Opening the Black Box of Interaction in Visualization

Hans-Jörg Schulz[1], Tatiana v. Landesberger[2], Dominikus Baur[3]

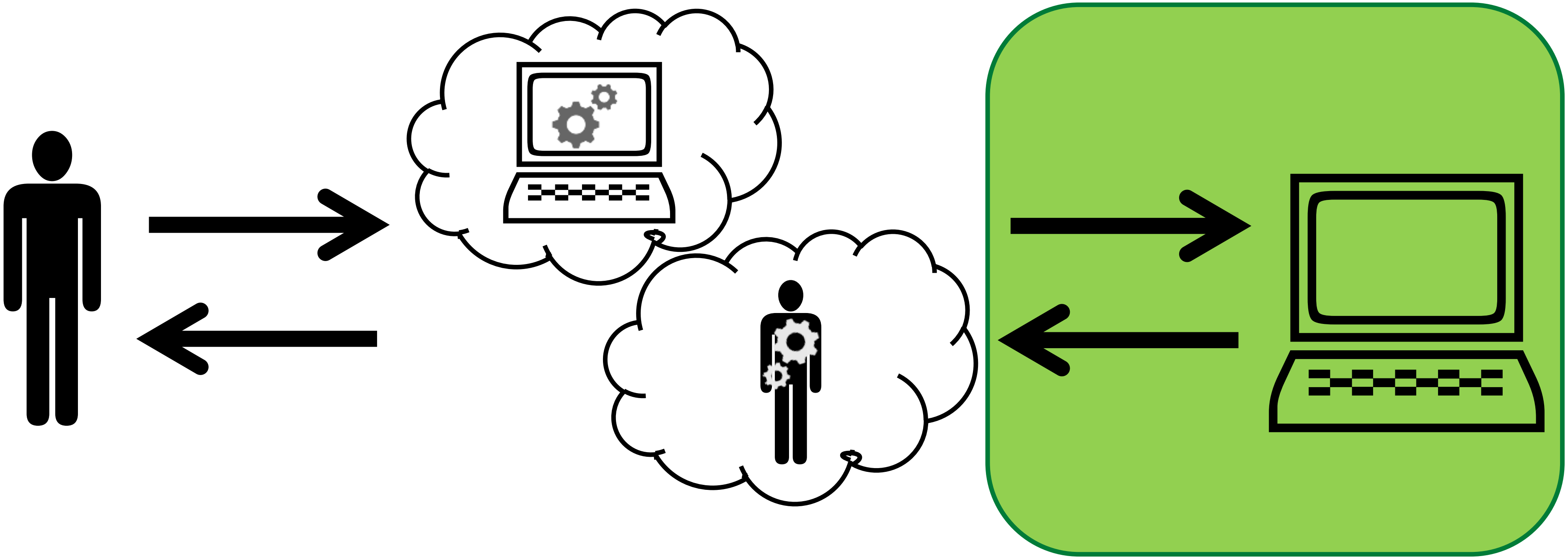
VIS Tutorial 2014

1. Fraunhofer IGD, Rostock, Germany
2. TU Darmstadt, Darmstadt, Germany
3. Dominikus Baur Interfacery

PART II: INTERACTION ARCHITECTURE

Speaker: Hans-Jörg Schulz

Interaction Architecture



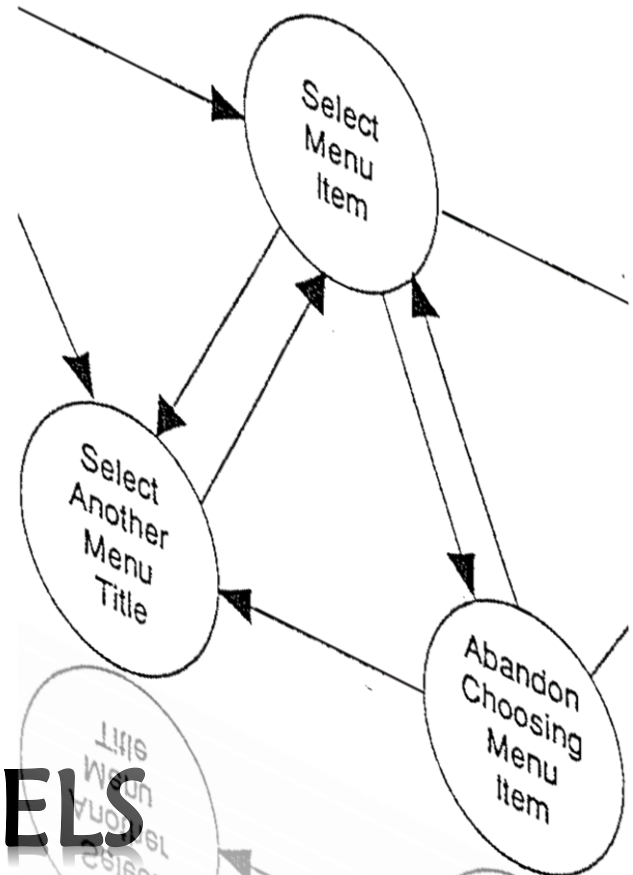
Interaction Architecture

What happens between the computer receives an input and sends off an output?

1. Designer's Perspective
   – Interaction Models (UML, UAN,…)
2. User's Perspective
   – History and Interruption Management
3. Software Engineer's Perspective
   – Patterns for Interactive Software (MVC, DCI,…)

Designer’s Perspective

INTERACTION MODELS



Interaction Models

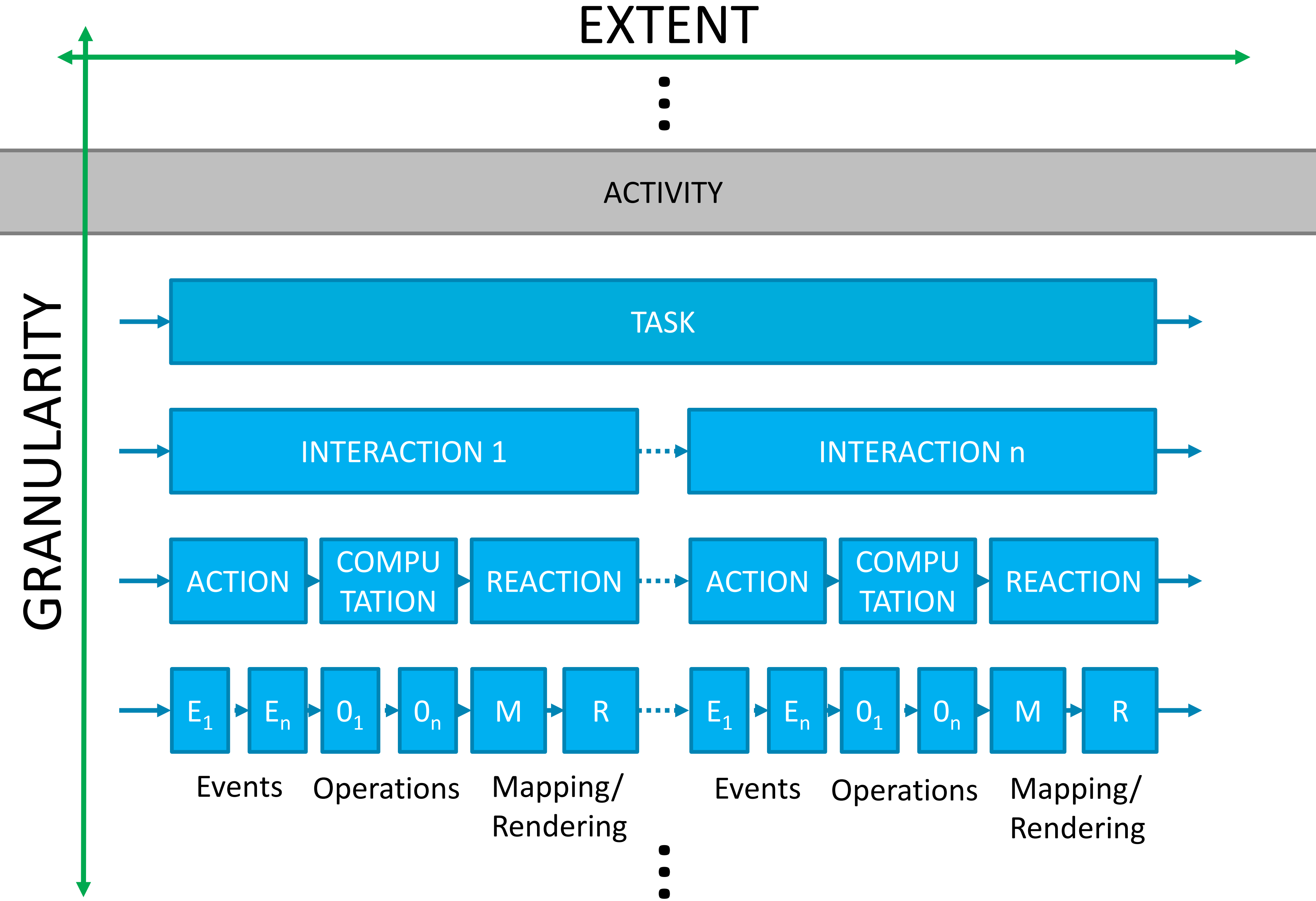
Why do we need to model interaction?

- to capture requirements (*when the user does this, then the system should do that*)
  - to develop by
  - to evaluate against
- to build workflows for passing on interaction knowledge and providing user guidance
- to automatically generate UIs

Types of Interaction Models

What is modeled?

- Extent: Single Action or whole Workflow
- Granularity: Concrete Events or General Task


EXTENT
ACTIVITY
GRANULARITY
TASK
INTERACTION 1
INTERACTION n
ACTION
COMPU TATION
REACTION
ACTION
COMPU TATION
REACTION
E_1
E_n
O_1
O_n
M
R
E_1
E_n
O_1
O_n
M
R
Events
Operations
Mapping/ Rendering
Events
Operations
Mapping/ Rendering

Types of Interaction Models

What is modeled?

- Extent: Single Action or whole Workflow
- Granularity: Concrete Events or General Task

How is it modeled?

- Diagrammatically: Sequence/Activity Diagram
- Symbolically: Notations

Diagrammatic Models (UML)

Activity Diagrams

- State/Transition Diagram
- Models the logical flow (of interaction)
- Focus on which activities can be performed in which order and under which constraints […]
- Overview character -> shows the whole flow

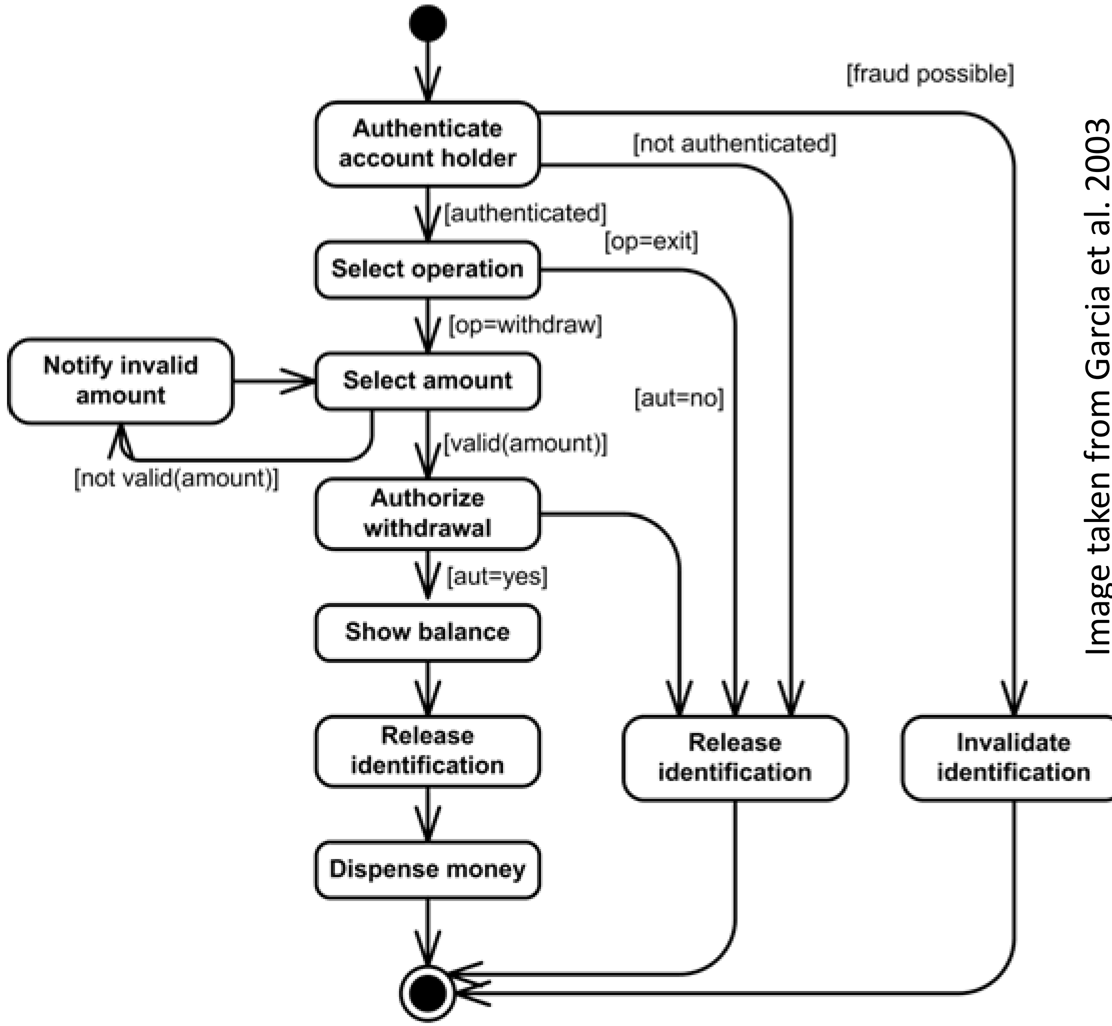


Image taken from Garcia et al. 2003

Diagrammatic Models (UML)

