DTD de UIML

```
<!--
   User Interface Markup Language (UIML)
   ======================================

   Developed by:

      Harmonia, Inc.

   Usage:

      <?xml version="1.0"?>
      <!DOCTYPE uiml PUBLIC "-//Harmonia//DTD UIML 2.0g Draft//EN"
      "http://uiml.org/dtds/UIML2_0g.dtd">

      <uiml>
        <head> ...      </head>
        <template> ...  </template>
        <peers> ...     </peers>
        <interface> ... </interface>
      </uiml>

   Description:

      This DTD corresponds to the UIML 2.0a specification and
      proposed changes posted at the URL http://www.uiml.org/docs/uiml20.

   Change History:

      14 Jun 2001 - Marc Abrams
              - Updated <!ELEMENT uiml...> to avoid message generated by
              - Xerces on certain UIML files.

      13 Jun 2001 - Cara Struble
              - Put in Alan's change to allow <template>, <interface>,
                and <peers> in any order.

      25 May 2001 - Cara Struble
              - Added <restructure> as a child of <template>
              - Made at-part not required for <restructure>

      25 May 2001 - Eric Shell
              - Removed the <d-event> element.

      23 May 2001 - Tom Kuo
              - Changed the "maps-to" attribute in <d-component>
                from #REQUIRED to #IMPLIED

       9 May 2001 - Marc Abrams
            - Updated definition of <action> to allow its body
              to contain any number of <property>, <call>,
              <restructure>, and <event> elements in any order.

       7 May 2001 - Eric Shell
              - Changed how=append to how=union
              - Added url-name attribute to <reference>

       3 May 2001 - Cara Struble
              - Added base attribute to <presentation>

      24 Apr 2001 - Marc Abrams
              - New element:  <listener> (child of <d-class>) to
                replace <d-event>
              - In <d-class>, deleted "event", "method", and
                "listener" and added "attribute" as possible
                values of attribute maps-type.  Added new
                attribute "used-in-tag".

      24 Apr 2001 - Eric Shell
              - Changed the name attribute to the id attribute in many elements.
```

23 Apr 2001 - Cara Struble
- Added <event> as a child of <param> and <property>

20 Apr 2001 - Eric Shell
- Changed the addMethod attribute of <d-event> to attacher

9 Apr 2001 - Cara Struble
- Made changes necessary for the latest version of the revised <peers> syntax proposal:
- New element: d-event
- Added d-event as a child of <d-class>
- Added "listener" as a possibility for maps-type in <d-class>
- Added "method" as a possibility for maps-type in <d-property>

7 Mar 2001 - Cara Struble
- added where and where-part attributes to the <part> tag (per the restructure proposal)

2 Mar 2001 - Kevin Rodriguez
- Added "class" as an acceptable value for the <d-class> maps-type attribute

1 Feb 2001 - Sumanth K. Lingam
- Introduced new Element <op>
- Added <op> as childelement of <condition>
- Added <op> as childelement of <param>
- Added <op> as childelement of <equal>
- Added <op> as childelement of <property>
- Introduced new Elements <when-true>, <when-false>, <by-default>
- Added <when-true> as childelement of <action>
- Added <when-false> as childelement of <action>
- Added <by-default> as childelement of <action>
- Introduced new Element <restructure>

17 Nov 2000 - A Batongbacal
- Changed "d-param" content from (CDATA)? to (#PCDATA)?

16 Jul 2000 - M Abrams
- Added new element: "d-class"
- Renamed element "attribute" to "d-property"
- Renamed element "component" to "d-component"
- Renamed element "method" to "d-method"
- Renamed element "mparam" to "d-param"
- Renamed attribute "returns-value" to "return-type", and made value #IMPLIED to represent a return type
- Added "d-class" as child of element "presentation"
- Added "d-param" as child of element "d-property" (formerly "attribute")
- Dropped "attribute" as child of element "d-component" (formerly "component")

17 May 2000 - M Abrams
- Added attribute "value" to constant element, and deleted PCDATA as a child of constant element

15 May 2000 - M Abrams
- Added attribute "model" to constant element

3 Apr 2000 - M Abrams
- Updated <uiml> to make <peers> last sub-element.

31 Mar 2000 - A Batongbacal
- Updated DTD to UIML spec version "2.0b"
- <component>:
  - Added "location" attribute
- <mparam>: new element split off from <param>
- <method>:
- Renamed attribute "return-value" to "returns-value"
- <system>: dropped because it was unused

