Inter-Widget Communication

Lecture series:
Mash-up Personal Learning Environments

Ivan Zuzak
University of Zagreb, Croatia
Lecture Outline

- Motivation

- Inter-Widget Communication and Web Browsers

- Dimensions of Inter-Widget Communication

- Example Systems

- Requirements Case Study

- Future Challenges and Conclusion
Motivation
Motivation

- Personal Learning Environments (PLEs)
  - Learning is dynamic and personalized
  - Flexible
    - Choice of learning elements
    - Interaction of learning elements
Motivation

- Mash-up PLEs
  - *Web widgets* as learning elements
    - Mash-ups as the methodology
    - Learning process = coordination of widget activities
  - Widget-to-widget communication
    - Automated execution of learning process
    - Social collaboration
Motivation

- Remember *Operating systems 101*?
  - Inter-process communication
  - Sockets, named pipes, …
- Inter-widget communication is similar
  - Higher level of abstraction
  - “*The browser is the new OS*”
Inter-Widget Communication and Web Browsers
Inter-Widget Communication

What is a widget?

- Intuitive definition
  - Self-contained functionality...
  - … accessible through a *small* GUI …
  - … embedded in a Web application
  - A “*mini Web application*”

- Operational, software engineering definition?
Web Application Execution Model
Web Application Execution Model

Web server

Web browser
Web Application Execution Model

Web server

URL

Web browser
Web Application Execution Model

HTML+JS+CSS → Web server → Web browser

URL
Web Application Execution Model

HTML+JS+CSS

Web server

URL

Web browser

Window context A.1 – Window or tab – Url\textsubscript{A.1} (HTML+CSS rendering, JavaScript execution)

Web application A
Web Application Execution Model

HTML+JS+CSS → Web server

URL → Web browser

Window context A.1 – Window or tab – URL

(HTML+CSS rendering, JavaScript execution)

Web application A

Web application B – URL

Web application N – URL
Web Application Execution Model

HTML+JS+CSS

Web server

URL

Window context A.1 – Window or tab – $Url_{A.1}$

(HTML+CSS rendering, JavaScript execution)

Window context A.1.1 – Iframe – $Url_{A.1.1}$

Window context A.1.2 – Iframe – $Url_{A.1.2}$

Window context A.1.3 – Iframe – $Url_{A.1.3}$

Web application A

Web application B – $Url_B$

Web application N – $Url_N$
Web Application Execution Model

- HTML+JS+CSS
- Web server
- URL

Web browser

Window context A.1 – Window or tab – $Url_{A.1}$
(HTML+CSS rendering, JavaScript execution)

- Window context A.1.1 – Iframe – $Url_{A.1.1}$
- Window context A.1.2 – Iframe – $Url_{A.1.2}$
- Window context A.1.1.3 – Iframe – $Url_{A.1.1.3}$

Web application A

Web application B – $Url_B$

Web application N – $Url_N$
Web Application Execution Model

// HTML source for Web application A
// loaded from http://urlA.com/

<html>
   <head> ... </head>
   <body>
      <div> Container content </div>
      <iframe src="http://wikipedia.com/" />
      <iframe src="http://google.com/" />
      <iframe src="http://kmi.open.ac.uk/" />
   </body>
</html>
Web Application Execution Model

- Web Application
  - Tree hierarchy of window contexts
  - Top-level contexts – Browser windows/tabs
  - Nested contexts – Iframes
Web Application Execution Model

- Web Application
  - Tree hierarchy of window contexts
  - Top-level contexts – Browser windows/tabs
  - Nested contexts – Iframes
- Window contexts
  - HTML rendering and JavaScript execution
  - Origin-based isolation (Same origin policy)
    - http://wikipedia.com/widget
    - http://kmi.open.ac.uk/widget
    - Cross-origin interaction is restricted
Web Application Execution Model

- Two approaches for implementing widgets and IWC

  - **Iframe contexts** (more common)
    - Widget
      - Iframe context, temporary entity in a Web browser
    - Widget specification
      - HTML+CSS+JS resources stored on a Web server
    - Inter-widget communication
      - Data transfer between two contexts
Web Application Execution Model