Sequence Diagrams

- Lifeline Diagram
- Models interaction between components as event sequences
- Each sequence can be seen as a path through the activity diagram
- Detail character -> shows one flow

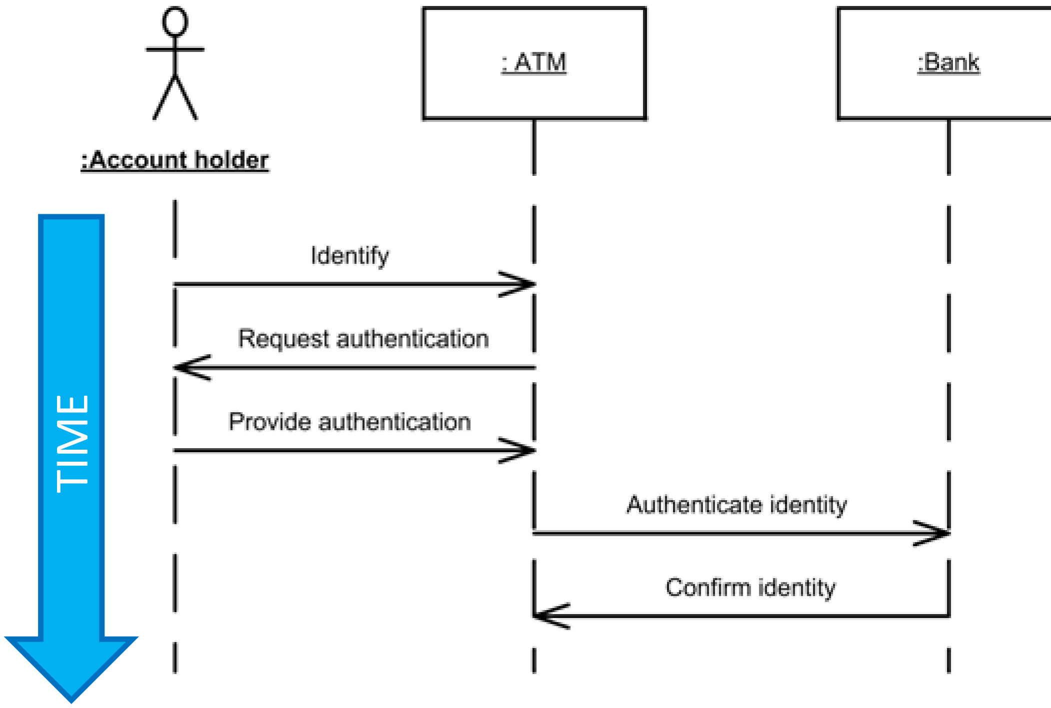


Image taken from Garcia et al. 2003

Diagrammatic Models (UML)

Interaction Overview Diagrams

- Nested/Compounded Diagram
- Combines Activity and Sequence Diagrams by embedding interaction for each activity
- Compartmentalization of Sequence Diagram
- Overview & Detail character combined -> somewhat crowded

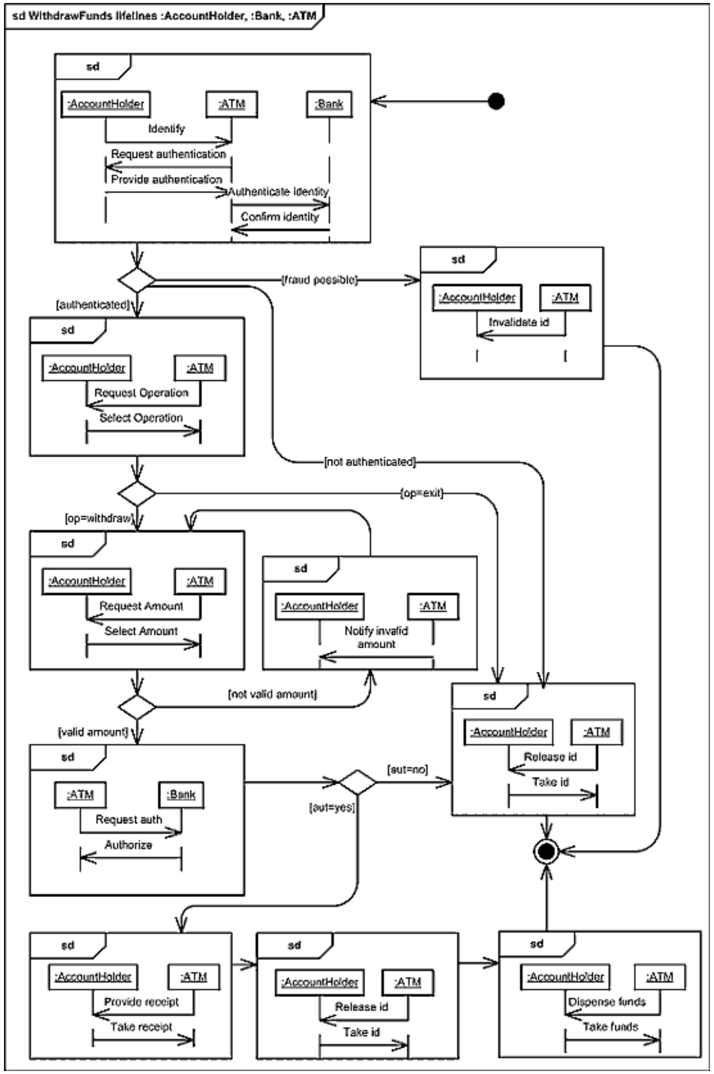


Image taken from Garcia et al. 2003

Symbolic Models

Notations (Task Design Space)

Design Dimensions

- WHY? **GOAL**
- HOW?
- WHAT?
- WHERE?

Design Choices

- Exploratory Analysis
  hypothesis generation through undirected search
- Confirmatory Analysis
  hypothesis testing through directed search
- Presentation
  communication of confirmed analysis results

adapted from Schulz et al. 2013

Symbolic Models

Notations (Task Design Space)

Design Dimensions

- WHY? **GOAL**
- HOW? **MEANS**
- WHAT?
- WHERE?

Design Choices

- Navigation
  changes the scope or granularity of the data
- (Re-)organization
  adjusts the data by reducing or enriching it
- Relation
  puts data in context by seeking similarities or differences

adapted from Schulz et al. 2013

Symbolic Models

Notations (Task Design Space)

Design Dimensions

- WHY? **GOAL**
- HOW? **MEANS**
- WHAT? **CHARACTERISTICS**
- WHERE?

- Low-level Characteristics
  observations about data objects and data values
  → *visual literacy*

- High-level Characteristics
  complex patterns in the data
  → *visual analysis*

Design Choices

adapted from Schulz et al. 2013

Symbolic Models

Notations (Task Design Space)

Design Dimensions

- WHY? **GOAL**
- HOW? **MEANS**
- WHAT? **CHARACTERISTICS**
- WHERE? **TARGET**

Design Choices

- Attribute Relations
  linking data objects to their attribute values – in particular:
  - Temporal Relations
  - Spatial Relations
- Structural Relations
  linking data objects with each other

adapted from Schulz et al. 2013

Symbolic Models

Notations (Task Design Space)

Design Dimensions

- WHY? **GOAL**
- HOW? **MEANS**
- WHAT? **CHARACTERISTICS**
- WHERE? **TARGET**

CARDINALITY

Design Choices

- Single Instance
  for highlighting details
- Multiple Instances
  for showing data in context
- All Instances
  for getting a complete overview

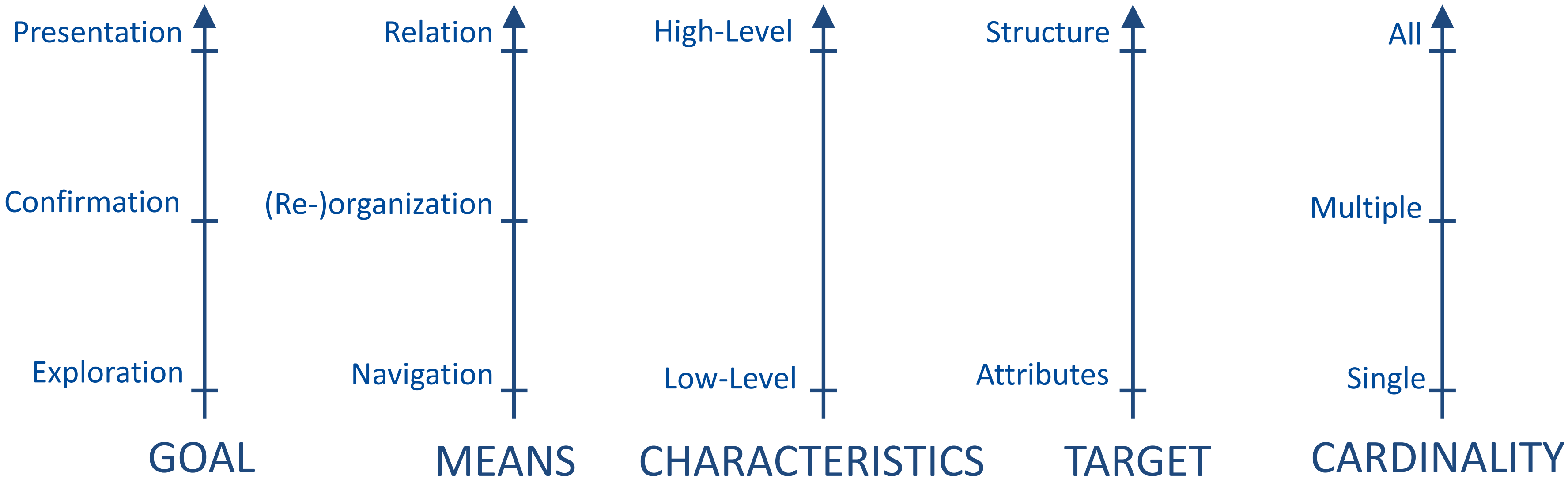
adapted from Schulz et al. 2013

Symbolic Models

Notations (Task Design Space)

Example: undirected search for a trend among all available temperature attribute values

(exploration, navigation(search), high-level(trend), attribute(temperature), all)



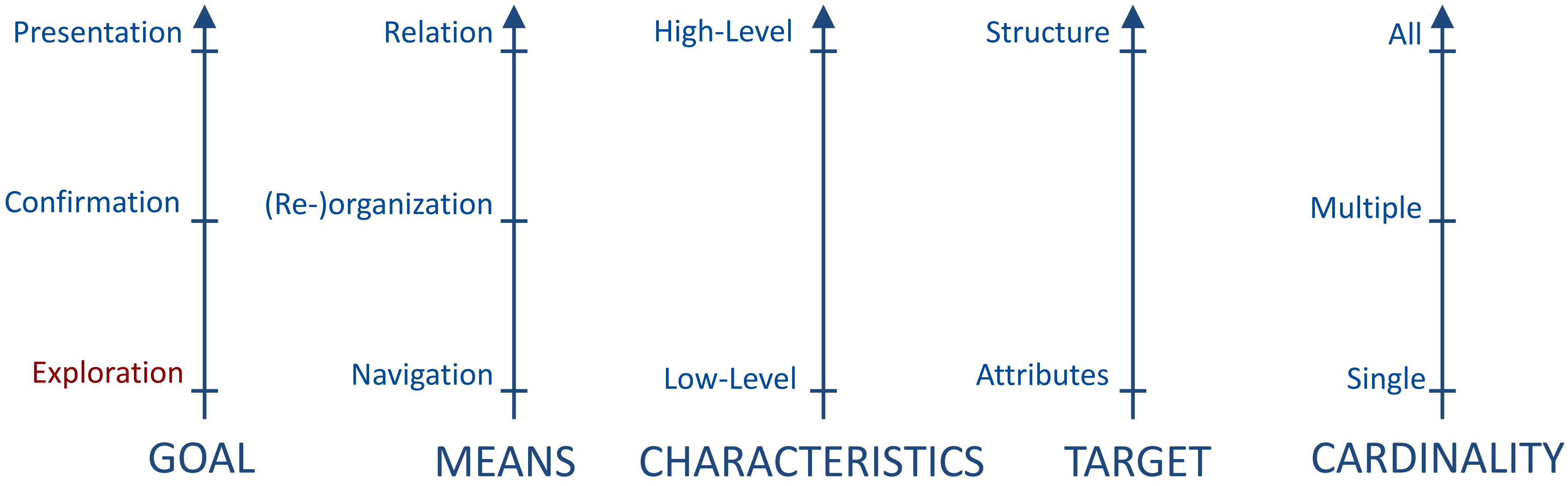
adapted from Schulz et al. 2013

Symbolic Models

Notations (Task Design Space)

Example: undirected search for a trend among all available temperature attribute values