22 Mar 2000 - M Abrams
- <component>:
  - Changed #IMPLIED to #REQUIRED for maps-to and name attributes
- <method>:
  - Changed #IMPLIED to #REQUIRED for maps-to and name attributes
  - Deleted attribute types

```
                - Added attribute return-value
                - Deleted <returns>
        16 Jan 2000 - M Abrams
                - Changed "href" attribute back to old name, "source"
                - Changed "task" tag back to old name, "call"
        08 Oct 1999 - C Phanouriou
                - Updated DTD to UIML spec version "2.0a"
                - Major changes and tag renaming
                - Added support for templates and peer components
        31 Jul 1999 - A Batongbacal
                - Updated DTD to UIML spec version "2.0"
        24 Jul 1999 - M Abrams
                - updated to revised language
        15 Jul 1999 - C Phanouriou
                - first draft
-->

<!-- ==================== Content Models ======================== -->

<!--
   'uiml' is the root element of a UIML document.
-->

<!ELEMENT uiml (head?,(template|interface|peers)*) >

<!--
   The 'head' element is meant to contain metadata about the UIML
   document.  You can specify metadata using the meta tag,
   this is similar to the head/meta from HTML.
-->

<!ELEMENT head (meta)*>
<!ELEMENT meta EMPTY>
<!ATTLIST meta
        name    NMTOKEN #REQUIRED
        content CDATA   #REQUIRED>

<!--
   The 'peers' element contains information that defines
   how a UIML interface component is mapped to the target platform's
   rendering technology and to the backend logic.
-->

<!ELEMENT peers (presentation|logic)*>
<!ATTLIST peers
        id     NMTOKEN                  #IMPLIED
        source CDATA                    #IMPLIED
        how    (union|cascade|replace)    "replace"
        export (hidden|optional|required) "optional">

<!--
   The 'interface' element describes a user interface in terms of
   presentation widgets, component structure and behavior specifications.
-->

<!ELEMENT interface (structure|style|content|behavior)*>
<!ATTLIST interface
        id     NMTOKEN                  #IMPLIED
        source CDATA                    #IMPLIED
        how    (union|cascade|replace)    "replace"
        export (hidden|optional|required) "optional">

<!--
   The 'template' element enables reuse of UIML elements.
   When an element appears inside a template element it can
   sourced by another element with the same tag.
-->

<!ELEMENT template (behavior|constant|content|d-class|d-component|interface
                |logic|part|peers|presentation|property|restructure|rule
                |script|structure|style)>
<!ATTLIST template
        id NMTOKEN #IMPLIED>
```

```
<!-- Peer related elements -->

<!--
   The 'presentation' element specifies the mapping between
   abstract interface parts and platform dependent widgets.
-->

<!ELEMENT presentation (d-class*)>
<!ATTLIST presentation
        id     NMTOKEN                    #IMPLIED
        source CDATA                      #IMPLIED
        base   CDATA                      #REQUIRED
        how    (union|cascade|replace)    "replace"
        export (hidden|optional|required) "optional">

<!--
   The 'logic' element specifies the connection between the interface
   and the backend application, including support for scripting.
-->

<!ELEMENT logic (d-component*)>
<!ATTLIST logic
        id     NMTOKEN                    #IMPLIED
        source CDATA                      #IMPLIED
        how    (union|cascade|replace)    "replace"
        export (hidden|optional|required) "optional">

<!--
   The 'd-component' element declares the interface to logic components
   in the backend (e.g., a class in an object oriented language or a
   function in a scripting langauge)
-->

<!ELEMENT d-component (d-method)*>
<!ATTLIST d-component
        id       NMTOKEN                  #REQUIRED
        source   CDATA                    #IMPLIED
        how      (union|cascade|replace)  "replace"
        export   (hidden|optional|required) "optional"
        maps-to  CDATA                    #IMPLIED
        location CDATA                    #IMPLIED>

<!--
   The 'd-class' element declares any name used in the "class"
   attribute of any element in UIML.
-->

<!ELEMENT d-class (d-method*, d-property*, event*, listener*)>
