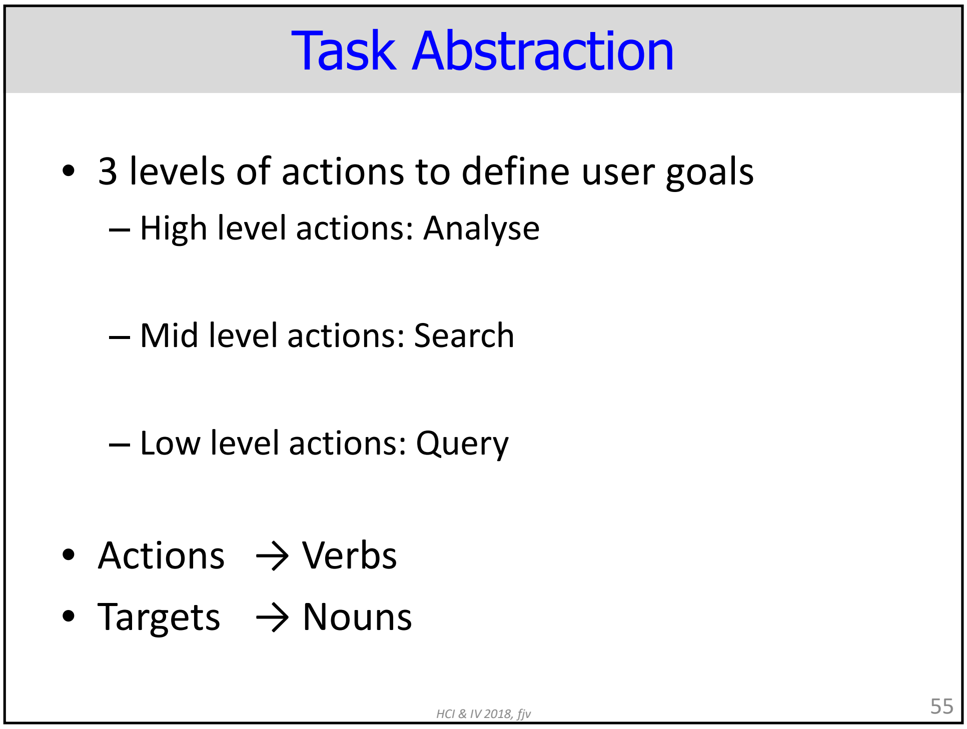
Consider the information visualization in an abstract form

