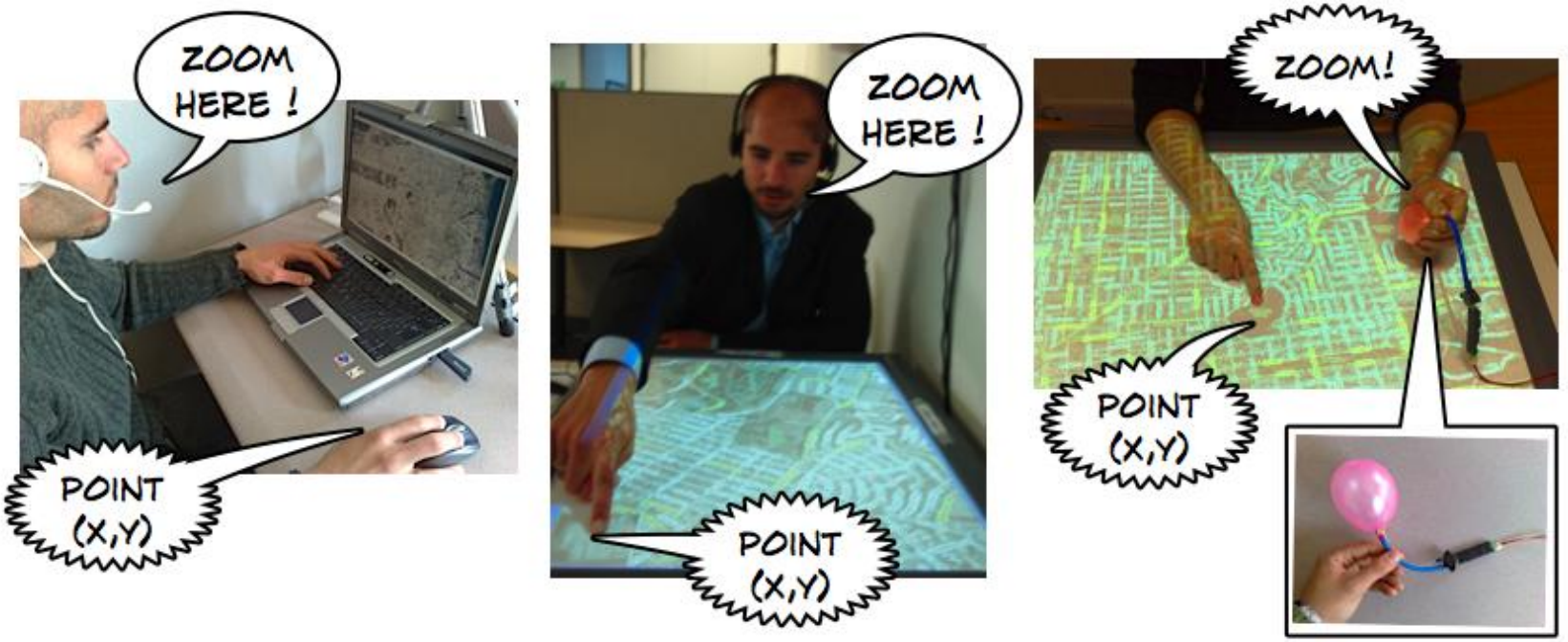
A Three-dimensional Characterization Space of Software Components for Rapidly Developing Multimodal Interfaces

Marcos Serrano, David Juras, Laurence Nigay
Laboratory of Informatics of Grenoble (LIG),
University of Grenoble, France

Illustrative example
Multimodal Map Navigator

- Put-that-there ([Bolt 80])



Component-based approach

- Integrate existing code
- Integrate heterogeneous components written in different languages
- Improve reusability
- Rapid modification of component’s assemblies
  - Fast prototyping of multimodal applications
- Graphical assembly of components: non developer users

Key point: component characteristics

- Improve reusability of components
- Ease the creation of component’s assemblies
- Allow people with different background to use a common component-based tool

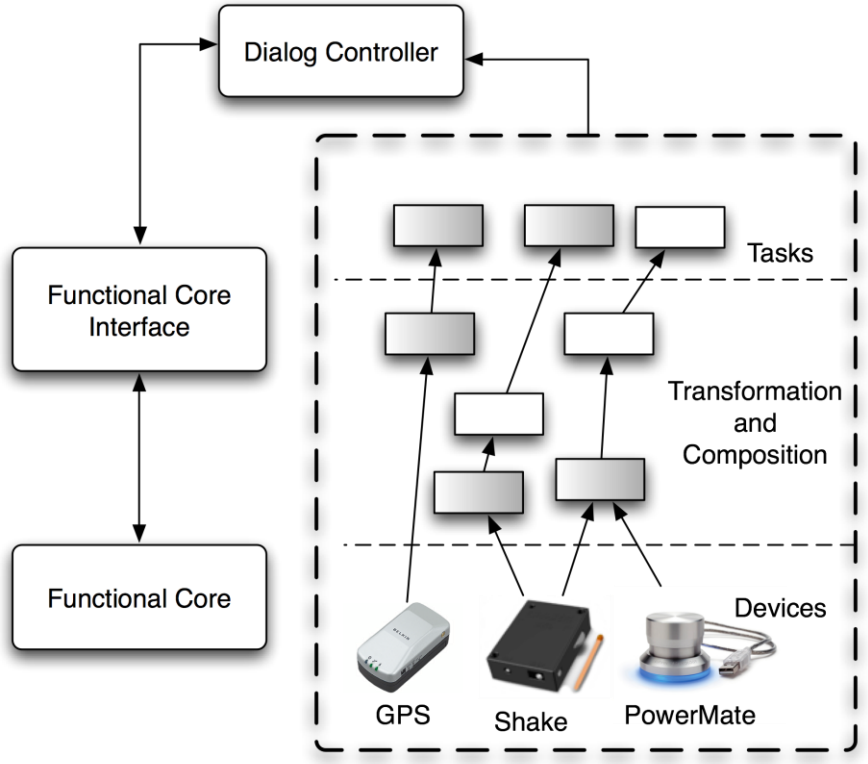
Scope and Overview of the Characterization Space