- Two approaches for implementing widgets and IWC

- **Iframe contexts** (more common)
  - Widget
    - Iframe context, temporary entity in a Web browser
  - Widget specification
    - HTML+CSS+JS resources stored on a Web server
  - Inter-widget communication
    - Data transfer between two contexts
- Virtual contexts (recent research)
Dimensions of Inter-Widget Communication
IWC Dimensions

- Inter-widget communication system
  - Part of a MUPPLE which supports IWC
  - Browser primitives, JavaScript libraries
IWC Dimensions

- Inter-widget communication **system**
  - Part of a MUPPLE which supports IWC
  - Browser primitives, JavaScript libraries

- Common questions
  - How to express requirements for IWC?
  - What are each IWC system's properties?
  - How to choose an IWC system?
IWC Dimensions

- Software architecture approach
- System design space
IWC Dimensions

- Software architecture approach
- System design space
  - Multidimensional space
IWC Dimensions

- Software architecture approach
- System design space
  - Multidimensional space
  - Dimensions describe system characteristics
  - Dimension values correspond to alternatives
IWC Dimensions

- Software architecture approach
- System design space
  - Multidimensional space
  - Dimensions describe system characteristics
  - Dimension values correspond to alternatives
  - A specific system corresponds to a point in the design space
IWC Dimensions

1) Examine existing IWC systems
2) Define IWC design space
IWC Dimensions

1) Examine existing IWC systems
2) Define IWC design space
3) Define MUPPLE requirements for IWC (dimension bounds)
IWC Dimensions

1) Examine existing IWC systems
2) Define IWC design space
3) Define MUPPLE requirements for IWC (dimension bounds)
4) Obtain acceptable systems
IWC Dimensions

1) Examine existing IWC systems
2) Define IWC design space
3) Define MUPPLE requirements for IWC (dimension bounds)
4) Obtain acceptable systems
   - 100% match – choose one acceptable system
   - <100% match – extend some existing system
IWC Dimensions

- User experience and high-level concerns
- Performance
- Security
- Basic communication concerns (Programming level)
IWC Dimensions

<table>
<thead>
<tr>
<th>Maximum message length</th>
<th>Reliability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Underlying transport system</td>
<td>Distribution scheme</td>
</tr>
<tr>
<td>Naming</td>
<td>Discovery</td>
</tr>
<tr>
<td>Cross-origin support</td>
<td>Cross-application support</td>
</tr>
<tr>
<td>Type of system</td>
<td>Communication model</td>
</tr>
</tbody>
</table>

User experience and high-level concerns

Performance

Security

Basic communication concerns (Programming level)
IWC Dimensions

<table>
<thead>
<tr>
<th>User experience and high-level concerns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Performance</td>
</tr>
<tr>
<td>Security</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Basic communication concerns (Programming level)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authorization of sender and receiver</td>
</tr>
<tr>
<td>Authentication of sender and receiver</td>
</tr>
<tr>
<td>Communication integrity</td>
</tr>
<tr>
<td>Communication confidentiality</td>
</tr>
<tr>
<td>Maximum message length</td>
</tr>
<tr>
<td>Reliability</td>
</tr>
<tr>
<td>Underlying transport system</td>
</tr>
<tr>
<td>Distribution scheme</td>
</tr>
<tr>
<td>Naming</td>
</tr>
<tr>
<td>Discovery</td>
</tr>
<tr>
<td>Cross-origin support</td>
</tr>
<tr>
<td>Cross-application support</td>
</tr>
<tr>
<td>Type of system</td>
</tr>
<tr>
<td>Communication model</td>
</tr>
</tbody>
</table>
IWC Dimensions

- Latency
- Size

| Authorization of sender and receiver | Authentication of sender and receiver |
| Communication integrity            | Communication confidentiality        |

<table>
<thead>
<tr>
<th>Maximum message length</th>
<th>Reliability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Underlying transport system</td>
<td>Distribution scheme</td>
</tr>
<tr>
<td>Naming</td>
<td>Discovery</td>
</tr>
<tr>
<td>Cross-origin support</td>
<td>Cross-application support</td>
</tr>
<tr>
<td>Type of system</td>
<td>Communication model</td>
</tr>
</tbody>
</table>

