v01.04
(Version 01, Draft 04)

DRAFT

The Relationship of the UIML 3.0 Spec. to Other Standards/Working Groups

July 26, 2004

Table of Contents

1 UIML

1.1 Introduction

UIML is an answer to the question of what a declarative language would look like that could provide a canonical representation of any user interface (UI) suitable for multi-platform, multi-lingual, and multi-modal UIs. This document describes the influences from W3C and other complimentary efforts on UIML, and comments on how UIML fits into these various technologies.

1.2 HTML, XML, CSS, WAI, and SOAP- Inspirations for UIML

Several W3C activities in 1997 -- XML, HTML, CSS, and WAI -- formed a catalyst of ideas that inspired the development of UIML. At that time a group of UI developers in Blacksburg, Virginia who were frustrated with the difficulty of creating UIs in traditional imperative languages (e.g., C, C++) starting work on UIML using a number of insights from these W3C activities.

The success of HTML by 1997 in allowing non-programmers to design UIs with a rich user experience was a beacon of light to the team that designed the original UIML language: Could we start fresh, and design a new declarative language powerful enough to describe UIs that historically were built only in imperative programming languages and toolkits (e.g., C with X-windows, C++ with MFC, Java with AWT/Swing)? Doing so would bridge the gap between HTML, which allows easy design of UIs with limited interaction, and imperative languages, which allow design of rich UIs but only in the hands of experienced programmers.

In 1997, the first XML conference was held. XML is a meta-language, to which a vocabulary of element and attribute names must be added [XML]. XML could be standardized once, and was extensible because many vocabularies could be created by different groups of people [BL]. In designing UIML we realized that if a UI language was a meta-language, then it could potentially serve as a canonical representation of any UI. Hence UIML is a meta-language. By separately creating vocabularies for UIML, UIML could be devoid of bias toward UI metaphors, target devices (e.g., PCs, phones, PDAs), UI toolkits (e.g., Swing, MFC), and could be translated to various target languages (e.g., Java, HTML, VoiceXML).

The world was clearly on a trend to untether users from the desktop computer, allowing them to use a plethora of devices via growing wireless technologies. UIML recognized that a meta-language enables the creation of UI descriptions in a device-independent form.

Another influence by 1997 was Cascading Style Sheets, which could be viewed as the first step to created UI descriptions that are separated, or factored, into orthogonal components [CSS]. The factoring was again a key to device-independent descriptions of UIs. The design of UIML started by asking what fundamentally are the orthogonal parts of a UI. The Model-View-Controller paradigm is a three-way separation. UIML arrived at a six-way separation (structure, style, content, behavior, APIs to components outside the UI, and mappings to UI toolkits) [PHAN].

The W3C's Web Accessibility Initiative [WAI], which also started in 1997, influenced UIML as well. The key to making documents and UIs accessible, according to WAI, is to capture the author's intent. A language like HTML has ingrained into it a certain metaphor based on the printed page. What authors need is the ability to represent a UI using abstractions representing the semantic information they have, which cannot be rediscovered easily from markup like HTML. Again, a meta-language appeared to be a key element for UIML, because an author could define and work with his or her own abstractions in a vocabulary that the author creates.

A second influence of WAI was the recognition that scripting in HTML pages presents an obstacle to making documents portable across devices. The lesson learned for UIML's designers was that the behavior of a user's interaction with a UI should clearly be a separable component in a UI description.

The original work on SOAP in 1998 also influenced UIML. When SOAP was first proposed, it suggested that remote calls to objects could be done using XML. Therefore the actions in UIML's syntax for behavior description were designed to allow invocation of SOAP or other XML-based remote calls.

1.3 HCI - Another Influence on UIML

Aside from W3C, there was one other key influence on UIML: the field of Human-Computer Interaction (HCI). The design of UIs that work across devices requires a good design methodology. Much work has been done in the HCI field in UI design. There is also a body of literature called UI Management Systems, which include notations to represent UIs, and these heavily influenced the design of UIML (especially the question of how to represent user interaction with a UI in a canonical form).

Our expectation is that work on design techniques for UIs will produce a number of tools and UI design languages (e.g., ConcurTaskTree [PAT]). UIML was not intended as a UI *design* language, but rather as a language for UI *implementation*. Therefore UI design tools could represent a design in a design language, and then transform a UI in a design language to a canonical representation for UI implementation, namely UIML. If Integrated Development Environments (IDEs) and Web page design tools could read UIML, then the world would have a complete path for computer-assisted design and implementation of multi-platform UIs.

1.4 Key Points in UIML

Here is a summary of the key facts about UIML that distinguish it from other XML languages for UIs:

- *UIML is a canonical representation of any UI.* There are many syntaxes to represent UIs, from Java to HTML. UIML simply offers a single syntax that is rich enough to represent the UI concepts in each of these languages. Therefore UIML must be rich enough to subsume the concepts from any target language and normalize their representation into a single syntax.

- *UIML is a meta-language.* A vocabulary must be added to UIML. Formal definitions (in UIML) of vocabularies for UIML are given here. Just as XML is a meta-language, tools can be created for UIML that are usable with any vocabulary. Vocabularies can be designed to capture UI metaphors, to represent abstractions to capture author intents, to work across devices, to describe controls specific to particular devices, and so on.

- *UIML separates a UI into six parts,* as stated earlier (structure, style, content, behavior, APIs to components outside the UI, and mappings to UI toolkits).

- *UIML is either compiled to a target language or interpreted.*

- *UIML can be freely implemented without license.*

1.5 How UIML Fits W3C Architecture Today

The best architectural picture of where UIML fits into the overall W3C architecture is the diagram in Figure 1 presented by Dave Raggett in his talk at the W3C Workshop on Web Device Independence (Bristol, Oct. 2000).

Figure 1. UIML's place in Dave Raggett's architecture diagram

[Note(rcc): Missing image for the architectural diagram. The Diagram has 4 levels. Top level is "Application database". Second level is "Content (e.g., XML, audio)". Third level is "Device adaption". Inside "Device adaption" is "UIML". A note attached to this level says "Using CC/PP". From the third layer (Device adaption) go four arrows to four sibling nodes on the bottom layer, labeled "XHTML", "VoiceXML", "XHTML", and "WML".

Where UIML Fits In*

L1: Application database

L2 : Content (XML, audio,…)

L3 : Device Adaption (using CC/PP) - UIML

L4: XHTML VoiceXML XHTML WML

Raggett proposed that there was a need for a layer (in green) that can adapt a UI to the particular XML language used by a target device. In Figure 1, we show UIML as a small box within the Device Adaptation layer. This is because UIML is an element of device adaptation, but not a complete solution. For example, there may be transform algorithms that transform the interface description (e.g., in UIML) to take into account device characteristics.

Without a single canonical language to represent UIs at this layer (regardless of whether it is UIML), then one must create transforms for multiple languages. Obviously if it is possible to have one language at this layer, the construction of reusable transforms is simplified.

