A Conceptual Model and Specification Language for Mixed Reality Interface Components

Conference Paper · March 2006

CITATIONS
3

READS
100

3 authors:

Pablo Figueroa
Los Andes University (Colombia)
111 PUBLICATIONS **386** CITATIONS
SEE PROFILE

Raimund Dachselt
Technische Universität Dresden
236 PUBLICATIONS **3,059** CITATIONS
SEE PROFILE

Irma Lindt
Technische Hochschule Köln
37 PUBLICATIONS **709** CITATIONS
SEE PROFILE

A Conceptual Model and Specification Language for Mixed Reality Interface Components

Pablo Figueroa*
Universidad de los Andes.
Colombia

Raimund Dachselt†
Dresden University of Technology.
Germany

Irma Lindt‡
Fraunhofer FIT, St. Augustin.
Germany

ABSTRACT

This paper presents a uniform approach for specifying mixed reality user interfaces, including 3D interaction techniques and 3D widgets. Our main goal is to facilitate the process of reusing previous work, so more complex applications can be built and documented in a formal and uniform way. Our work builds on previous experiences and taxonomies in related fields. Our specification method is described at two levels: a conceptual model and a specification language. The conceptual model provides means to generalize user interface components and to port them to different hardware settings and application contexts. An XML-based specification language implements the conceptual model and allows for automatic processing and tool support. Several examples show main features of the proposed representation and demonstrate the applicability of both model and language to different mixed reality user interfaces, including Desktop VR, Immersive VR, and Augmented Reality.

CR Categories: I.3.6 [Computer Graphics]: Methodology and Techniques—Languages D.2.1 [Software Engineering]: Requirements/Specifications—Languages;

Keywords: 3D User Interfaces, 3D Interaction Techniques, Mixed Reality, VR, AR, Desktop VR, 3D Widgets, Interface Description Language

1 INTRODUCTION

The interface of 3D applications is still a field with many uncertainties for designers and users. Despite of more than a decade of research in the field, designers still lack standard tools, methodologies, or guidelines to pursue their work. 3D user interfaces (3DUIs) are often developed from scratch, although useful solutions already exist. This is partially due to the limited capabilities that implementation environments offer to their users, but it is also due to the lack of standards for 3DUIs, which results in a wide variety of non comparable results in the field. End users also suffer from this situation. Since 3D applications are rather application and hardware-specific, it is difficult to anticipate how to interact with a 3D application. This impacts the usability of a 3D application and affects users' learning curves.

Results in the field are promising and diverse, but there are still big difficulties in the creation of complex, portable mixed reality (MR) [22] applications. It is usually difficult to reuse previous solutions for new developments. 3DUIs are generally described in plain text, mostly source code, which leads to an incomplete description of the interaction techniques involved, the intended method of use, and the context in which the interfaces were tested. The lack of a standard description language makes difficult the processes of analyzing, comparing, and reusing techniques. It is also difficult to abstract ideas from previous experiences and use them in new applications, which may be running in totally different hardware setups. Finally, the lack of a common description language also precludes the creation of supporting tools, guidelines, and methodologies, which are important in the professional work of 3D application designers.

This work attempts to provide a unifying method for the description of 3DUIs. Our goals with this work are the following:

- Increase the accessibility of 3DUI concepts. We would like to build a repository of 3DUI descriptions that can be used to categorize, compare, and to find 3DUI components in order to design new mixed reality applications.
- Lay the foundations for further developments in the field of 3DUIs. A unified description of generic 3DUI components might encourage the development of new and more complex components through combination or adaptation of existing ones.
- Push the development of 3DUI guidelines and tools. Guidelines and tools can be developed based on a unified description, e.g. to ease the selection of appropriate 3DUI components or to combine and adjust them.
- Push the standardization of 3DUI concepts and their descriptions. We are aiming at a standardized description of 3DUIs that can be referenced from different frameworks and that might serve as a general specification and exchange format.

The following sections are organized as follows: Section 2 discusses previous and related work in the field of 3D user interface components and description languages. Section 3 presents our model for the formal specification of 3DUI components along with several examples. Section 4 introduces the XML-based specification language ICDL, again providing an example. Section 5 concludes this paper and points out future directions.

2 RELATED WORK

As the basis for our work we have analyzed previous research on 3DUI with the focus on classifying and describing components of mixed reality interfaces. In the field of VR numerous 3D interaction techniques have been developed by various researchers during the past decade. The survey by Hand [14] examines some techniques for object manipulation, navigation, and application control. A more recent overview is given in the book on 3D user interfaces by Bowman et al. [5], where techniques are classified in terms of task decomposition using the following categories: selection, manipulation, travel, wayfinding, system control, and symbolic input. Other classification approaches exist such as the subdivision in egocentric and exocentric manipulation techniques by Poupyrev et al. [23]. However, they do not include specification approaches for specific techniques. Figueroa et al. presented InTml [11], an XML-based specification of interaction techniques that was used to create

*e-mail:pfiguero@uniandes.edu.co
†e-mail:dachselt@inf.tu-dresden.de
‡e-mail: irma.lindt@fit.fraunhofer.de

a set of reusable components and influenced this work. Within the Open Tracker project [27] an object-oriented architecture is used, which also employs an XML-based approach to achieve a ’write once, input anywhere’ VR application development.

3D interaction in the field of Desktop VR is usually done with the help of 3D widgets, since they allow the subdivision of higher-dimensional interaction tasks into sub tasks being suitable for lower-dimensional input devices. However, most of the many existing 3D widget solutions were developed within the context of immersive VR systems. In 1997 Leiner et al. presented a first overview on existing 3D widgets, roughly classifying them into geometric manipulation and application control [18]. Dachselt and Hinz presented in [9] a unifying widget classification according to the criteria interaction purpose / intention of use. They identified the following main categories: direct 3D object interaction, 3D scene manipulation, exploration and visualization, and system/application control. This classification can be found online [37] and contains more than 70 basic widget types from the literature. For the specification of those widgets an XML-based language was also proposed within the Contigra research project [10, 37]. Although it is mainly suitable for Desktop VR, tailored to Web3D applications and not sufficiently general, it influenced the language proposed here.

