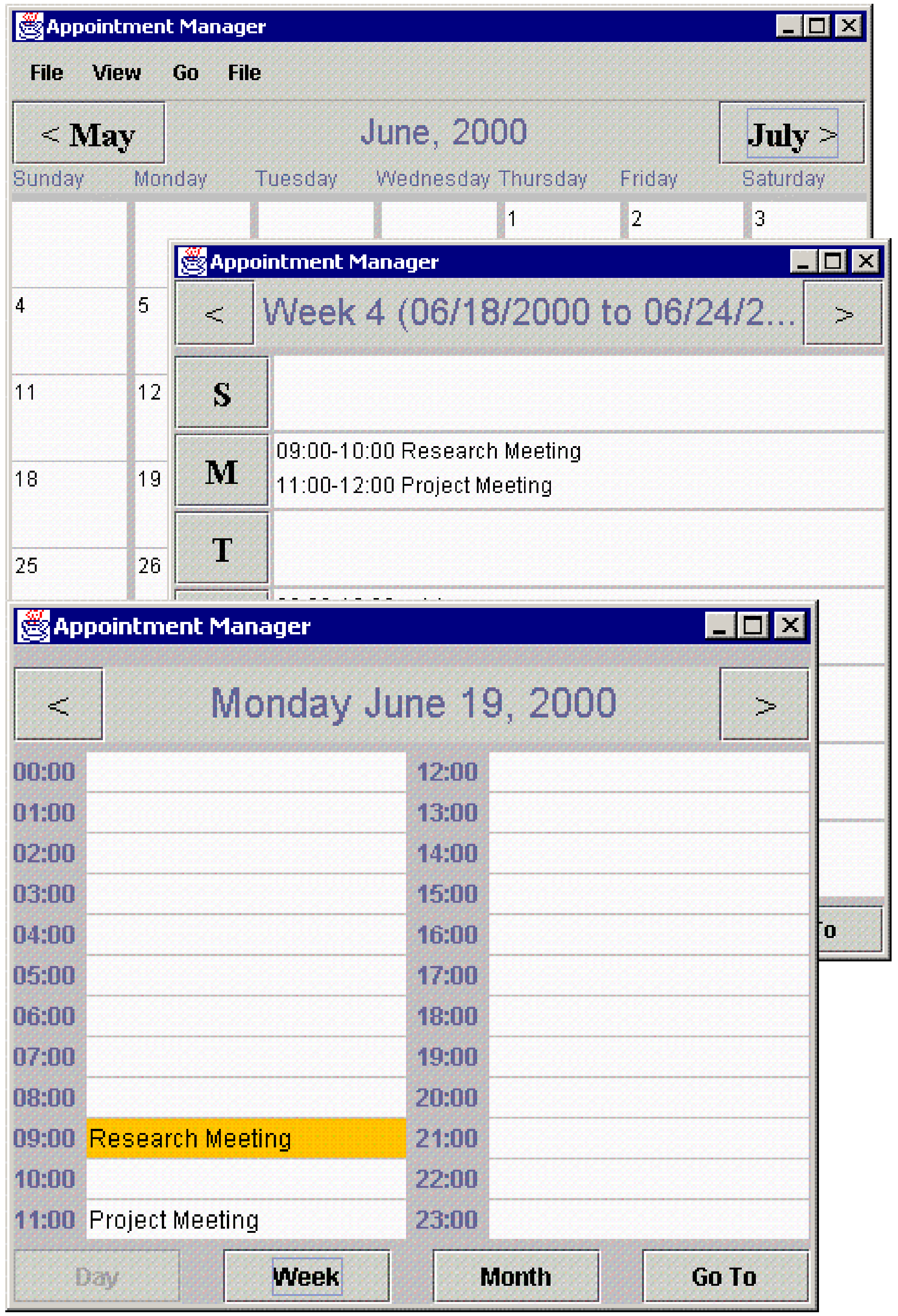
UIML
User Interface Markup Language

Introduction and Overview

Gabriel Vögler

Motivation: One Application, Multiple User Interfaces



Appointment Manager

File View Go File

< May | June, 2000 | July >

Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
				1	2	3
4	5					
11	12					
18	19					
25	26					

Appointment Manager

< Week 4 (06/18/2000 to 06/24/2... >

S

M 09:00-10:00 Research Meeting
11:00-12:00 Project Meeting

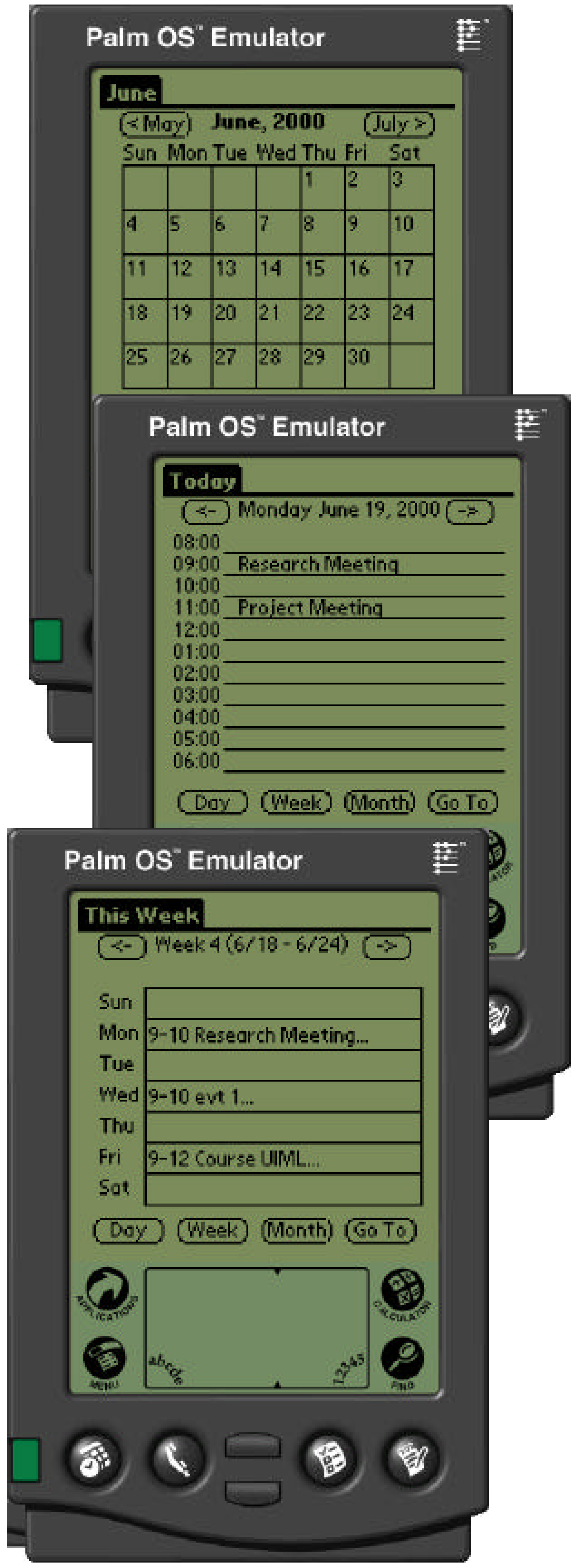
T

Appointment Manager

< Monday June 19, 2000 >

00:00		12:00	
01:00		13:00	
02:00		14:00	
03:00		15:00	
04:00		16:00	
05:00		17:00	
06:00		18:00	
07:00		19:00	
08:00		20:00	
09:00	Research Meeting	21:00	
10:00		22:00	
11:00	Project Meeting	23:00	