TASK ABSTRACTION



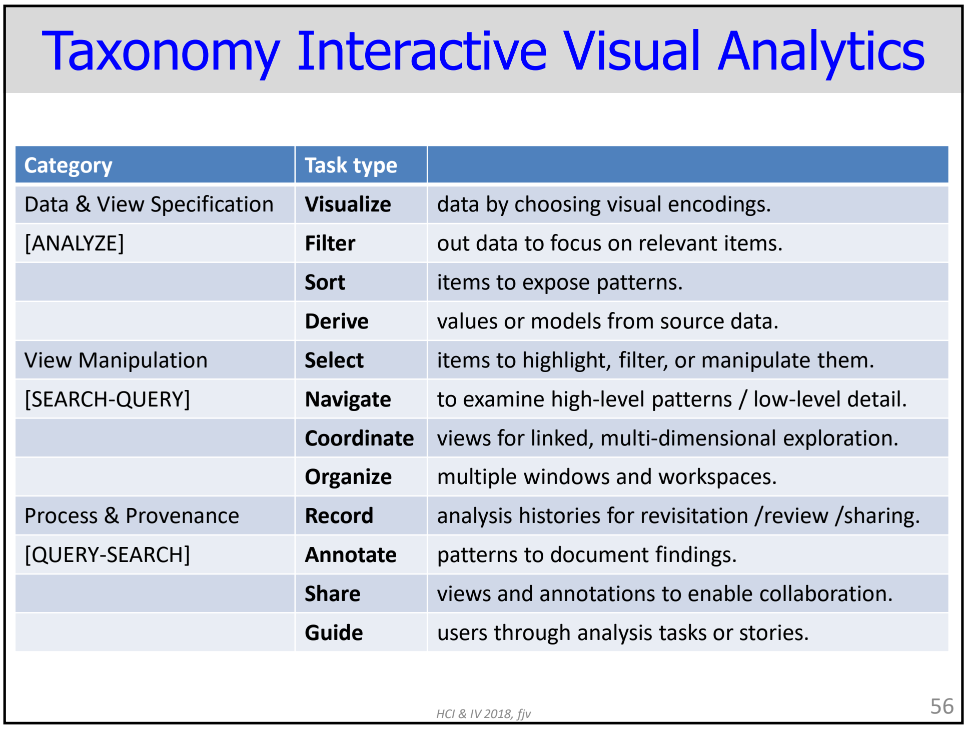
Building an Information Display

- User domain ~ Designer domain
- Tasks User intends to perform with data
  - Why visualize!
- Abstract away from user task
  - Generalize
  - Produce tools
  - Produce support
- Task Abstraction:
  - generalize from domain specific to abstract form. Helps finding similarities in vis-applications



Task Abstraction

- 3 levels of actions to define user goals
  - High level actions: Analyse
  - Mid level actions: Search
  - Low level actions: Query
- Actions → Verbs
- Targets → Nouns



Taxonomy Interactive Visual Analytics

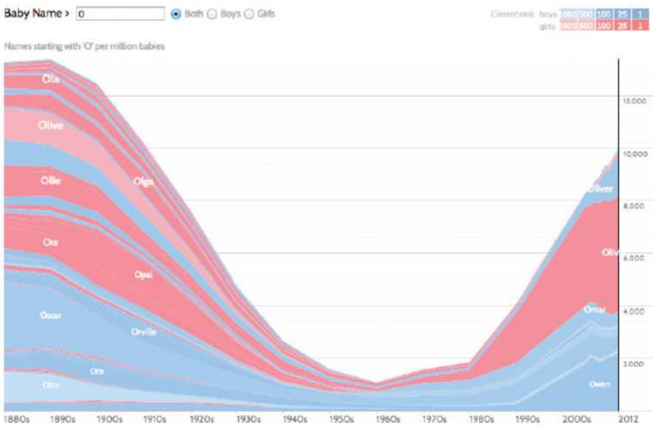
Category	Task type	
Data & View Specification	**Visualize**	data by choosing visual encodings.
[ANALYZE]	**Filter**	out data to focus on relevant items.
	Sort	items to expose patterns.
	Derive	values or models from source data.
View Manipulation	**Select**	items to highlight, filter, or manipulate them.
[SEARCH-QUERY]	**Navigate**	to examine high-level patterns / low-level detail.
	Coordinate	views for linked, multi-dimensional exploration.
	Organize	multiple windows and workspaces.
Process & Provenance	**Record**	analysis histories for revisitation /review /sharing.
[QUERY-SEARCH]	**Annotate**	patterns to document findings.
	Share	views and annotations to enable collaboration.
	Guide	users through analysis tasks or stories.

Analyze Abstraction, actions

- Consume information
  - Discover, use InfoVis to find new knowledge
    - Hypothesis testing and generation
  - Present
    - Communication, storytelling
  - Enjoy
    - Casual (playfull) encounters with Infovis

- Produce information
  - Annotate, augment InfoVis with annotation (domain)
  - Record, saves/captures InfoVis elements
    - e.g. For Dashboards, Infographics
  - Derive, produce new data elements
    - derived attributes