Various interaction techniques were also developed in the field of Augmented Reality (AR) in the past few years. The work of Azuma et al. presents a number of advanced solutions [3]. To our knowledge there exist no detailed taxonomies nor generalized specification languages for AR, partly because of the novelty of this research area. However, some of the objectives pursued by MR authoring systems such as DART [21] and APRIL [17] are similar to those presented here. DART extends a 2D authoring system with plugins and scripts to allow for the creation of AR applications; APRIL describes AR presentations based on state charts. Typically, these systems offer a set of best practice interaction techniques and 3D widgets targeting only a specific application domain.

It can be summarized that individual interface techniques and 3D widgets were developed in all fields, whereas classification approaches and especially generalized specification languages only rarely exist. Moreover, there is no unifying model or descriptive language available for the whole mixed reality continuum by now.

In addition to that it can be observed that XML-based user interface description languages [34] are gaining attention in the field of GUIs, especially for mobile devices. The advantages of using XML-based languages for flexible and parameterized interface descriptions have become obvious with languages such as the User Interface Markup Language (UIML) [2]. One of its extensions, CUIML [28] uses XSL transformations to convert a CUIML-based user interface description to markup languages that can be displayed on various I/O devices, including markup languages such as VRML97 and HTML.

3 A Conceptual Model for 3D Interface Components

This section presents our model for the formal specification of interface components. 3DUIs are described as a set of typed components. Some operations defined over the type system might allow us to generalize and port components to different hardware platforms, so we can reuse definitions at an abstract level. For the design of the conceptual model we studied the design space of 3D interface components by comparing existing 3DUIs and their properties. Based on the design space, we have identified several dimensions of interest for the formal specification of interface components including the following:

- The required interaction devices. Interface components process input data received from input devices and produce some output sent to output devices. Devices required by an interface component might be rather exchangeable or there might be a strong dependency on specific device types.

- The involved modalities. The input and output data of an interaction component might support different sensory perceptions.

- The application type addressed in the mixed reality continuum. Certain interaction components only make sense in the context of an AR, VR, or Desktop VR application, such as the GoGo arm extension technique [24] in an immersive VR setting.

- The relationships with other components. Some interface components can, should, or need to be combined with other interface components.

- The behavior of the component. Interface components implement different behaviors, meaning that they are producing different outputs for the same input values.

- The target group. Interface components might be rather universally applicable or they might be designed to meet the needs of specific user groups e.g. by employing metaphors or interaction devices that are well-known to this group.

- The user task supported. Typically, interface components support one or several tasks, such as navigation or selection.

- The required user skills. Interface components require a user to perform certain actions ranging from simple button clicks to actions that are physically challenging.

In this first attempt for a conceptual model, we concentrate on the first six aspects: devices, modalities, application types, relationships, target groups, and behaviors. Further work needs to be done in order to specify user tasks and required user skills.

The following subsections present some definitions, the formal language, the treatment of devices and events, and some examples of use.

3.1 Definitions

A 3D Interaction Component (3DIC) is an identifiable element in the user interface of a mixed reality application employing 3D content. We want to use this term in order to unify and include others currently used in the literature, such as 3D interaction technique, 3D widget [8], or 3D gadget [29]. Such modular elements are currently implemented as manipulators [32], PROTOs [7], behaviors [33], or callback functions [30, 16], depending on the implementation technology used.

The description of any 3DIC includes the following elements, explained in more detail later in the paper: a name, the devices involved in its execution, the abstract events that could be extracted from the devices, a state machine, a set of parameters, the context in which it has been run, and a description of its execution. Not all elements may be required in particular cases, and current descriptions usually cover a subset of the ideal description for any 3DIC. The following subsection shows a notation for this concept, and Section 4.2 shows the implementation as an XML schema.

A particular 3D widget, gadget, or interaction technique could be represented by one or several 3DICs, depending on the level of abstraction of the description. We propose the following lifecycle for 3DICs:

1. First, a 3DIC is used to describe an interaction technique or widget as it is shown in its origin, which may be a paper, a book, an application, or an API. We call this first stage Specific 3DIC (S-3DIC). The input and output of a S-3DIC is usually dependent of specific devices, and its execution may be described directly in terms of input and output events from its devices, which precludes its generality.

2. A generalization work can be done on the S-3DIC description, so input and output events are identified without references to particular devices, mapping from devices to events are described, and execution description is based in events, without references to devices. Further work can be done so this new description is decomposed into *basic* components. We call these basic 3DICs Generic 3DICs (G-3DICs), and they represent the abstract and reusable representation of any widget or interaction technique described.

The set of G-3DICs represents the reusable and platform independent set of 3DICs, which may be the most important contribution of an unifying representation framework such as this one. Initially, we are assuming that the transition from S-3DICs to G-3DICs is made by experts in the field, which certify its quality. We envision supporting tools that will guide future developers in the process of identifying suitable G-3DICs from their S-3DICs.

3.2 Formal Description

Any 3DIC has a name, and refers to a tuple of the form $\{St, Par, InpDev, OutDev, InpEv, OutEv, DevMap, Exec, Cont\}$. Each element in the tuple is a set of a certain type, which may have elements, be empty, or be unknown, if there is not enough information.

St stands for *States*, a state machine that describes the behavior of a 3DIC. We use the state machine definition in UML [4], although our examples here show only the state names, for space and simplicity reasons.

