A Generic Framework for Modeling User Interface Devices with UML

Zeljko Obrenovic

ABSTRACT

In this note we describe ideas about a generic modeling framework for specification of user interface devices. Our modeling framework does not define any specific input or output device but instead defines a generic approach for modeling such devices. It also enables description of a wider class of user interface devices than existing solutions. Proposed framework is based on the Unified Modeling Language (UML), a widely adopted standard, familiar to many practitioners. There is a good conceptual match between the object paradigm of UML and user interface devices concepts. With UML, we can make easier use of device models for ordinary computing engineers, so results of user interface device researches can have broader and more practical effects. By providing a standard means for representing multimodal interaction, we can seamlessly transfer UML models of user interface devices between design and specialized analysis tools. Using formal models it is also possible to develop tools for repurposing of existing user interfaces content to other platforms.

KEYWORDS: User-interface devices, Human-computer interaction, Model-driven development, UML

INTRODUCTION

After almost two decades of WIMP (windows, icons, menus, and pointer) paradigm dominance, with mice, keyboards and monitors as standard user interface devices, a bewildering variety of devices for human-computer interaction again exists on the market. Hundreds of device profiles are available for accessing online content and more are introduced everyday [1]. All of these devices, such as Internet-enabled cell phones, PDAs, desktop, laptop, and wearable PCs have quite different requirements and interaction capabilities. This variety significantly complicates development of content and user interfaces, especially in case of a multiplatform development. For example, handcrafting content for each device and its usage, as well as all of their combinations is just not manageable; among other problems, this approach is too expensive, takes too much time; and leads to multiple, inconsistent versions of the content. In order to make sense of this variety, it is necessary to create solutions that can provide a generic and unified view on various classes of user interface devices. In this note we present a generic framework for specification of user interface devices with the Unified Modeling Language (UML). Our modeling framework does not define any specific device - such as a mouse or keyboard - but instead defines a generic approach for modeling such devices. The model, therefore, focuses on the notion of an abstract device, which defines the common characteristics of user interface devices regardless of their specific manifestations. We used UML as it is a widely adopted standard that is familiar to many practitioners, widely taught in undergraduate courses, supported by many books, training courses, and tools from different vendors.

In next section we briefly discuss some of the existing solutions. Then, we describe the proposed modeling framework, where we present the metamodel of user interface devices, and introduce UML modeling extensions. After that, we illustrate our approach on several examples of widely used user interface devices. In the end, we give short discussion and conclusions.

EXISTING SOLUTIONS

In last two decades computer science researchers have proposed and implemented various solutions for creating frameworks for description of user interface devices. Most of those solutions focus on input devices. Other approaches, such as those from multimedia field, tried to separate content form devices, allowing generic description of content. Both of these areas have influenced our work.

Card, Mackinlay and Robertson proposed a framework for characterization of computer input devices [2]. They did the systematization of input devices through morphological design space analysis, in which different input devices designs, are taken as a points in a parametrically described abstract design space. Authors defined a primitive movement vocabulary, as well as a set of composition operators for combining primitives from the vocabulary. This input device framework is well suited for user-manipulated event based interactive tools such as mice, trackballs, tablets and light pens, but it does not easily handle temporal and linguistic structures such as speech

[3]. Moreover, the authors used non-standard graphical notation for the description of input devices, which makes it less practical.

Allanson proposed the new sensor interaction model that is more suitable for streaming-based input devices such as physiological electrodes or microphones [4]. In this model, a stream of raw data (such as electroencephalogram) from the sensing hardware passes through up to two levels of signal preprocessing before it is either passed to an application or is presented directly to the user. The first layer is the standard device layer and is mandatory. The second command layer is optional, but it enables integration of complex signal processing components such as neural networks or pattern recognition modules. The main purpose of the proposed framework is to alleviate combining computing with physiological sensing technologies, but the model has a potential to be used in a wider range of applications. The main weaknesses of the model are that it is a high-level approach, and primary dedicated to electrophysiological devices.