Day Week Month Go To



Palm OS Emulator

June

(< May) June, 2000 (July >)

Sun	Mon	Tue	Wed	Thu	Fri	Sat
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	

Palm OS Emulator

Today

(<-) Monday June 19, 2000 (->)

08:00
09:00 Research Meeting
10:00
11:00 Project Meeting
12:00
01:00
02:00
03:00
04:00
05:00
06:00

(Day) (Week) (Month) (Go To)

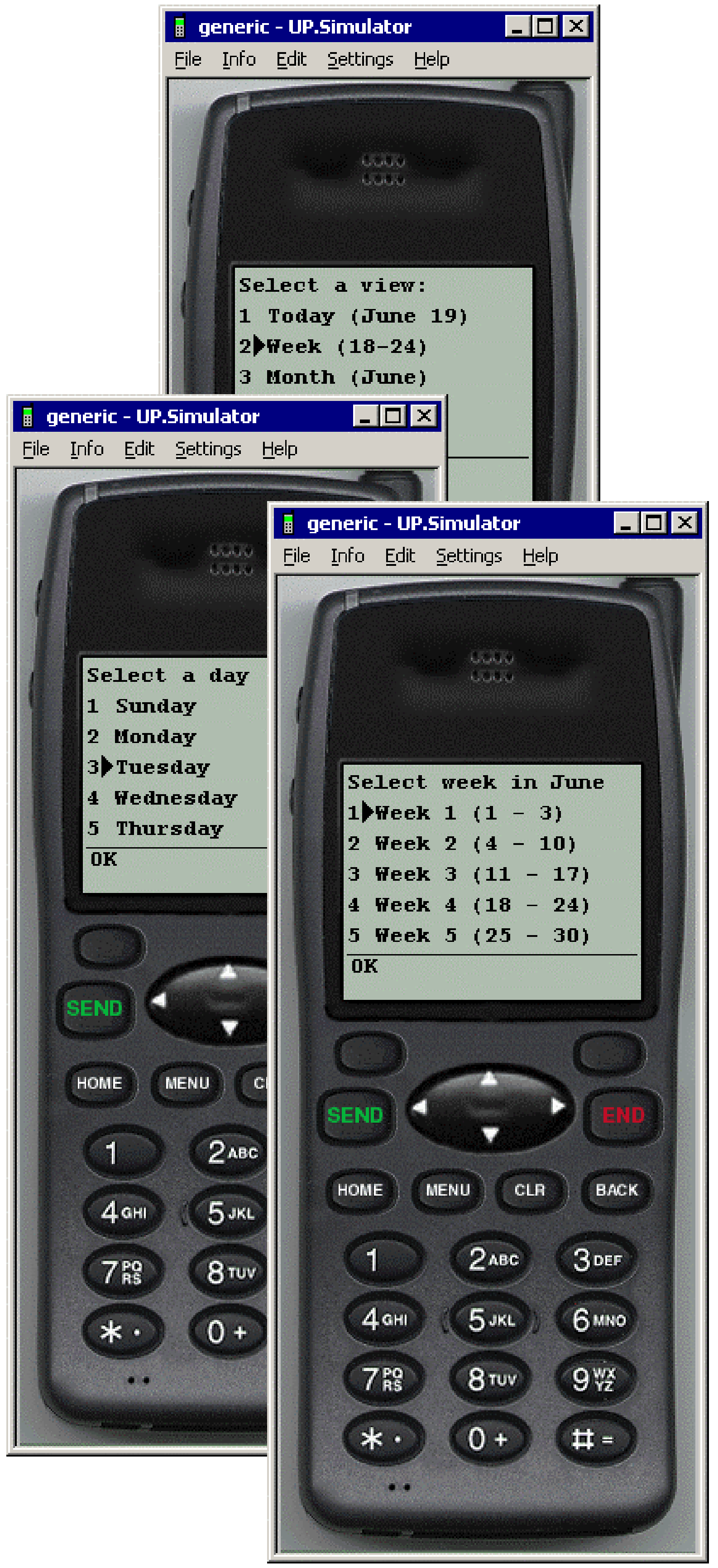
Palm OS Emulator

This Week

(<-) Week 4 (6/18 - 6/24) (->)

Sun
Mon 9-10 Research Meeting...
Tue
Wed 9-10 evt 1...
Thu
Fri 9-12 Course UIML...
Sat

(Day) (Week) (Month) (Go To)



generic - UP.Simulator

File Info Edit Settings Help

```
Select a view:
1 Today (June 19)
2▶Week (18-24)
3 Month (June)
```

generic - UP.Simulator

File Info Edit Settings Help

```
Select a day
1 Sunday
2 Monday
3▶Tuesday
4 Wednesday
5 Thursday
OK
```

generic - UP.Simulator

File Info Edit Settings Help

```
Select week in June
1▶Week 1 (1 - 3)
2 Week 2 (4 - 10)
3 Week 3 (11 - 17)
4 Week 4 (18 - 24)
5 Week 5 (25 - 30)
OK
```

What is UIML?

One language to create UI for

- Any device
- Any language (Java, HTML, ...)
- Any operating system
- Any UI metaphor

UIML is a declarative, XML-compliant <u>meta language</u> for describing user interfaces

Background

- Work started 1997
- First idea by Marc Abrams at Virginia Tech, HCI faculty
- PhD dissertation about UIML by Phanouriou Constantinos
- Current version is UIML 2, UIML 3 available as draft (http://www.uiml.org)
- Standardization at OASIS in the future

UIML Spans Multiple Worlds

Web UIs (e.g., HTML)

Desktop UIs

(e.g., Java, C++, Visual Basic)

Handheld UIs

(e.g., WAP phones)

Voice UIs

Soon: multi-modal UIs

(e.g., Voice + GUI)

UIML Provides Canonical Representation of UIs

There are so many syntaxes for UIs!