Par stands for *Parameters*, a set of values used for initial setup. Each element in *Par* has the form $\{id, type\}$, where type is a defined type in the parameter space.

InpDev and *OutDev* are the sets of input and output devices. Each device is described as the tuple $\{Par, InpEv, OutEv\}$, which are the device's setup parameters, input events, and output events, respectively.

InpEv and *OutEv* are the input and output events of a 3DIC. *InpEv* can be defined implicity as the union of all *InpEv* in the 3DIC's input devices, and accordingly for *OutEv*. In the case of G-3DIC, which do not have devices, *Par*, *InpEv*, and *OutEv* are defined explicitly and they represent the interface of such a component. *InpEv* and *OutEv* are represented also as tuples with the same type as parameters, and they define the space of events of a component.

DevMap stands for *DeviceMapping*, which are functions that relate 3DIC input and output events to device input and output events. In other words, they define how a 3DIC input event is related to output events from the 3DIC input devices, and accordingly to 3DIC output events.

Exec describes the execution as an algorithm in pseudocode. We decided to use a pseudocode that closely follows the syntax of ECMAScript [15], and in particular the bindings that this language has in X3D [35], due to the availability of its specification and its closeness to our purpose.

Cont stands for *Context*, and describes the required elements for the full specification of a 3DIC, on top of the previously mentioned. It is composed of the types that define the space for parameters, devices, and events, as well as additional types and objects used in the description of the 3DIC's execution. Since it is already included in other parts of the description, we do not explicitly extract its representation, but it is important to be aware of its existence.

3.3 Devices, Events, and Parameters

The definition of a 3DIC is based on types for several items, in particular for devices, events, and parameters. Seminal work in the field of HCI [13, 6] present theories of how we can categorize input information. In this previous work devices are presented as a collection of sensors, each one from a small set of possible types. Less work has been done in the standardization of types for events and parameters of 3DICs, but it is our belief that any list will always be incomplete, since it is always possible to use any data structure as a parameter or event type. Nevertheless we believe a basic set of types is required, in order to unify descriptions. We decided to re–use the set of basic types in X3D [36] (excluding SFNode and MFNode), since it is a well known reference.

Device types are constructed on top of simple types. We decided to define a simple set of devices instead of just leaving input event types, since we consider such types a more concrete idea for future 3DIC designers. Again, this simple set should be enhanced with more device definitions. Our current set of generic devices is shown in Figure 1, in a notation similar to UML classes. Although the actual mapping from this abstract definition to a specific device is unknown, we believe the actual implementation could be straightforward.

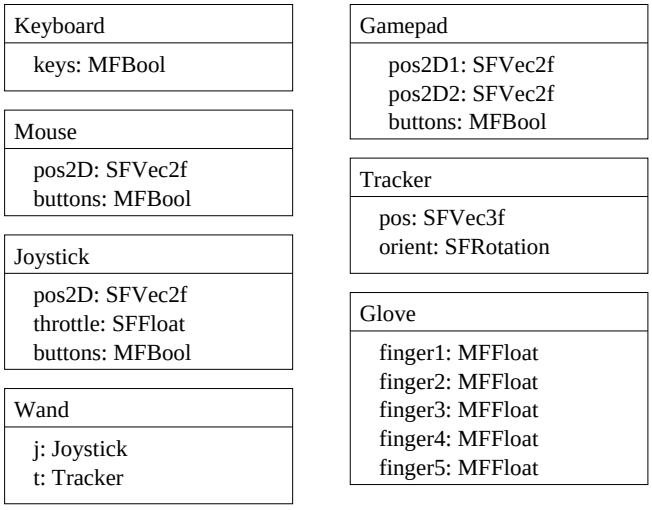

Figure 1: Basic Device Types

3.4 Examples

Here we show several examples of use of this description language, each one showing different types of applications in the mixed reality continuum, featuring component abstraction and subdivision, in VR, Desktop VR, and AR scenarios.

3.4.1 Generic Component from a Specific One

We show here our description of the GoGo Technique [24], a technique for lengthen the user's hand in the virtual world. From the description in the paper, we extracted the S-3DIC shown in Listing 1. You can notice the use of trackers for input, and the use of formulas from the paper.

Go-Go [24] :
St{ LINEAR_MOV NON_LINEAR_MOV }
Par{ $k_{SFFloat}$ $D_{SFFloat}$ $dht_{SFVec3f}$ $virtualHand_{Object3D}$ }
InpDev{ $head_{Tracker}$ $hand_{Tracker}$ }
Exec{

```
  initialize() {
    headReceived = false;
    handReceived = false;
  }
  head.pos( value, timestamp ) {
    headPos = value;
    headReceived = true;
    computeOutput();
  }
  hand.pos( value, timestamp ) {
    handPos = value;
    handReceived = true;
    computeOutput();
  }
  function computeOutput() {
    if( headReceived && handReceived )
    {
      chestPos = headPos - dht;
      handInPolar = cartesian2Polar(handPos - chestPos);
      if( Rr.length() < D )
        Rv = Rr;
      else
        Rv = Rr + k*(Rr-D)*(Rr-D);
      virtualHand.pos = polar2Cartesian(Rv, rho, theta)+
                        chestPos;
      headReceived = handReceived = false;
    }
  }
}
```

Listing 1: S-3DIC Description for Go-Go

The listing is read as follows: Go-Go is a S-3DIC defined in [24]. It has two states, for linear and non–linear movement. As parameters, it requires a distance for the change of movement (D), a coefficient between 0 and 1 (k), the distance between the user’s torso and her head (dht), and a geometrical representation of the user’s hand (virtualHand). The technique is executed with the use of two trackers, one for her head and one for her hand. The execution is divided in several functions. Initialize is executed once, at the beginning of the program execution. head.pos is a function that is called once an event *pos* from the device *head* is received. computerOutput is a user defined function.