Multimedia researchers have attempted to separate knowledge content from devices and media. To this end, Ashwin Ram and his colleagues used the Procedural Markup Language (PML) [5]. PML lets developers specify knowledge structures, the underlying physical media, and the relationships between them using cognitive media roles. Edward Posnak, Greg Lavender, and Harrick Vin proposed a framework that simplifies multimedia software component development by facilitating reuse of code, design patterns, and domain expertise [6]. Their solution lets components dynamically adapt presentation quality to the available resources and devices in heterogeneous environments

USER INTERFACE DEVICES MODELING FRAMEWORK

Trends and industry standards in software engineering, such as model driven development, open new possibilities for improving analysis, design, and implementation of interactive systems. In this note we propose a generic framework for specification of user interface devices with Unified Modeling Language (UML). UML (see www.omg.org/uml) is a good choice for modeling interactive devices for several reasons. It is a widely adopted standard that is familiar to many practitioners, widely taught in undergraduate courses, supported by many books, and training courses. In addition, many tools from different vendors support UML. Our approach is tightly coupled and inspired by the model-driven development, where software development's primary focus and products are models rather than computer programs. In this way, it is possible to use concepts that are much less bound to the underlying implementation technology and are much closer to the problem domain [7]. As user interface devices can be viewed as objects with complex structure, there is a good conceptual match between the object paradigm of UML and user interface devices concepts.

Our modeling framework does not define any specific device - such as a mouse or a keyboard - but instead defines a generic approach for modeling such devices. The model, therefore, focuses on the notion of an abstract device, which defines the common characteristics of user interface devices regardless of their specific manifestations. In order to create this generic framework we have explored two problems:

- Formal definition of user interface device concepts, and
- Definition of UML extension for modeling user interface devices.

To define the concept of device, we have created the metamodel of user interface devices. This metamodel represents an abstract, higher-level, view on various aspects of device specification. Based on this metamodel, we introduced UML extensions, and used them for modeling various input and output devices.

The Metamodel of User Interface Devices

In order to define device models, we need a vocabulary of modeling primitives. Therefore, we defined a metamodel where we formally described basic concepts of user interface (UI) devices. Figure 1 shows simplified definition of a UI device, from our metamodel, based on the composite software designed pattern. According to our model, a UI device can be input, output, or complex. A complex UI device integrates other devices to create simultaneous use of them, while a simple input and output devices carry out more elementary actions.

Input devices transfer human output, such as hand movement or speech, into a form suitable for computer processing. We classified input devices into event-based and streaming-based classes. Event-based input devices produce discrete events in reaction to user actions. For example, user input via a keyboard or a mouse, represents event-based input style. Streaming-based devices sample input signals, with some resolution and frequency, producing a time-stamped array of sampled values. For example, a computer detects a user's voice or psychological signals by sampling input signals with sensors such as a microphone or an electrode.