<!ATTLIST d-class
        id          NMTOKEN                  #REQUIRED
        source      CDATA                    #IMPLIED
        how         (union|cascade|replace)  "replace"
        export      (hidden|optional|required) "optional"
        used-in-tag (event|listener|part)    #REQUIRED
        maps-type   (attribute|tag|class)    #REQUIRED
        maps-to     CDATA                    #REQUIRED>

<!--
  The 'd-property' element associates a specific property with the
  methods that set and get its value.
-->

<!ELEMENT d-property (d-method*, d-param*)>
<!ATTLIST d-property
        id           NMTOKEN                          #REQUIRED
        maps-type    (attribute|getMethod|setMethod|method)  #REQUIRED
        maps-to      CDATA                            #REQUIRED
        return-type  CDATA                            #IMPLIED>

<!--
  The 'method' element describes a routine that forms part
  of a component's callable interface (i.e., the component's "API").
```

```
-->

<!ELEMENT d-method (d-param*, script?)>
<!ATTLIST d-method
        id            NMTOKEN                  #REQUIRED
        source        CDATA                    #IMPLIED
        how           (union|cascade|replace)  "replace"
        export        (hidden|optional|required) "optional"
        maps-to       CDATA                    #REQUIRED
        return-type   CDATA                    #IMPLIED>

<!--
   'd-param' denotes either a single formal parameter to a callable
    routine or attribute in a markup language
-->

<!ELEMENT d-param (#PCDATA)>
<!ATTLIST d-param
        id   NMTOKEN #IMPLIED
        type CDATA   #IMPLIED>


<!--
   The 'script' element contains data passed to an embedded scripting
   engine. The type specifies the scripting language (see HTML4.0)
-->

<!ELEMENT script (#PCDATA)>
<!ATTLIST script
        id     NMTOKEN                  #IMPLIED
        type   NMTOKEN                  #IMPLIED
        source CDATA                    #IMPLIED
        how    (union|cascade|replace)  "replace"
        export (hidden|optional|required) "optional">


<!-- Interface related elements -->

<!--
   The 'structure' element describes the initial logical relationships
   between the components (i.e., the "part"s) that comprise the user
   interface.
-->

<!ELEMENT structure (part*)>
<!ATTLIST structure
        id     NMTOKEN                  #IMPLIED
        source CDATA                    #IMPLIED
        how    (union|cascade|replace)  "replace"
        export (hidden|optional|required) "optional">

<!--
   A 'part' element describes a conceptually complete component of the
   user interface.
-->

<!ELEMENT part (style?, content?, behavior?, part*)>
<!ATTLIST part
        id        NMTOKEN                  #IMPLIED
        class     NMTOKEN                  #IMPLIED
        source    CDATA                    #IMPLIED
        where     (first|last|before|after)  "last"
        where-part NMTOKEN                 #IMPLIED
        how       (union|cascade|replace)  "replace"
        export    (hidden|optional|required) "optional">

<!--
   A 'style' element is composed of one or more 'property' elements,
   each of which specifies how a particular aspect of an interface
   component's presentation is to be presented.
-->

<!ELEMENT style (property*)>
```

```
<!ATTLIST style
        id     NMTOKEN                  #IMPLIED
        source CDATA                    #IMPLIED
        how    (union|cascade|replace)    "replace"
        export (hidden|optional|required) "optional">

<!--
   A 'property' element is typically used to set a specified
   property for some interface component (or alternatively,
   a class of interface components), using the element's
   character data content as the value.  If the 'operation'
   attribute is given as "get", the element is equivalent to
   a property-get operation, the value of which may be "returned"
   as the content for an enclosing 'property' element.
-->

<!ELEMENT property (#PCDATA|constant|property|reference|call|op|event)*>
<!ATTLIST property
        name       NMTOKEN                  #IMPLIED
        source     CDATA                    #IMPLIED
        how        (union|cascade|replace)    "replace"
        export     (hidden|optional|required) "optional"
        part-name   NMTOKEN                  #IMPLIED