From the S-3DIC description, we produced a G-3DIC description of Go-Go. The fundamental idea in this technique is the use of a smooth linear and non-linear mapping of user input. In this case, such a quantity is abstracted and it is left alone, since all other elements are complimentary to the technique. Note the use of a generic type *AType*, in order to generalize the type of information that is processed. Listing 2 shows our generic version for Go-Go.

3.4.2 Several G-3DICs from one S-3DIC

One text–based description of a VR application is found in [31], for NICE. NICE is an environment that allows geographically–distant kids to interact within a virtual world. The interaction techniques of the application are described in categories, first navigation, then manipulation. We created two 3DICs, *NICENavigation* and *NICEManipulation*. From those specific components we abstracted the basic techniques used. For example, Figure 2 shows the set of generic components we created from *NICENavigation*. Notice that several levels of abstraction are possible. Listing 3 shows excerpts for *NICENavigation*.

Gengogo :
St{ LINEAR_MOV NON_LINEAR_MOV }
Par{ $k_{SFFloat}$ $D_{SFFloat}$ }
InpEv{ $inputVal_{AType}$ }
OutEv{ $result_{AType}$ }
Exec{

```
  inputVal( val, timestamp ) {
    value = val;
    computeOutput();
  }
  function computeOutput() {
    if( value.length() <  D ) )
      result = value;
    else
      result = value + k*(distance(value)-D)*
               (distance(value)-D);
  }
}
```

Listing 2: G-3DIC Description for Go-Go

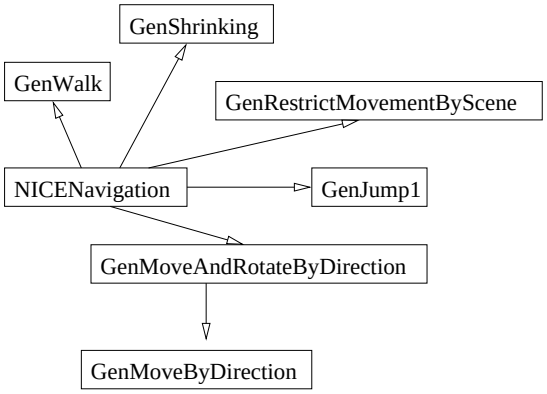

Figure 2: Generic 3DICs from NICENavigation. Lines with arrows show how G-3DICs are abstracted

3.4.3 An Augmented Reality Technique

Our first example in the field of AR comes from an early tangible user interface developed by Fitzmaurice et al. in the Bricks project [12]. The specific interface component moves and rotates a virtual object displayed on a horizontal surface by manipulating a physical handle (brick) on top of it. To attach the physical handle to the virtual object the physical handle is placed directly above the virtual object. To release the virtual object, the physical handle is lifted up. Listing 4 shows the interaction component as a S-3DIC.

The second interaction component (see Listing 5) is one of several interaction techniques realized in the Tiles project [26] by Poupyrev et al. The proximity copying interaction component copies 3D graphics from a menu. The copy action is activated if a physical artefact remains in the proximity of a menu item for a certain time frame.

Both interaction components can be used together in a user interface, e.g. by using a tracker that serves as input for both interaction components or by automatically attaching a copied object to the tracker. In the latter case, the graspable object interaction component would need to be extended with the input event newContentCreated that is an output event of the proximity copying interaction component.

NICENavigation [31] :
St{ WALKING STANDING MOVING_BY_WAND JUMPING SHRINKING }
Par{ $\text{initialPos}_{\text{SFVec3f}}$ $\text{initialOrient}_{\text{SFRotation}}$ $\text{angleThreshold}_{\text{SFFloat}}$ $\text{speedOfRotation}_{\text{SFFloat}}$ $\text{speedOfMovement}_{\text{SFFloat}}$ $\text{avatar}_{\text{Object3D}}$ }
OutEv{ $\text{avatarPos}_{\text{3DValue}}$ $\text{avatarOrient}_{\text{3DOrientation}}$ }
InpDev{ $\text{wand}_{\text{Wand}}$ $\text{headTracker}_{\text{Tracker}}$ }
Exec{

```
  initialize() {
    avatarPos = initialPos;
    avatarOrient = initialOrient;
    previousPos = headTracker.pos;
    previousOrient = headTracker.orient;
  }
  wand.buton[0] { // Jump button
    inJump = true; computeOutput();
  }
  wand.button[1] { //shrinking
    inShrinking = true; computeOutput();
  }
  function computeOutput() {
    if(inJump) {
      // Jump ten times its height and slowly float down
      // Move avatar according to the parabolic movement
      finishJump = jump( avatar, deltaPos, deltaOrient )
      if( finishJump )
        inJump = false;
    }
    if( inShrinking ) {
      // Move avatar towards the closest 'floor' object
    }
    // Compute delta movement and if move can be done
    ...
  }
}
```

Listing 3: S-3DIC Description for NICENavigation

GraspableObjectMovement [12] :
St{ ATTACHED_OBJECT DETACHED_OBJECT }
Par{ $\text{content}_{\text{SetOfObject3D}}$ $\text{displayRegion}_{\text{Region3D}}$ }
InpEv{ $\text{trackedObjectPos}_{\text{SFVec3f}}$ $\text{trackedObjectOrient}_{\text{SFRotation}}$ }
OutEv{ $\text{virtualObjectGrabbed}_{\text{Object3D}}$ }
InpDev{ $\text{brick}_{\text{Tracker}}$ }
Exec{

```
  initialize() {
    selectedObj = null;
  }
  trackedObjectPos( value, timeStamp ) {
    prevGrabbed = selectedObj;
    if( displayRegion.hasInside( value ) ) {
      selectedObj = getSelectedObject( content );
      if( selectedObj != null ) {
        mapPos2Object( selectedObj, value );
        if( prevGrabbed != selectedObj )
          virtualObjectGrabbed = selectedObj;
      }
    }
    else selectedObj = null;
  }
  trackedObjectOrient( value, timeStamp ) {
    if( selectedObj != null )
      mapOrient2Object( selectedObj, value );
  }
}
```