One way to apply UIML at this layer is to use multiple vocabularies with UIML, and transform from UIML using one vocabulary to UIML using another vocabulary. For example, one may start with a UI description using a generic vocabulary (e.g., a vocabulary whose abstractions can be mapped to a variety of devices). Perhaps the UI was authored with this generic vocabulary to facilitate accessibility. A transform algorithm, guided by a rule base that takes into account characteristics of different devices, can then be used to map UIML with the generic vocabulary to UIML with a vocabulary specific to a particular device. This technique has been implemented to adapt UIs to various versions of Web browsers (e.g., to give a similar appearance to UIs for HTML 3.2 vs. HTML 4.0 browsers).

The UIML produced by the green Device Adaptation box can then be rendered to a particular XML language (e.g., by a rendering program that compiles UIML into XHTML, or UIML into VoiceXML).

1.5.1 The Path Toward Separation in User Interfaces

The evolution of W3C specifications in the UI area has followed a path of gradually separating a UI description into orthogonal parts:

- Up until HTML 3.2, there was no separation.
- In HTML4, the style was separated (via CSS and XSL-FO).
- In XForms, the portion of a document that represents a form was separated.
- In XML Events, events were separated.

As stated earlier, UIML separates a UI into six parts, answering these six questions:

1. What are the parts that constitute the structure of the UI?
2. What is the presentation style of the parts?
3. What is the content associated with the parts?
4. What is the behavior of the UI when a user interacts with the UI?
5. What is the API of components outside the UI with which the UI interacts?
6. What is the mapping of the vocabulary to a target UI toolkit or markup language?

These six questions are answered in UIML's structure, style, content, behavior, logic, and presentation elements, respectively.

Therefore this fundamental design decision in UIML is compatible with the path being followed by W3C. UIML should provide W3C working groups with an example of what will ultimately be reached as this path toward separation is followed in the future.

2 Relations to W3C Activities and Working Groups

2.1 W3C Device Independence Working Group (DIWG)

http://www.w3.org/2001/di/

2.1.1 Overview

2.1.1.1 Authoring for Device Independence

See *Authoring Challenges for Device Independence*, www.w3.org/TR/acdi/. Latest working draft is dated October 18, 2002.

According to the DIWG web site, DIWG is looking at considerations that Web authors face in supporting access to their sites from a variety of different devices.

Here are the goals of this effort:

- "Identify the difficulties that authors face in an environment in which there is an increasingly diverse set of devices used to access web sites.
- Identify the implications for authoring techniques that may assist authors in creating sites that can be accessed using a variety of devices and networks with different capabilities. These

implications will form the basis for further work in identifying and developing appropriate techniques."

The work is very interesting. It analyzes the different type of user interface authors, and their roles. It discusses a number of considerations and implications of allowing authoring user interfaces in a way that is independent of the device used to render the interface.

An authoring tool that adheres to the information in www.w3.org/TR/acdi/ could generate UIML as output. Therefore the authoring work in the DIWG is related to UIML, in the sense that the UIML TC must verify that UIML can support all needs in www.w3.org/TR/acdi/.

2.1.1.2 Presentation Attributes

See *Delivery Context Overview for Device Independence*, www.w3.org/TR/di-dco/. Latest working draft is dated December 13, 2002.

According to the DIWG web site, DIWG is setting out the requirements for defining a set of Core Presentation Attributes. The purpose of defining these Core Presentation Attributes is to provide a common set of attribute definitions that can be reused in many contexts in which the presentation capabilities of a presentation device need to be considered. This effort appears to go a step further than the CC/PP working group (www.w3c.org/Mobile/CCPP/).

"The intended purpose of the Core Presentation Attributes recommendation will be to define

- a common set of presentation attributes
- that can be reused in different delivery context vocabularies
- but share a common semantics
- in order to simplify the task of interpreting these attributes when adapting content for presentation in different delivery contexts."[From www.w3.org/TR/di-dco/]

2.1.2 Relation to UIML

UIML is a meta-language, which means that one must add a vocabulary to UIML to make it useful. The DIWG is going to the hard intellectual work to figure out what appears to be a common vocabulary. Therefore the OASIS UIML TC should track this effort and identify whether the DIGW presentation attributes can become a vocabulary for UIML. Furthermore, the TC should investigate whether there are any concepts in the presentation attributes that cannot be represented in UIML, and update UIML to include the concepts.

2.2 Web Accessibility Initiative (WAI)

http://www.w3.org/WAI/

2.2.1 Overview

"The World Wide Web Consortium's (W3C) commitment to lead the Web to its full potential includes promoting a high degree of usability for people with disabilities.

WAI, in coordination with organizations around the world, pursues accessibility of the Web through five primary areas of work: technology, guidelines, tools, education and outreach, and research and development." [http://www.w3.org/WAI/]

2.2.2 Relation to UIML

UIML can facilitate the design of accessible UIs. WAI members have articulated the importance of capturing author intents in UI design. Because UIML is a meta-language, one can use with UIML a vocabulary that captures the abstractions that a UI author uses in a design.

Consider the design of a web site that represents a collection of documents. Perhaps each document has four parts: a title, an abstract, a body, and navigation (e.g., to return to an index of documents). One could use a UIML vocabulary that uses class names of *title, abstract, body,* and *navigation.* The UIML document would then represent the UI in terms of the high level abstractions important to the web site designer. For example, the <structure> element in UIML, which answers the question of what parts constitute the structure of the UI, could look like this:

```
<structure>
  <part class="document">
    <part class="title">
    <part class="abstract">
    <part class="body">
    <part class="navigation">
  </part>
</structure>
```

The class names in UIML are part of a UIML vocabulary. A UIML document that uses a vocabulary expressing author intent can be rendered to a target language through transformation. One school of thought in the HCI field is to design UIs through a process of applying transforms to UI designs. Transforms could be designed that map UIML documents to UIML documents. Such a transform could map a UIML document using the class names above to another UIML document using a vocabulary whose classes represent abstractions in a UI metaphor suitable for a family of devices or languages (e.g., for a graphical UI metaphor, navigation might be mapped to a button). The resultant UIML might then be transformed again to UIML with a device-specific vocabulary (e.g., those listed on uiml.org/toolkits). This third UIML document could then be compiled in an efficient and straightforward manner to the target language.

The important point is that with a UI design represented in UIML, the UIML is a very malleable form that permits transformation and changes in vocabulary without worrying about the syntax of the target language. Use of a single canonical UI language permits libraries of transforms to be created and reused. This saves labor over writing transforms specifically for Java UIs, other transforms specifically for HTML, and so on.

In contrast, HTML would represent the site in terms of low-level HTML markup. For example, the title might be represented as * The Title
Subtitle .* In this example, the markup does not use the HTML heading tags (e.g., <h1>), and interpreting tags like complicates the job of software that tries to display the UI in alternate forms (e.g., vocalizing web pages). In essence UIML through appropriate vocabularies can preserve more of the semantics of the original UI design, while programs like screen readers that try to turn HTML into voice try to rediscover semantics.