(exploration, navigation(search), high-level(trend), attribute(temperature), all)



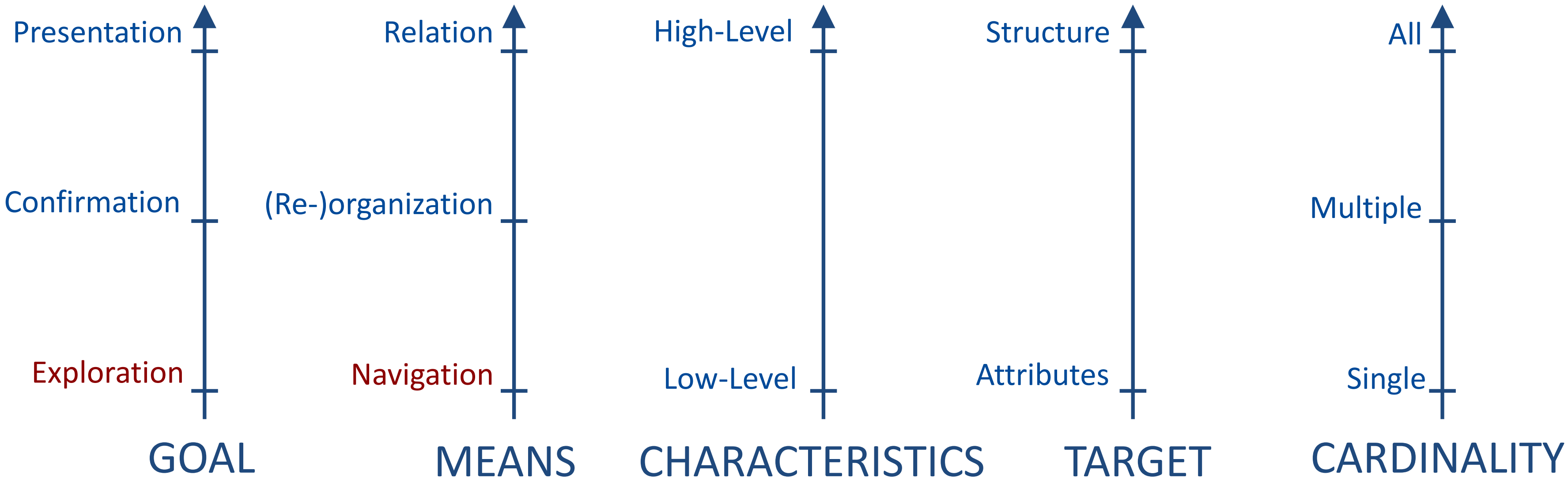
adapted from Schulz et al. 2013

Symbolic Models

Notations (Task Design Space)

Example: undirected search for a trend among all available temperature attribute values

(exploration, navigation(search), high-level(trend), attribute(temperature), all)



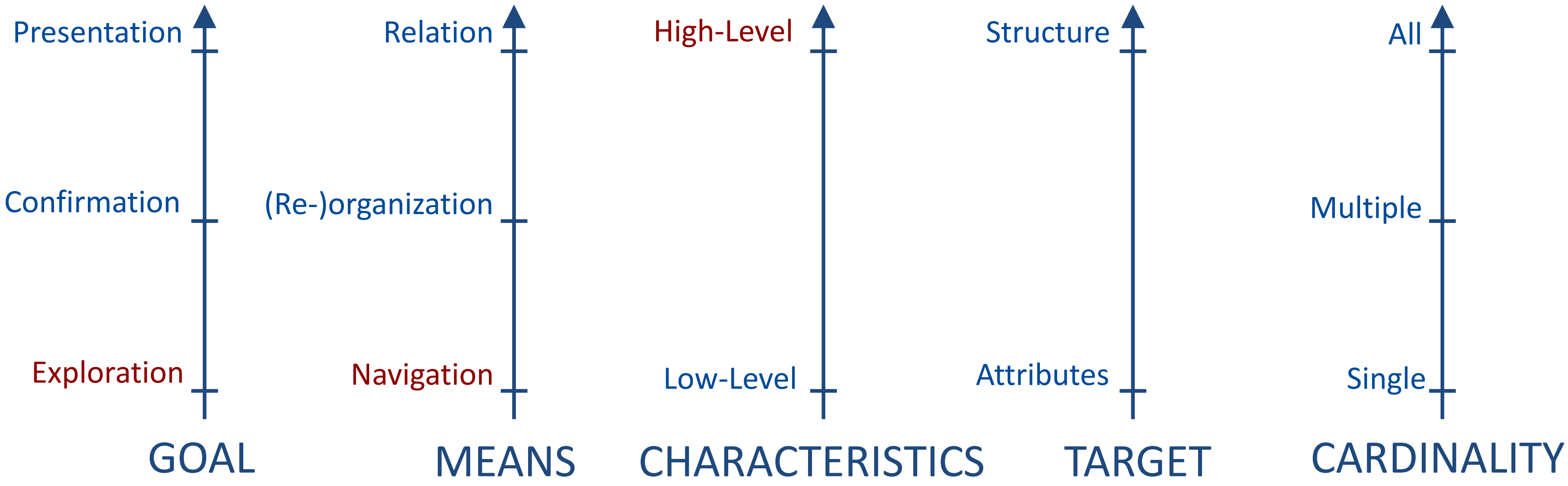
adapted from Schulz et al. 2013

Symbolic Models

Notations (Task Design Space)

Example: undirected search for a trend among all available temperature attribute values

(exploration, navigation(search), high-level(trend), attribute(temperature), all)



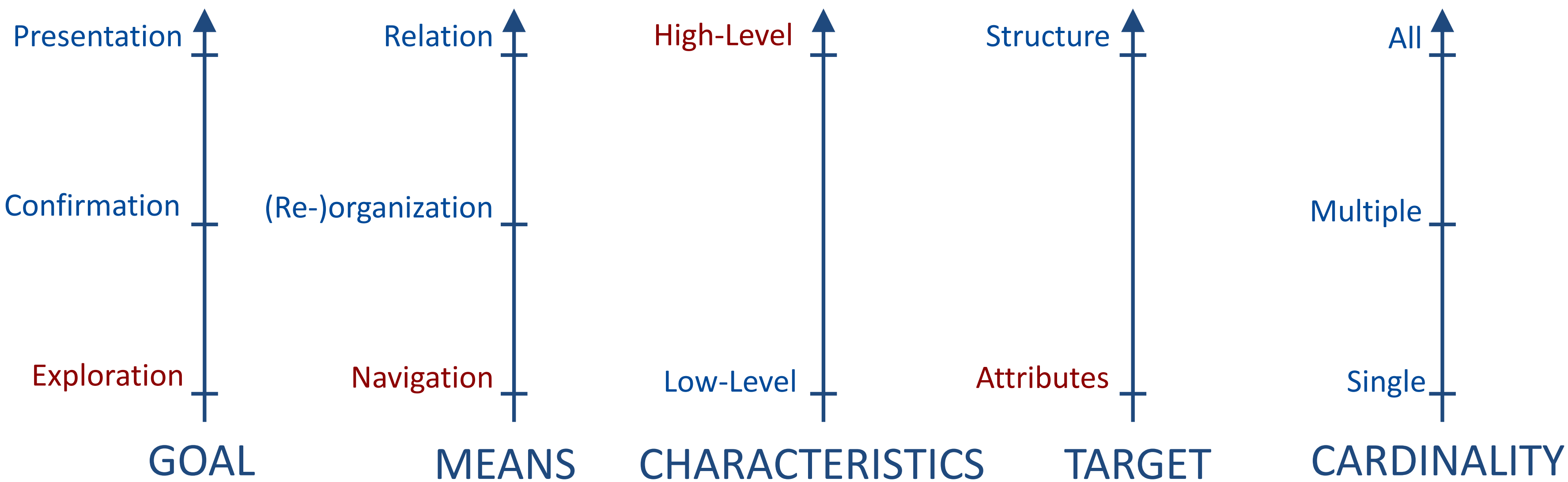
adapted from Schulz et al. 2013

Symbolic Models

Notations (Task Design Space)

Example: undirected search for a trend among all available temperature attribute values

(exploration, navigation(search), high-level(trend), attribute(temperature), all)



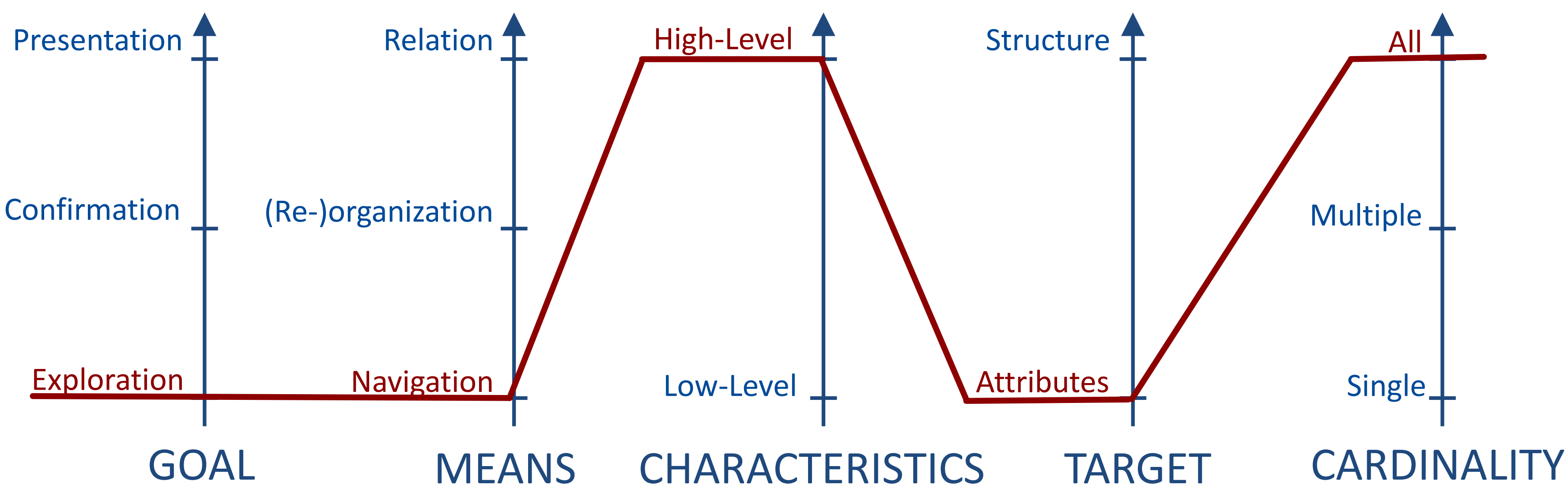
adapted from Schulz et al. 2013

Symbolic Models

Notations (Task Design Space)

Example: undirected search for a trend among all available temperature attribute values

(exploration, navigation(search), high-level(trend), attribute(temperature), all)



adapted from Schulz et al. 2013

Symbolic Models

Notations (Task Design Space) – Domain-Specific Examples

task	
compare variable distribution	$bute_1)\|attrib(attribute_2)$, *all)*
find model input/output relati	*ns,* $attrib(*_{input})\|attrib(*_{output})$, *all)*
gain overview of whole datas	
analyze trends	
visual (climate) model validat	$)\|attrib(*_{time})\|struct(*)$, *all)*
visual data cleansing / find da	*), all)*
analyze periodicities	*ll)*
analyze outliers	
compare measurements with	$)\|attrib(*_{simulation})$
present data for general audie	**), multiple)*