Listing 4: S-3DIC Description for graspable object movement

ProximityCopying [25] :
St{ DISTANCE_CALCULATION COPYING }
Par{ $\text{copyDistance}_{\text{SFFloat3f}}$ $\text{timeThreshold}_{\text{SFTime}}$ $\text{menu}_{\text{Menu}}$ }
InpEv{ $\text{timeChanged}_{\text{Trigger}}$ }
OutEv{ $\text{newContentCreated}_{\text{Trigger}}$ $\text{newObject}_{\text{Object3D}}$ }
InpDev{ $\text{handle}_{\text{Tracker}}$ }
Exec{

```
  initialize() {  previousSel = null; }
  handle.pos( value, timestamp ) {
    selection = getSel( menu, value, copyDistance );
    if( selection != previousSel ) {
      initTime = getCurrentTime();
      previousSel = selection;
    }
  }
  timeChanged( value, timestamp ) {
    if( previousSel != null )
      if( getCurrentTime() - initTime > timeThreshold )
        copyObject();
  }
  function copyObject() {
    newContentCreated = true;
    newObject = copy(previousSel); previousSel = null;
  }
}
```

Listing 5: S-3DIC Description for proximity copying

3.4.4 A Desktop VR Technique

The example depicted in Listing 6 shows a formal description of a 3D ring menu widget in the area of Desktop VR. The original ring menu [19] is rotated directly using the input device. In this Desktop VR example two additional button widgets are used to activate rotation in one or the other direction with the help of a 2DOF device, usually a mouse. Also note various geometry parameters to configure the appearance of the widget. We created another 3DIC from the implementation at [19], and a G-3DIC from both of them, not shown here but available online [1].

4 ICDL: A Unified XML-based Specification Language

In addition to the underlying conceptual model for 3DICs and their formal description presented in Section 3 it is desirable to have a consistent specification language for 3DICs.

4.1 Requirements and Goals

Whereas the formal description already supports a unique identification of 3DICs, an extended specification language should allow a consistent description of 3DICs at a high level of detail to avoid ambiguities and to realize the model presented in Section 3. It must be possible to describe arbitrary 3DICs – modular components rather than an interface as a whole – in current and future mixed reality applications. That is why the language should be extensible to permit further enhancements, additions and new meta data. Another goal was a reduced learning curve along with an easy and familiar syntax, at the same time allowing for automatic processing and tool support. Independence from existing frameworks and specific exchange formats for 3DUIs should be guaranteed, translation to other description languages and classification approaches ought to be feasible. To meet these requirements we decided to develop a specification language based on XML schema. We propose the *Interface Component Description Language* (ICDL), which is presented in the following subsections. It can also be found online at [1] along with various ICDL instance documents.

RingMenu [19, 37] :
St{ OBJECT_SELECTED MENU_ROTATING }
Par{ $\text{itemList}_{\text{MFObject3D}}$ $\text{firstSelected}_{\text{Object3D}}$ $\text{rotationSpeed}_{\text{SFFloat}}$ $\text{itemRatio}_{\text{SFFloat}}$ $\text{menuRadius}_{\text{SFFloat}}$ $\text{rotationWidgets}_{\text{Object3D}}$ $\text{additionalGeometry}_{\text{Object3D}}$ $\text{highlightGeometry}_{\text{Object3D}}$ }
InpEv{ $\text{directItemSelection}_{\text{Object3D}}$ $\text{activateRotationLeft}_{\text{SFBool}}$ $\text{activateRotationRight}_{\text{SFBool}}$ }
OutEv{ $\text{currentItemChanged}_{\text{SFBool}}$ $\text{selectedItem}_{\text{Object3D}}$ }
InpDev{ $\text{mouse}_{\text{2DOF}}$ }
Exec{

```
  initialize() {
    arrangeObjectsInRing(itemList, menuRadius, itemRatio);
    directItemSelection( firstSelected, 0 );
  }
  directItemSelection( value, timestamp ) {
    rotateToObject(value); selectedItem = value;
    currentItemChanged = true;
  }
  activateRotationRight( value, timestamp )
  {
    selectedItem = rotateRight();
    currentItemChanged = true;
  }
  ...
}
```

Listing 6: S-3DIC Description for a Ring Menu

4.2 Basic Structure

The Interface Component Description Language allows for the description of generic and specific 3D interface components in terms of separate root elements. For example, Figure 3 shows the basic structure of the S-3DIC type.

An *Identification* element provides a name and ID attribute as well as elements to uniquely identify a 3DIC. This includes several meta data, among them author and version within the *Modification* element. The relation to other 3DICs, either informal or in terms of inheritance, is expressed with the following elements *Related3DIC* and *ExtensionOf* referring to other 3DIC specifications. Some of the dimensions of the conceptual model described in 3 are expressed with *SuitableDomain* and *TargetGroup*. In order to provide means of extending the identification with user-specific meta data or to allow future enhancement there is also a generic *MetaData* element.

To identify the origin of a specific 3D interface technique one or more references to the literature can be included in the *Origin* element as BibTeX entries. To conclude with the informal part of a 3DIC description the element *Documentation* may contain a verbal and some figurative illustrations of the S-3DIC as well as links to Websites providing additional information and generic meta data.