Java:

```
JButton jb = new JButton(“Go”);
```

HTML:

```
<INPUT  TYPE= “BUTTON” NAME=“jb” VALUE=“Go!”>
```

XForms:

```
<button><caption>Go</caption></button>
```

UIML uses one syntax for every UI language:

```
<part name=“myButton” class=“Button”/>
<property name=“content”>Go!</property>
```

UIML is a “Meta” Language

UIML

- Doesn’t define tool-kit specific tags (<JTable>,...)
  - <part id=“myButton” class=“????”) / >
- Uses a few powerful tags (<part>, <property>,...)
- Must add toolkit vocabulary to make it useful
- No need to change UIML as new devices invented

Vocabularies

- Defines tool-kit specific class and property names
  - <part id=“myButton” class=“JButton”) / >
- 11 vocabularies available at www.uiml.org
- E.g. Java AWT, Java Swing, HTML, WML VoiceXML
- One generic vocabulary

Special Features

UI metaphor *in*dependent

No <p>, <window>, OnClick, ...

Simple tag set (~2 dozen)

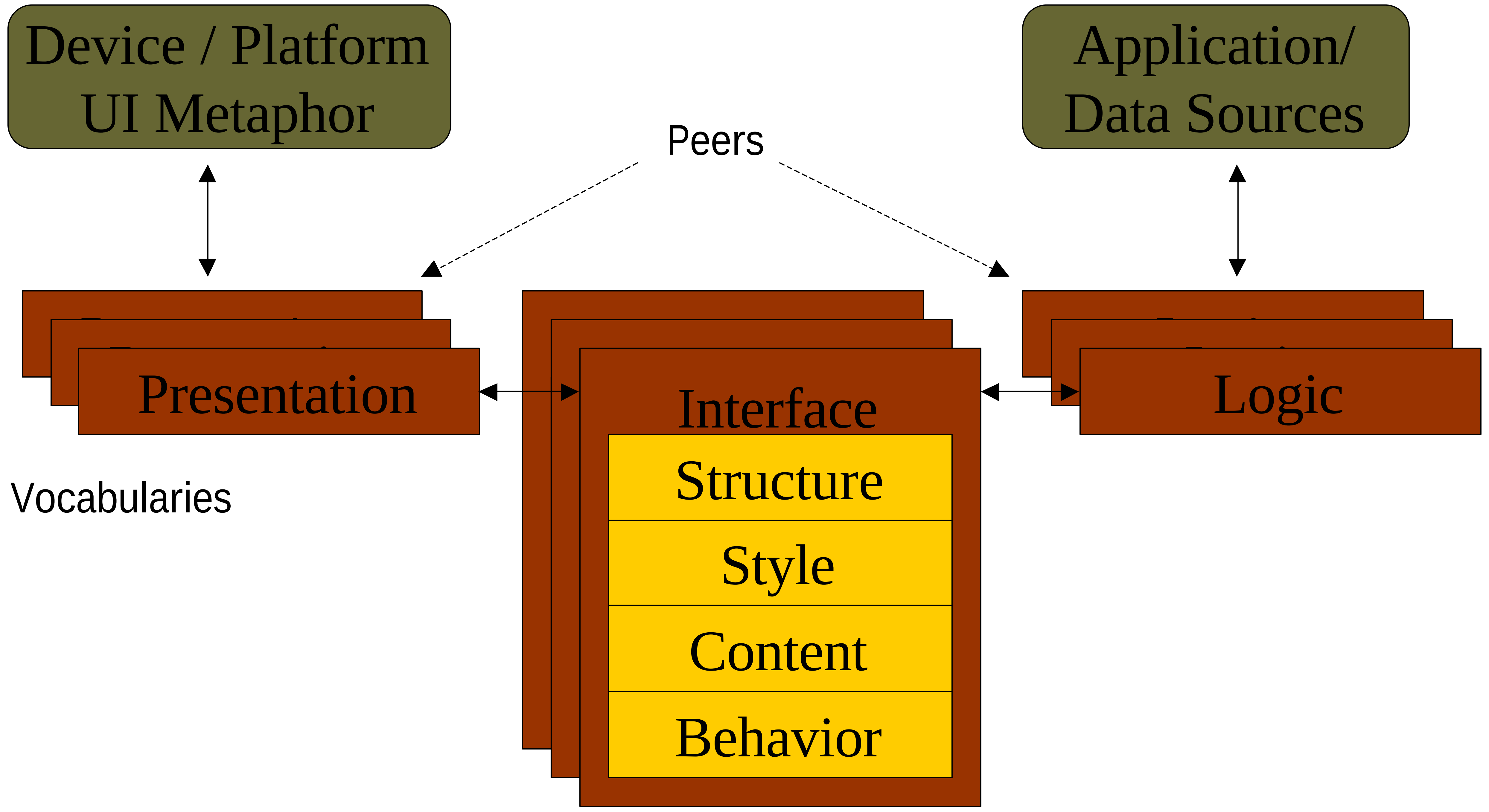
Compiles to everything else

HTML, XHTML, WML, Java, C++, VoiceXML, ...

Separates content to support XML, XSL, internationalization, customized UIs:

HTML	= (structure, content) / style
ideal	= structure / content / style

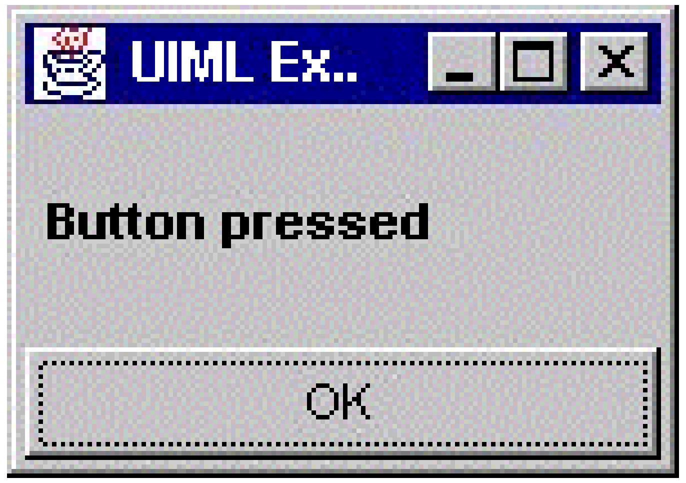
UIML Model



6-way separation of UI description

Example

- Description of a small Java GUI with UIML



- Using the vocabulary for Java, specifying AWT and Swing toolkits

  --> this UIML description is not for multiple platforms

UIML Skeleton

```
<?xml version="1.0">
<uiml>
  <interface>
    <structure>...</structure>
    <style>...</style>
    <content>...</content>
    <behavior>...</behavior>
  </interface>
  <peers>
    <logic>...</logic>
    <presentation>...</presentation>
  </peers>
</uiml>
```

<structure> & <part>

```
<?xml version="1.0">
<uiml>
  <interface>
    <structure>
      <part   id="TopLevel" class="JFrame">
        <part id="L"        class="JLabel"/>
        <part id="Button"   class="JButton"/>
      </part>
    </structure>
    <style>...</style>
    <content>...</content>
    <behavior>...</behavior>
  </interface>
  <peers>
    <logic>...</logic>
    <presentation>...</presentation>
  </peers>
</uiml>
```

What parts comprise the UI?