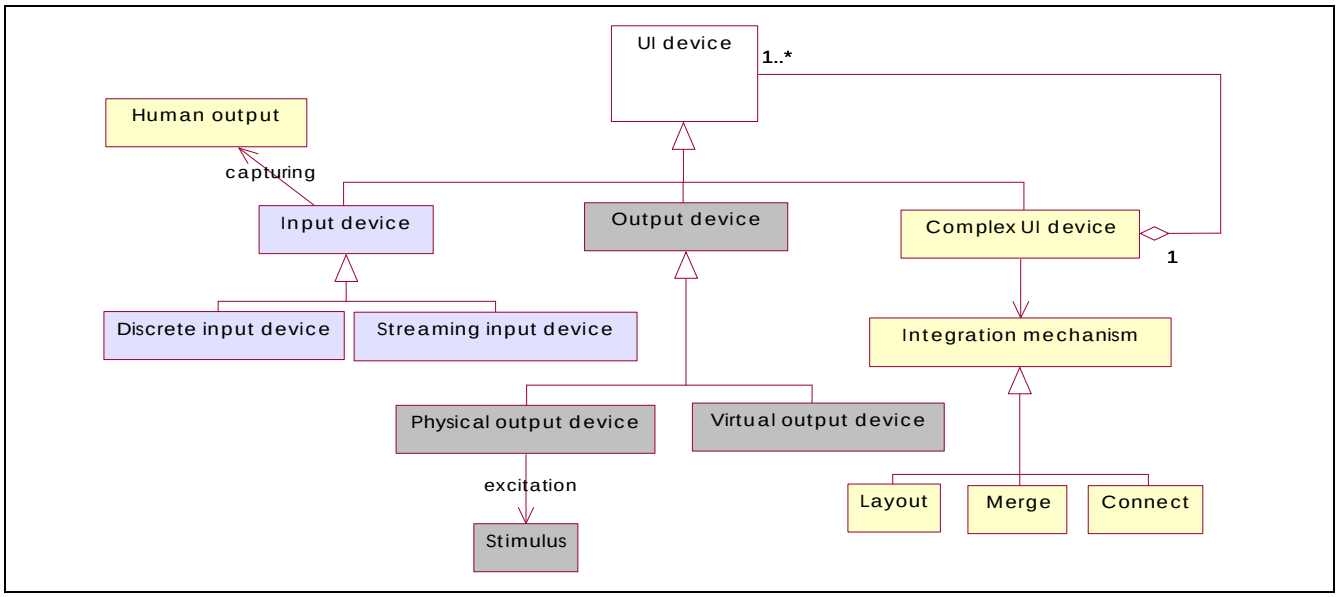

Figure 1. Simplified metamodel of UI devices.

Output devices present data to a user. We introduced two classes of output device: physical output device and virtual output device. A physical output device is ordinary device such as a monitor or speaker, and it is characterized by type of stimulus it produce (light, sound, movement, pressure, scent). A virtual output device creates an illusion of more devices or bigger devices than devices actually used. Examples of virtual devices include virtual displays, which create illusions of desktop size monitors using devices that are only few centimeters big [8], and phantom speakers, which, based on how a brain interprets the ears' input, led to virtual surround, in which two speakers sound like many more [9].

A complex device integrates many input or output devices using some of integration mechanisms. Some complex devices can be composed of only input or output devices only, but some combine input and output elements. For example, touch-screen device combines visual output with haptic input. We defined three basic integration mechanisms, based on framework proposed by Card et al [2]: layout, merge, and connect. With layout integration mechanism, complex devices are spatially grouped on a common panel, for example, keys on a keyboard, or mouse buttons. In this case, integration includes only physical layout, while an application can treat each devices independently. Merge layout mechanism joins two or more devices, to produce a new one in which resulting output domain of a device merges output domains of constituting devices. For example, mouse merges two orthogonal movement sensors to produce 2D sensor. In this case mouse always sends integrated X and Y values, so outputs of two 1D sensors cannot be viewed as two independent channels. Connect mechanism connect output domain of one device to input domain of another. For example, a mouse is connected to a mouse cursor.

UML Extensions

Although our metamodel's general nature makes it independent of a specific modeling language, for our purposes we wanted to apply it to UML. UML is a general-purpose modeling language, but it includes built-in facilities that allow customizations—or profiles—for a particular domain. A profile fully conforms to the semantics of general UML but specifies additional constraints on selected general concepts to capture domain-specific forms and abstractions. To address this purpose, a formal extension mechanism was defined to allow practitioners to extend the semantics of the UML. The mechanism allows us to define stereotypes, tagged values and constraints that can be applied to model elements. A *stereotype* is an adornment that allows us to define a new semantic meaning for a modeling element. *Tagged values* are key value pairs that can be associated with a modeling element that allow us to "tag" any value onto a modeling element. *Constraints* are rules that define the well-formedness of a model. They can be expressed as free-form text or with the more formal Object Constraint Language (OCL).