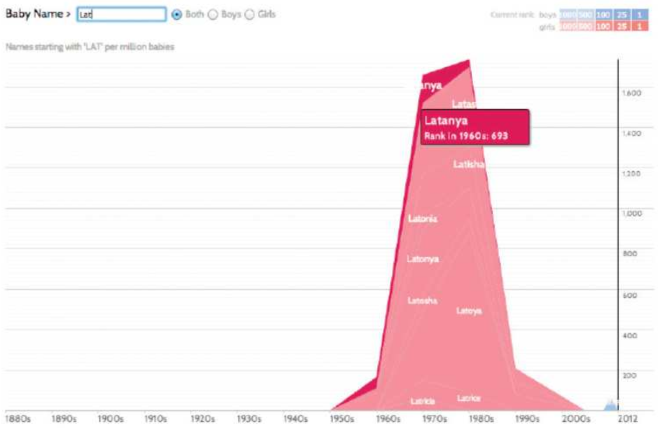
Example Enjoy, baby-names



Baby Name > O

● Both ○ Boys ○ Girls

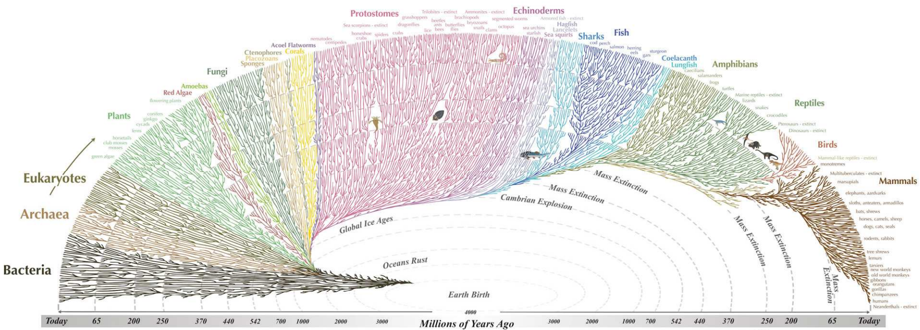
InfoVis Mantra, Focus & Zoom



Search Abstraction, actions

- Look-Up
  - Know where to look for
  - Using InfoVis as knowledge table
- Locate
  - Known target, at unknown location
  - Helps in understanding relations
- Browse
  - Unknown target, location is group of attributes
- Explore
  - Characteristics from parts of InfoVis
  - Typically Focus and Zoom, start from Overview

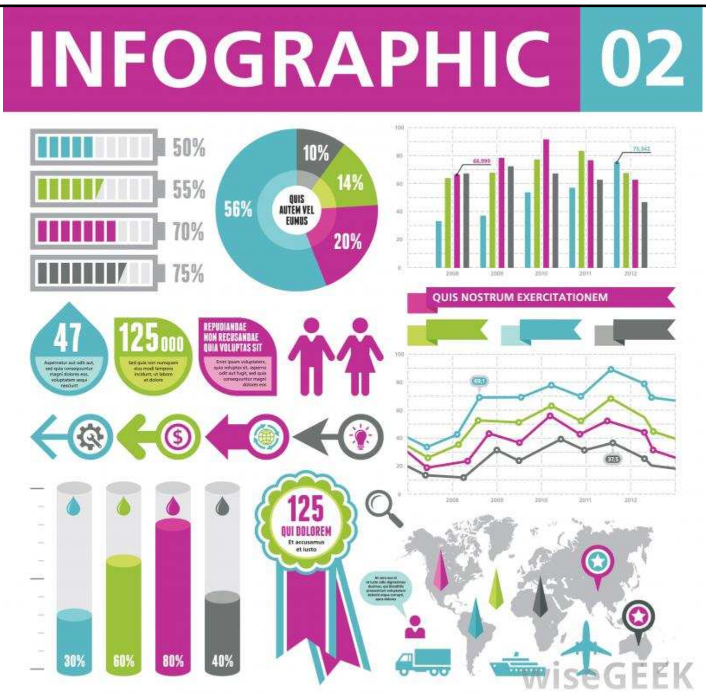
Example Tree - Hierarchy



All the major and many of the minor living branches of life are shown on this diagram, but only a few of those that have gone extinct are shown. Example: Dinosaurs - extinct

Query Abstraction, actions

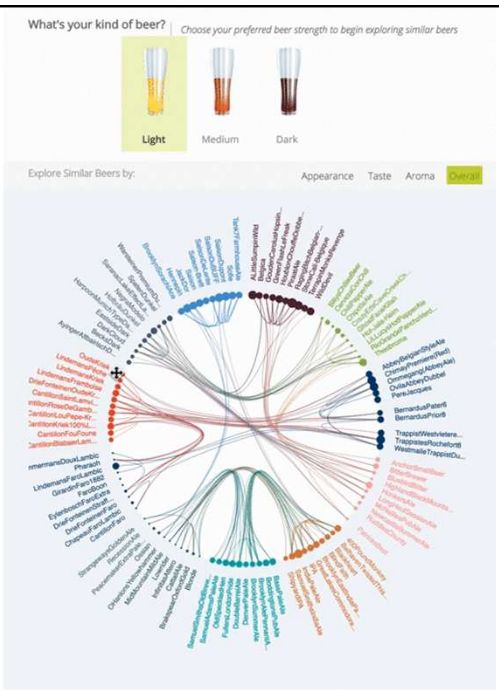
- Identify
  - Query on a single target
  - Chararteristics identified from result
- Compare
  - Query on multiple targets
  - Possibillity to identify relations
- Summarize
  - Query all possible targets
  - Produce an overview
  - Starting point for exploration or comparison