While the use of properly designed style sheets with HTML facilitates mapping HTML to different devices, HTML still frequently requires scripting. UIML's <behavior> element contains rules for events and their actions that again can exploit the meta-language nature with a vocabulary of events chosen to match the

author's intent. The <behavior> ultimately can be compiled to whatever scripting languages are used by a target.

In summary, UIML can provide a piece of the puzzle that WAI is solving. UIML could be a component in a set of methodologies, design tools and languages that collectively improve accessibility. UIML provides WAI with a new avenue to explore ways to preserve author intent and reduce the obstacles in designing universally accessible documents.

2.3 XML/XSLT Working Group

http://www.w3.org/Style/XSL/

2.3.1 Overview

"XSL is a language for expressing stylesheets. It consists of three parts: XSL Transformations (XSLT): a language for transforming XML documents, the XML Path Language (XPath), an expression language used by XSLT to access or refer to parts of an XML document. (XPath is also used by the XML Linking specification). The third part is XSL Formatting Objects: an XML vocabulary for specifying formatting semantics. An XSL stylesheet specifies the presentation of a class of XML documents by describing how an instance of the class is transformed into an XML document that uses the formatting vocabulary. "[http://www.w3.org/Style/XSL/]

2.3.2 Relation to UIML

To display XML documents in a browser, XSL is often written to transform XML to a target language via XSLT. The most common target languages today are HTML and XHTML.

If one wants to map an XML document to *N* target languages, then *N* XSL transforms are required. The author of each transform must be familiar with the syntax of each of the *N* target languages. Furthermore, if an organization uses *M* XML schemas, then *M*N* XSL transforms are required. During the lifetime of the system, the *N* target languages may change as new versions are introduced. This may require maintenance of the *M*N* transforms over a period of years. Finally, if in the future a new target language is introduced, one must write *M* new XSL transforms; there is no way to reuse any of the existing *M*N* transforms.

UIML can reduce the multiplicative effort required to provide display of *M* documents on *N* devices. One would write *M* transforms in XSL to map documents in each of the XML schemas into UIML. The ability of UIML to be mapped to various target languages could then be exploited so that the authors of XSL need only learn one syntax, that of UIML, rather than the syntax of each target language. Moreover, as new versions of the target languages appear, *no* change to the XSL transforms are required -- only the software that renders UIML to target languages would need updating.

In addition, UIML can be rendered to imperative languages (e.g., Java, C++, C). Therefore someone can exclusively use XML, XSL, and UIML to deploy documents to any imperative target language, in addition to XML-compatible target languages.

In essence, the "distance" which an XSL transform must span is reduced using UIML.

2.4 XHTML Working Group

http://www.w3.org/MarkUp/

2.4.1 Overview

"XHTML 2 is a general purpose markup language designed for representing documents for a wide range of purposes across the World Wide Web. To this end it does not attempt to be all things to all people, supplying every possible markup idiom, but to supply a generally useful set of elements, with the possibility of extension using the **span** and **div** elements in combination with stylesheets." [http://www.w3.org/TR/2003/WD-xhtml2-20030131/introduction.html#s_intro]

2.4.2 Relation to UIML

XHTML is a target language to which UIML can be mapped. All that would need to be created is a vocabulary and renderer to map UIML to XHTML. XHTML is one of the many languages that represent concretely the UI's expressed in UIML.

2.5 XForms Working Group

http://www.w3.org/MarkUp/Forms/

2.5.1 Overview

"Key Goals of XForms: (1) Support for handheld, television, and desktop browsers, plus printers and scanners; (2) Richer user interface to meet the needs of business, consumer and device control applications; (3) Decoupled data, logic and presentation; (4) Improved internationalization; (5) Support for structured form data; (6) Advanced forms logic; (7) Multiple forms per page, and pages per form; (8) Suspend and Resume support; (9) Seamless integration with other XML tag sets." [http://www.w3.org/MarkUp/Forms/]

2.5.2 Relation to UIML

On the surface, UIML and XForms appear to be very similar technologies and there are some areas where they address similar problems. Therefore, it is instructive to compare their solutions. In terms of a Venn diagram, the range of UIs (user interfaces) you can describe with XForms is a strict subset of those you can describe using UIML. In fact, UIML can be used to define interfaces that represent not only the XForms sections of the UI, but also the XHTML in which the XForms tags are embedded. Basically UIML provides flexibility in form and function that can be used to represent any UI. However a synergy should exist between UIML and XForms, so it will be useful to explore the differences between the two technologies and discover where they complement each other.

UIML and XForms were designed from different worldviews:

XForms originated to enhance forms in Web pages; to overcome many of the limitations in look, feel, and functionality of the <form> and related tags from HTML; to provide a more sophisticated model of interaction between the server and the browser; to introduce types so that form input can be validated; and later to support form deployment on different devices. XForms had to build on the concepts in HTML forms to be familiar to the Web community and to provide a smooth transition from today's authoring tools, browsers, and servers to XForms enabled pages.

UIML's initial design perspective emerged from a 'clean sheet of paper' unhindered by any metaphors, and was formed by asking very fundamental questions of what information is required to describe any user interface, regardless of device, interface metaphor, or widget set. The language design goal of UIML was to provide a meta-language powerful enough to subsume the concepts in every language ever designed to describe or implement user interfaces. This means that UIML would be capable of defining a

canonical representation of any UI. The ability to create the canonical representation of any UI allows a vision of a world in which every researcher or product company whose technologies fuel e-business can interchange and interoperate by using UIML as the common interchange language. This is very important to facilitating the use of results coming out of the HCI community in models and transforms. For example, if HCI people designing transforms created UIML-to-UIML transforms, they could potentially be applied no matter whether the UI is ultimately translated to Java or XForms. Similarly, UIML can be used to serve as an intermediary so that transforms designed by one researcher from a language or model to UIML can be used in conjunction with transforms designed by another researcher from UIML to another language or model. An example of this is shown in work by Ralph Miller and Scott P. Overmyer, where they translate a meta-language called Requirements Element Language (REL) into UIML and back. In contrast, XForms has a more immediate objective, and is not powerful enough to serve as this universal interchange representation.

UIML touches on issues that were not familiar for the HTML world.

First, UIML provides strict separation or factorization of the user interface elements. For example, the words, pictures, videos and other content of the UI should be separated from the UI structure for other concerns. This is not done in XForms. Instead, XForms enforces a separation of instance data from UI controls. The UI controls still encapsulate all content of the UI, meaning that label text and UI control content are still integrated into the UI control. This can make internationalization difficult. More generally, UIML is based on an extension to the Model/View/Controller model, called the Meta Interface Model (See http://scholar.lib.vt.edu/theses/available/etd-08122000-19510051/unrestricted/PhanouriouETD.pdf for details). The MIM provides a six-way separation: UI structure, presentation style, content, behavior rules, connection of the UI to whatever code is outside of the UI (e.g., business logic), and mapping of the vocabulary used for class/property/event names to widget set or XML tags to which the UI is mapped.