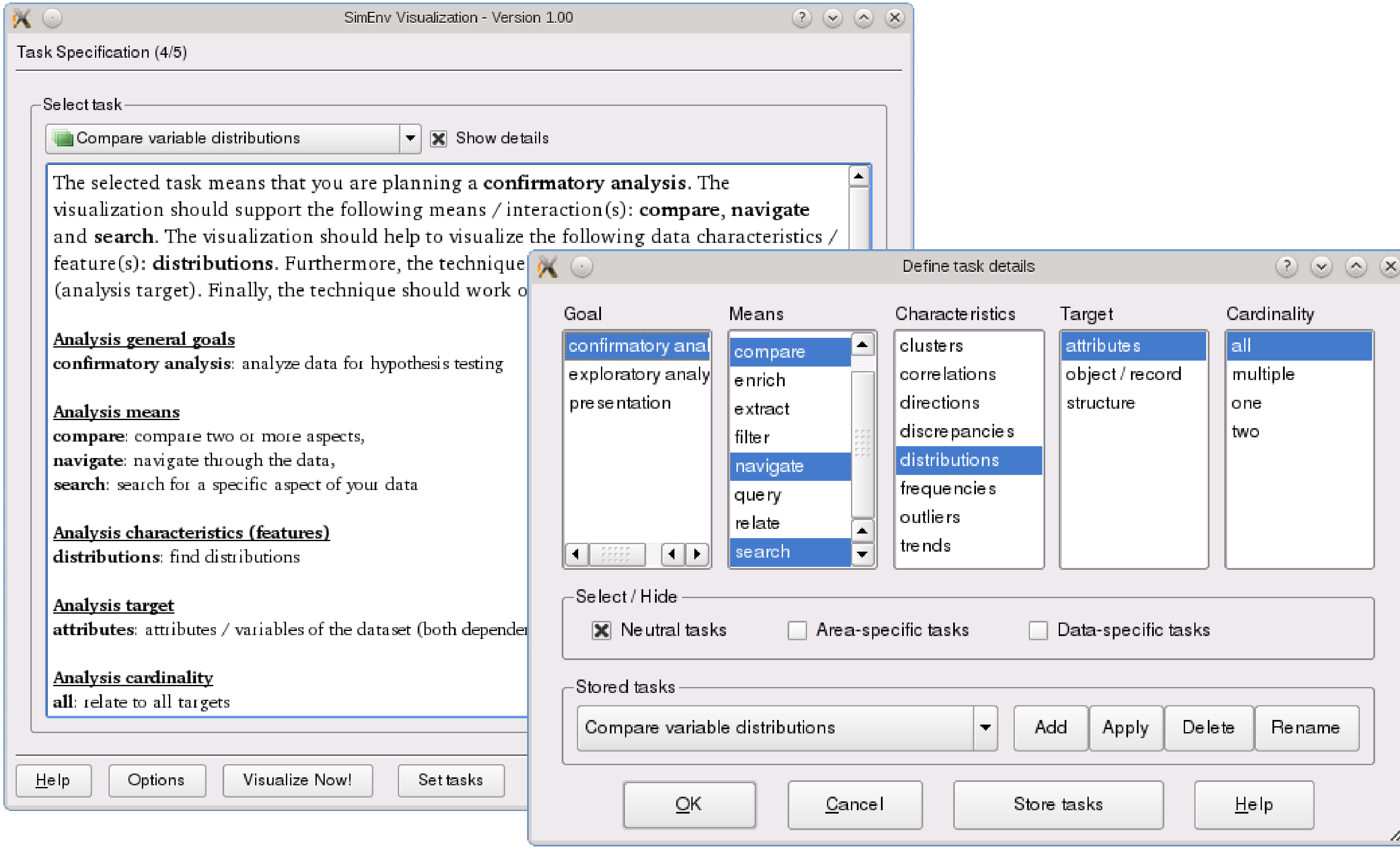
```
<TASK name = "find model input/output relations">
  <GOAL v0 = "exploratory analysis" v1 = "confirmatory analysis"/>
  <MEANS v0 = "relate" v1 = "search" v2 = "enrich"/>
  <CHARACTERISTICS v0 = "trends" v1 = "correlations"/>
  <TARGET v0 = "attributes(input)" v1 = "attributes(output)"/>
  <CARDINALITY v0 = "all"/>
</TASK>

<TASK name = "compare variable distributions">
  <GOAL v0 = "confirmatory analysis"/>
  <MEANS v0 = "search" v1 = "compare" v2 = "navigate"/>
  <CHARACTERISTICS v0 = "distributions"/>
  <TARGET v0 = "attributes(1)" v1 = "attributes(2)"/>
  <CARDINALITY v0 = "all"/>
</TASK>

<TASK name = "analyze trends">
  <GOAL v0 = "confirmatory analysis" v1 = "exploratory analysis"/>
  <MEANS v0 = "*"/>
  <CHARACTERISTICS v0 = "trends"/>
  <TARGET v0 = "attributes(*)" v1 = "attributes(time)"/>
  <CARDINALITY v0 = "all"/>
</TASK>
```

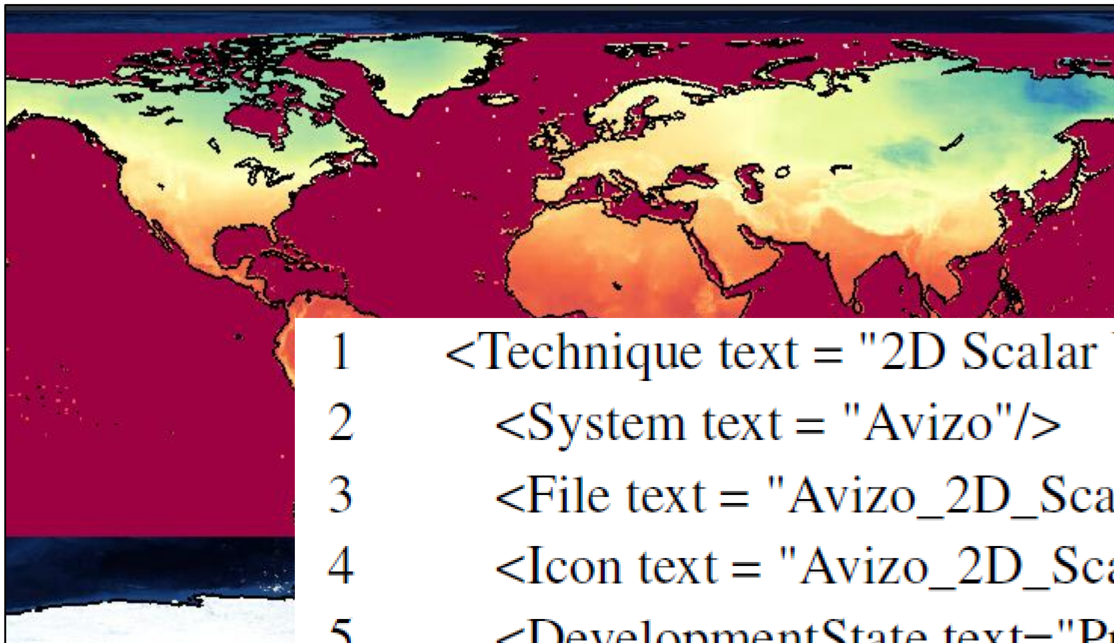
[Schulz et al. 2013]

Symbolic Models



[Schulz et al. 2013]

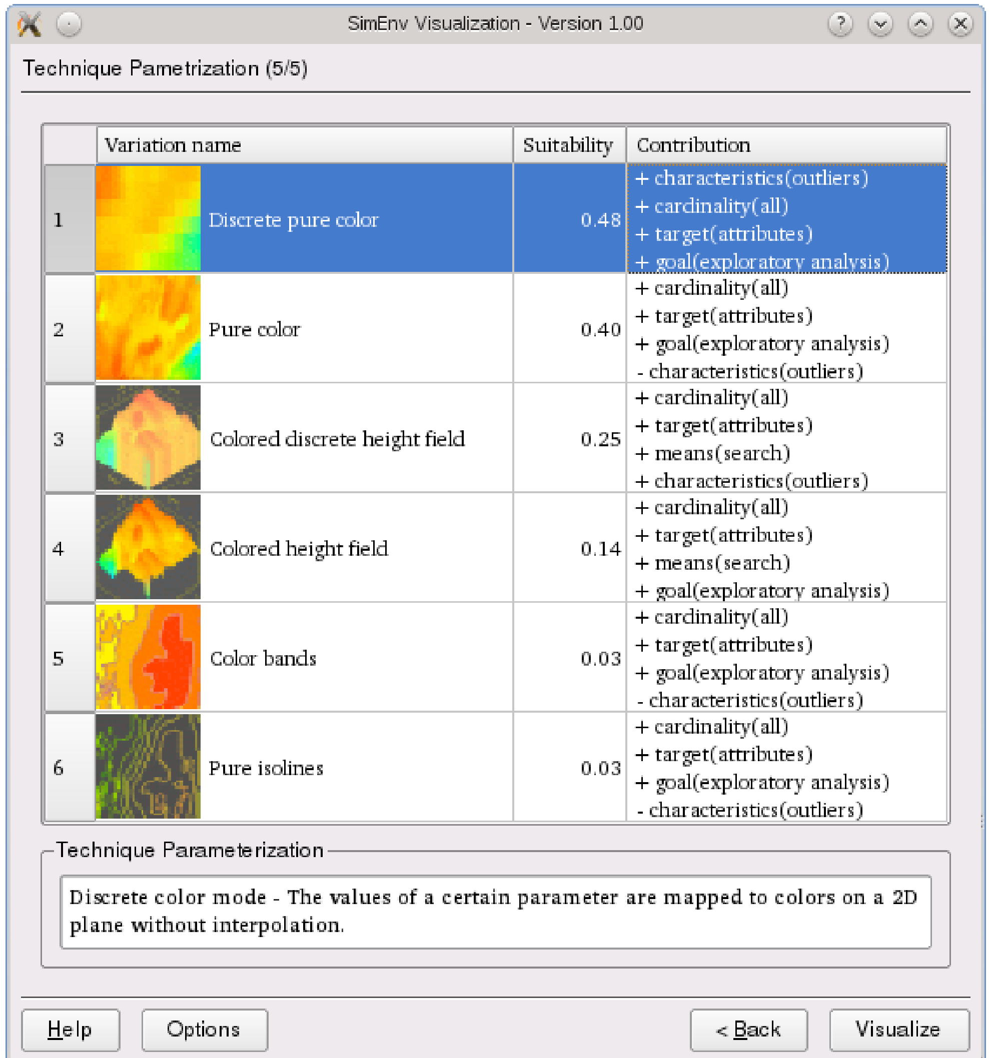
Symbolic Models



```
<Technique text = "2D Scalar Visualization (Avizo)">
   <System text = "Avizo"/>
   <File text = "Avizo_2D_Scalar_Visualization.hx"/>
   <Icon text = "Avizo_2D_Scalar_Visualization.png"/>
   <DevelopmentState text="PreRelease"/>
   <DimensionsDescriptor>
      <spatialDimensionality>
```

[Schulz et al. 2013]

Symbolic Models



[Schulz et al. 2013]

Symbolic Models

Notations (UAN – User Action Notation)

	ACTION		FEEDBACK
~	move cursor	!	highlight object
Xv	depress button "X"	-!	dehighlight object
X^	release button "X"	!-!	blink object
Xv^	click button "X"	$(!-!)^n$	blink object n times
( )	grouping of actions	X > ~	object follows cursor
*	performed 0 or more times	@x,y	at point x,y
+	performed 1 or more times	display(X)	show object X
{ }	enclosed action is optional	erase(X)	hide object X
¥	for all	outline(X)	outline object X

adapted from [Hartson et al. 1990]

And some more…

Symbolic Models

Notations (UAN – User Action Notation)

TASK: move a file icon			
USER ACTIONS	INTERFACE FEEDBACK	INTERFACE STATE	COMPUTATION
~[file icon] Mv	file_icon-!: file_icon! ¥file_icon‘!: file_icon‘-!	selected = file	
~[x,y]* ~[x‘,y‘]	outline(file_icon) > ~		
M^	@x‘,y‘ display(file_icon)		location(file_icon) = x‘,y‘

adapted from [Hartson et al. 1990]

Combining the What and the How

We can use these diagrams (and notations) on all levels of granularity!