Overview and Scope of the space Component-based models

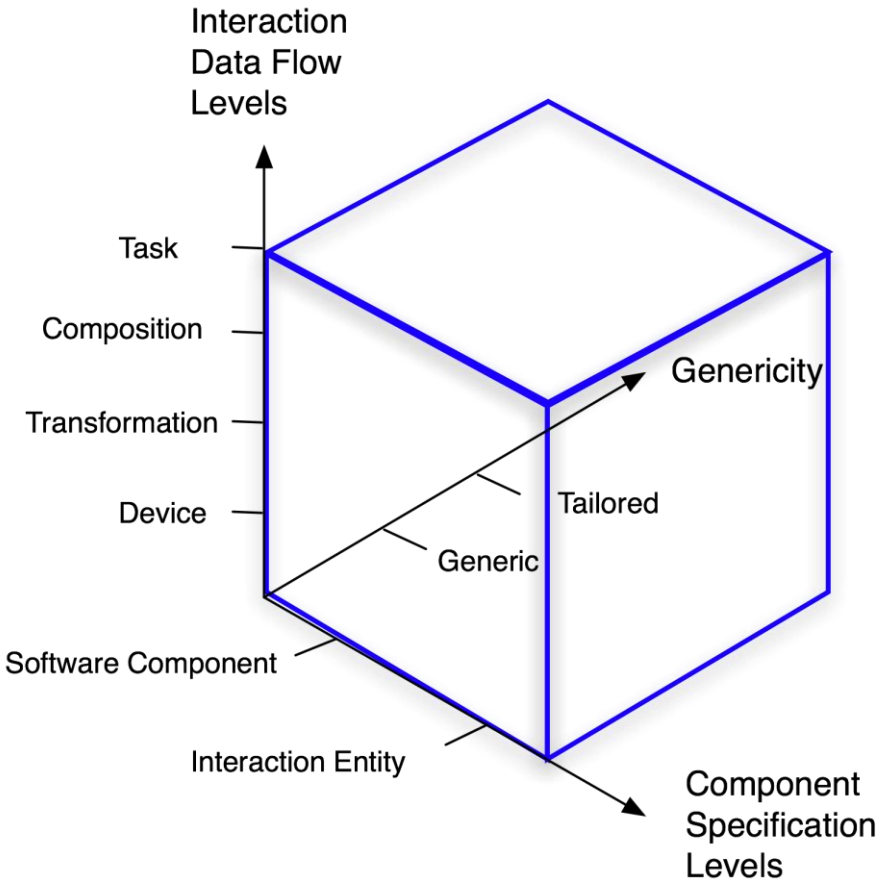
- Space defines set of characteristics of components
- Component: any type of software unit (software component, service)
- DOES NOT define execution behaviour of components
- Define high-level characteristics of components, implemented in any component-based technology
  - Corba Component Model
  - Component Object Model (COM)
  - JavaBeans
  - OSGi

Overview and Scope of the space Interactive System

- Focus on the interactive part of an interactive system in an ARCH software model

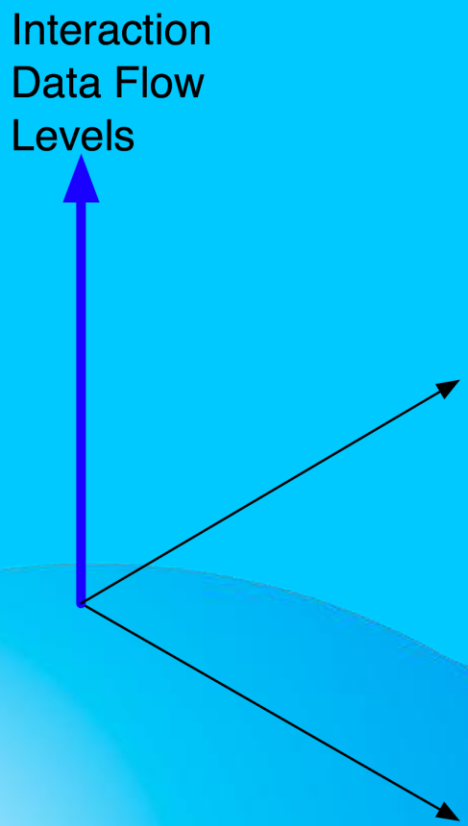


Overview and Scope of the space
Three dimensions

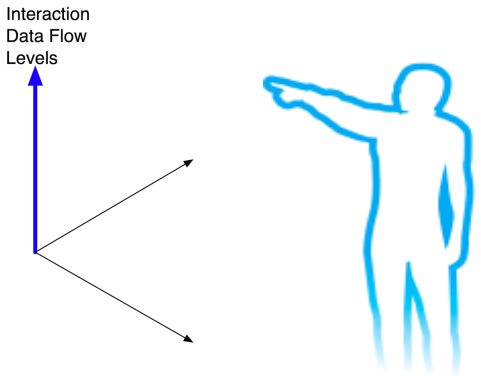


- 3 Dimensions:
  - Data Flow from devices to tasks
  - Specification level
  - Component genericity