(logical, hierarchical model of the UI)

<style>

```
<?xml version="1.0">
<uiml>
  <interface>
    <structure>
      <part   id="TopLevel" class="JFrame">...</part>
    </structure>
    <style>
      <property part-name="TopLevel”
                  name=”title">UIML Example</property>
    </style>
    <content>...</content>
    <behavior>...</behavior>
  </interface>
  <peers>
    <logic>...</logic>
    <presentation>...</presentation>
  </peers>
</uiml>
```

What presentation style for each part?

(rendering, font size, color, …)

<content>

What content for each part?

(text, sounds, image, …)

```
<?xml version="1.0">
<uiml>
  <interface>
    <structure>...</structu
    <style>
      <property part-     ="TopLevel” name=”title">
        <reference constant-name=”titleText"/>
      </property>
    </style>
    <content id="German">
      <constant id=”titleText">UIML Beispiel</constant>
    </content>
    <content id=”English">
      <constant id=”titleText">UIML Example</constant>
    </content>
    <behavior>...</behavior>
  </interface>
  <peers>...</peers>
</uiml>
```

<behavior>

```
<?xml version="1.0">
<uiml>
  <interface>
    <structure>
      <part   id="TopLevel" class="JFrame">
        <part id="L"        class="JLabel"/>
        <part id="Button"   class="JButton"/>
      </part>
    </structure>
    <behavior>
      <rule>
        <condition>
          <event class="actionPerformed" part-name="Button"/>
        </condition>
        <action>
          <property part-name="L" name="text">Button pressed
                                                  </property>
        </action>
      </rule>
    </behavior>
  </interface>
  <peers>...</peers>
</uiml>
```

What behavior do parts have?

(UI interaction logic)

<logic>

```
<?xml version="1.0">
<uiml>
  <interface>
    <structure>...</structure>
    <style>...</style>
    <content>...</content>
    <behavior>...</behavior>
  </interface>
  <peers>
    <logic>
      <d-component id="Counter" maps-to="org.something.SimpleCounter">
        <d-method id="increment" return-type="int" maps-to="count"/>
      </d-component>
    </logic>
    <presentation>...</presentation>
  </peers>
</uiml>
```

How to connect to outside world?

(business logic, data sources, UI toolkit)

Can be used as `Counter.incremenet` within `<behavior>`

<presentation>

```
<?xml version="1.0">
<uiml>
  <interface>
 <structure>
      <part   id="TopLevel" class="JFrame">
        <part id="L"         class="JLabel"/>
        <part id="Button"    class="JButton"/>
      </part>
    </structure>
    <style>...</style>
    <content>...</content>
    <behavior>...</behavior>
  </interface>
  <peers>
    <logic>...</logic>
    <presentation base="Java_1.3_Harmonia_1.0”/>
  </peers>
</uiml>
```

Java vocabulary

UI Specification - Summary

What parts comprise the UI?
(logical, hierarchical model of the UI)

What presentation style for each part?
(rendering, font size, color, ...)

What content for each part?
(text, sounds, image, ...)

What behavior do parts have?
(UI interaction logic)

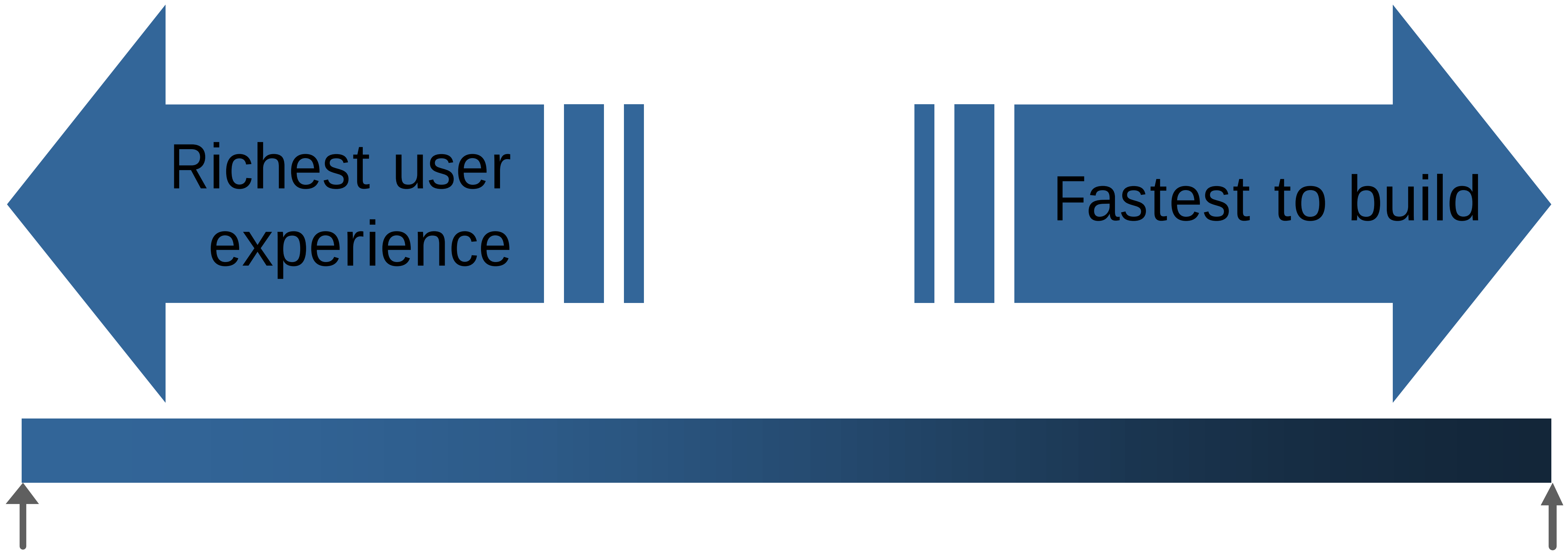
How to connect to outside world?
(business logic, data sources, UI toolkit)

Multi Device User Interfaces

Three ways to achieve multi platform user interfaces:

- **Multiple UIML documents (toolkit specific vocabulary)**
- **Generic vocabulary**
- **Vocabularies for device families**

Permit Continuum of Effort



Use
device-specific
vocabulary:

<part Class="JPopupMenu">

Use
generic
vocabulary:

<part Class="Menu" >

Multiple UIML Documents

- **One document for every platform (using toolkit specific vocabularies)**
- **What is the advantage over traditional approaches?**
  - One language for every platform (no toolkit specific skills needed)
  - One authoring tool for every platform (?)