Second, UIML can describe any method of connection between the UI and whatever the UI "talks to". In particular, UIML should describe the wiring of the UI to business logic that uses a procedure call model, or a Web services server that uses SOAP; or a web server using HTTP GET and PUTs; or a publish/subscribe protocol using events to push content to the UI; and so on. In contrast, XForms builds on the HTTP GET/PUT model of the Web.

Third, UIML allows any UI metaphor (e.g., card-based, Web form-based, desktop GUI application, voice dialog, 3D immersive environments, etc.). In contrast, XForms is restricted to the web-forms model of interaction.

Fourth, UIML should capture author intent. This is possible by allowing descriptions of UIs expressed using abstractions of the author's choice, rather than the choice of the language designer or widget designer. For example, someone designing a course or training manual may want to think in terms of a UI that has an outline, a list of objectives, prerequisites, etc. A UI designer for autos may think in terms of steering wheel buttons, driver alerts, etc. A UI designer for displays in factory automation may think in terms of the flow of the manufacturing process. If these UI designers use a traditional UI development tool (e.g., Visual Basic, Dreamweaver, Forte), they are forced to translate their thinking to the level of VB or HTML form or Java widgets, because the palette they use to design the UI contains widgets. In contrast, because UIML is a meta-language to which one adds a vocabulary, one can devise vocabularies specific to course designers versus auto designers versus an industrial engineer and so on, which represent the author intent independently of the particular widgets used with a particular target language or device. This view is not really captured in XForms.

Fifth, UIML should support behaviors, so that much of the JavaScript in traditional web pages can be described in a device-independent form. In contrast, XForms uses XEvents. The event-based description format used in UIML was based on research in the UIMS literature such as Hartson and Hix's 1993 book User Interface Development: Ensuring Usability through Product and Process. In UIML, interface authors

can define a set of rules, like a rule-based system, that are stated in terms of conditions and actions. These rules are (1) separated from the rest of the UI, and (2) described in a device-independent format. These rules define a rulebase from which rules are selected as their conditions are met.

Sixth, UIML vocabularies have been created to represent UIs in many languages: Java, C++ for QT widgets in Linux, C++ for an embedded system, C with PalmOS, VoiceXML, WML, HTML, CSS, and Visual Basic. This may be a wider range of targets than envisioned for XForms.

To summarize, UIML is an XML schema targeted to serve as a universal interchange format that can represent any UI - regardless of UI metaphor, language, operating system, and device. On the other hand, XForms is an "XML application that represents the next generation of forms for the Web" [http://www.w3c.org/XForms]. XForms provides a vast improvement on HTML Forms and begins to address many of the problems with facilitating interaction with remote systems. Ultimately XForms is tremendously helpful for UIML, because mapping UIML to XForms is much easier mapping UIML to HTML forms.

2.6 CSS

http://www.w3.org/Style/CSS/

2.6.1 Overview

"Cascading Style Sheets (CSS) is a simple mechanism for adding style (e.g. fonts, colors, spacing) to Web documents." [http://www.w3.org/Style/CSS/]

2.6.2 Relation to UIML

UIML contains a style element as one of its six-way separations of UIs (see *The Path Toward Separation in User Interfaces*). The style element contains one or more property elements. Each property element specifies the following:

- *Property name:* Because UIML is a meta-language; the vocabulary of property names is not part of UIML. One could use the vocabulary of, say, CSS2 with UIML, or the vocabulary of XSL-FO.
- *Property value:* The property value can be a constant, a reference to some entity (e.g., a URL), or the return value of a call to some component external to the UI.
- *The entity to which the property applies:* This is a part instance, a part class, an event instance or class.

The UIML1 language originally used CSS for style. There were three reasons this was abandon in UIML 2 and 3:

1. UIML1 users were confused by why they had to switch from an XML syntax for UIML to a non-standard syntax for CSS. This issue was resolved in UIML2 by simply changing syntax. Whereas in CSS one might write

   ```
   H1 { color: blue }
   ```

   In UIML this is written

   ```
   <property part-class="H1" name="color">blue</property>
   ```

   The property name *color* is defined in the vocabulary (in UIML's presentation element).

1. CSS's semantics of inheritance (that a property applied to a part instance also applied to any children of the part instance, unless explicitly reset for a child) does not correspond to the semantics of object-oriented imperative languages (e.g., Java). In fact, *for UIML to be a canonical representation of any UI, a semantic rule requiring inheritance could not be part of UIML.* When a property is set for a part instance in UIML, it applies to just that instance and no others.

2. CSS has not used with imperative programming languages, and would be unfamiliar to a developer who only creates applications for languages like Java, Visual Basic, and C++. To such a developer writing only UIML is preferable to writing UIML plus CSS.

What was critical to retain from CSS in UIML is the ability to cascade style sheets. Cascading facilitates accessibility and the design of reusable style sheets. UIML has language mechanisms to control how a base style sheet and additional style are combined together, which help avoid the conflicts possible in CSS (e.g., from combining cascading and inheritance). UIML allows UI descriptions to be packaged into reusable components through a <template> element, and this element is used for all language elements to allow cascading, not just for style elements.

Note UIML is used *with* CSS, and does not replace CSS. To illustrate, suppose a developer has a UI description in UIML, and this UI is to be displayed on some clients as an HTML web page, and on some devices using Java. A renderer for the Web page case could emit HTML with style represented by CSS, while a renderer for Java could generate Java source code with style represented by Java method calls. Therefore UIML simply serves as an intermediate and canonical representation of style that maps to the syntax of various target languages, including CSS.

2.7 Voice Working Groups

http://www.w3.org/Voice/

2.7.1 Overview

The W3C Voice Browser Working Group is defining a suite of markup languages covering dialog, speech synthesis, speech recognition, call control, and other aspects of interactive voice response applications. The following suite of languages forms the "W3C Speech Interface Framework":

- VoiceXML 2.0: Describes audio dialogs involving synthesized speech, digitized audio, recognition of spoken voice input, recognition of DTMF [the touch tone keys you press on a phone], and recorded spoken input.

- Speech Synthesis Markup Language (SSML): Markup language to describe voice prompts. Allows specification of name, gender, age, pitch, speed, volume, and "emphasis".

- Speech Recognition Grammar (SRGS): Describes a grammar (rules for parsing inputs) for DTMF and voice input. Lets a grammar processor find a sequence of words that best matches what a speech recognizer hears when someone speaks.

- Call Control Markup Language (CCML): Developers can use this markup language to screen calls, transfer calls, conditionally answer calls, place outbound calls, and similar functions.

- Semantic Interpretation: I may have this wrong, but I believe SRGS describes SYNTAX, while "Semantic Interpretation" is a set of tags that specify semantic processing done when a grammar

processor is trying to parse someone's voice input using SRGS. (If you are familiar with compiler design for programming languages, then a parser uses both syntax and semantic rules.)

