OpenInterface: SIMILAR platform
openinterface@similar.cc

Lionel Lawson
lawson@tele.ucl.ac.be
Université Catholique de Louvain, Belgium
Outline

• What
  ➢ OpenInterface goals
• Why build a new platform
• How
  ➢ Platform requirements
  ➢ Platform design overview
  ➢ Development cycle
  ➢ What OpenInterface is not
• Questions
OpenInterface Goals

• Example: Testing a new user interaction model
  – Available Materials:
    • Several interaction devices and miscellaneous libraries as heterogeneous software code (Matlab, C/C++, Java, Python).

  – Don’t want to:
    • Build glue code yourself (might include substantial code modifications)
    • Think of modality fusion or fission mechanism
Testing a new user interaction model
Testing a new user interaction model

• OpenInterface would:

  – Allow seamless integration of heterogeneous soft
    ➢ middleware
  – Allow rapid prototyping of new multimodal application
    ➢ Bundled generic fission and fusion mechanism
    ➢ Easy software connection
Outline

✓ What
  ➢ OpenInterface goals
  • Why build a new platform
Why building a new Platform?

- **State-of-the-art**
  - Technology to achieve heterogeneous program communication. (CORBA, EJB, various middleware)
  - Multimodal platform. (Galaxy, Kirusa, etc.)

- **Why**
  - Middleware are heavy-weighted, or technology is platform dependent.
  - Multimodal platforms come with fixed number of modalities.
Outline

✓ Why a new platform
  • How
    ➢ Platform requirements
    ➢ Platform design overview
    ➢ Development cycle
    ➢ What OpenInterface is not
Platform Requirements

- Useful high level programming language support:
  - C/C++, Java, Matlab, Python

- Plug n Play behavior.

- Lightweight skeleton platform.
Platform Requirements

- Example of front-end applications
  - Components database for targeted purpose
    - User Interaction
    - Benchmarking
    - ...
  - « Just In Time » platform for heterogeneous software composition.
  - ...

7/22/05
How

✓ Platform requirements
➢ Platform design overview
➢ Development cycle
➢ What OpenInterface is not
Platform Design Overview

• Component Oriented.
• Easy Heterogeneous components Integration
  – Component Interface Description Language
  – Proxies and Stubs
• Multimodal interaction and multimodal data oriented feature
  – Bundles Fission and Fusion components
• Component Composition
  – Pipeline Description and Configuration Language.
Component Architecture

• Step towards reusable software.
• Components behave like objects.
• Software delivered as reusable independent unit with exported/imported I/O interfaces.
Heterogeneous components Integration

• Standard description of interfaces and properties.
• Automatic conversion towards a common language.
• For the kernel, all components are virtually in the same language.
Multimodal aware

- Extended-CIDL to express Fission and Fusion Properties.

- Generic Fission and Fusion mechanism bundled with the kernel.
Component Composition

- Pipeline description language to setup a running application.
How

✔ Platform requirements
✔ Platform design overview
➢ Development cycle
➢ What OpenInterface is not
A «rather technical» tutorial is available.
Current Functionalities

• Supported Platform
  – Linux

• Supported Languages (see tutorial for minor restrictions)
  – Java
  – C/C++
  – Matlab (One way communication)

• Description languages
  – CIDL, interfaces description
  – PDCL, pipeline description

• See tutorial for more details. (I’ll be around too)
What OpenInterface is not

- Graphical programming language
  - Mainly aims at connecting component
  - Large interaction granularity
  - Design tool
- A repository of miscellaneous programming tools (socket, etc…)
  - Available tools are the ones that have been integrated by the user.
OpenInterface: SIMILAR platform
openinterface@similar.cc

Lionel Lawson
Université Catholique de Louvain, Belgium
Example CIDL

\[\text{scalar} = \text{matlab\_foo}(b, \text{str})!\]

<Interface type="function">
  <Name value="matlab\_foo"/>
  <Argument>
    <Param name="b">
      <Descr>human readable description for 'b' here</Descr>
      <PrimitiveType name="int"/>
    </Param>
    <Param name="str">
      <PrimitiveType name="string"/>
    </Param>
  </Argument>
  <Return>
    <Param name="ret">
      <PrimitiveType name="float"/>
    </Param>
  </Return>
</Interface>
Example CIDL

```c
float** getArray(int * dims);
```

```xml
.Interface type="function"
  <Name value="getArray"/>
  <Argument>
    <Param name="dims">
      <Descr>array dimensions</Descr>
      <ArrayType>
        <PrimitiveType name="int"/>
        <DimSpec type="custom">
          <NDims value="1"/>
          <DimList>
            <NDims value="2"/>
          </DimList>
        </DimSpec>
      </ArrayType>
    </Param>
  </Argument>
  <Return>
    <Param name="result">
      <ArrayType>
        <PrimitiveType name="float"/>
        <DimSpec type="custom">
          <NDims value="2"/>
          <DimList ref="dims"/>
        </DimSpec>
      </ArrayType>
    </Param>
  </Return>
</Interface>
```
Example Pipeline

Matlab

Image Misc

Simple Viewer

initViewer

Networked Viewer

Java

[Diagram of a pipeline with nodes labeled Matlab, Image Misc, Simple Viewer, initViewer, Networked Viewer, and Java, connected by arrows indicating the flow of data.]
Example Pipeline

<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE Pipeline SYSTEM
"/home/lawson/work-development/openinterface/prototype2/common/pipeline.dtd">
<Pipeline>
  <FacetList>
    <Facet id="simpleViewer" name="simpleViewer" config="ImageViewer_cfg0.xml"/>
    <Facet id="imgserver" name="imgserver" config="testmatlab_cfg0.xml"/>
    <Facet id="imgdisplay" name="imgdisplay" config="testmatlab_cfg0.xml"/>
  </FacetList>
  <PinList>
    <Pin id="pin7" facet="simpleViewer" name="imageReader"/>
    <Pin id="pin8" facet="imgserver" name="imageserver"/>
    <Pin id="pin9" facet="imgdisplay" name="imagedisplay"/>
    <Pin id="pin10" facet="simpleViewer" name="imageDisplay"/>
    <Pin id="pin11" facet="simpleViewer" name="injectArray"/>
    <Pin id="pin12" facet="ctest" name="testc_fun"/>
  </PinList>
  <Pipe>
    <Plug source="pin7" sink="pin8">
      <Filter>
        <TargetValue target="filename">
          <SourceValue source="imagefile"/>
        </TargetValue>
      </Filter>
      <OutFilter>
        <TargetValue target="pixels">
          <SourceValue source="image"/>
        </TargetValue>
      </OutFilter>
    </Plug>
  </Pipe>
</Pipeline>
Example Pipeline

<Plug source="pin10" sink="pin9">
  <Filter>
    <TargetValue target="imArray">
      <SourceValue source="pixels"/>
    </TargetValue>
  </Filter>
</Plug>

<Plug source="pin11" sink="pin12">
  <Filter>
    <TargetValue target="dims">
      <SourceValue source="dims"/>
    </TargetValue>
    <TargetValue target="array">
      <SourceValue source="data"/>
    </TargetValue>
  </Filter>
</Plug>
</Pipe>
</Pipeline>