Generic Vocabulary

- **One vocabulary for all platforms**
- **Two objectives**
  - powerful enough to accommodate a family of devices
  - generic enough to use without having to be an expert in all the various platforms
- **Only a subset of all available GUI elements can be supported**

Sample of Generic Vocabulary

Generic UI Element	Java Swing	PalmOS	WML	HTML
GButton	JButton	Command Button	<input type= “Button”>	<input type= “Button”>
GArea	JFrame | JWindow | JPanel | …	Window	<card>	<form> | <table>
GTop-Container	JFrame	Form	<wml>	<html> and <body>

Each Generic Class Has Properties & Events

Generic Name	GButton
Properties	name, title, size, location, foreground, background, layout, font, border
Events	actionPerformed

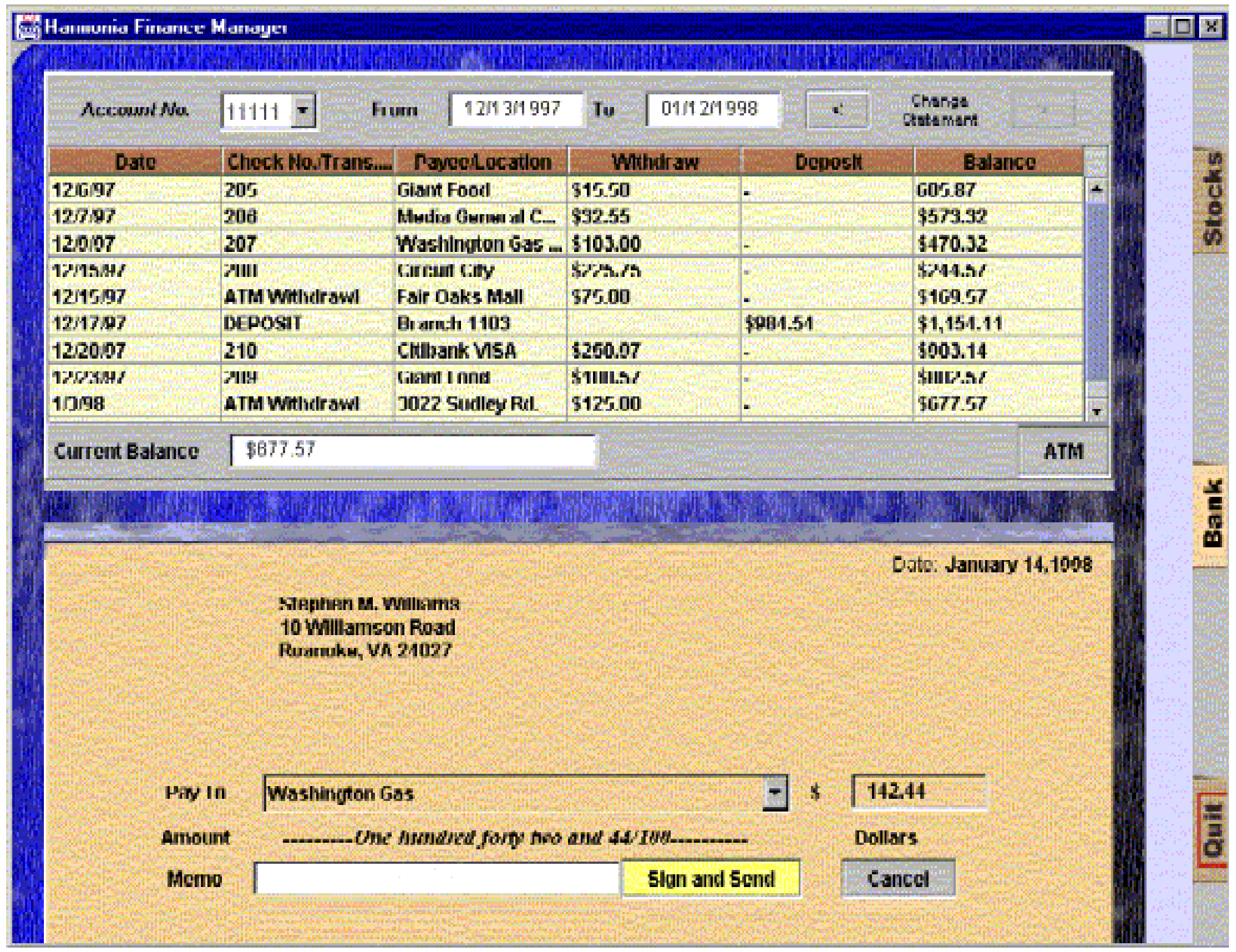
Mapping of Generic Vocabulary to Java and HTML

```
...

<part   id=”myButton" class=”GButton">

...

<presentation name=“Java”>
  <d-class name=”GButton" … maps-to="javax.swing.JButton">
    …
  </d-class >
</presentation>

<presentation name=“HTML”>
  <d-class name="GButton" … maps-to=“html:input">
    …
  </d-class >
</presentation>
```

This stuff is written once, like a device driver for an OS.

Events and calls to outside world handled similarly.

UIML Describes Family of UIs...



Harmonia Finance Manager

Account No. 11111 From 12/13/1997 To 01/12/1998 Change Statement

Date	Check No./Trans...	Payee/Location	Withdraw	Deposit	Balance
12/6/97	205	Giant Food	$15.50	-	605.87
12/7/97	206	Media General C...	$32.55		$573.32
12/9/97	207	Washington Gas ...	$103.00	-	$470.32
12/15/97	208	Circuit City	$225.75	-	$244.57
12/15/97	ATM Withdrawl	Fair Oaks Mall	$75.00	-	$169.57
12/17/97	DEPOSIT	Branch 1103		$984.54	$1,154.11
12/20/97	210	Citibank VISA	$250.07	-	$903.14
12/23/97	209	Giant Food	$100.57	-	$802.57
1/3/98	ATM Withdrawl	3022 Sudley Rd.	$125.00	-	$677.57