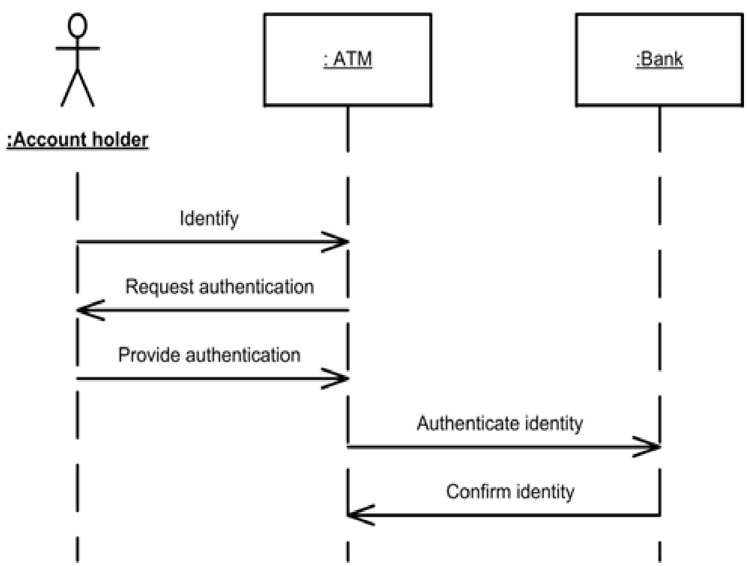


Image taken from Garcia et al. 2005

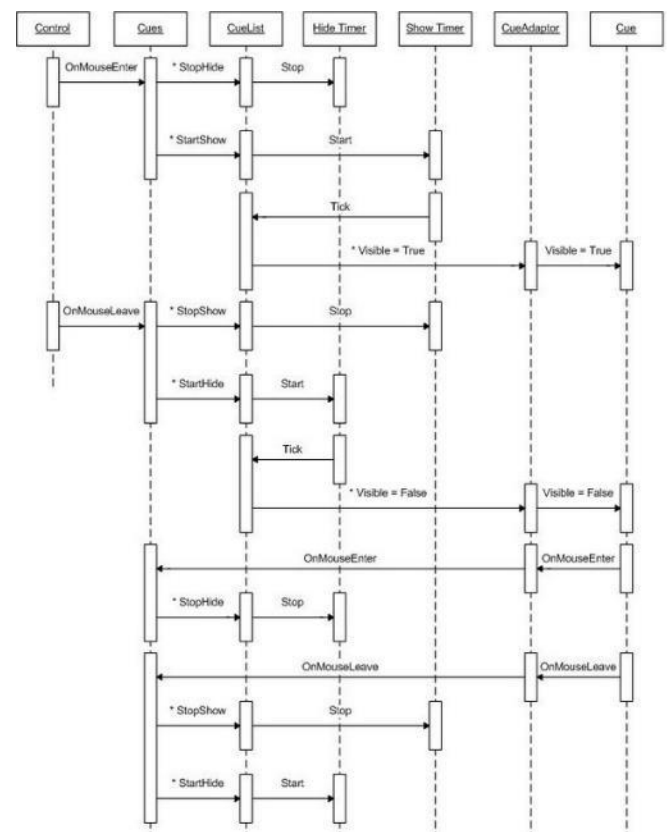
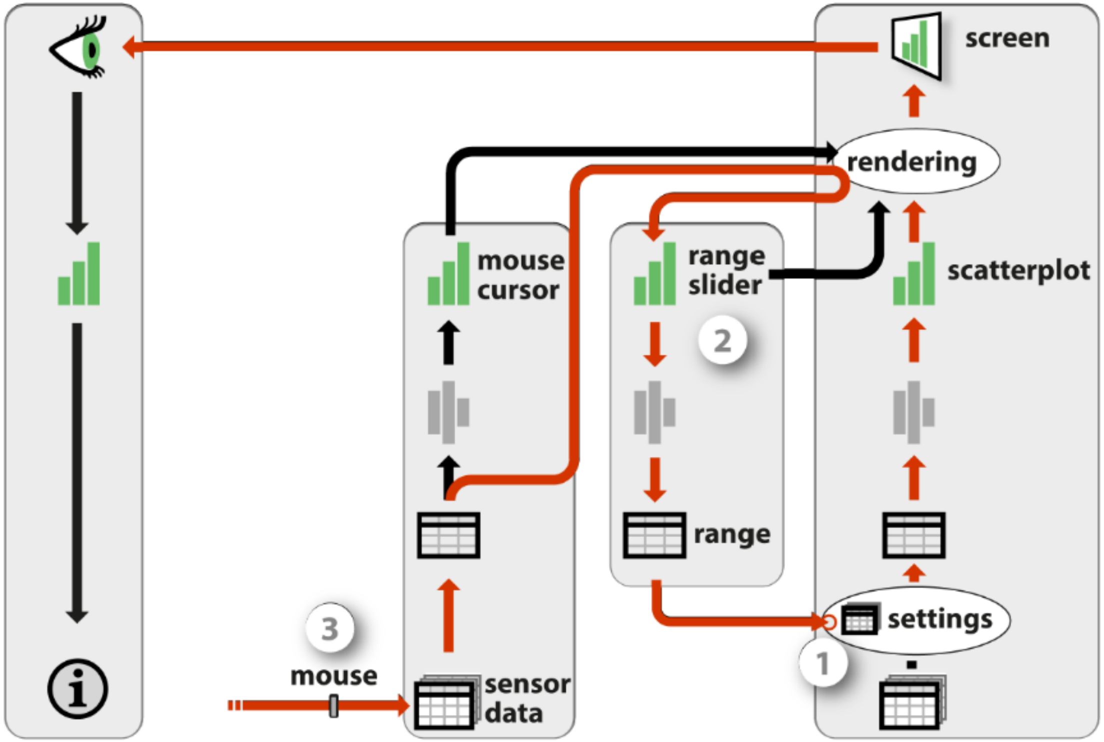


Image source Mark Belton 2006,
http://www.codeproject.com/Articles/13027/

Combining the What and the How

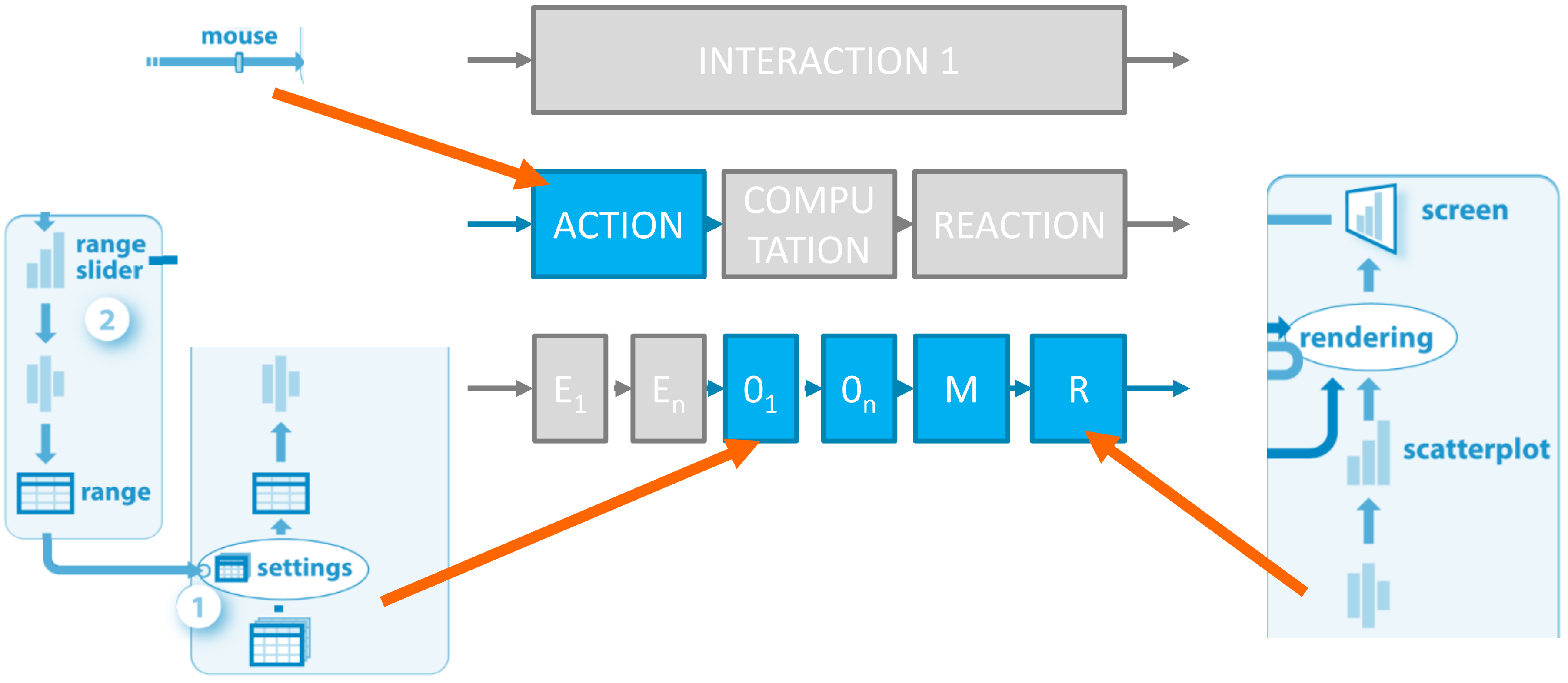
Example 1: An Extended Infovis Pipeline



Source: [Jansen + Dragicevic 2013]

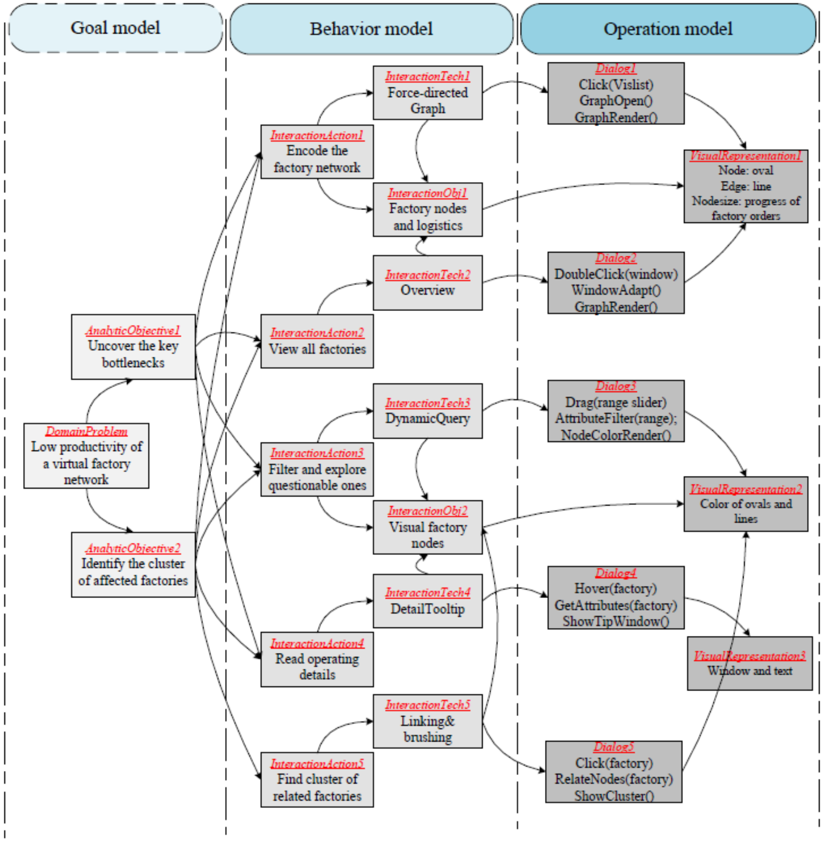
Combining the What and the How

Example 1: An Extended Infovis Pipeline



Combining the What and the How

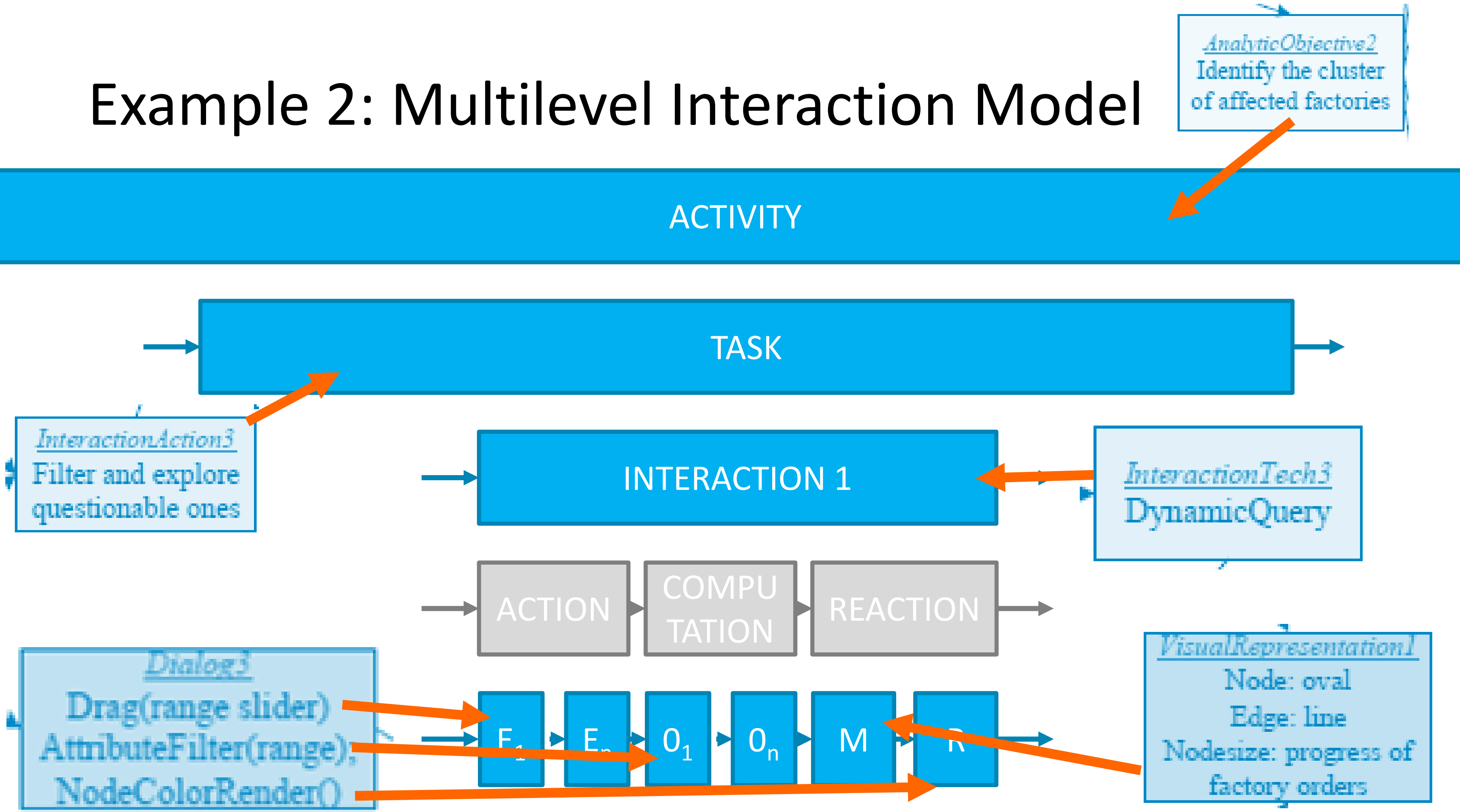
Example 2: Multilevel Interaction Model



Source: [Ren et al. 2013]

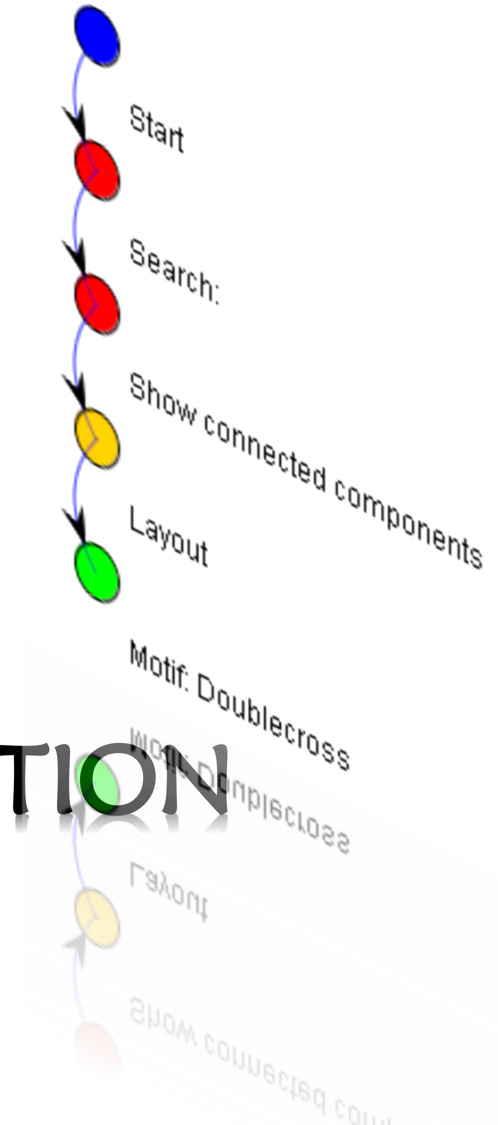
Combining the What and the How

Example 2: Multilevel Interaction Model



User’s Perspective

HISTORY AND INTERRUPTION MANAGEMENT



History Management

Interaction History

"Keep a history of actions to support undo, replay, and progressive refinement."