The Voice group wants to make these languages work with XHTML, XForms, and SMIL. There has been some controversy in W3C as to whether any member of W3C would have intellectual property rights on work created by the Voice group. I believe that the result is the Voice group has been designated as a "royalty free" working group under a new patent policy at W3C.

2.7.2 Relation to UIML

The vision of UIML is to simplify the world for user interface authors by permitting them to use one markup language (UIML), irrespective of target device. The work of the Voice Browser Working Group is to create multiple markup languages for voice. The effort of the Voice group is very important to think through the many issues of how to build voice interfaces. I feel that UIML plays a complementary role, because UIML can provide a way for user interface authors to use the Voice languages that W3C develops without learning the syntactic details of those languages.

The role of UIML then would be to provide vocabularies to support these voice markup languages. In addition, organizations implementing UIML tools would then provide compilers or interpreters to implement a UIML document using these voice vocabularies using the Voice Browser Working Group's new languages.

If we step back a moment and look at the overall work at W3C to develop many markup languages to carefully describe user interfaces for voice, web browser, and other platforms, then the need for UIML becomes even stronger. User interface authors are faced with an increasing number of implementation languages (such as the ones from the Voice browser group), and need the simplicity of one language.

2.8 W3C Graphics Activity

http://www.w3c.org/Graphics/Activity

2.8.1 Overview

The W3C's Graphics activity represents the collaboration of several working groups within the organization. The activity has been the driving catalyst for the last six years in Web Graphics. Included in their accomplishments are the Portable Network Graphics (PNG) format, WebCGM (computer Graphics Metafiles), and Scalable Vector Graphics (SVG).

2.8.1.1 WebCGM (Computer Graphics Metafile)

WebCGM was built to extend the CGM standards to the web. CGM has been used by industry for years to provide precise vector data to a market that had this need. Many vendors had already standardized on CGM, but lacked a vendor-neutral and interoperable hyperlinking mechanism. CGM is an ISO standard for vector graphics. The WebCGM profile adds additional constraints to improve interoperability, defines how hyperlinking works, and defines mechanisms for use in HTML.

WebCGM has primarily been used to represent technical documentation for industry on the web. Typically this type of information is presented with no desire for restylable graphics or for graduated fills, but instead precise specification of line and hatch styles.

2.8.1.2 Portable Network Graphics (PNG):

PNG's original purpose was to replace Graphics Interchange Format (GIF) for web graphics. PNG is another alternate image format that supports true color and indexed color and provides loseless compression.

2.8.1.3 Scalable Vector Graphics (SVG)

See section 2.9.

2.8.2 Relation to UIML

All of the graphics formats developed by the Graphics Activity should be support by UIML. This means that UIML should be able to represent any interaction that can be defined in these formats as well as being able to defines where graphical content resides within a UI.

2.9 Scalable Vector Graphics (SVG)

http://www.w3.org/Graphics/SVG/Overview.htm8

2.9.1 Overview

SVG is a language, being developed by the W3C's Graphics Activity, for describing two-dimensional graphics in XML. SVG can describe three types of graphic objects: vector graphic shapes (e.g., paths consisting of straight lines and curves), images, and text. SVG can group, style, transform and composite graphical objects. Text can be in any XML namespace suitable to the application, which enhances searchability and accessibility of the SVG graphics. The feature set includes nested transformations, clipping paths, alpha masks, filter effects, template objects and extensibility.

SVG drawings can be dynamic and interactive. The Document Object Model (DOM) for SVG, which includes the full XML DOM, allows for straightforward and efficient vector graphics animation via scripting. A rich set of event handlers such as onmouseover and onclick can be assigned to any SVG graphical object. Because of its compatibility and leveraging of other Web standards (See section 1.4 of the SVG 1.1 specification introduction), features like scripting can be done on SVG elements and other XML elements from different namespaces simultaneously within the same Web page.

A pocket computer and cell phone specification is being developed called SVG Basic and SVG tiny respectively (collectively SVG Mobile).

SVG is currently in version 1.1 and its 1.1, 1.0, and mobile profiles work are all Web standards (W3c recommendations.)

2.9.2 Relation to UIML

SVG is a graphics format that can be used to describe fairly complex interfaces. UIML must be able to describe any interface that can be similarly described in SVG. In this way, a vocabulary for SVG can be developed that will make rendering from UIML to SVG possible.

3 Other XML-based UI languages

3.1 USer Interface eXtensible Markup Language (UsiXML)

http://www.usixml.org/

3.1.1 Standardizing Organization

UsiXML is mainly supported by the Cameleon research project (http://giove.cnuce.cnr.it/cameleon.html) under the umbrella of the European Fifth Framework Programme (FP5-2000-IST2) and the SIMILAR network of excellence (http://www.similar.cc), the European research task force creating human-machine interfaces similar to human-human communication of the European Sixth Framework Programme (FP6-2002-IST1-507609).

The researchers behind UsiXML, led by Dr. J. Vanderdonckt, are working with the OASIS TC on UIML to conceptually reconcile USiXML and UIML.

3.1.2 Overview

Excerpted from usixml.org:

> UsiXML (which stands for USer Interface eXtensible Markup Language) is a XML-compliant markup language that describes the UI for multiple contexts of use such as Character User Interfaces (CUIs), Graphical User Interfaces (GUIs), Auditory User Interfaces, and Multimodal User Interfaces. In other words, interactive applications with different types of interaction techniques, modalities of use, and computing platforms can be described in a way that preserves the design independently from peculiar characteristics of physical computing platform.
>
> UsiXML is intended for non-developers, such as analysts, specifiers, designers, human factors experts, project leaders, novice programmers,... Of course, UsiXML can be equally used by experienced developers. Thanks to UsiXML, non-developers can shape the UI of any new interactive application by specifying, describing it in UsiXML, without requiring programming skills usually found in markup languages (e.g., HTML) and programming languages (e.g., Java or C++).
>
> UsiXML consists of a User Interface Description Language (UIDL), that is a declarative language capturing the essence of what a UI is or should be independently of physical characteristics.
>
> UsiXML describes at a high level of abstraction the constituting elements of the UI of an application: widgets, controls, containers, modalities, interaction techniques, ...
>
> UsiXML allows cross-toolkit development of interactive application.
>
> A UI of any UsiXML-compliant application runs in all toolkits that implement it: compilers and interpreters.
>
> UsiXML supports device independence: a UI can be described in a way that remains autonomous with respect to the devices used in the interactions such as mouse, screen, keyboard, voice recognition system,... In case of need, a reference to a particular device can be incorporated.
>
> UsiXML supports platform independence: a UI can be described in a way that remains autonomous with respect to the various computing platforms, such as mobile phone, Pocket PC, Tablet PC, laptop, desktop,... In case of need, a reference to a particular computing platform can be incorporated.
>
> UsiXML supports modality independence: a UI can be described in a way that remains independent of any interaction modality such as graphical interaction, vocal interaction, 3D interaction, or haptics. In case of need, a reference to a particular modality can be incorporated.
>
> UsiXML allows reuse of elements previously described in anterior UIs to compose a UI in new applications.