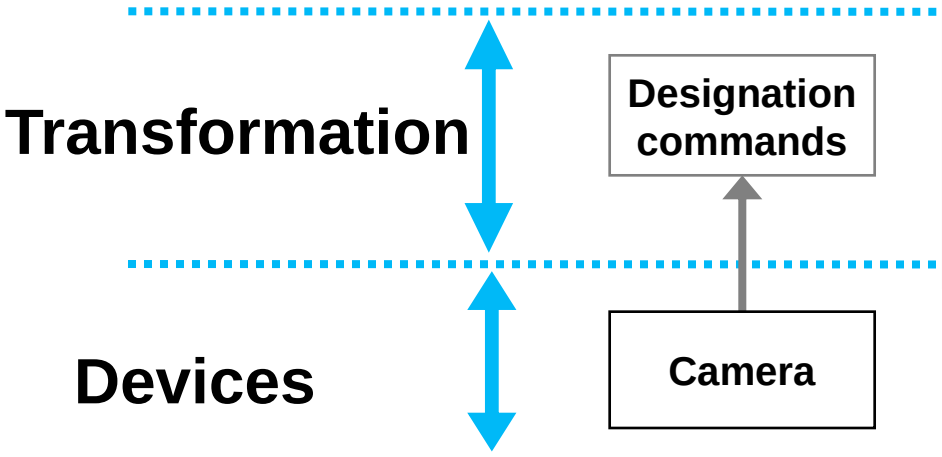
From input devices to tasks



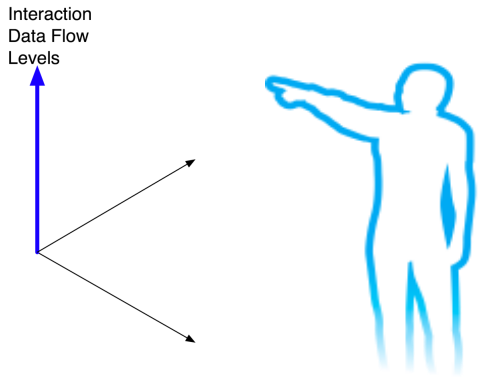
From input devices to tasks



Device + Transformation = Interaction Modality



From input devices to tasks



Tasks

Task:
Center and Zoom Point (x,y)

CARE
Composition

Complementarity

Transformation

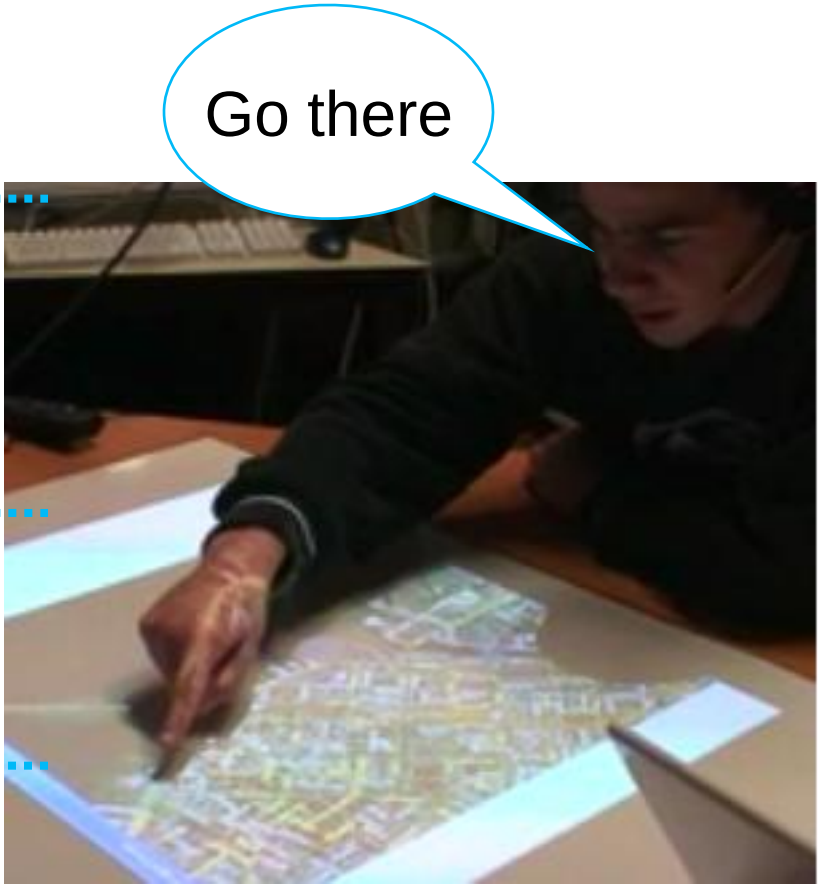
Designation commands

Speech command

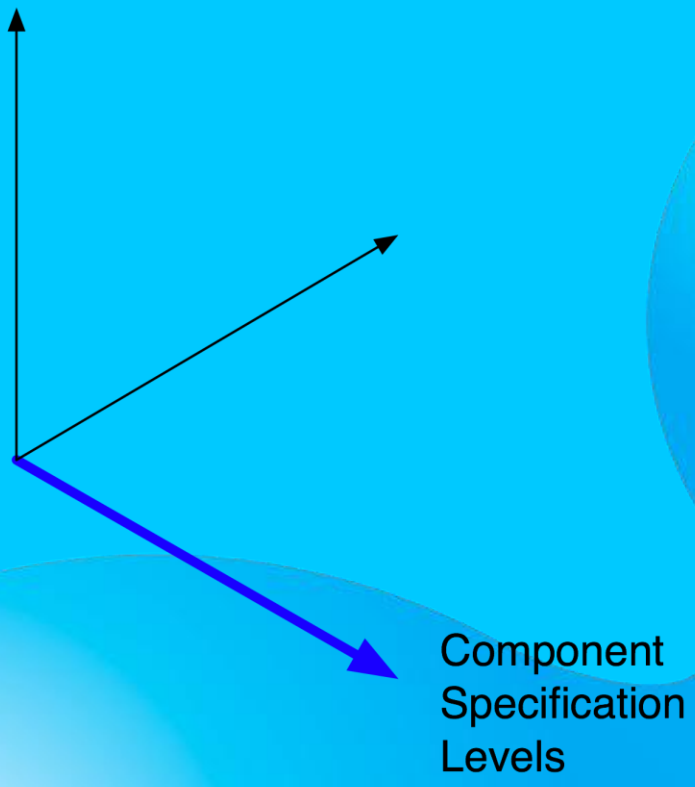
Devices

Camera

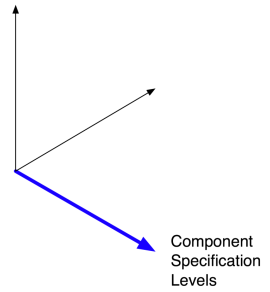
Microphone



Specification Levels



Specification Levels



- Designers should be able to build multimodal applications
- Key point: description level of the components
- Designer interested in
  - Human sense involved
  - Weight of the device
  - Dimensions, etc..
- Developer interested in
  - Driver library
  - Programming language
  - Types of the parameters