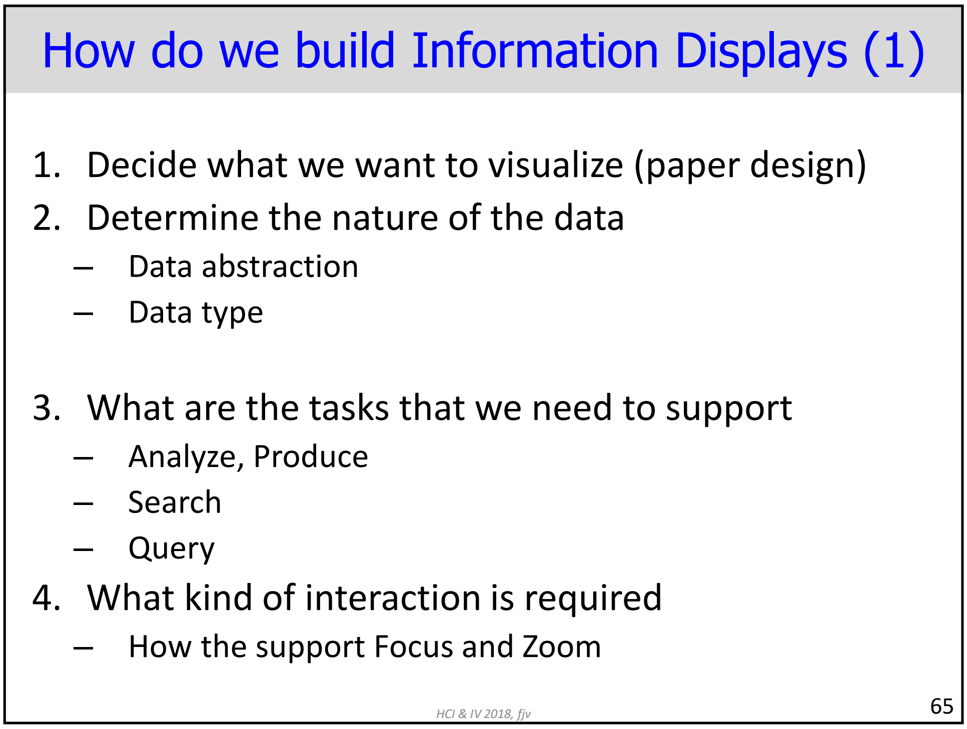


- Beer recommender system

- Choose a general style
- Show similar beers

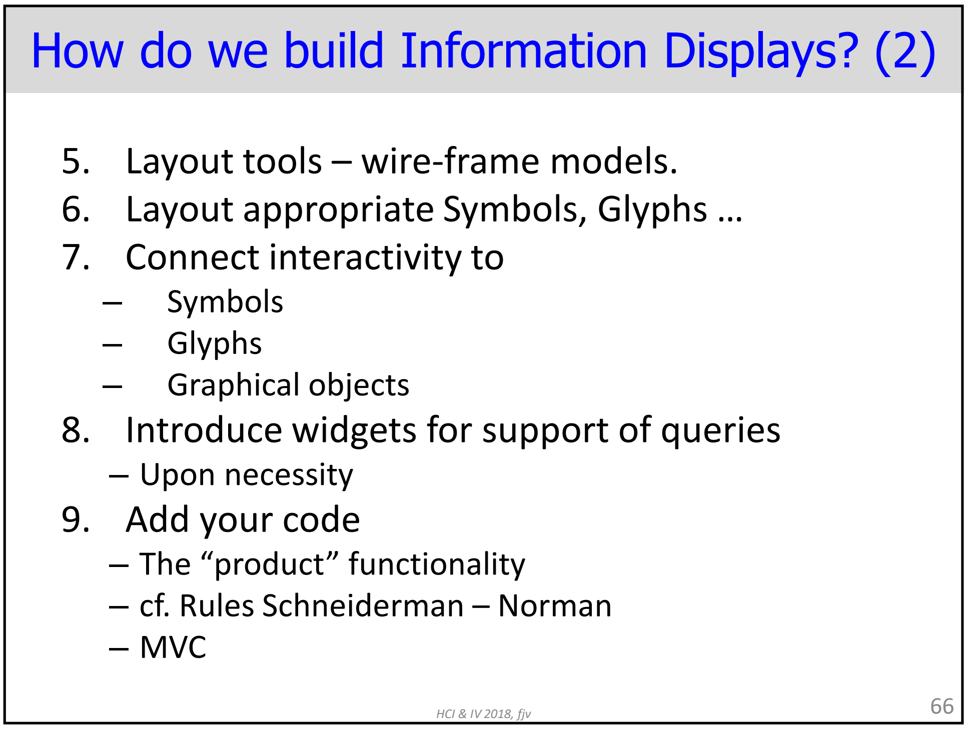
- No focus/zoom?