Current Balance $677.57 ATM

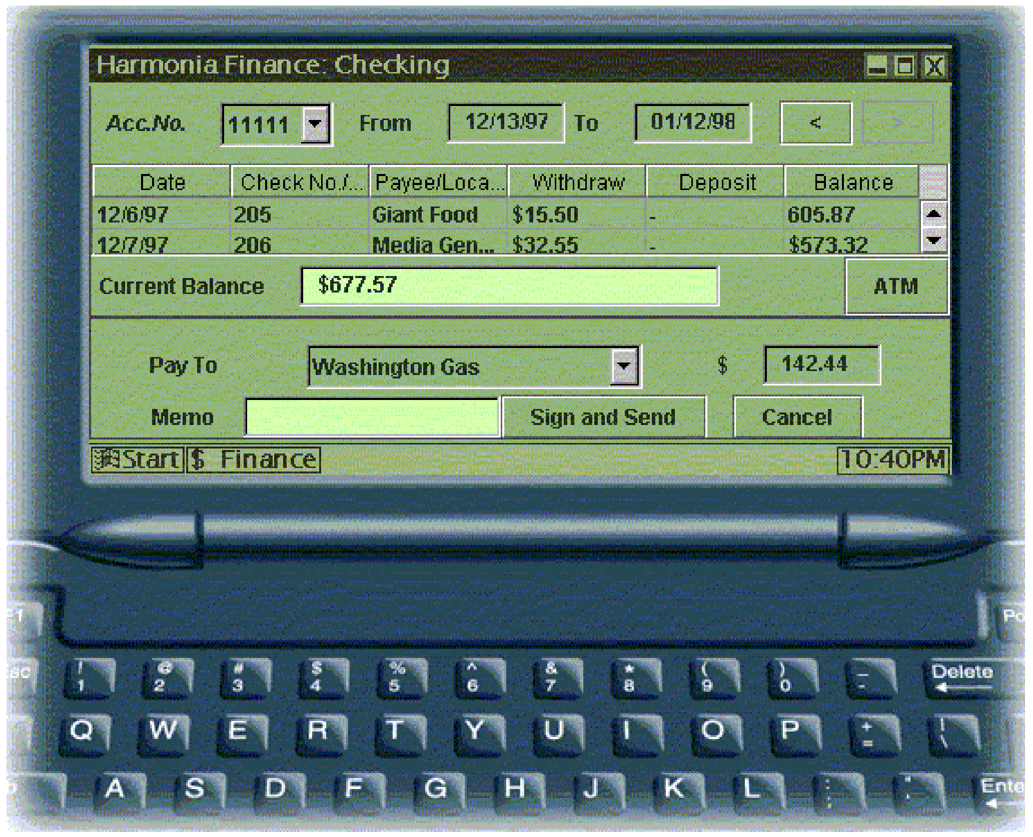
Date: January 14, 1998

Stephen M. Williams
10 Williamson Road
Roanoke, VA 24027

Pay to Washington Gas $ 142.44

Amount ---------One hundred forty two and 44/100--------- Dollars

Memo Sign and Send Cancel



Harmonia Finance: Checking

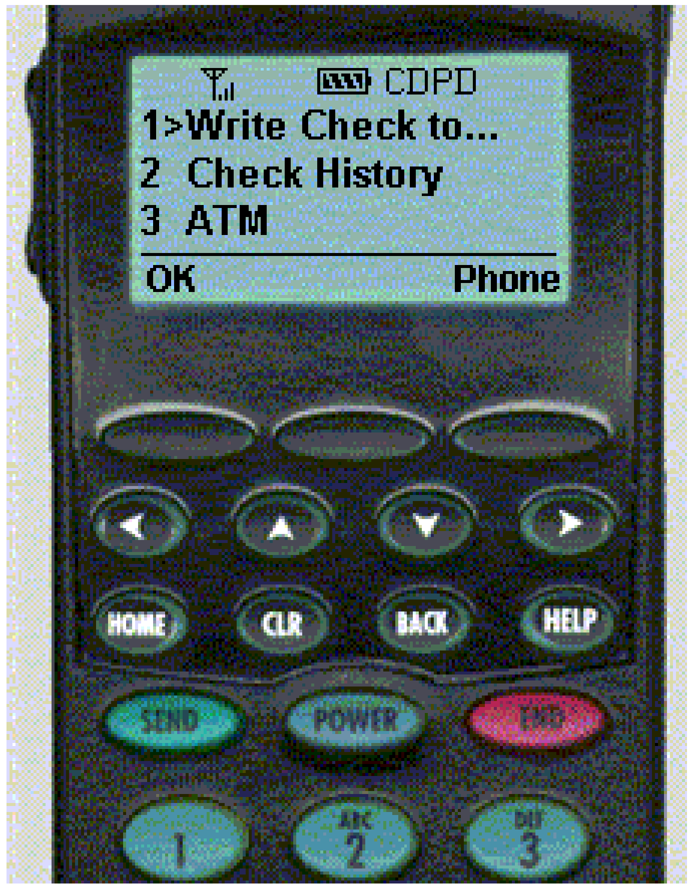
Acc.No. 11111 From 12/13/97 To 01/12/98

Date	Check No./...	Payee/Loca...	Withdraw	Deposit	Balance
12/6/97	205	Giant Food	$15.50	-	605.87
12/7/97	206	Media Gen...	$32.55	-	$573.32