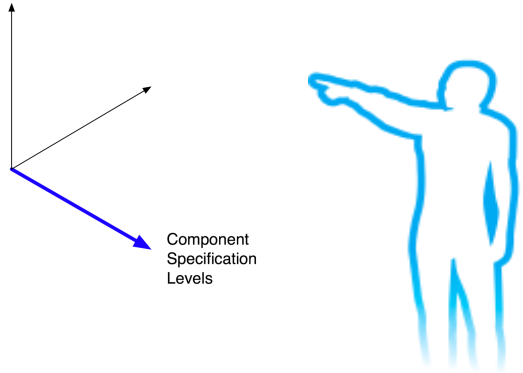
Designer Level

Interaction entities

Developer Level

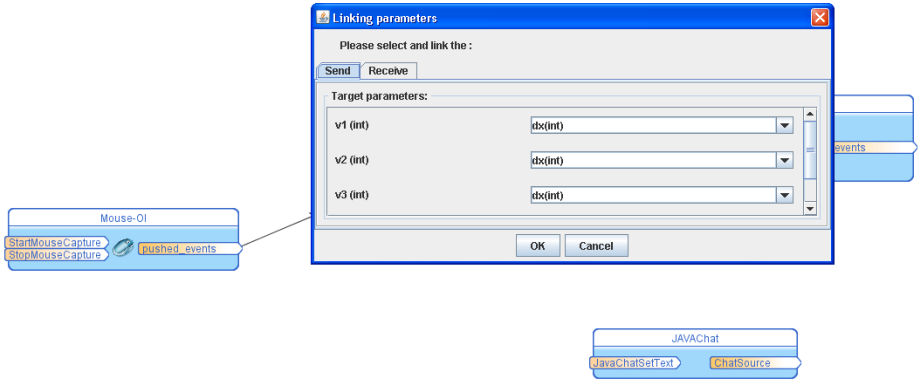
Software components

Specification Levels

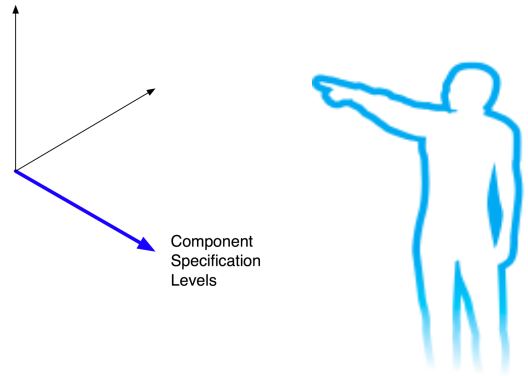


Designer Assembly

Developer Assembly

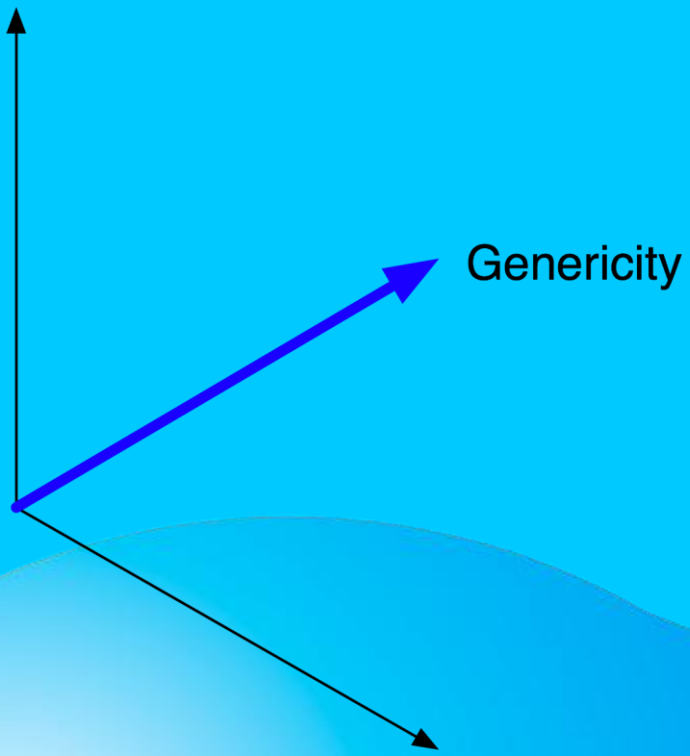


Specification Levels

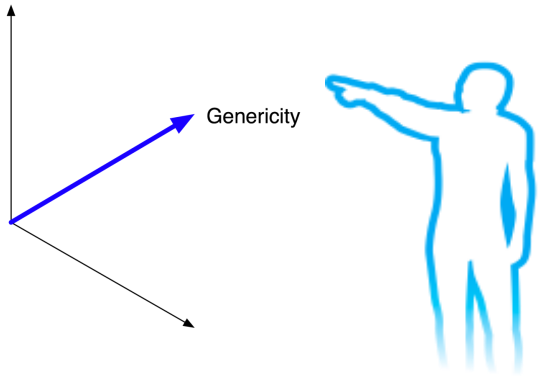


- Developers
  - Reuse existing components
  - Software engineering criteria
- Application designers
  - Rapid and Easy development of multimodal applications (focus on multimodal interaction)
    - Low threshold (easy to use) while providing a high ceiling (how much can be done with the platform)
  - More iterations as part of an iterative design method for achieving usable multimodal applications