3.1.3 Relation to UIML

UIML and UsiXML seem to share several common goals. Both seek to provide device and platform independence for any type of user interface (graphical or non). Both avoid creating a new user interface implementation language and instead focus on providing integration with existing implementation

technologies. Both abstract the description of a user interface to a more model-based representation that is suitable for examination and creation by non-programmers. What is unclear is the extent to which UsiXML and UIML overlap in the methods by which they attempt to achieve these goals. To answer this question and to verify that the goals of UsiXML truly do coincode with those of UIML, Dr. J. Vanderdonckt and Mr. Q. Limbourg have begun participating in the UIML TC discussions. One of the aims of their participation is to reconcile UsiXML and UIML at the conceptual level and determine is a syntactic reconciliation would be appropriate/possible.

3.2 XML for Abstract User Interfaces (AUI) used by TERESA

http://giove.cnuce.cnr.it/teresa/teresa_xml_aui.html

3.2.1 Standardizing Organization

This XML format is being used in research related to the TERESA project.

3.2.2 Overview

This format (referred to here as XAUI) was developed to represent user interfaces designed with the Transformation Environment for inteRactivE Systems representAtions (TERESA) system being researched by Dr. Fabio Paterno. TERESA uses two XML representations to represent models of the user interface: ConcurTaskTree (CTT) and XAUI. CTT describes user task models while XAUI describe the abstract user interface design.

XAUI specifies how various Abstract Interaction Objects (AIO) that compose the UI are organized, and specifies the dialogue of the UI itself. AIOs are defined in terms of types of user interaction (e.g. output only, single choice selection, and multiple choice selections). XAUI is completely divorced from the details of platform or device, and would require transformation to produce a concrete user interface.

3.2.3 Relation to UIML

The relationship between UIML and XAUI is unclear. When combines with an abstract vocabulary, UIML can be used to represent abstract interface designs. Given this it may be appropriate to use XAUI as a starting point for defining a new abstract vocabulary for UIML. Before this can be definitively determined, an analysis of the goals of XAUI should be undertaken to see if UIML can adequately encompass the descriptive power of XAUI.

3.3 XML User Interface (XUL)

http://www.mozilla.org/projects/xul/

3.3.1 Standardizing Organization

Independent consortium

3.3.2 Overview

The "XML-based User Interface Language" or XUL was first built by the Mozilla project (www.mozilla.org) for use in their Mozilla web-browser. It leverages several W3C standards, including XML 1.0, HTML 4.0; Cascading Style Sheets (CSS) 1 and 2; Document Object Model (DOM) Levels 1 and 2; and JavaScript

1.5, including ECMA-262 Edition 3 (ECMAscript). This combination of standards makes for a powerful level of expression.

3.3.3 Relation to UIML

XUL and UIML have several key important similarities. Both languages are XML–based, providing for ease of transport and recognition across platforms. Both provide desktop platform portability by abstracting the user interface definition away from concrete languages such as Java, C++, or HTML. Both provide ease of localization, customization, and personalization by separating content from structure. And perhaps the most important similarity is that UIML and XUL both separate the business logic from the presentation logic. This separation is key to allowing user interface definitions to be maintained independently of the backend and letting user interface designers and application developers apply their skills where they are most needed without interfering with each other.

Despite these similarities there are several key differences. For one, to achieve the flexibility and benefits listed in the paragraph above, a developer must combine XUL, JavaScript, XBL, CSS, and DTD of string bundles. The distribution of languages requires the developer to know and understand all 5 technologies whereas UIML packages this all into one language. The ease of learning one technology over five should provide UIML developers a quicker path to designing interfaces.

The key difference between UIML and XUL is that UIML is a meta-language. UIML seeks to break the definition of a user interface into its six fundamental components: structure, content, style, behavior, presentation, and logic. The structure, which defines the *parts* of an interface, does not adhere to any metaphor for interaction (e.g. Windows GUI, WML Cards, etc.). Instead the metaphor is provided by a vocabulary defined in the presentation. Vocabulary is not a part of the UIML specification and can be customized for whatever domain and/or widget set is appropriate to the application. This also implies that one vocabulary could potentially map to all widgets on all platforms. On the other hand, XUL has elements such as <hbox> (horizontally-oriented box), <button>, and <menupopup> which are intrinsically grounded in a traditional GUI metaphor. Mapping such widgets to an alternate, non-traditional device, for example a voice browser, is troublesome at best and typically requires a re-write of the user interface for the alternate device.

Now one obvious concern is since things like <hbox>, <button>, and <menupopup> are not defined in the UIML tag set, can they be represented in UIML? The answer of course is yes. In the UIML case, each of these objects would be defined in a vocabulary and each <part> in the interface would have an attribute set indicating which object that part corresponded to. This allows UIML to easily represent interfaces on traditional and non-traditional devices simply by applying the appropriate vocabulary.

One course of action to make UIML and XUL complimentary would be to formulate a vocabulary that would map UIML parts to XUL tags. Then, UI designers could potentially get the best of both worlds.

3.4 Alternate Abstract Interface Markup Language (AAIML)

http://xml.coverpages.org/userInterfaceXML.html

3.4.1 Standardizing Organization

INCITS V2 Committee

3.4.2 Overview

The V2 technical committee of INCITS is developing an Alternate Use Interface Access standard. This standard includes the development of an XML language for communicating abstract user interface definitions for a service or device to a user's personal device allowing it to act as a Universal Remote Console (URC). This console would allow users to then interact with the target service or device using a tool that they are familiar/able to use. This is especially useful for making devices accessible to people with disabilities. The idea is that anyone could connect to a device or service using any personal device. The target device would provide an abstract definition of its user interface to the user's personal URC. The URC would then emulate the interface using whatever interaction techniques apply for that URC (point and click for desktop, hear and speak for phone, brail, etc.)

The language strives to be sufficiently abstract so that the individual URCs have the freedom to display the interface however they want. The language defines a set of "abstract interactors" for interaction. These have semantics and function, but no concrete realization. These are then mapped to "concrete interactors" much like UIML. The language is based on an event model, built on eventing capabilities found in Microsoft's Universal Plug and Play and Jini/Java.

No published language specification was discovered for AAIML.

3.4.3 Relation to UIML

AAIML and UIML are complimentary in that they both strive to describe interfaces at abstract levels. The focus of AAIML is to enhance accessibility across devices and services so that people with URC devices can interact with these devices and services in a manner appropriate to their individual URC. In many ways UIML can serve the same purpose. Further investigation should be performed into AAIML to determine the level of overlap with UIML. UIML may be able to subsume AAIML by defining a vocabulary that allows UIML to be transformed into AAIML.