-- Ben Shneiderman 1996

Three aspects:

1. Recording history (*logging*)
2. Utilizing the current history (*undo/redo*)
3. Utilizing a collection of histories (*guidance*)

History Management:
1. Recording Interaction History

Approaches differ in:

- What is captured (actions vs. states)
  -> states easier to log, actions allow more diverse use
- How the information is aligned (linear vs. branching time)
- Which and how many levels of detail are captured (low vs. mid vs. high)
  -> highly problematic to discern when one action ends and the next begins
  -> use of taxonomy/ontology can help to define actions more clearly
- Scope (local vs. global)
  -> from system-wide logging (e.g., Glass box by Cowley et al.)
  to object-specific logging (e.g., per spreadsheet or per cell)

adapted from Heer et al. 2008

History Management: 2. Utilizing the Current History

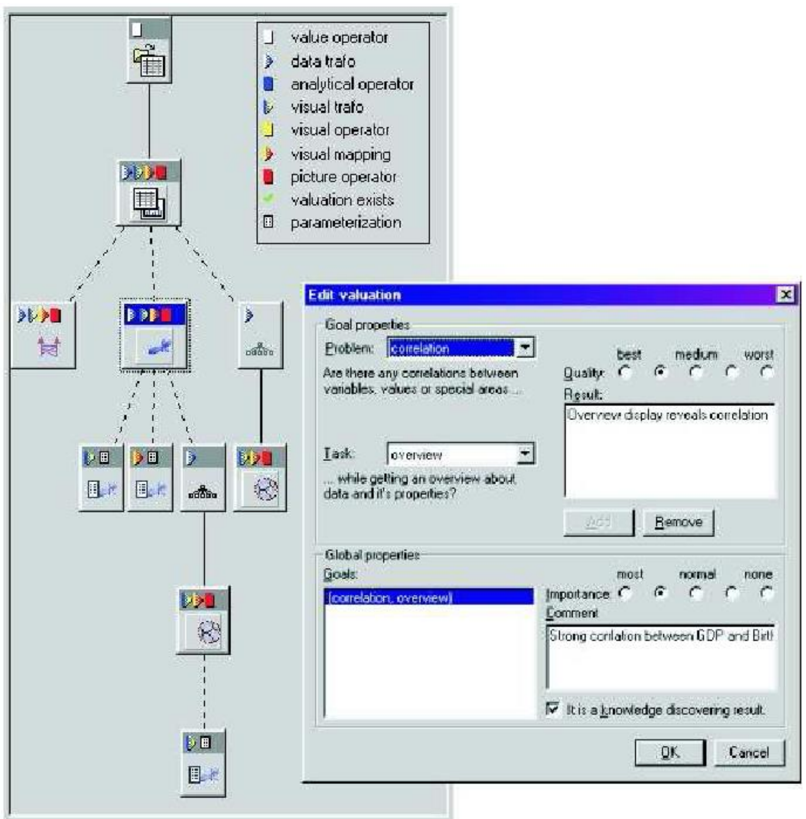
Operations on the history:

- **Reflect:** passively show the history as it evolves alongside the visualization
- **Replay:** recap the history for presentation/validation purposes
- **Retrace:** undo/redo an action or reestablish a prior state
- **Reuse:** reapply a sequence of actions like a makro
- **Reconfigure:** selective undo/redo, reordering, refining, reparametrization
- **Report:** annotate, share

1_history.avi

History Management

Example for History Management



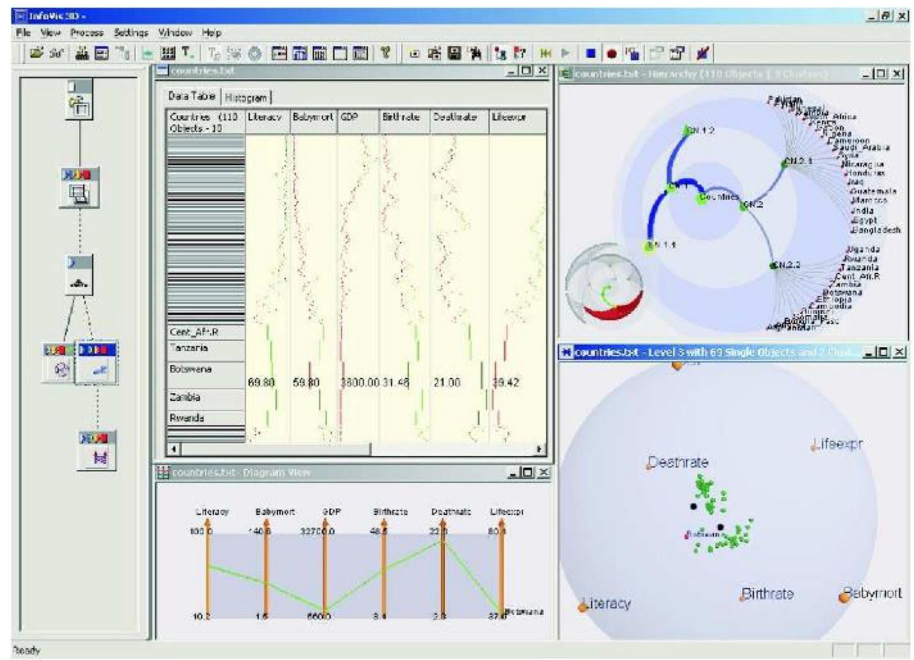


Image source Kreuseler et al. 2004

History Management: 3. Utilizing a Collection of Histories

Analyze/mine past histories to provide guidance:

- **Guidance context:** prior knowledge of the user
  -> nothing, goal (desired final state), path (activity sequence), full
- **Guidance domain:** the matter on which guidance shall be provided
  -> data, views, infrastructure, users
- **Guidance target:** how the aim or goal of the guidance is declared
  -> direct (directed search), indirect (query by example), inverse indirect (discover the unexpected)
- **Guidance degree:** how much freedom to deviate is still allowed
  -> orienteering -> steering -> storytelling -> animated animation

taken from Schulz et al. 2014

History Management

Example for Guidance:

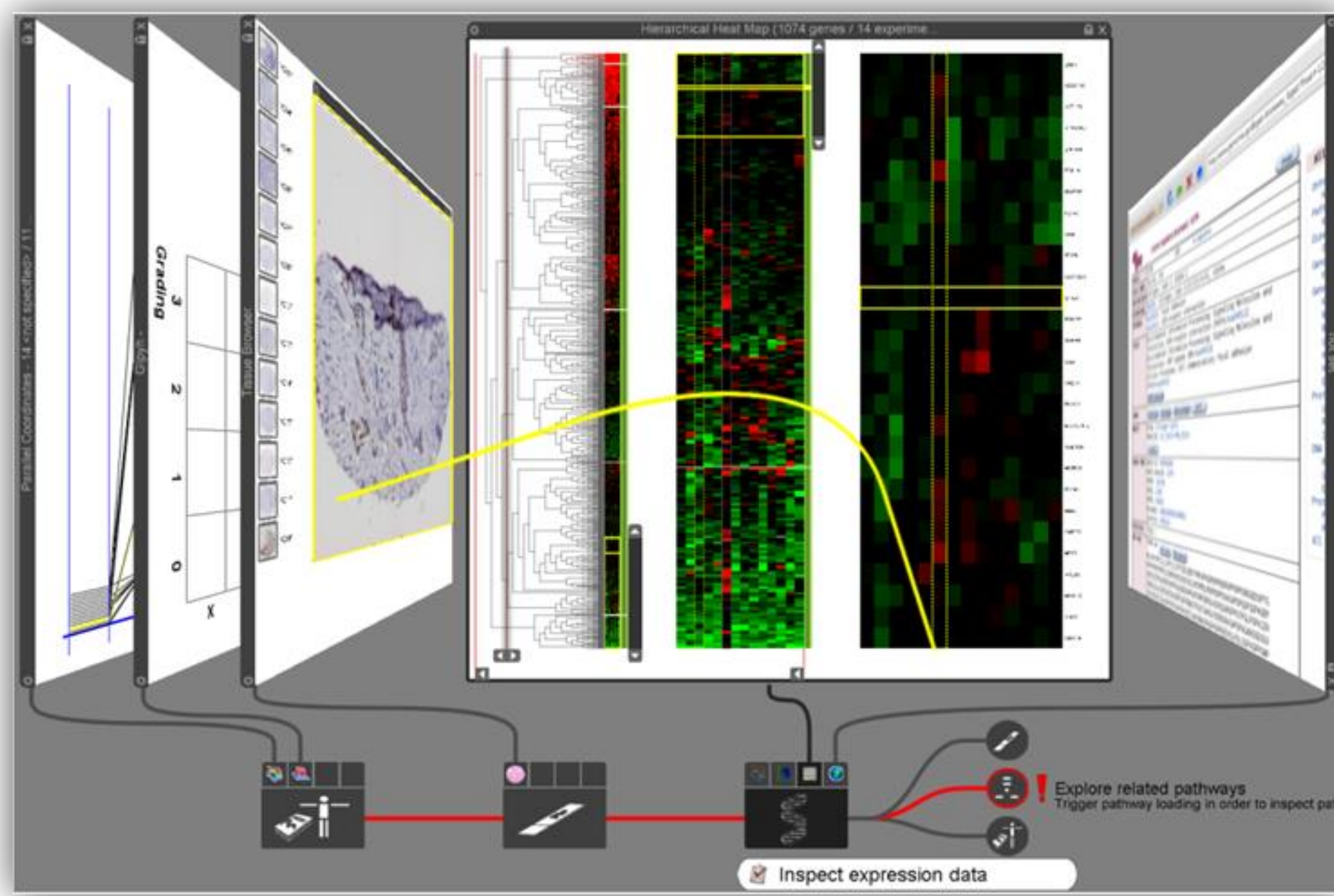
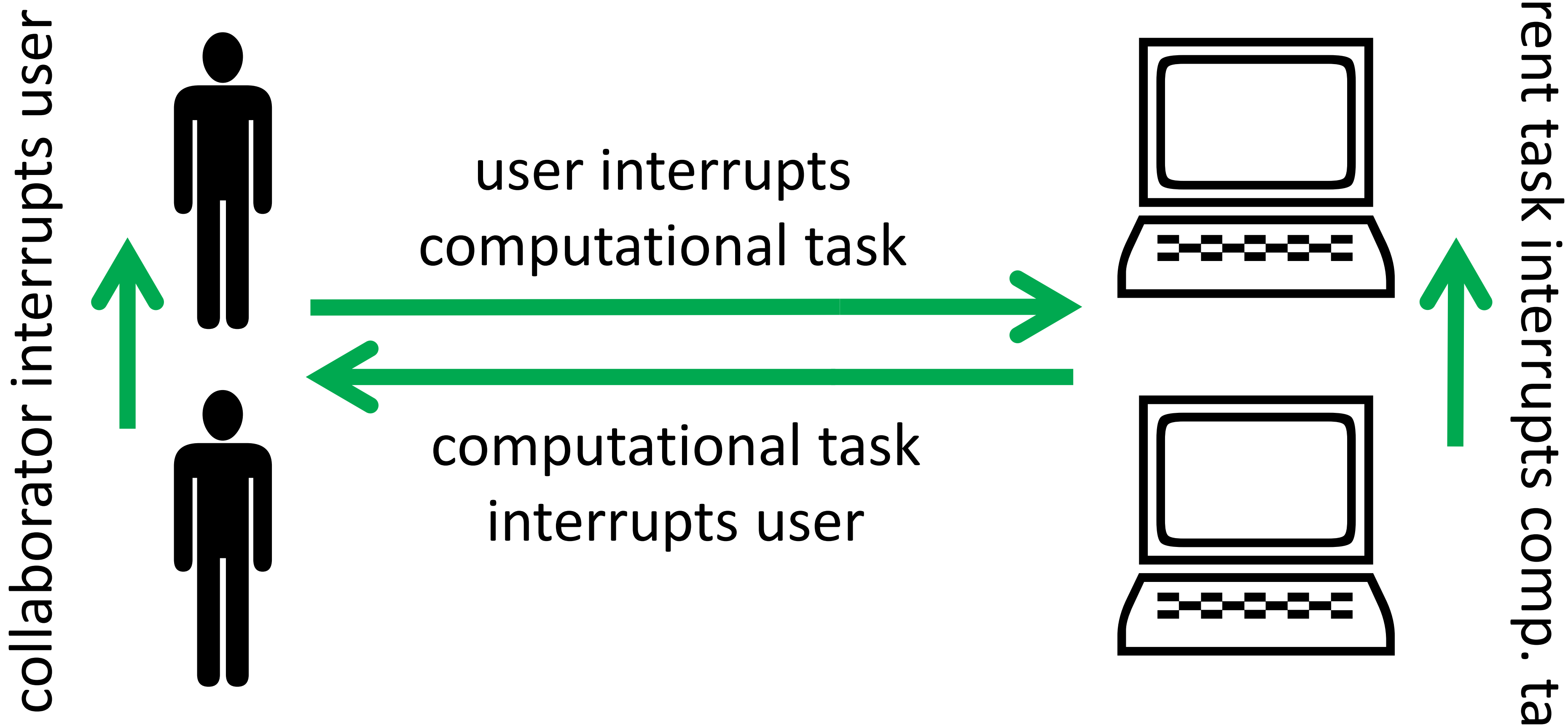