- User experience and high-level concerns
- Performance
- Security
- Basic communication concerns (Programming level)
### IWC Dimensions

<table>
<thead>
<tr>
<th>User involvement</th>
<th>Visual display of communication</th>
<th>Semantic interoperability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Latency</td>
<td>Size</td>
<td></td>
</tr>
<tr>
<td>Authorization of sender and receiver</td>
<td>Authentication of sender and receiver</td>
<td></td>
</tr>
<tr>
<td>Communication integrity</td>
<td>Communication confidentiality</td>
<td></td>
</tr>
</tbody>
</table>

#### Basic communication concerns (Programming level)

- Maximum message length
- Underlying transport system
- Naming
- Cross-origin support
- Type of system

#### Security

- Communication confidentiality
- Communication integrity
- Authorization of sender and receiver
- Authentication of sender and receiver

#### Performance

- Reliability
- Distribution scheme
- Discovery

#### User experience and high-level concerns

- User involvement
- Visual display of communication
- Semantic interoperability
## IWC Dimensions

<table>
<thead>
<tr>
<th>Latency</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authorization of sender and receiver</td>
<td>Authentication of sender and receiver</td>
</tr>
<tr>
<td>Communication integrity</td>
<td>Communication confidentiality</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Maximum message length</th>
<th>Reliability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Underlying transport system</td>
<td>Distribution scheme</td>
</tr>
<tr>
<td><strong>Naming</strong></td>
<td><strong>Discovery</strong></td>
</tr>
<tr>
<td>Cross-origin support</td>
<td>Cross-application support</td>
</tr>
<tr>
<td><strong>Type of system</strong></td>
<td>Communication model</td>
</tr>
</tbody>
</table>

- **User involvement**
- **Visual display of communication**
- **Semantic interoperability**

**User experience and high-level concerns**

**Performance**

**Security**

**Basic communication concerns (Programming level)**
IWC Dimensions

- Type of IWC system (IWC system span)

- Web browser
- Web server
IWC Dimensions

- Type of IWC system (IWC system span)

Web browser

In-browser mechanism

Web server

Browser primitives
IWC Dimensions

- Type of IWC system (IWC system span)

Web browser

- In-browser mechanism
- Client-side logic

Web server

Browser primitives

Client-side frameworks
IWC Dimensions

- Type of IWC system (IWC system span)

![Diagram showing IWC Dimensions]

- Web browser
  - In-browser mechanism
  - Client-side logic

- Web server
  - Server-side coordination logic

- Browser primitives
- Client-side frameworks
- Server-mediated coordination frameworks
IWC Dimensions

- Type of IWC system (IWC system span)

**Web browser**

- In-browser mechanism
- Client-side logic

**Web server**

- Server-side coordination logic
- Server-side data transfer

Browser primitives

Client-side frameworks

Server-mediated coordination frameworks

Server-mediated communication frameworks
IWC Dimensions

- Communication model
IWC Dimensions

- Communication model

**Message-oriented**

Sender widget \(\xrightarrow{\text{Send msg}}\) Receiver widget

**Shared memory**

Sender widget \(\xrightarrow{\text{Write}}\) Shared space \(\xrightarrow{\text{Read}}\) Receiver widget
IWC Dimensions

- Communication model

**Message-oriented**

Sender widget → **Send msg** → Receiver widget

**Remote procedure call (RPC)**

Sender widget → **Invoke proc** → Receiver widget

**Shared memory**

Sender widget → **Write** → Receiver widget

**Publish-subscribe (pubsub)**

Virtual channel → **Publish** → Sender widget

Receiver widget → **Subscribe** → Receiver widget

Receiver widget → **Notify** → Receiver widget
IWC Dimensions

- Naming
  - How are widgets addressed when using the IWC system?
IWC Dimensions

• Naming
  • How are widgets addressed when using the IWC system?
  • *JavaScript object references to contexts*
    - A reference to the window context, with an API
    - Contexts have references to parent and nested contexts
      • `window.parent`, `window.frames`
IWC Dimensions