Need for genericity



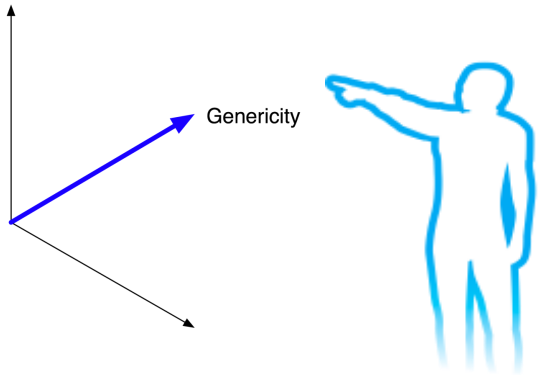
Need for genericity



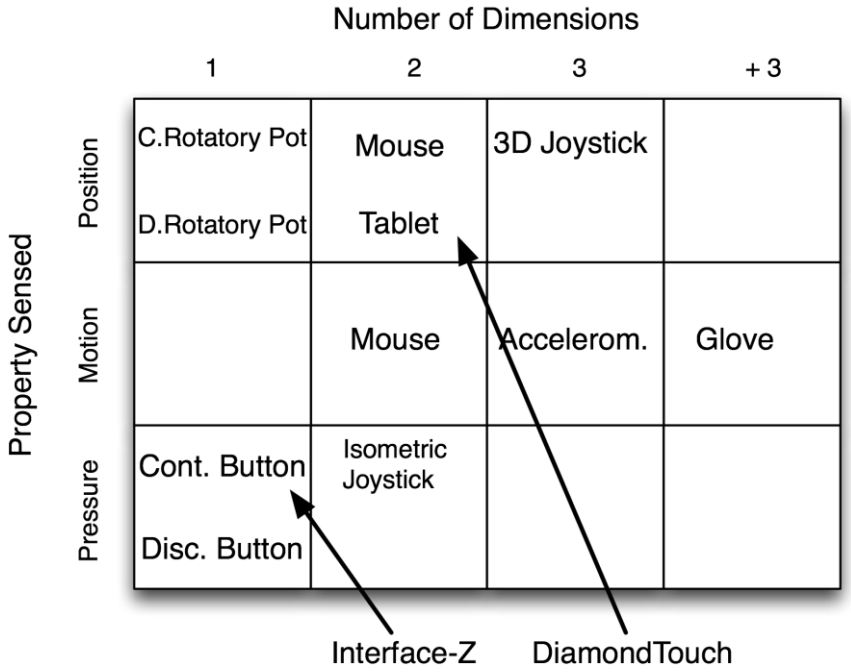
- Tailored components
  - Implemented in ad-hoc ways for the needs of a specific application or interaction
- Generic components
  - Implement generic mechanisms

- Using generic components improve expressive leverage and reusability
  - Expressive leverage: « tool reduces total number of choices a designer must make to express a desired solution » [Olsen 07]

Need for genericity
Generic Devices

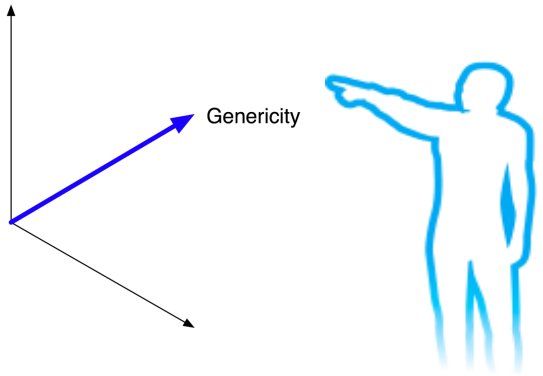


- Buxton's taxonomy [Buxton 83]
  - Hand-controlled input devices



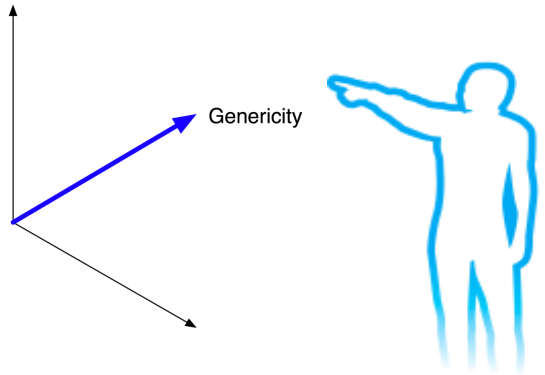
- Speech and Image analysis devices
  - Difficult to classify

Need for genericity



- Generic Transformation
  - Reusable operations usually performed on data from generic devices
- Generic Composition
  - CARE properties [Nigay 97]
  - Complementarity, Assignation, Redundancy and Equivalence
- Generic Tasks
  - Foley’s interactive tasks [Foley 84]