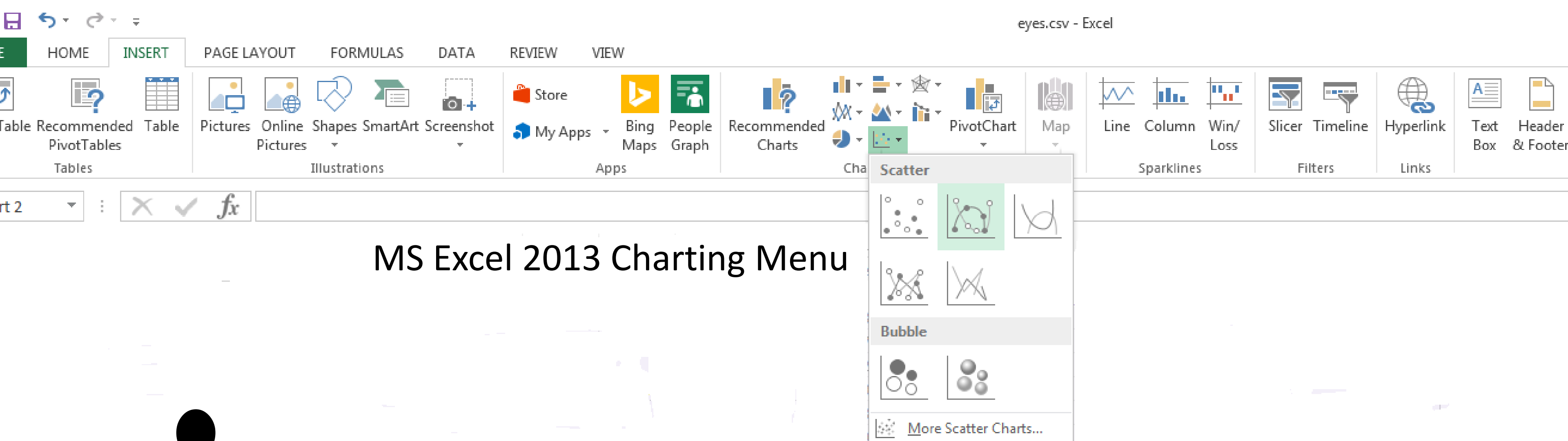
Image source: Streit et al. 2012

Interruption Management

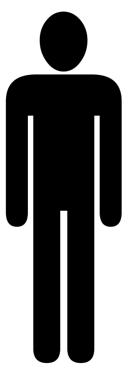
Who interrupts whom?



Mixed-Initiative Interaction: Who Starts/Leads the Communication?

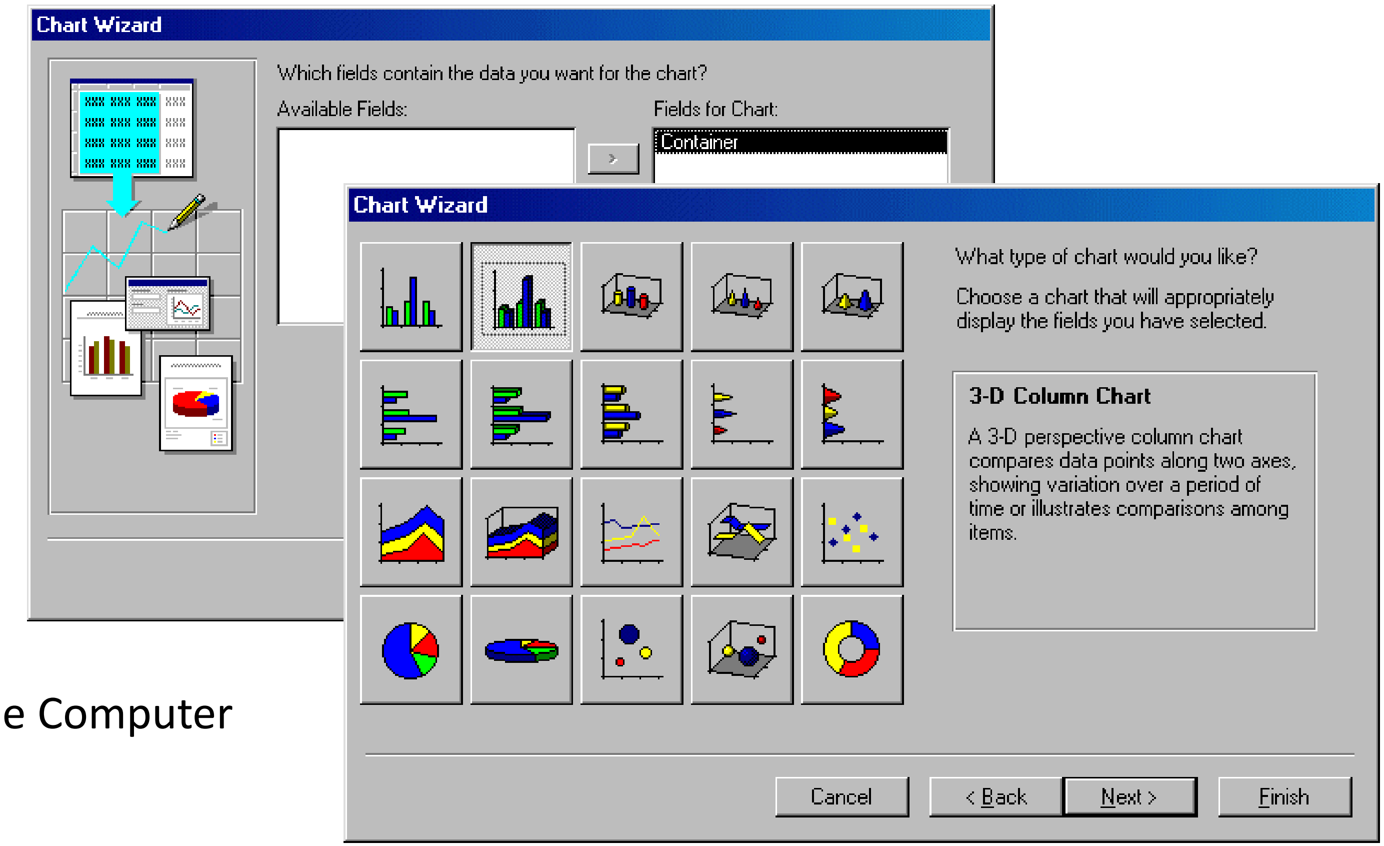


MS Excel 2013 Charting Menu



Interaction lead by the User

Mixed-Initiative Interaction: Who Starts/Leads the Communication?



Interaction lead by the Computer

MS Access 2003 Charting Wizard

Interruption Management

Possible Interruption Responses

- **Oblivious dismissal**
  -> interruption goes unnoticed
- **Unintentional dismissal**
  -> interruption is noticed, but its significance w.r.t. the current task not evaluated/understood
- **Intentional dismissal**
  -> interruption is deemed less important than the ongoing task
  -> interruption queued for later, ongoing task resumed
- **Preemptive integration**
  -> interruption is deemed more important than the ongoing task
  -> remainder of ongoing task is queued, start work on interruption
- **Intentional integration**
  -> interruption and ongoing task are subsequently worked on together

source: McFarlane + Latorella 2002

Interruption Management

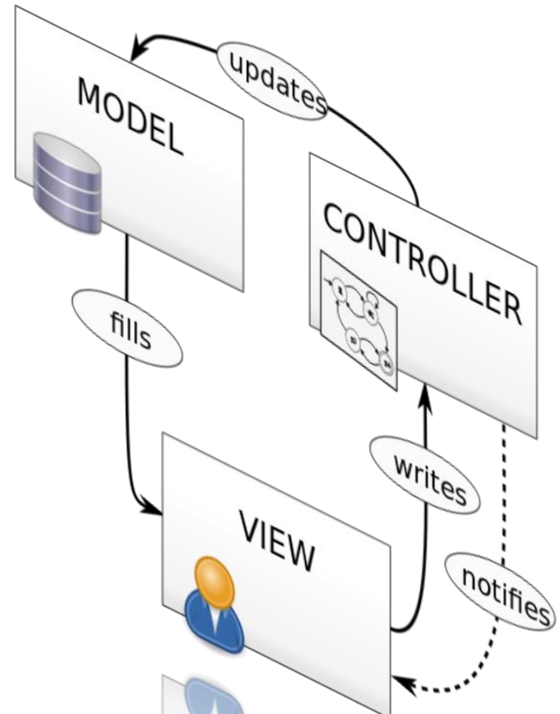
Common Interruption Management Strategies

- **Immediate interruption**
  -> e.g., error messages (BSOD, Browser 404,…)
- **Negotiated interruption**
  -> modeled after human-human interruption strategies:
  alert to interruption, but let the user accept, decline, or ignore it
- **Mediated interruption**
  -> aims to predict the user‘s interruptability (e.g., by determining the current cognitive load of the screen content) and time interruptions accordingly
- **Scheduled interruption**
  -> the user specifies in advance which interruptions are permitted – e.g., for the next 2 hours only interruptions that can be handled within 1 minute are allowed

source: McFarlane + Latorella 2002

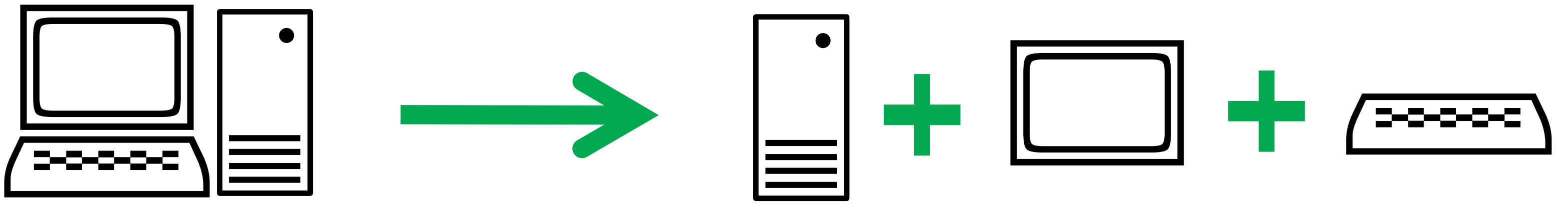
Software Engineer’s Perspective

PATTERNS FOR INTERACTIVE SOFTWARE

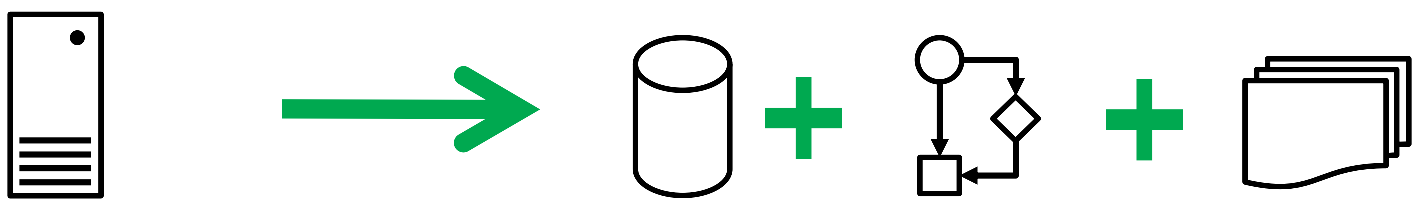


Patterns for Interactive Software

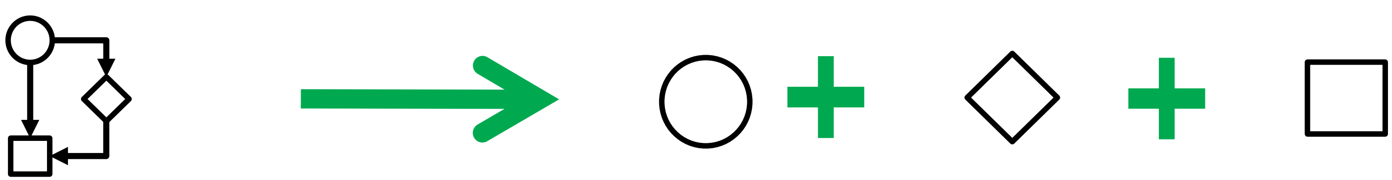
- Functional: Decoupling into Model-View-Controller