What's your kind of beer?
Choose your preferred beer strength to begin exploring similar beers
Light
Medium
Dark
Explore Similar Beers by:
Appearance
Taste
Aroma
Overall



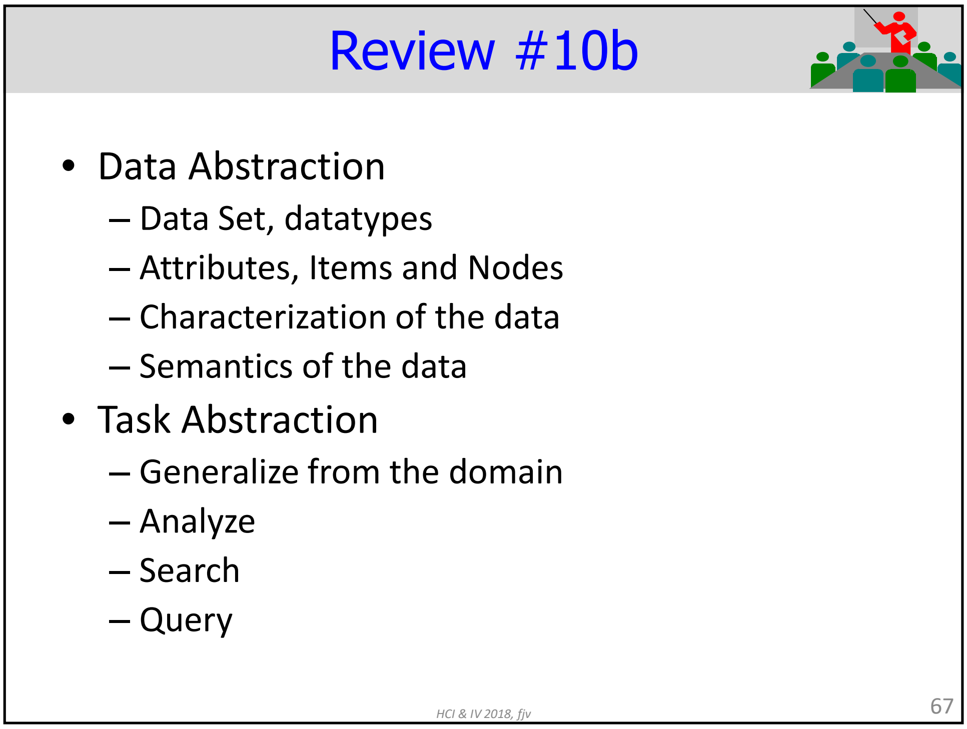
How do we build Information Displays (1)

1. Decide what we want to visualize (paper design)
2. Determine the nature of the data
   - Data abstraction
   - Data type
3. What are the tasks that we need to support
   - Analyze, Produce
   - Search
   - Query
4. What kind of interaction is required
   - How the support Focus and Zoom



How do we build Information Displays? (2)

5. Layout tools – wire-frame models.
6. Layout appropriate Symbols, Glyphs …
7. Connect interactivity to
   - Symbols
   - Glyphs
   - Graphical objects
8. Introduce widgets for support of queries
   - Upon necessity
9. Add your code
   - The “product” functionality
   - cf. Rules Schneiderman – Norman
   - MVC



Review #10b

- Data Abstraction
  - – Data Set, datatypes
  - – Attributes, Items and Nodes
  - – Characterization of the data
  - – Semantics of the data
- Task Abstraction
  - – Generalize from the domain
  - – Analyze
  - – Search
  - – Query