- Naming
  - How are widgets addressed when using the IWC system?
  - *JavaScript object references to contexts*
    - A reference to the window context, with an API
    - Contexts have references to parent and nested contexts
      - `window.parent`, `window.frames`
  - *Custom strings*
    - E.g. “*map coordinates*” for publish-subscribe channel names, shared memory location or remote procedure name
    - Resolved internally by IWC system
IWC Dimensions

- Discovery support
  - Widget names may not be known in advance
  - Widgets may be added/removed at runtime
IWC Dimensions

- Discovery support
  - Widget names may not be known in advance
  - Widgets may be added/removed at runtime
  - No support
  - Supported
    - Traversal of Web application context hierarchy
    - Hidden communication between contexts
      - RPC and pubsub channel discovery
IWC Dimensions

• Reliability support
  • Widgets may be added/removed/changed at runtime
    - Receiver may not be fully loaded yet
    - Procedures (RPC) or channels (pubsub) may be added dynamically
IWC Dimensions

- Reliability support
  - Widgets may be added/removedchanged at runtime
    - Receiver may not be fully loaded yet
    - Procedures (RPC) or channels (pubsub) may be added dynamically
  - No support
  - Handshake + queuing
    - Widget-level handshake
    - Message-level handshake
Example Inter-Widget Communication Systems
Example IWC Systems

- Today's IWC ecosystem
  - More than 30 browser primitives and client-side frameworks
  - Many more server-side frameworks
    - Twitter or Facebook messages for IWC? Why not?
Example IWC Systems

- HTML5 postMessage API
  - Standardized browser primitive

<table>
<thead>
<tr>
<th>Browser</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://sender.origin.com/widget">http://sender.origin.com/widget</a></td>
<td>Sender widget code</td>
</tr>
<tr>
<td><a href="http://receiver.origin.com/widget">http://receiver.origin.com/widget</a></td>
<td>Receiver widget code</td>
</tr>
</tbody>
</table>
Example IWC Systems

- HTML5 postMessage API
  - Standardized browser primitive

```javascript
http://sender.origin.com/widget

function handler(msg, sender) {
  // verify sender origin;
  // deserialize and use msg;
}

http://receiver.origin.com/widget

postMessage.attachHandler(handler);
```

**Sender widget code**

**Receiver widget code**
Example IWC Systems

- HTML5 postMessage API
  - Standardized browser primitive

```
http://sender.origin.com/widget

*Sender widget code*

postMessage.sendMessage("mapcoordinates#31.0#45.0", destinationContextObject, "http://destination.origin.com");

http://receiver.origin.com/widget

*Receiver widget code*

postMessage.attachHandler(handler);

function handler(msg, sender) {
  // verify sender origin;
  // deserialize and use msg;
}
```
Example IWC Systems

- HTML5 postMessage API
  - Communication model – Message-oriented system
    - String messages
  - Naming – Destination context object
  - Cross-origin support
  - Secure – Origin-based authorization
  - No discovery support
  - No reliability support
Example IWC Systems

- jsChannel
  - Client-side framework – Built on top of postMessage

Browser

<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Sender widget code</em></td>
<td><em>Receiver widget code</em></td>
</tr>
</tbody>
</table>
Example IWC Systems

- jsChannel
  - Client-side framework – Built on top of postMessage

```javascript
http://sender.origin.com/widget

Sender widget code

http://receiver.origin.com/widget

Receiver widget code

jsChannel.attachRPCHandler("refreshMap", origin, handler);

function handler(parameters) {
  // use parameters;
  return result;
}
```
Example IWC Systems

- **jsChannel**
- Client-side framework – Built on top of postMessage

### Sender widget code
```
jsChannel.invokeProcedure(
    "refreshMap", // procedure name
    ["31.0", "45.0"], // parameters
    destinationContextObject,
    "http://destination.origin.com",
    returnValueHandler
);

function returnValueHandler(result) {
    // use result
}
```