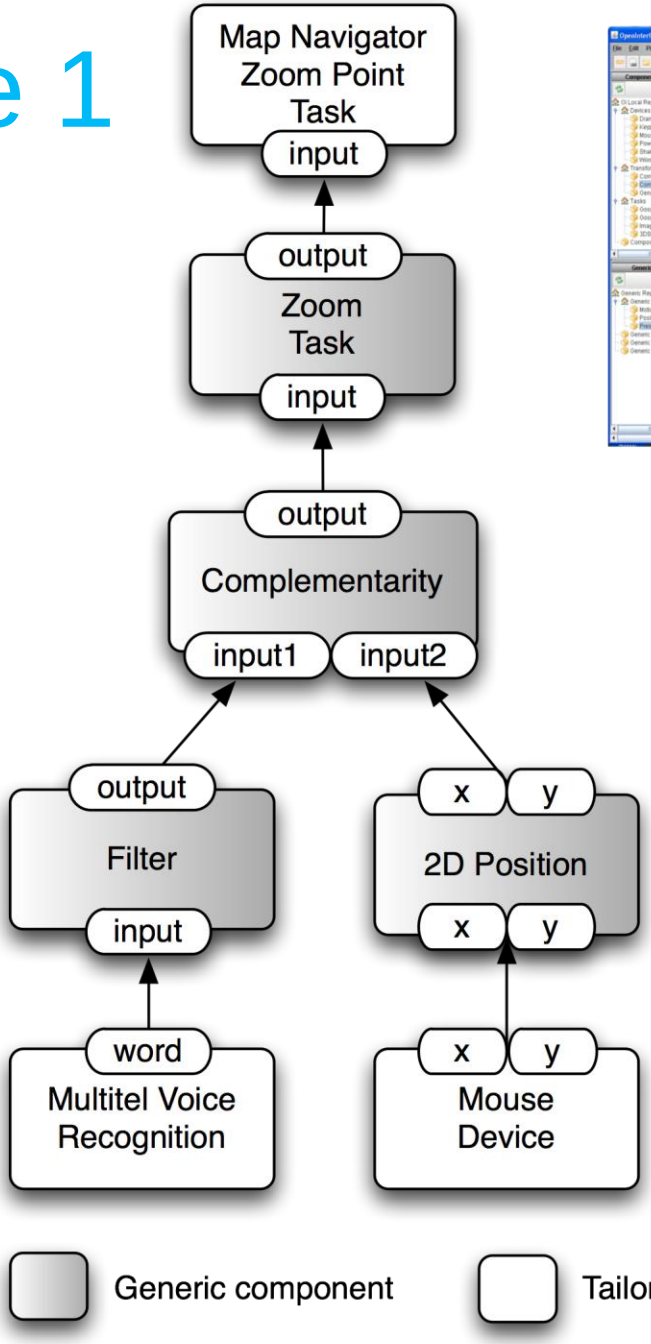
Need for genericity

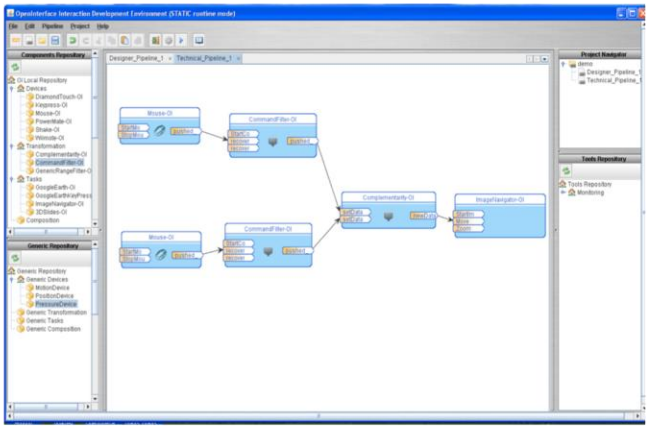


	What ?	How ?
Task	Application independent	Foley's task Configurable generic task
Composition	Modality independent	CARE properties
Transformation	Device and application independent	General Transformations: mathematical, mapping operations
Device	Abstraction of device data in terms of interaction actions, Buxton properties	Buxton's properties