We defined a new UML profile where we have introduced several UML extensions based on the proposed metamodel. With these extensions, we can describe a device at different levels of abstraction, with various levels of details. Table 1 shows some of introduced UML class and association stereotypes. As we do not describe concrete devices, but some class of devices, such as mouse or monitors, we propose modeling of UI devices with class diagrams, In next section, we will demonstrate usage of these stereotypes in class diagrams.

Table 1. *UML stereotypes for modeling of user interface devices.*

Package stereotype	UI device	*All classes that describe some device may be grouped into a package with this stereotype.*
Class stereotypes	discrete input device	*Discrete input device such as a mouse or keyboard.*
	streaming input device	*Streaming input device such as a microphone or EEG electrode.*
	physical output device	*Physical output device such as speaker.*
	virtual output device	*Virtual output device such as virtual display.*
	complex device	*A device that integrates more other devices. For example, a monitor integrates pixels, while a mouse integrates a movement sensor with mouse buttons.*
	layout panel	*A panel used for layout of devices that complex device integrates.*
	human output	*Human output captured by some input device, such as movement or speech.*
	stimulus	*Stimulus produced by output device, such as light or sound.*
	data structure	*A data structure used for description of the device.*
	device state	*A state maintained by device. This is important for devices such as a mouse which detects only relative changes.*
Attribute stereotypes	device property	*An attribute that describes some characteristics of a device. For example, a monitor has width and height dimensions, white mouse may be defined by size, resolution and C:D ratio.*
Association stereotypes	merge	*Connects a complex device with other devices it integrates by merging.*
	layout	*Connects a complex device with other devices it integrates by layout on common panel.*
	connect	*Connects a complex device with other devices it integrates by connecting.*
	panel	*Connects a complex device with a panel used for layout of devices it integrates.*
	media	*Connects a physical output device with a stimulus it produces.*
	capturing	*Connects input device with human output it captures.*
	data	*Connects a device with a data structure that describes it.*
	state maintenance	*Connects a device with a state it maintains.*

MODELING USER INTERFACE DEVICES

Our modeling framework defines a generic approach for modeling user interface devices, where actual devices are described with UML models defined with extensions explained in the previous section. To illustrate some of the possibilities of the proposed modeling framework, we have applied it to several examples.

Figure 1a shows a model of a raster screen device, such as monitor, developed using proposed UML stereotypes. Basic element of a raster screen is a pixel, which can be viewed as a complex device that *merges* three pixel parts (red, green, and blue). Each pixel part is a physical output device which produces light of some frequency, and with variable intensity. Each pixel part is described with a pixel part data, which simply describes intensity of light that the pixel part produces. A pixel is defined with its shape and size. A pixel data is described as an aggregation of three such pixel part data structures. A raster screen is a complex device that layouts pixels in rows and columns. A data structure for whole raster screen is simply an array of pixel data structures.