### Receiver widget code
```
jsChannel.attachRPCHandler(
    "refreshMap", origin, handler);

function handler(parameters) {
    // use parameters;
    return result;
}
```
Example IWC Systems

- jsChannel
  - Communication model – Remote procedure call
    - Procedures, procedure call parameters
  - Naming – Destination context object, procedure name
  - Secure – Origin-based authorization
  - No discovery support
  - Reliability support
    - Widget-level handshake
Example IWC Systems

- Faye – Messaging server and browser library
  - Requires a server component for message routing
Example IWC Systems

- Faye – Messaging server and browser library
- Requires a server component for message routing

Server

Browser A

http://receiver.origin.com/widget

Receiver widget code

Faye.subscribeToChannel(serverUrl, "map coordinates", handler);

Browser B

http://sender.origin.com/widget

Sender widget code
Example IWC Systems

- Faye – Messaging server and browser library
  - Requires a server component for message routing

```
Browser A
http://receiver.origin.com/widget

Faye.subscribeToChannel(serverUrl, "map coordinates", handler);

Browser B
http://sender.origin.com/widget

Faye.publishToChannel(serverUrl, "map coordinates", "31.0, 45.0");
```
Example IWC Systems

- Faye
  - Server-mediated communication framework
    - Ajax for browser-server communication
  - Communication model – Publish-subscribe
    - String messages on channels
  - Naming – String channel name
  - Cross-origin support
  - Security
    - Password protected channels
    - HTTPS communication with server
  - No discovery support
  - No reliability support
Example IWC Systems

- Analysis results of over 30 IWC systems
  - Browser primitives, client-side and server-mediated coordination frameworks
Example IWC Systems

- Analysis results of over 30 IWC systems
- Browser primitives, client-side and server-mediated coordination frameworks

![Bar chart showing the number of systems for different communication models:]

- Msg: 17 systems
- Sh. mem.: 8 systems
- RPC: 8 systems
- Pubsub: 6 systems

![Pie chart showing the number of communication models supported by systems:]

- 1 model: 25 systems
- 2 models: 4 systems
- 3 models: 2 systems
Example IWC Systems

- Analysis results of over 30 IWC systems
Requirements Case Study for Inter-Widget Communication
IWC Requirements Case Study

- Geppeto
  - Consumer programming environment
  - User-defined application logic based on GUI actions
IWC Requirements Case Study

- Geppeto
  - Consumer programming environment
  - User-defined application logic based on GUI actions
IWC Requirements Case Study

- Geppeto
  - Consumer programming environment
  - User-defined application logic based on GUI actions
IWC Requirements Case Study

- Geppeto
  - Consumer programming environment
  - User-defined application logic based on GUI actions
IWC Requirements Case Study

- Geppeto
  - Spreadsheet view of user-defined logic

<table>
<thead>
<tr>
<th>wait for click Planvoznje1 at ivan2</th>
<th>wait for click Osvjezi at ivan2</th>
</tr>
</thead>
<tbody>
<tr>
<td>text@ivan2 =&gt; text@ZET-Planvoznje</td>
<td>click Osvjezi at GPSpolozaj</td>
</tr>
<tr>
<td>click Osvjezi at GPSpolozaj</td>
<td>copy element3 at GPSpolozaj to element3 at ivan2</td>
</tr>
<tr>
<td>gpsaddress@GPSpolozaj =&gt; text1@ZET-Planvoznje</td>
<td></td>
</tr>
<tr>
<td>click Planvoznje@ZET-Planvoznje</td>
<td></td>
</tr>
<tr>
<td>element2@ZET-Planvoznje =&gt; element2@ivan2</td>
<td></td>
</tr>
<tr>
<td>copy element3 at GPSpolozaj to element3 at ivan2</td>
<td></td>
</tr>
</tbody>
</table>
IWC Requirements Case Study

- Geppeto – IWC requirements
IWC Requirements Case Study

- Geppeto – IWC requirements
  - *Type of system* – Browser primitive or client-side framework
IWC Requirements Case Study