The elements *Parameters*, *Events* and *Devices* reproduce the elements of the formal description introduced in Section 3.2. All parameters possess an ID, value and type. If possible, the type is chosen according to the X3D types to ensure consistency and a standardized definition. A documentation attribute also belongs to the *Parameter* element. This attribute is also occurring in other places of the specification. The *Events* section is divided into *InputEvents* and *OutputEvents*. *InputDevices* and *OutputDevices* are listed in the following main section. Both events and devices possess a unique id, a type and a description. The *Devices* element might also contain *InDeviceEventMappings* and *OutDeviceEventMappings* with so called event maps. Each *EventMap* maps a device event to an interface event and vice versa.

Finally, the *Execution* and *States* parts provide several ways to illustrate the basic functionality and algorithm(s) behind a 3DIC. *Execution* includes elements and attributes for an informal imperative description, references to Web pages including code or additional descriptions as well as direct notations of pseudo code or even program code. By now the different states are only described as simple entries. However, it should not be difficult to add formal state machine descriptions on the basis of XML such as XML Metadata Interchange (XMI) for UML state machines [38].

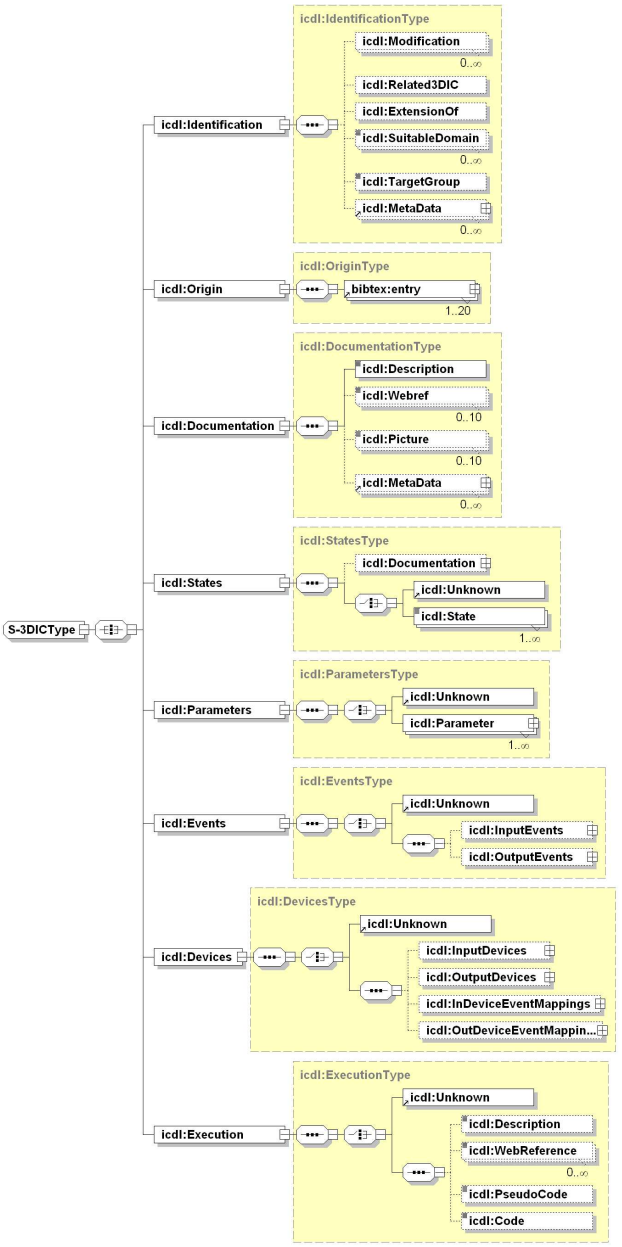

Figure 3: Basic Structure of the Specific-3DIC Description with the XML specification language ICDL

Generic 3DICs are very similar in structure and only differ in minor aspects. Due to their generalized nature they do not include devices at all, thus being independent of concrete realizations. Within the *Identification* element references to several other S-3DICs can be included to establish a link to the basis of the conducted abstraction. In addition to that the *Origin* is an optional element, since generic descriptions of 3D interaction techniques or widgets are only rarely to be found within the literature by now.

4.3 3DIC XML Example

Listing 7 shows the main elements of the XML definition of a ring menu [19]. This is an example of a specific 3DIC implementation adapted for the area of Desktop VR. Please note the rich description within the *Identification* element including some additional meta data. Also note that a few documentation attributes or elements are omitted for better readability and can be found online [1].

The presented specification language ICDL and sample instance documents lay the ground for further improvements, which may eventually lead to future standards for developing mixed reality applications. To evaluate the language we have written a number of example specifications for specific (see for example Listing 7) and generic 3DICs including all 3DICs presented in Section 3.4. Due to the XML format and its flexible structure a number of operations can be performed on the language and instance documents, such as translation to other formats by XSLT, consistent code generation, uniform web–based presentation [1], graphical visualization schemes for 3DIC relationships, creation of authoring tools, and extensions such as translations to other specification languages (e.g. InTml [11]) or meta data languages (e.g. XMI [38]).

5 Conclusions and Future Work

We have presented a conceptual model and a specification language that can be used to describe components of 3D user interfaces in a more formal and extensive way than it is currently common practice. Relevant properties and characteristics of 3DICs have been identified and a conceptual model for 3DICs, including parameters, input and output events, devices and device mappings, states, and execution has been proposed. Based on the model the XML-based specification language ICDL has been developed that is human-readable and that allows for automatic processing and tools support. A detailed and extensive repository of generalized 3DICs may increase the accessibility of existing concepts and may support their selection and reuse. Several examples describing well-known interface components for immersive VR, Desktop VR and Augmented Reality show the applicability of the formal notation and the XML-based language. The model and the XML-based language ICDL provide the basics for a growing specification. Nevertheless, a number of issues remain open and need to be addressed in further research.