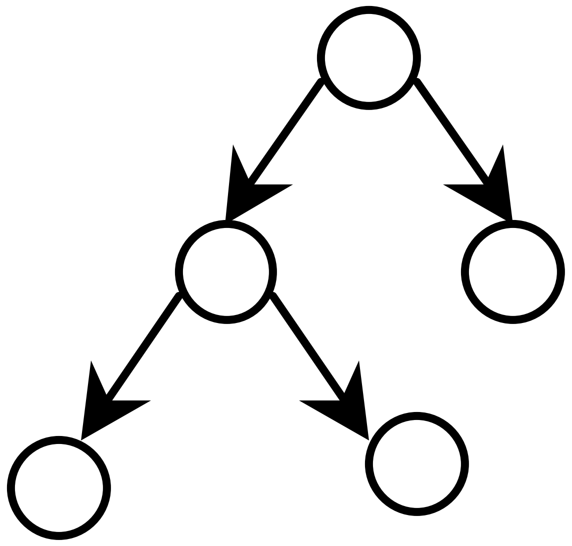
Current Balance $677.57 ATM

Pay To Washington Gas $ 142.44

Memo Sign and Send Cancel



CDPD
1>Write Check to...
2 Check History
3 ATM
OK Phone

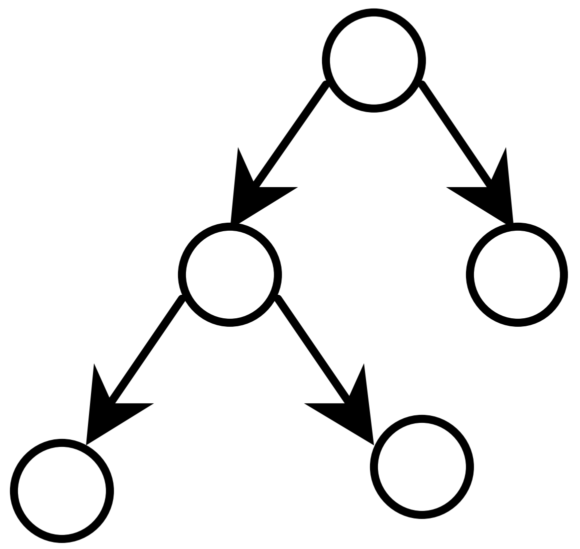


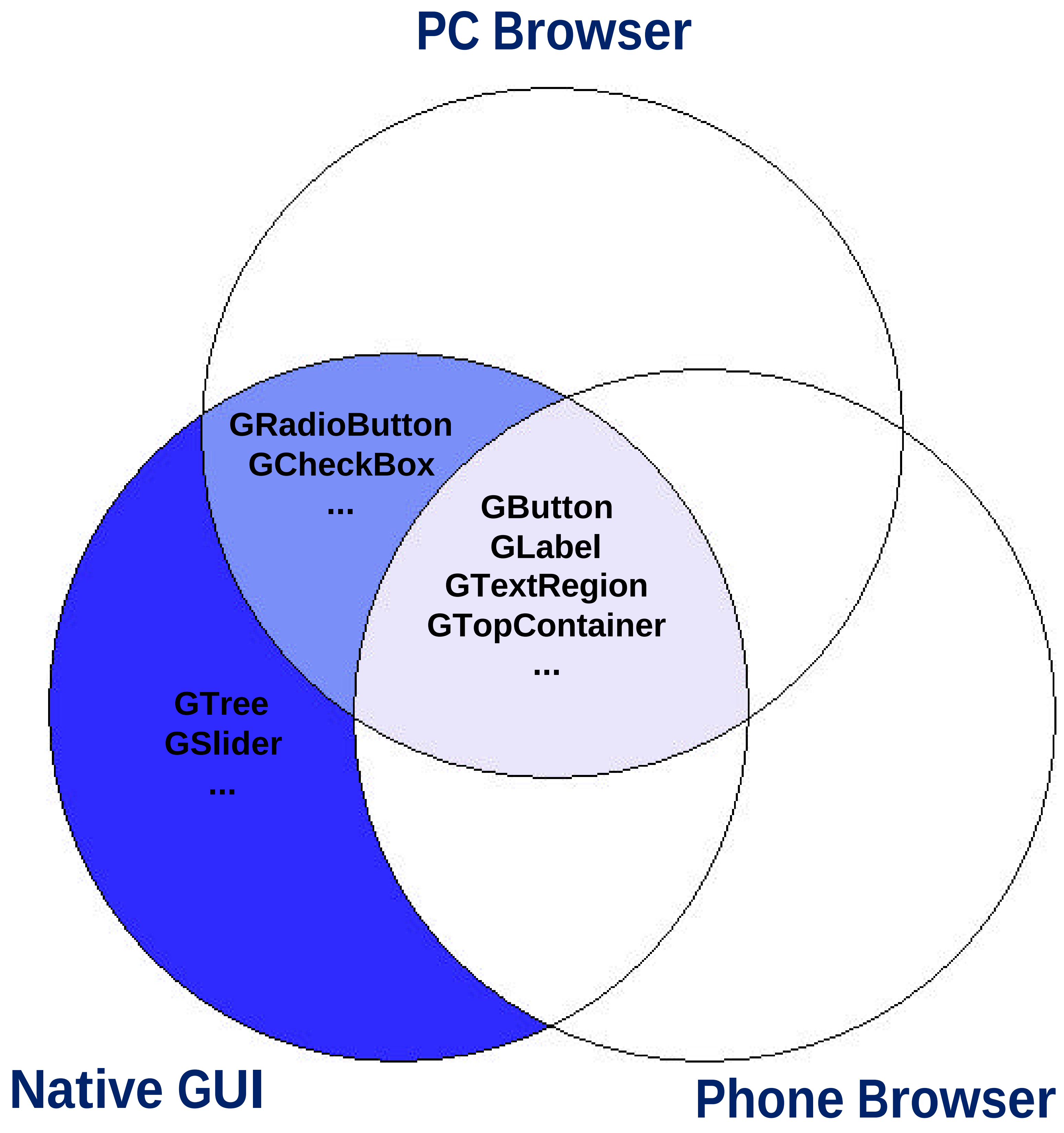
Describing User Interface Families

Problems with generic vocabulary:

- Too little elements which are supported by every device (e.g. voice vs. native GUIs)
- Layout very different

Solution: vocabularies for device families





Multiple <structure> Elements

Description of multiple structure elements within one document

```
<structure id="ComplexUI">
  <part class="c2" id="n3">
    <part class="c1" id="n2"/>
  </part>
</structure

<structure id="SimpleUI">
  <part class="c1" id="n1"/>
</structure>

<structure id="default">
  <part class="c1" id="n1"/>
  <part class="c2" id="n2"/>
</structure>
```

--> simple reuse (?)

What About Different Layouts?

What is needed is a *design process*.

Process first describes UI independent of

- UI metaphor (GUI, voice dialogs, ...)
- Layout
- UI widgets to be used
- Partitioning into screens (graphical) or dialogs (voice)

This top level UI is then refined into groups

Each group is set of platforms with common layout, different UI widgets

Then refine group members by choosing UI widgets (if generic vocabulary allows multiple choices)

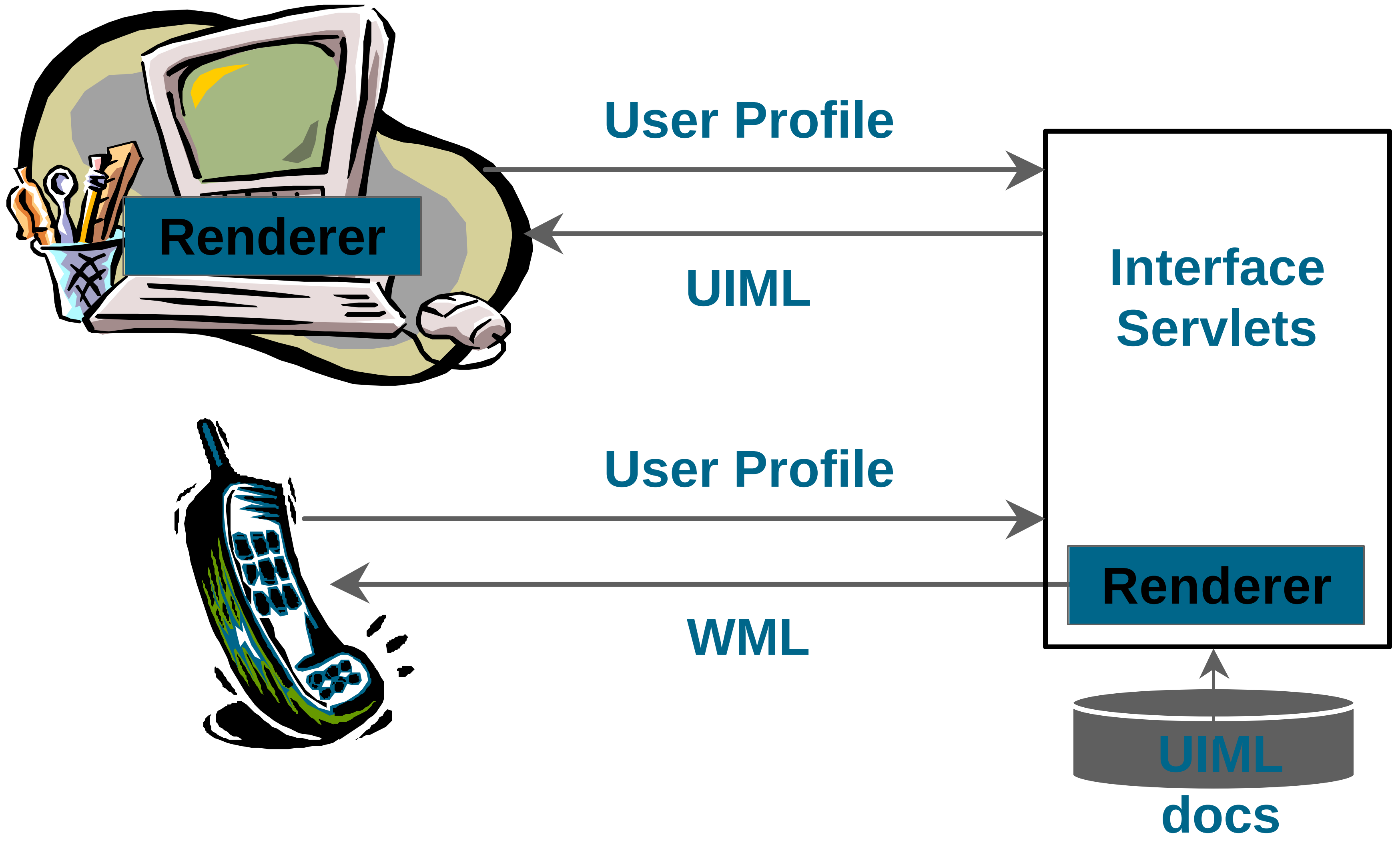
--> Good design methodologies for multi platform UIs is needed