• Geppeto – IWC requirements
  • *Type of system* – Browser primitive or client-side framework
  • *Communication model* – Remote procedure call
  • *Distribution scheme* – 1:1 unicast
IWC Requirements Case Study

- Geppeto – IWC requirements
  - *Type of system* – Browser primitive or client-side framework
  - *Communication model* – Remote procedure call
  - *Distribution scheme* – 1:1 unicast
  - *Maximum message length* – Unlimited
IWC Requirements Case Study

- Geppeto – IWC requirements
  - *Type of system* – Browser primitive or client-side framework
  - *Communication model* – Remote procedure call
  - *Distribution scheme* – 1:1 unicast
  - *Maximum message length* – Unlimited
  - *Cross-origin communication* – Yes
IWC Requirements Case Study

- Geppeto – IWC requirements
  - Type of system – Browser primitive or client-side framework
  - Communication model – Remote procedure call
  - Distribution scheme – 1:1 unicast
  - Maximum message length – Unlimited
  - Cross-origin communication – Yes
  - Naming – Visible widget names (string) + procedure names (string)
    - Translation from widget names to context references
IWC Requirements Case Study

- Geppeto – IWC requirements
  - *Type of system* – Browser primitive or client-side framework
  - *Communication model* – Remote procedure call
  - *Distribution scheme* – 1:1 unicast
  - *Maximum message length* – Unlimited
  - *Cross-origin communication* – Yes
  - *Naming* – Visible widget names (string) + procedure names (string)
    - Translation from widget names to context references
  - *Discovery* – No preference
  - *Reliability* – Yes
IWC Requirements Case Study

- Geppeto – IWC requirements
  - *Type of system* – Browser primitive or client-side framework
  - *Communication model* – Remote procedure call
  - *Distribution scheme* – 1:1 unicast
  - *Maximum message length* – Unlimited
  - *Cross-origin communication* – Yes
  - *Naming* – Visible widget names (string) + procedure names (string)
    - Translation from widget names to context references
  - *Discovery* – No preference
  - *Reliability* – Yes
  - *Authorization of sender and receiver* – Yes
Future Challenges and Conclusion
Virtual Contexts

- Driven by security concerns
  - Google Caja, ECMAScript5 strict mode
Virtual Contexts

- Driven by security concerns
  - Google Caja, ECMAScript5 strict mode
- Virtual context
  - Sanitized, *safe* subset of context content (HTML, CSS and JavaScript)
  - Embedded within host window context
Virtual Contexts

- Driven by security concerns
  - Google Caja, ECMAScript5 strict mode
- Virtual context
  - Sanitized, safe subset of context content (HTML, CSS and JavaScript)
  - Embedded within host window context
  - Interaction with environment possible only if given references by host context
    - IWC = method invocation
User Involvement in IWC

- Degree of user intervention required for IWC
  - a) Integrated into widgets, no user involvement required
  - b) Requires some degree of user involvement
User Involvement in IWC

- Degree of user intervention required for IWC
  - a) Integrated into widgets, no user involvement required
  - b) Requires some degree of user involvement
    - Drag-and-drop of elements
    - Wiring of elements (e.g. Yahoo Pipes)
    - Visual proximity of widgets
    - User-defined GUI-level scripts (e.g. Geppeto)
Conclusion

- Inter-widget communication in MUPPLEs
  - Tied to Web application architecture
  - Many dimensions for IWC system classification
  - Large ecosystem of existing systems

- Future research
  - Security and performance dimensions
  - User experience dimensions
  - Semantic interoperability
Resources

- Virtual contexts
  - http://code.google.com/p/google-caja/
- HTML5 Web Messaging (postMessage API)
  - http://www.w3.org/TR/webmessaging/
- Software architecture design space
- Usability
  - Isaksson, Palmer: “Usability and Inter-widget Communication in PLE”, MUPPLE'10
- List of existing IWC systems and systematization
Questions?

Do we need more powerful IWC systems?
What about peer-to-peer inter-browser IWC communication?
But, I've heard that there are “desktop widgets” and “web widgets”?
Is IWC being standardized?

Thank you!

Ivan Zuzak
izuzak@gmail.com
@izuzak