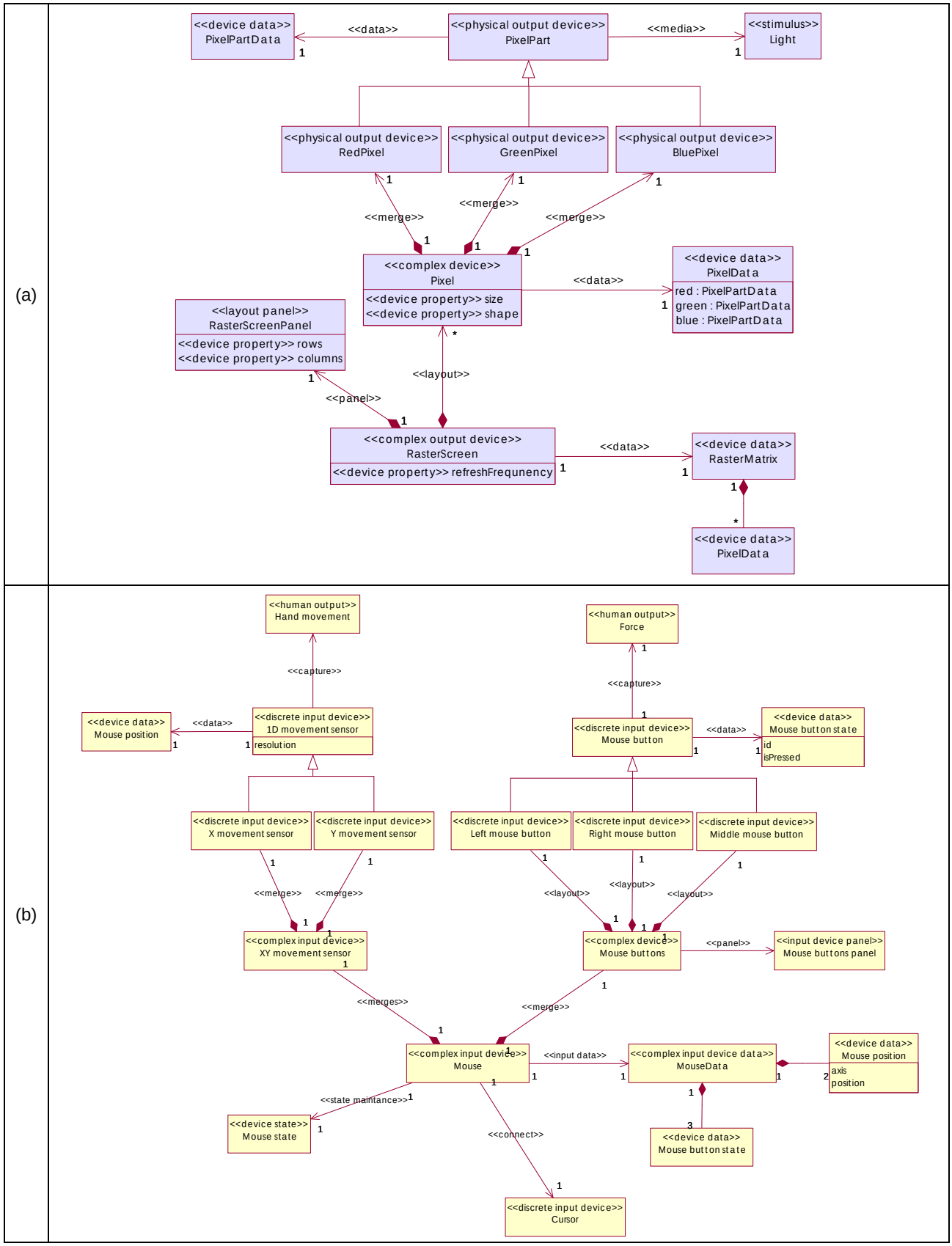

Figure 1. Describing a raster screen (a) and a mouse (b) on a class diagram using proposed UML stereotypes.

Figure 1b shows a UML description of a mouse. A mouse merges two complex devices: a XY movement sensor and mouse buttons. The XY movement sensor merges two orthogonally placed 1D movement sensors, capturing hand movement. Each 1D movement sensor is a discrete input device characterized with its resolution. A mouse buttons panel layouts, usually three, mouse button devices, which capture human force output. A mouse is connected to a cursor, modeled as a virtual output device.

The model of a keyboard is shown of picture 2. A keyboard is a complex device that layouts keys on common panel. Each key is modeled as a discrete input device that detects pressure. Each key is described by its code and current state. Keyboard data represent a union of currently used keys. A keyboard is additionally described by a keyboard state, which, for example, keeps num lock, caps lock, and scroll lock keys state. There are many types of keyboards, according to the keys layout, such as QWERTY or numerical keyboards.