Illustrative Examples

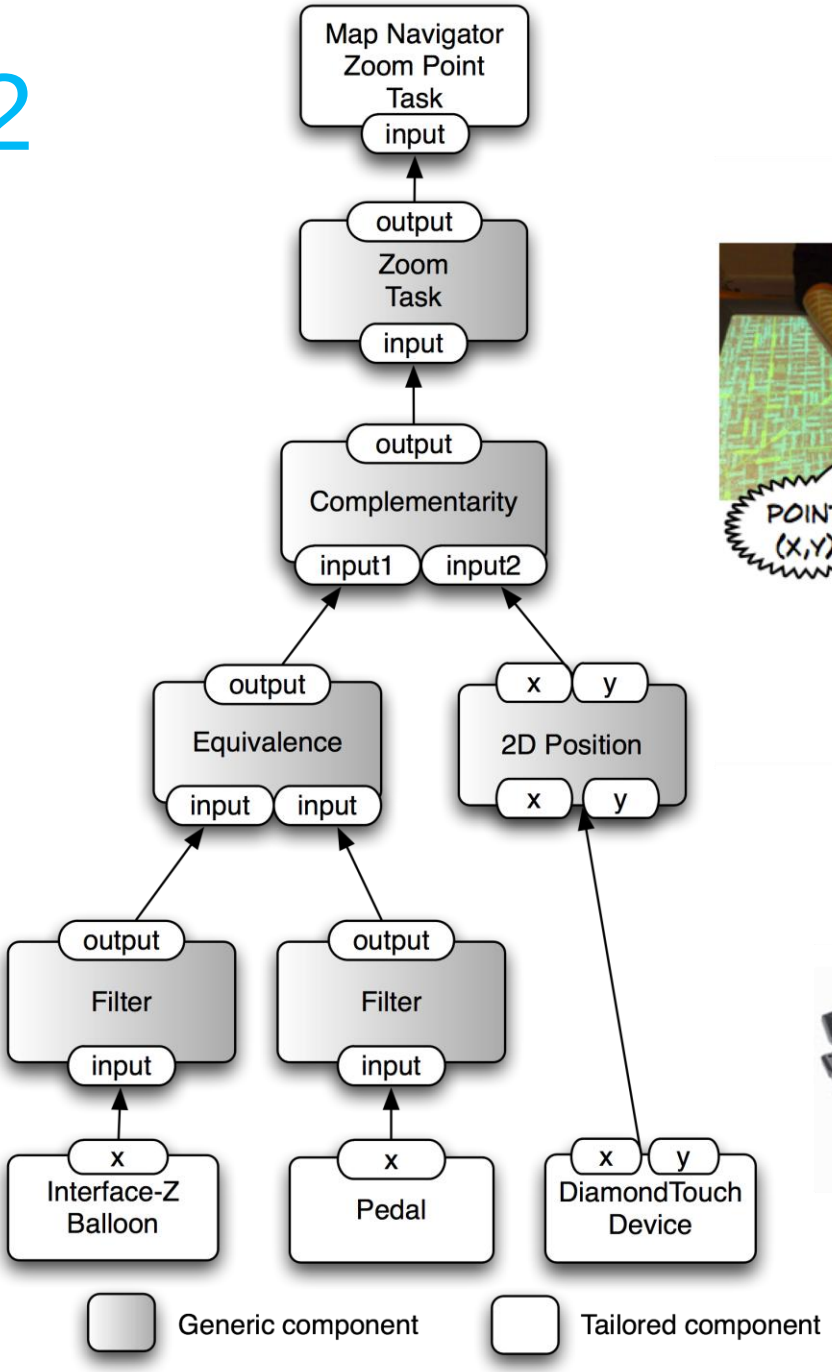
Example 1

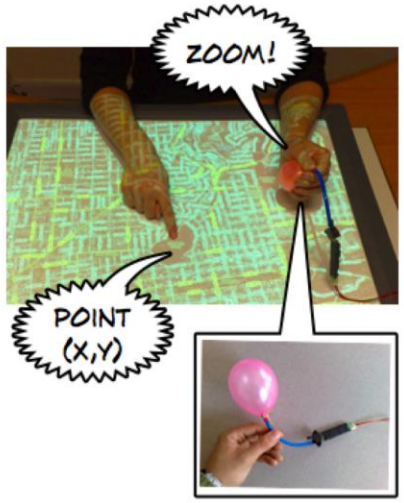

Map Navigator
Zoom Point
Task
input
output
Zoom
Task
input
output
Complementarity
input1
input2
output
Filter
input
x
y
2D Position
x
y
word
Multitel Voice
Recognition
x
y
Mouse
Device
Generic component
Tailored component




ZOOM
HERE !
POINT
(X,Y)

Example 2


Map Navigator
Zoom Point
Task
input
output
Zoom
Task
input
output
Complementarity
input1
input2
output
Equivalence
input
input
x
y
2D Position
x
y
output
Filter
input
output
Filter
input
x
Interface-Z
Balloon
x
Pedal
x
y
DiamondTouch
Device
Generic component
Tailored component


ZOOM!
POINT
(X,Y)


OR

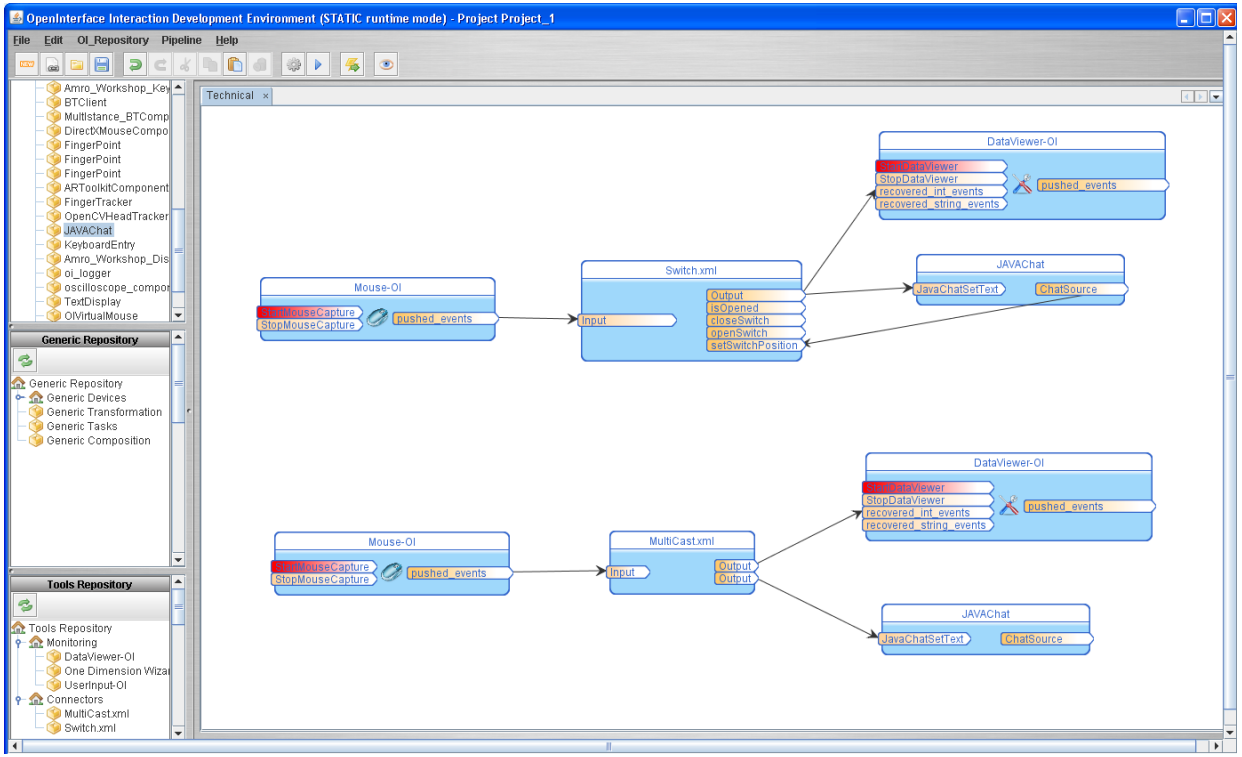
Implementation OpenInterface Framework

OpenInterface Framework
A tool for multimodal interaction

- www.oi-project.org
- The component-based OI framework is made of:
  - 1) Runtime kernel: underlying platform
    - Heterogeneous components:
      Java, C++, Matlab, Python, .NET
  - 2) Repository of interaction modalities
  - 3) OIDE
    OpenInterface Interaction Development Environment
    - Graphical Construction tool
    - Debugging / Logging tool