UIML Renderer

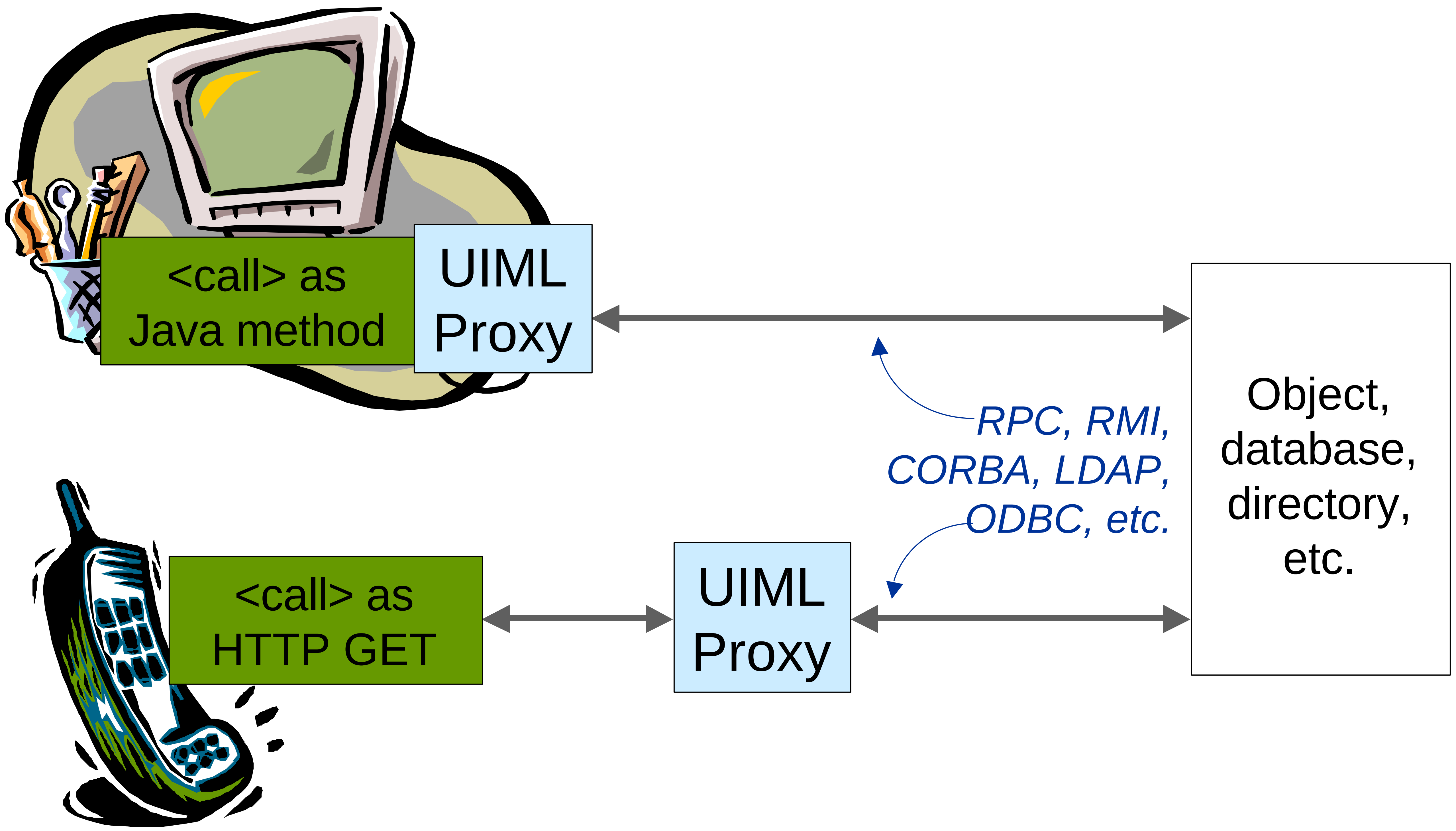
A UIML renderer will either

Interpret UIML on client device
(as Web browser does for HTML)

Compile UIML to another language
(WML, C++, ...)



After Rendering, How UI Talks to the World


<call> as Java method
UIML Proxy
<call> as HTTP GET
UIML Proxy
RPC, RMI, CORBA, LDAP, ODBC, etc.
Object, database, directory, etc.

UIML Renderers

Harmonia is creating a tool suite called LiquidUI™ for UIML:

- **Java Renderer:** Interpreter that renders UIML user interfaces in Java AWT/Swing.
- **HTML Renderer:** Compiler that renders UIML user interfaces to HTML for viewing through web browsers.
- **WML Renderer:** Compiler that renders UIML user interfaces to WML for viewing through WML-compliant devices.
- **VoiceXML Renderer:** Compiler that renders UIML user interfaces to VoiceXML for deployment through voice-only devices.
- **InterfaceServer: c**ucustomize UIML based on a profile and compile UIML for a target platform.

(see http://www.harmonia.com)

Other implementations at http://www.uiml.org

Additional Products

Harmonia

- **Authoring Tool**: allow users to graphically compose and edit UIML user interfaces (Currently In Development)
- **UIML Proxis**: allow UIML user interfaces to call back-end applications that support standard protocols. UIML proxies are available for:
  - CORBA (Common Object Request Broker Architecture)
  - EJB (Enterprise Java Beans)
  - LDAP (Lightweight Directory Access Protocol)
  - RMI (Remote Method Invocation)

UIML is Not a “Silver Bullet”

“One language to create UI for any device”

Does not mean

Write one UI description that magically adapts itself to any device

Does mean

If you properly design multiple UIs, the set may be expressed in one language, UIML

And …

If you also use a

- ***Proper design methodology and***
- ***Multi-platform vocabulary***

set of UIs may be described in *one* document.

The Future...

UIML Standardization at OASIS

Integration of UIML with existing W3C standards

Good design methodologies for multi-platform UIs is needed

--> For further information: http: / /www.uiml.org