3.5 Abstract User Interface Markup Language (AUIML)

http://xml.coverpages.org/userInterfaceXML.html

3.5.1 Standardizing Organization

None

3.5.2 Overview

An intent-based interface definition language, primarily developed by IBM, that allows UI designers to design the intent of the interface without tying the tasks to any concrete, device specific realizations. Since AUIML is an XML-based language, platform independent encoding of interfaces is possible.

No published specification for AUIML was discovered and it appears that work on this language has slowed.

3.5.3 Relation to UIML

Since AUIML uses tags like <Group>, <Choice>, <Caption>, and <String> to represent data, a vocabulary can be defined for UIML that maps directly to AUIML's tag set. This will allow UIML to easily represent any interface that can be defined in AUIML.

3.6 Extensible Interface Markup Language (XIML)

http://www.ximl.org

3.6.1 Standardizing Organization

The XIML Forum (Independent Consortium – see below)

3.6.2 Overview

The eXtensible Interface Markup Language (XIML) is an XML-based language for representing user interface geared toward allowing universal support of functionality across the entire interface lifecycle. This includes the phases of "design, development, operation, management, organization, and evaluation" (http://xml.coverpages.org/userInterfaceXML.html).

Gleaning information on XIML from the freely available sources on the Internet is difficult. From what this author has been able to uncover, it seems that XIML is a very abstract interface definition language. It divides the definition of an interface into *components*, high-level building blocks of an interface. These components can be of several specialized types such as *Task* (business processes)*, Domain* (defines a hierarchy of components)*, User* (defines a hierarchy of end-users)*, Presentation,* and *Dialog* (defines actions within the interface). Components are then mapped to *Elements (*concrete representations such as widgets).

Along with components four other objects are used to define interfaces in XIML. These are *Relations, Attributes, Statements,* and *Definitions*. It is unclear at this time exactly how these objects interact.

The difficulty in studying XIML is that most of the available information (including the language specification) is open only to member of the XIML Forum. In order to join this forum, one has to agree to a license agreement that prohibits wide dissemination of any information contained therein. Given this restriction, the TC at OASIS decided not to join the Forum as any interaction between the Forum and TC could endanger UIML's royalty free status.

The XIML Forum is described in the XIML Coverpages as "an industry organization dedicated to the research, dissemination, adoption, and standardization of the eXtensible Interface Markup Language (XIML), a comprehensive specification language for user interfaces. The Forum aims to provide a common specification and development infrastructure for user interface professionals of all types from interaction designers, to software engineers, to usability experts. It also seeks to afford human-computer interaction researchers in academia, industry, and the military a representation mechanism for the study of advanced technologies for user interfaces."

3.6.3 Relation to UIML

UIML and XIML seem to be both overlapping and complimentary at the same time. The overall goal of both languages is to serve as a universal, canonical language for specifying user interfaces on computing devices. However, XIML's emphasis on the early design level, generic components of a user interface and UIML's ability as a meta-language to apply to late-design and implementation leaves room for both languages to compliment on another. In order to better define the relation between UIML and XIML, more documents on the nature of XIML will have to be made publicly available.

4 Miscellaneous XML languages from the AVI 2004 workshop

In May 2004 a Workshop on Developing User Interfaces with XML: Advances on User Interface Description Languages (UIDL) was organized to be help at the Advanced Visual Interfaces 2004 (AVI'04) conference in Gallipoli, Italy. The workshop had a marvelous turnout and several XML-based UIDLs were presented. Information on the participants and their submissions is included in the table below. These languages will be investigate in more detail as the TC progresses.

Table 1: Accepted papers at the AVI'04 workshop

[abrams2004]	*Retrospective on UI Description Languages, Based on 7 Years Experience with the User Interface Markup Language (UIML)*, Marc Abrams, Jim Helms. [abstract] [more information] [presentation (ppt)]
[berti2004]	*The TERESA XML Language for the Description of Interactive Systems at Multiple Abstraction Levels*, Silvia Berti, Francesco Correani, Fabio Paternò, Carmen Santoro. [abstract][more Information]
[bleul2004]	*Multimodal Dialog Description for Mobile Devices*, Steffen Bleul, Wolfgang Mueller, Robbie Schaefer. [abstract] [more information] [presentation (pdf)]
[braun2004]	*Extending XML UIDLs for MultiDevice Scenarios*, Elmar Braun, Max Mühlhäuser. [abstract]
[crowle2004]	*Into the mangle, Software engineers run creases through a user interface metaphor*, Simon Crowle. [abstract] [more information] [presentation (ppt)]
[cuppens2004]	*VRIXML : A User Interface Description Language for Virtual Environments*, Erwin Cuppens, Chris Raymaekers, Karin Coninx. [abstract] [more information] [presentation (ppt)] [movie 1] [movie 2]
[fogli2004]	*IM2L: A User Interface Description Language Supporting Electronic Annotation*, Daniela Fogli, Giuseppe Fresta, Andrea Marcante, Piero Mussio. [abstract] [presentations (ppt)] [presentation with movies (zip)] [movie 1] [movie 2] [movie 3] [movie 4] [movie 5] [movie 6]
[haya2004]	*Extending an XML environment definition language for spoken dialogue and web-based interfaces*, Pablo A. Haya, Germán Montoro, Xavier Alamán, Rubén Cabello, Javier Martínez. [abstract] [more information] [presentations (pdf)]
[jia2004]	*Incorporating UIDLs into Model-Driven Development*, Xiaoping Jia, Adam Steele. [abstract]
[koskimies2004]	*Practical experiences with device independent authoring concepts*, Oskari Koskimies, Michael Wasmund, Peter Wolkerstorfer, Thomas Ziegert. [abstract][more information] [presentations (pdf)]
[kost2004]	*Dynamically generated multimodal application interfaces*, Stefan Kost. [abstract] [more information] [presentation (html+javascript)]

[leite2004] *Extensibility and Reusability of Web User Interface Components using XICL*, Jair C. Leite, Lirisnei Gomes de Sousa. [abstract] [more information] [presentation (pdf)]

[limbourg2004] *USIXML: A User Interface Description Language for Context-Sensitive User Interfaces*, Quentin Limbourg, Jean Vanderdonckt, Benjamin Michotte, Laurent Bouillon, Murielle Florins, Daniela Trevisan. [abstract] [more information]

[menkhaus2004] *Adaptation for Device Independent Authoring*, Guido Menkhaus, Sebastian Fischmeister. [abstract] [presentation (pdf)]

[merrick2004] *Abstract User Interface Markup Language*, Roland Merrick, Brian Wood, William Krebs. [abstract][more information]

[muller2004] *Best of both worlds - linking of XUL to x3usgp*, Andreas Muller, Peter Forbrig. [abstract]

[nichols2004] *Describing Appliance User Interfaces Abstractly with XML*, Jeffrey Nichols, Brad A. Myers, Kevin Litwack, Michael Higgins , Joseph Hughes , Thomas K. Harris. [abstract][more information]