OpenInterface Framework

- OIDE: Graphical Interface
  - www.oi-project.org



Devices

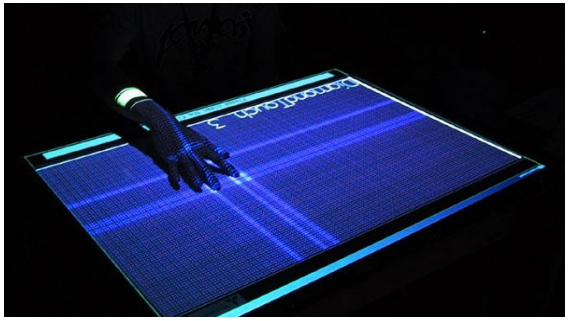
Transformation

Logging and Monitoring

Composition

Interactive Tasks

Implementation: OpenInterface Devices



Diamond Touch

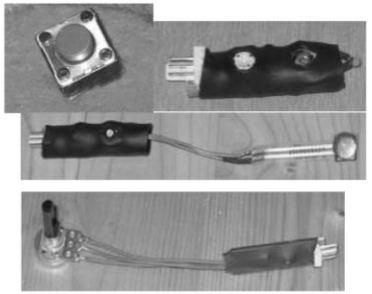
GPS

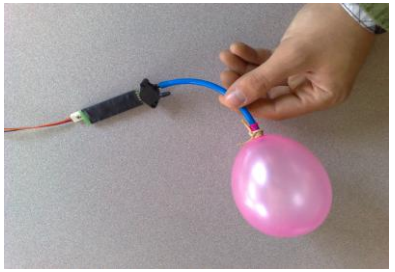
Shake

Wiimote

Track-IR

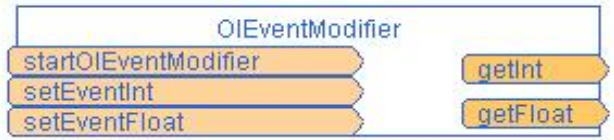
Power Mate



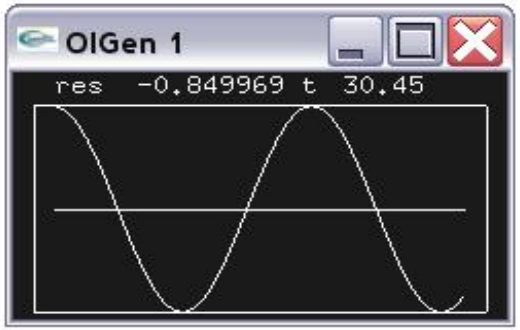


Interface-Z

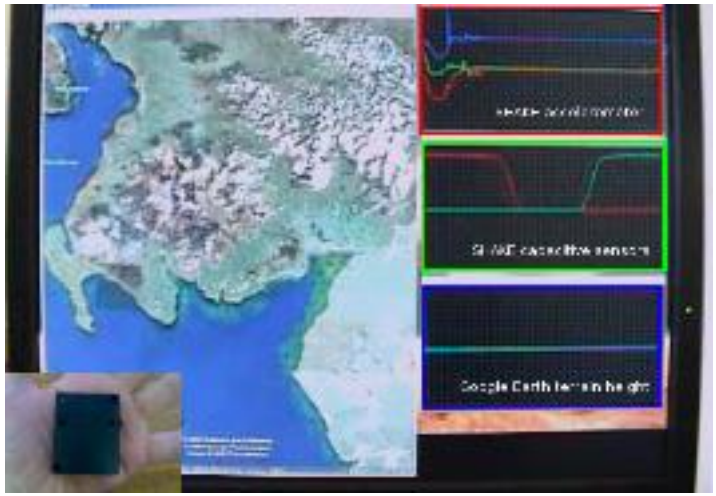
OpenInterface framework Adapters



Event Modifier
Python

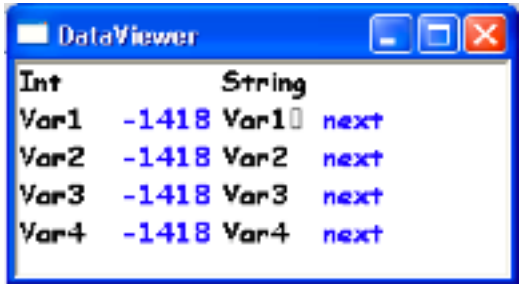


Automatic
Event Generator



Oscilloscope

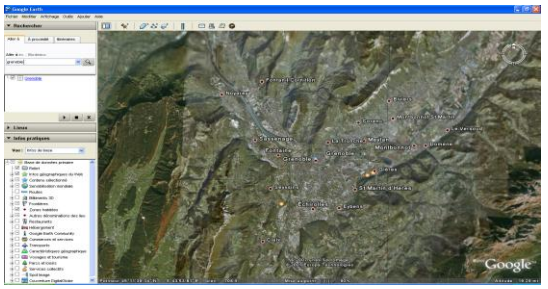
Manual Event
Generator



Data Viewer

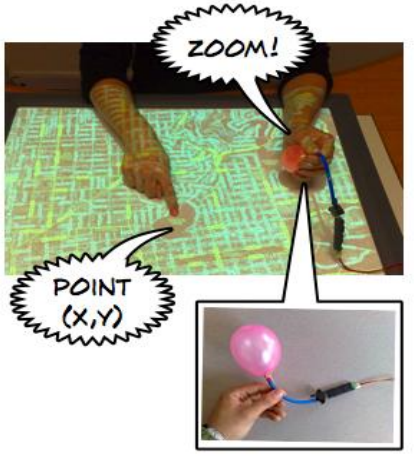
OpenInterface Applications

Game on mobile phone



Google Earth

Game on pc: speedycopter



Multimodal Map Navigator

Conclusion and Future Work

Conclusion

- Characterization Space of Software Components for multimodal interaction
  - Supports generic and tailored components
  - Different levels of abstraction
  - Data flow from input devices to interactive tasks
- Allows users with different technical backgrounds to use a tool implementing these characteristics
- Implemented in the OpenInterface (OI) framework
  - www.oi-project.org
- Several testbeds have been developed:
  - Map-based application
  - Game

Future Work

- Create the designer view in the graphical tool
- Relation between designer assembly and developer assembly
  - Iterative design process for multimodal interaction
- Enrich the framework with new generic components
- Focus on output multimodal response
  - Multiple display set-up
- Links between input and output components

Thank you for your attention

+ OpenInterface demo today

Questions ?