        part-class  NMTOKEN                  #IMPLIED
        event-name  NMTOKEN                  #IMPLIED
        event-class NMTOKEN                  #IMPLIED
        call-name   NMTOKEN                  #IMPLIED
        call-class  NMTOKEN                  #IMPLIED>


<!--
   A 'reference' may be thought of as a property-get operation,
   where the "property" to be read is a 'constant' element defined
   in the UIML document's 'content' section.
-->

<!ELEMENT reference EMPTY>
<!ATTLIST reference
        constant-name  NMTOKEN #IMPLIED
        url-name       NMTOKEN #IMPLIED>

<!--
   The 'content' element is composed of one or more 'constant'
   elements, each of which specifies some fixed value.
-->

<!ELEMENT content (constant*)>
<!ATTLIST content
        id     NMTOKEN                  #IMPLIED
        source CDATA                    #IMPLIED
        how    (union|cascade|replace)    "replace"
        export (hidden|optional|required) "optional">

<!--
   'constant' elements may be hierarchically structured.
-->

<!ELEMENT constant (constant*)>
<!ATTLIST constant
        id     NMTOKEN                  #IMPLIED
        source CDATA                    #IMPLIED
        how    (union|cascade|replace)    "replace"
        export (hidden|optional|required) "optional"
        model  CDATA                    #IMPLIED
        value  CDATA                    #IMPLIED>

<!--
   The 'behavior' element gives one or more "rule"s that
   specifies what 'action' is to be taken whenever an associated
   'condition' becomes TRUE.
-->

<!ELEMENT behavior (rule*)>
```

```
<!ATTLIST behavior
        id     NMTOKEN                  #IMPLIED
        source CDATA                    #IMPLIED
        how    (union|cascade|replace)    "replace"
        export (hidden|optional|required) "optional">


<!ELEMENT rule (condition,action)?>
<!ATTLIST rule
        id     NMTOKEN                  #IMPLIED
        source CDATA                    #IMPLIED
        how    (union|cascade|replace)    "replace"
        export (hidden|optional|required) "optional">

<!--
   At the moment, "rule"s may be associated with two types of
   conditions: (1) whenever some expression is equal to some other
   expression; and (2) whenever some event is triggered and caught.
-->

<!ELEMENT condition (equal|event|op)>

<!ELEMENT equal (event,(constant|property|reference|op))>

<!ELEMENT op (constant|property|reference|call|op|event)*>
<!ATTLIST op
        name   CDATA                    #REQUIRED>

<!ELEMENT action (((property|call|restructure)*,event?)|(when-true?,when-false?,by-default?))>

<!ELEMENT when-true ((property|call)*,restructure?,op?,equal?,event?)>

<!ELEMENT when-false ((property|call)*,restructure?,op?,equal?,event?)>

<!ELEMENT by-default ((property|call)*,restructure?,op?,equal?,event?)>

<!ELEMENT restructure (template)?>
<!ATTLIST restructure
        at-part NMTOKEN                     #IMPLIED
        how    (union|cascade|replace|delete) "replace"
        where  (first|last|before|after)      "last"
        where-part NMTOKEN                     #IMPLIED
        source  CDATA                       #IMPLIED>

<!ELEMENT call (param*)>
<!ATTLIST call
        name  NMTOKEN #IMPLIED
        class NMTOKEN #IMPLIED>

<!--
   'event' denotes one of three things:
   (1) When a child of <condition> or <op>, denotes that when the named
      event is fired, the condition should be evaluated.
   (2) When a child of <action>, denotes that the named event should
      be fired.
   (3) Inside <d-class>, denotes that the named event can occur for
      the part class named by the <d-class>.
-->

<!ELEMENT event EMPTY>
<!ATTLIST event
        name       NMTOKEN #IMPLIED
        class      NMTOKEN #IMPLIED
        part-name  NMTOKEN #IMPLIED
        part-class NMTOKEN #IMPLIED>

<!--
   'param' denotes a single actual parameter to a call-able routine.
-->

<!ELEMENT param (#PCDATA|property|reference|call|op|event)*>
<!ATTLIST param
        name NMTOKEN #IMPLIED>
```

```
<!--
   'listener' denotes that a name defined with d-class
   used-in-tag="listener" should be attached as a listener to the
   d-class which contains this <listener> element.
-->

<!ELEMENT listener EMPTY>
<!ATTLIST listener
        class    NMTOKEN #IMPLIED
        attacher CDATA   #IMPLIED>
```