[samaan2004] *The AMF Architecture in a Multiple User Interface Generation Process*, Kinan Samaan, Franck Tarpin-Bernard. [abstract]

[stavness2004] *Supporting Workflow in User Interface Description Languages*, Nicole Stavness, Kevin Schneider. [abstract]

[vandenbergh2004] *Evaluation of High-Level User Interface Description Languages for Use on Mobile and Embedded Devices*, Jan Van den Bergh, Kris Luyten, Karin Coninx. [abstract] [more information] [presentation (pdf)]

[zuehlke2004] *useML: A Human-Machine Interface Description Language*, Detlef Zuehlke, Kizito Mukasa, Alexander Boedcher, Achim Reuther. [abstract] [presentation (pdf)]

5 Potential users of UIML

5.1 SEESCOA Project [Software Engineering for Embedded systems using a Component-Oriented Approach]

http://xml.coverpages.org/userInterfaceXML.html

5.1.1 Standardizing Organization

Independent consortium (Flemish)

5.1.2 Overview

The SEECOA project is seeking to adapt Component Based Development (CBD) technology to the needs of embedded systems. As part of this approach, the project has need to define user interfaces for the embedded systems they are creating. They have decided that an XML-based language for defining user interfaces is desirable because it would allow the interfaces to be easily migrated to emerging technology. Embedded systems are constantly being updates and replaced with newer versions. An XML-based language will allow the interfaces to migrate from existing machines to the machines that will replace them, creating a considerable savings in avoided re-training and lost productivity. Early experiments by the project have used XIML as the user interface definition language.

The project focuses on using a component based approach to software engineering for developing embedded systems. A secondary focus is on combining component-based design with a UI description language to get more flexible and adaptable UIs on embedded systems. This UI should be able to migrate from on device to another and adapt to the new device constraints. This increases the re-usability and extensibility of the design. The idea was conceptualized to avoid the problem of re-inventing UIS whenever new technology came to market. The project uses and interaction/task based approach. One of the major things they want is to separate UI design from low-level programming.

5.1.3 Relation to UIML

The SEECOA project would be an excellent testbed for UIML and could provide a good environment to apply UIML into.

5.2 Human Markup TC

5.2.1 Standardizing Organization

OASIS

5.2.2 Overview

Human Traits and Expression through XML"...The HumanMarkup TC is set forth to develop the HumanML and associated specifications. HumanML is designed to represent human characteristics through XML. The aim is to enhance the fidelity of human communication.

5.2.3 Relation to UIML

Pending further investigation

6 Complimentary technologies to UIML

6.1 Web Services for Remote Portals (WSRP) TC

6.1.1 Standardizing Organization

OASIS

6.1.2 Overview

Version 0.88 dated January 2003, prepared jointly by the WSRP and WSIA Technical Committees (TCs).

The Web Services for Remote Portlets (WSRP) specifies a protocol to allow plug 'n' play of user-facing Web services with portalets or other intermediate web applications. Compliant portals can aggregate information from multiple sources and present it to the end users in a plug-and-play manner, without the need for specific presentation logic or API programming. The specification covers both application logic and presentation logic. It supports multi-step transactions, preserving information across a multi-step conversation. Typical applications would be to send the same purchase order to multiple suppliers, or collect information from multiple sources and present it to the user on a single web page.

The WSRP describes a conversation between conforming Producers and Consumer on behalf of end users, where:

- Producers are web services that serve up transaction code or information in a manner that can be used by the consumer
- Consumers are portals, eCommerce applications, or other user-facing applications that aggregate information and/or transactions from multiple producers for presentation to the end user. The Consumer is responsible for re-formatting incoming data for presentation to the end user.

In this dialog, the consumer is responsible for identifying its presentation requirements to the producer. The producer is responsible for rendering its service in a manner that can be consumed by the requestor. The producer may have many renderings of the same service to accommodate different types of consumers.

The protocol supports HTML, XHTML, and XHTML Basic. It provides a common Cascading Style Sheet (CSS style sheet) that portlets SHOULD use in order to avoid generating Consumer-specific markup while preserving the uniform display to end user. WSRP defines the following CSS elements:

Style Element	**Examples**
Font	Portlet-font = normal
Messages	Status, error, alert
Sections	Header/footer, footnote, table
Forms	Input field, button text, form-field, form-field label
Menus	Portlet menu, menu item, selected menu item, sub-menu item, description

6.1.3 Relation to UIML

WSRP and WSIA use HTML dialects and CSS for presentation. At the present time, they recommend use of a common CSS to reduce the effort involved in creating a common presentation style for aggregated data. UIML could streamline this function while providing greater flexibility.

6.2 Web3D

http://www.web3d.org/

6.2.1 Standardizing Organization

Web3D

6.2.2 Overview

Web3D is a consortium formed to provide a community for the creation of open standards for Web3D specifications. This group oversees the evolution of standards such as VRML 1.0 and 2.0. They also seek to educate users on the use of Web3D technologies and advance the adoption of these products. The consortium provides the structure necessary to standardize Web3D technology. The consortium consists of several working groups who oversee various aspects of Web3D technologies.

6.2.3 Relation to UIML

The Web3D consortium is a community focused on creating new user experiences by immersing them in three-dimensional interfaces over the web. The TC's mission should be to ensure that UIML can represent the information required to create such interfaces. Liaising with Web3D will probably be desirable to ensure that UIML keeps up with the technological evolution of the Web3D technologies.

7 Summary

There is much implementation experience with device independence using UIML from people around the world that can be leveraged. Our hope is that UIML concepts can assist with/complement W3C and other standards bodies' activities.

8 Efforts still requiring investigation

- Open GL
- Web Services for Interactive Applications Technical Committee (OASIS)
- XML Localization Interchange File Format Technical Committee (OASIS)
- Open Mobile Alliance (W3C)
- W3C Multimodal Activity
- Internationalization and Localization Working Group (W3C)
- Xpath (W3C)
- XForms Models (W3C)
- XML Events (W3C)

9 References

- [BL] Berners-Lee, T., and Connolly, D., Web Architecture: Extensible Languages, 10 Feb 1998 (www.w3.org/TR/1998/NOTE-webarch-extlang-19980210)
- [CSS] Cascading Style Sheets (www.w3.org/Style/CSS/)
- [PHAN] Phanouri, C., *UIML - A Device-Independent User Interface Markup Language,* Ph.D. dissertation, Virginia Tech, September 2000.
- [PAT] Paterno, F, *Model-Based Design and Evaluation of Interactive Applications,* Springer, 1999.
- [UIML3] UIML3 specification (www.uiml.org)
- [WAI]Web Accessibility Initiative (www.w3.org/WAI/)
- [XML]Extensible Markup Language (www.w3.org/XML/)
- [XE] XML Events (www.w3.org/TR/2001/WD-xml-events-20011026/)
- [XF] XForms - The Next Generation of Web Forms (www.w3.org/MarkUp/Forms/)
- [XH] XHTML Roadmap (www.w3.org/Markup/xhtml-roadmap/)