The current description of 3DICs concentrates on a restricted set of properties and characteristics. Certain aspects that might be important for the selection and the automatic processing of 3DICs, such as supported tasks or required user skills have not been addressed yet. In addition, further work needs to be done on the data types. The types used for devices, events and parameters need to be extended to support the consistent description of a wide range of 3DICs. Guidelines and standards for the pseudocode would also contribute to a more uniform and consistent description of 3DICs among several authors.

Advanced user interface concepts, such as adaptive 3D user interfaces in general or run-time-exchangeable interaction components [20] in particular require a formal and machine-readable notation of interface components. The specification language might need to be extended to support these developments.

To learn about the advantages and drawbacks of the current version of the conceptual model and the specification language and to improve them, they have to be used to describe further interface components. We are therefore aiming at a publicly available repository, where people can access and upload specifications of 3DICs. A first version of such a repository can be found online at [1]. These community efforts along with concluding refinement and extension of the proposed model and language might eventually lead to their standardization.

```
<Specific-3DIC ...>
  <Identification name="RingMenu">
    <Modification version="4.0" date="..." author="..."
    <Related3DIC id="LiangRingMenu"/>
    <ExtensionOf id="GenericMenu"/>
    <SuitableDomain>Desktop-VR</SuitableDomain>
    <SuitableDomain>Fishtank-VR</SuitableDomain>
    <TargetGroup>Anybody</TargetGroup>
    <MetaData name="bestNumber" type="range">3-15</MetaData>
    ...
  </Identification>
  <Origin>
    <bibtex:entry bibtex:id="Liang94">...</bibtex:entry>
  </Origin>
  <Documentation>
    <Description>A ring menu is composed of...</Description>
    <Picture>RingMenuPicture.jpg</Picture>
  </Documentation>
  <States>
    <State>OBJECT_SELECTED</State>
    <State>MENU_ROTATING</State>
  </States>
  <Parameters>
    <Parameter id="itemList" type="MFObject3D" doc="..."/>
    <Parameter id="firstSelected" type="Object3D" doc="..."/>
    <Parameter id="rotationSpeed" type="SFFloat"/>
    <Parameter id="itemRatio" type="SFFloat"/>
    ...
    <Parameter id="highlightGeometry" type="Object3D"/>
  </Parameters>
  <Events>
    <InputEvents>
      <Event id="directItemSelection" type="Object3D"/>
      <Event id="activateRotationLeft" type="SFBool"/>
      <Event id="activateRotationRight" type="SFBool"/>
    </InputEvents>
    <OutputEvents>
      <Event id="currentItemChanged" type="SFBool"/>
      <Event id="selectedItem" type="Object3D"/>
    </OutputEvents>
  </Events>
  <Devices>
    <InputDevices>
      <Device id="mouse" type="2DOF"/>
    </InputDevices>
    <InDeviceEventMappings>
      <EventMap doc="Menu rotated while mouse over...">
        <DeviceEvent>mouse.pos2D</DeviceEvent>
        <InterfaceEvent>ActivateRotationLeft</InterfaceEvent>
      </EventMap>
      ...
    </InDeviceEventMappings>
  </Devices>
  <Execution>
    <WebReference>...</WebReference>
    <PseudoCode>
      ...
      activateRotationRight( value, timestamp )
      {
        selectedItem = rotateRight();
        currentItemChanged = true;
      }
      ...
   </PseudoCode>
  </Execution>
</Specific-3DIC>
```

Listing 7: XML example of a specific 3DIC (Ring Menu)

REFERENCES

[1] Website with additional material for this paper (contains the ICDL XML Schema and several examples of 3DICs). http://www.3D-components.org/3DIC, 2005.

[2] Marc Abrams, Constantinos Phanouriou, Alan L. Batongbacal, Stephen M. Williams, and Jonathan E. Shuster. UTML: An appliance-independent XML user interface language. In *WWW '99: Proceeding of the eighth international conference on World Wide Web*, pages 1695–1708, New York, NY, USA, 1999. Elsevier North-Holland, Inc.

[3] Ronald Azuma, Yohan Baillot, Reinhold Behringer, Steven Feiner, Simon Julier, and Blair MacIntyre. Recent advances in augmented reality. *IEEE Comput. Graph. Appl.*, 21(6):34–47, 2001.

[4] Grady Booch, James Rumbaugh, and Ivar Jacobson. *The Unified Modeling Language User Guide*. Addison-Wesley, 1999.

[5] Doug Bowman, Ernst Kruijff, Jr. Joseph J. LaViola, and Ivan Poupyrev. *3D User Interfaces: Theory and Practice*. Addison Wesley, July 2004.

[6] Stuart K. Card, Jock D. Mackinlay, and George G. Robertson. The design space of input devices. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 117–124. ACM Press, 1990.

[7] Rikk Carey and Gavin Bell. *The Annotated Vrml 2.0 Reference Manual*. Addison-Wesley, 1997.

[8] D. Brookshire Conner, Scott S. Snibbe, Kenneth P. Herndon, Daniel C. Robbins, Robert C. Zeleznik, and Andries van Dam. Three-dimensional widgets. In *ACM Symposium on Interactive 3D Graphics*, pages 183–188, Cambridge, Massachusetts (USA), 1992. ACM Press, New York.

[9] Raimund Dachselt and Michael Hinz. Three-dimensional widgets revisited - towards future standardization. In *IEEE VR 2005 Workshop 'New Directions in 3D User interfaces'*. Shaker Verlag, 2005.

[10] Raimund Dachselt, Michael Hinz, and Klaus Meissner. Contigra: An XML-based architecture for component-oriented 3d applications. In *Seventh International Conference on 3D Web Technology*, pages 155–163, Tempe, Arizona (USA), 2002. ACM Press, New York.