- Logical: Decoupling into Data-Context-Interaction

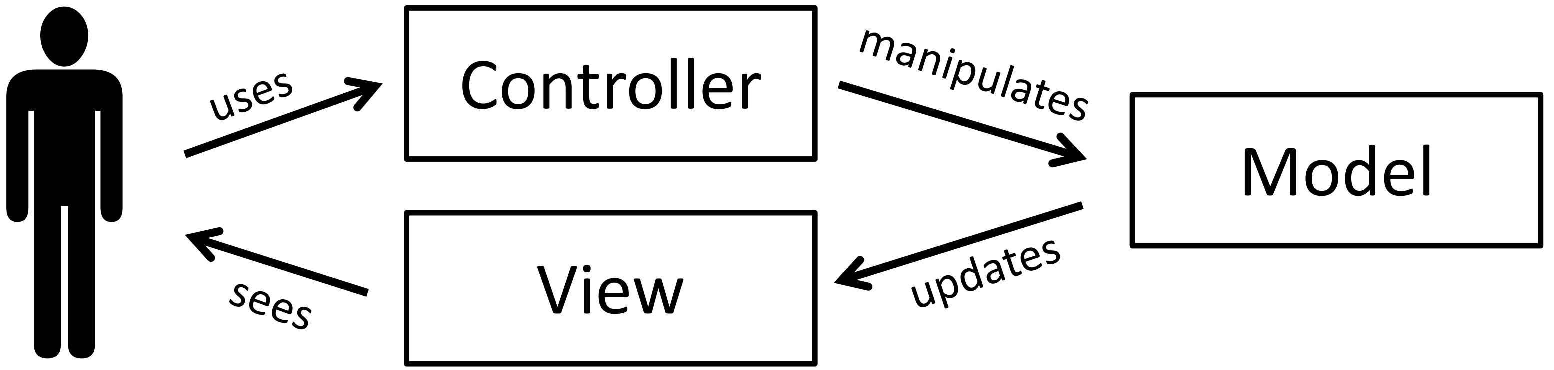


- Technical: Decoupling into Threads



The Model-View-Controller Pattern

- invented by Trygve Reenskaug in 1978/79 at Xerox PARC
- de-facto standard for implementing UIs



- decoupling allows, for example, for syncing multiple views via a single model

The Data-Context-Interaction Pattern

- invented by Trygve Reenskaug around 2006
- helps mainly to untangle the model into
  - **Data:** the pure data with base functionality
  - **Context:** the processes/workflows in which the data is used
  - **Interaction:** the roles the data plays in the processes
    -> same data can assume different roles
    -> for each, role-specific functionality is added to the data
- decouples data from behavior (roles)
  -> data objects can play many roles over their lifetime
  -> while roles persist only for the duration of the process

A Multi-Threaded Architecture for Continuous Interaction

Discrete interaction = click, key stroke, etc.

Continuous interaction = drag slider/layout back and forth
-> must be sped-up to be interactive (10-20 frames/sec)

Bottleneck @ Retrieval/Computing: precompute affected pixels/objects for each pixel of a slider [Tanin et al. 1996]

Bottleneck @ Networking/Rendering: progressive output with constant refinement – e.g., progressive vis. [Stolper et al. 2014], per-iteration vis. [Choo et al. 2014], online vis. [Angelini et al. 2014]

For both: limit distance the user is allowed to travel per time interval w.r.t. available hardware [Chan et al. 2008]

A Multi-Threaded Architecture for Continuous Interaction

Step 1: One Thread per View

- Separate Application thread and Visualization thread(s)
- Each View gets its own Visualization thread
- Visualization thread progressively refines the view until finished or cancelled
- Use Early Thread Termination

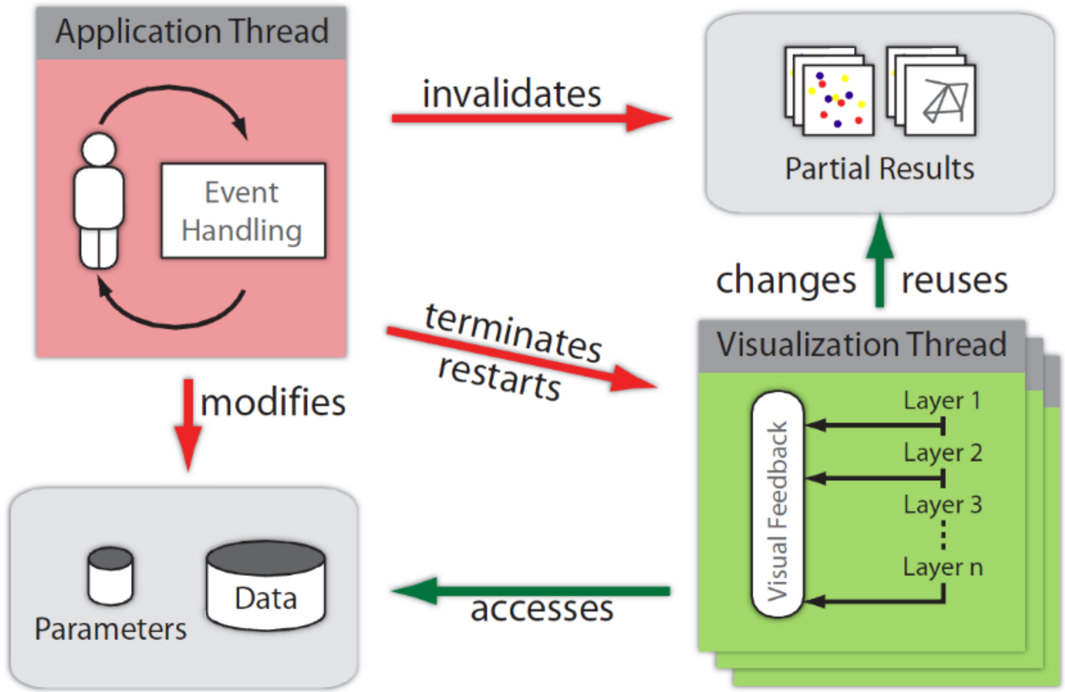


image source [Piringer et al. 2009]

A Multi-Threaded Architecture for Continuous Interaction

Step 2: Multiple Threads per View by Layering

Common Layering Mechanisms (from Piringer et al.):

Semantic layers: background (map, grid,…), coordinate axes, data items, labels,…
-> sort by decreasing relevance or increasing effort

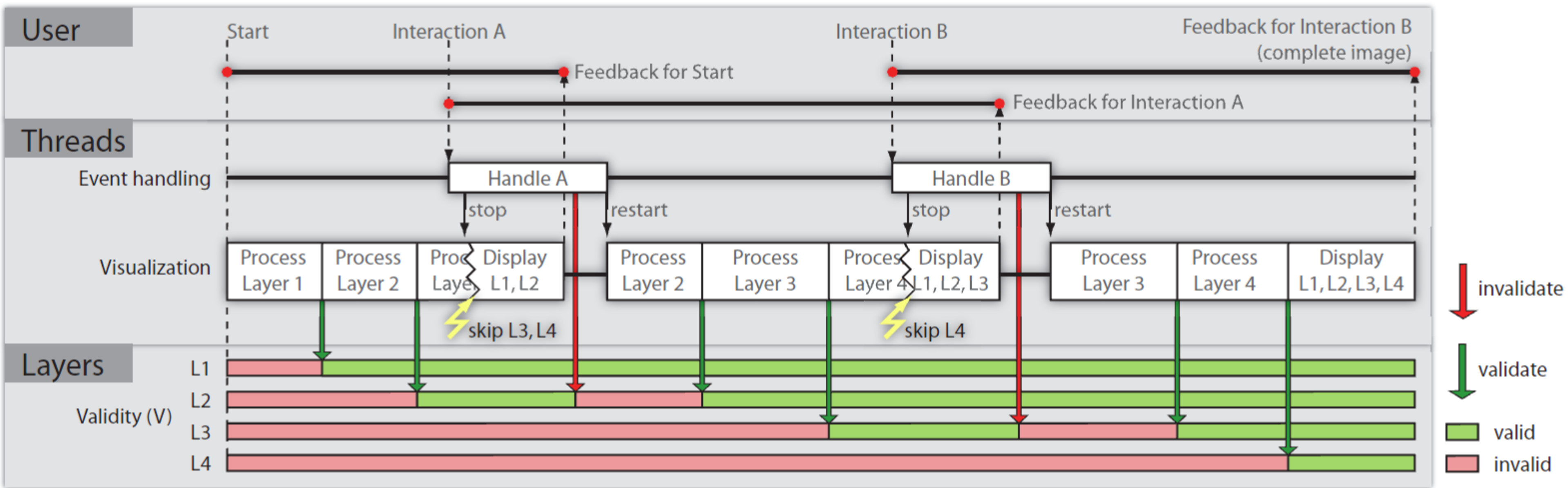
Incremental layers: sampled data items – e.g., every 100th, every 10th, and finally every item

Level-of-Detail layers: on data level – clusters, subclusters,…
on image level – first without anti-aliasing at low resolution, then with anti-aliasing at high resolution
-> here: layers get replaced, not blended onto drawn ones

A Multi-Threaded Architecture for Continuous Interaction

Step 2: Multiple Threads per View by Layering

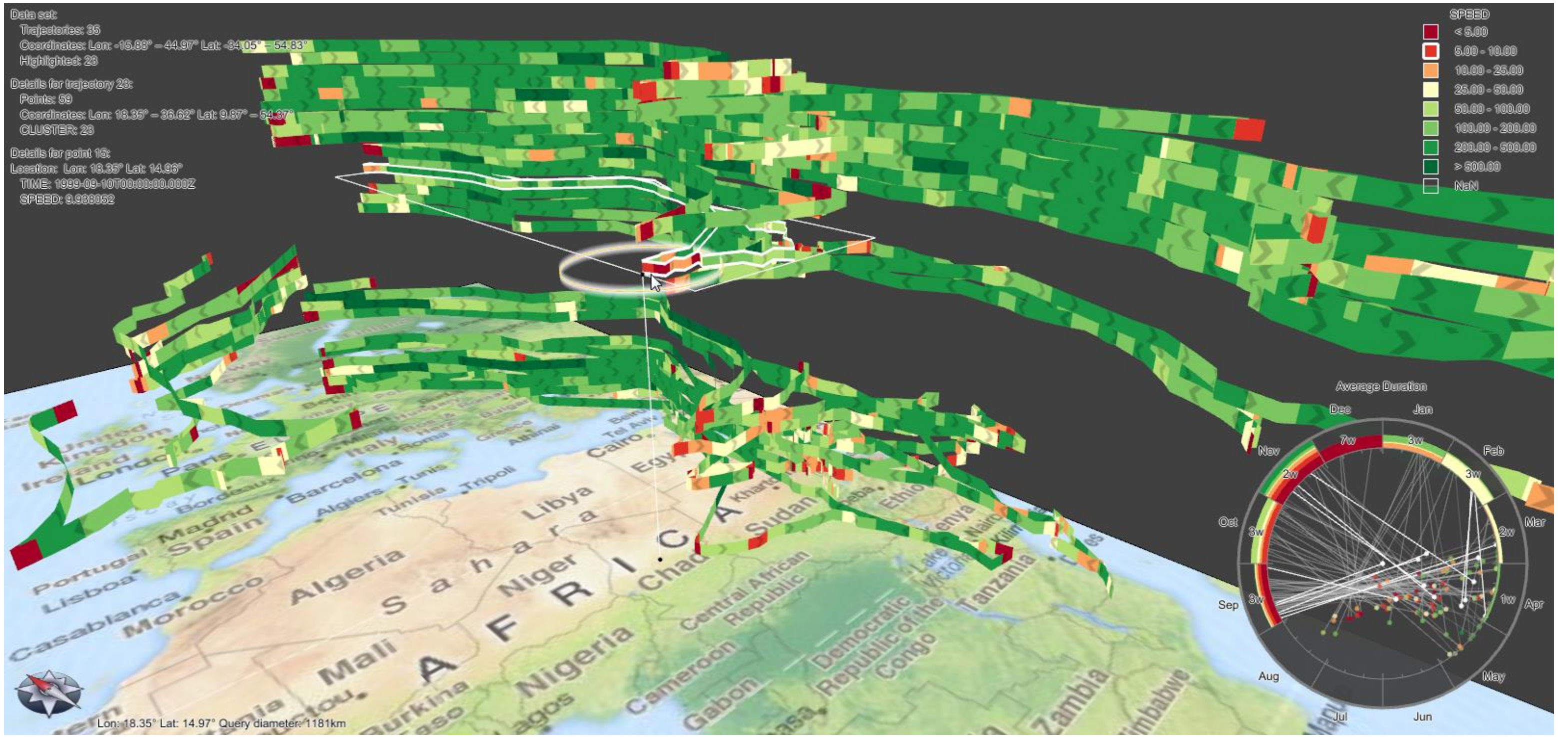
Layered Visualizations allow for reuse of already drawn elements



Picture source: [Piringer et al. 2009]

Interactive Visualization in Action

A short practical interlude:



Christian Tominski, University of Rostock, Germany