[11] Pablo Figueroa, Mark Green, and H. James Hoover. InTml: a description language for vr applications. In *Web3D '02: Proceeding of the seventh international conference on 3D Web technology*, pages 53–58. ACM, ACM Press, 2002.

[12] George W. Fitzmaurice, Hiroshi Ishii, and William Buxton. Bricks: Laying the foundations for graspable user interfaces. In *CHI*, pages 442–449, 1995.

[13] James D. Foley and Victor L. Wallace. The art of natural man-machine conversation. In *Proceedings of the IEEE*, pages 417–426, 1974.

[14] Chris Hand. A survey of 3d interaction techniques. In *Eurographics*. Blackwell Publishers, 1997.

[15] ECMA International. Ecma – 262: Ecmascript language specification. http://www.ecma-international.org/publications/standards/Ecma-262.htm.

[16] G. Drew Kessler, Doug A. Bowman, and Larry F. Hodges. The simple virtual environment library: An extensible framework for building VE applications. *Presence: Teleoperators and Virtual Environments*, 9(2):187–208, 2000.

[17] Florian Ledermann and Dieter Schmalstieg. APRIL a high-level framework for creating augmented reality presentations. In *Proceedings of IEEE Virtual Reality Conference 2005 (VR'05)*, pages 187–194, 2005.

[18] Ulrich Leiner, Bernhard Preim, and Stephan Ressel. Development of 3d-widgets - Overview (only in German). In *Simulation und Animation*, pages 170–188, Magdeburg, 1997. SCS Europe, Erlangen.

[19] Jiandong Liang and Mark Green. JDCAD: A highly interactive 3d modeling system. *Computers and Graphics*, 18(4):499–506, 1994.

[20] Irma Lindt. Exchangeability of 3d interaction techniques. In *Proceedings of the IEEE Workshop on New Directions in 3D User Interfaces*, pages 93–94. Shaker Verlag, 2005.

[21] Blair MacIntyre, Maribeth Gandy, Steven Dow, and Jay David Bolter. Dart: a toolkit for rapid design exploration of augmented reality experiences. *ACM Trans. Graph.*, 24(3):932–932, 2005.

[22] Paul Milgram and Fumio Kishino. A taxonomy of mixed reality visual displays. *IEICE Transactions on Information Systems*, E77-D(12), December 1994.

[23] I. Poupyrev, S. Weghorst, M. Billinghurst, and T. Ichikawa. Egocentric object manipulation in virtual environments: Empirical evaluation of interaction techniques. In *Eurographics*, pages 41–52. Blackwell Publishers, 1998.

[24] Ivan Poupyrev, Mark Billinghurst, Suzanne Weghorst, and Tadao Ichikawa. The go-go interaction technique: non-linear mapping for direct manipulation in vr. In *Proceedings of the 9th annual ACM symposium on User interface software and technology*, pages 79–80. ACM, ACM Press, 1996.

[25] Ivan Poupyrev, Desney Tan, Mark Billinghurst, Hirokazu Kato, Holger Regenbrecht, and Nobuji Tetsutani. Developing a generic augmented-reality interface. *Computers and Graphics*, 35(3):44–50, 2002.

[26] Ivan Poupyrev, Desney S Tan, Mark Billinghurst, Hirokazu Kato, Holger Regenbrecht, and Nobuji Tetsutani. Tiles: A mixed reality authoring interface. In *INTERACT 2001 Conference on Human Computer Interaction*, Tokyo, Japan, 2001.

[27] Gerhard Reitmayr and Dieter Schmalstieg. An open software architecture for virtual reality interaction. In *Proceedings of the ACM symposium on Virtual reality software and technology*, pages 47–54. ACM Press, 2001.

[28] Christian Sandor and Thomas Reicher. CUIML: A language for the generation of multimodal human-computer interfaces. In *Proceedings of the European UIML conference*, 2001.

[29] Bastiaan Schönhage and Anton Eliëns. Dynamic and mobile vrml gadgets. In *VRML '99: Proceedings of the fourth symposium on Virtual reality modeling language*, pages 47–52, New York, NY, USA, 1999. ACM Press.

[30] Chris Shaw, Jiandong Liang, Mark Green, and Yunqi Sun. The decoupled simulation model for virtual reality systems. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 321–328. ACM Press, 1992.

[31] William R. Sherman and Alan Craig. *Understanding Virtual Reality : Interface, Application, and Design*. Morgan Kaufmann, 2003.

[32] Paul S. Strauss and Rikk Carey. An object-oriented 3d graphics toolkit. In *SIGGRAPH '92: Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, pages 341–349, New York, NY, USA, 1992. ACM Press.

[33] Sun Microsystems. Java 3D Home Page. http://java.sun.com/products/ java-media/3D/index.html, 1997.

[34] Shari Trewin, Gottfried Zimmermann, and Gregg Vanderheiden. Abstract user interface representations: how well do they support universal access? In *CUU '03: Proceedings of the 2003 conference on Universal usability*, pages 77–84, New York, NY, USA, 2003. ACM Press.

[35] Web3D Consortium. ISO/IEC FDIS 19777-1:2005. extensible 3D (X3D) language bindings part 1: ECMAScript. http://www.web3d.org/x3d/specifications/ISO-IEC-19777-1-X3DLanguageBindings-ECMAScript/.

[36] Web3D Consortium. Extensible 3D (X3D^{TM}) Graphics. Home Page. http://www.web3d.org/x3d.html, 2003.

[37] Online 3d widget classification. http://www.3d-components.org.

[38] XML metadata interchange (XMI) official page. http://www.omg.org/technology/documents/formal/xmi.htm, 2005.