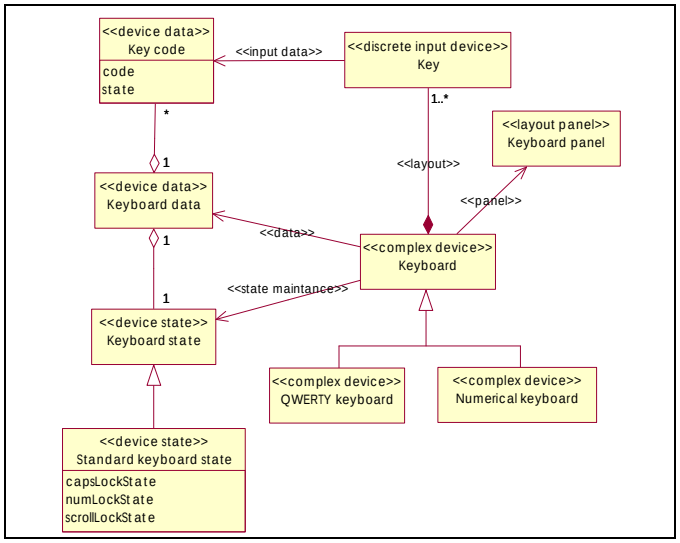

Figure 2. A model of keyboards.

Figure 3 shows a model of a 5.1 sound system. This system merges sound produced by five ordinary speakers and one subwoofer. A normal speaker is a complex device that merges sound of woofer, which produces sounds of lower frequencies, and tweeter, which produces higher frequencies.

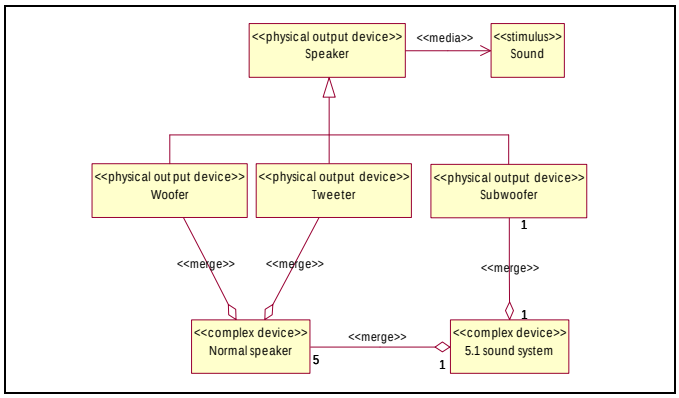

Figure 3. A model of a 5.1 sound system.

Figure 4 presents the model of a virtual display [8]. A virtual display simulates a raster screen device of some width and height. A virtual device is produced by miniature imager, a complex device that layouts LED pixels. Minature imager is usually only few centimeters wide and high.

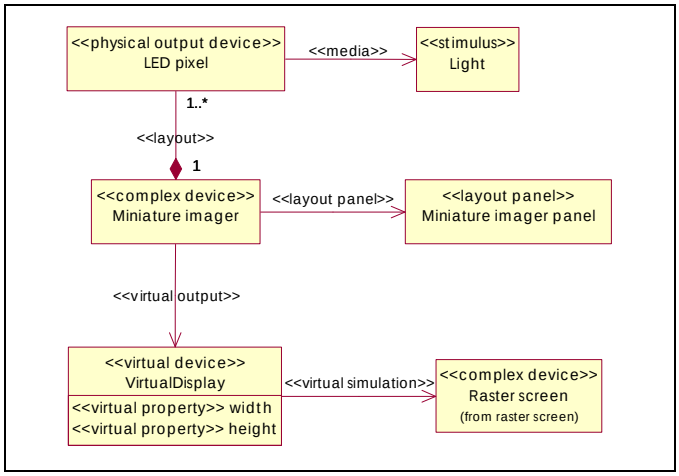

Figure 4. A model of a virtual display.

At the end, in figure 5 presents a model of an EEG input device. A basic element of this system is an EEG electrode, a streaming input device which captures human EEG signals. EEG electrodes can be connected to some on-board processing elements, such as FFT processor. More complex processing using neural processing is also often part of these systems [10].

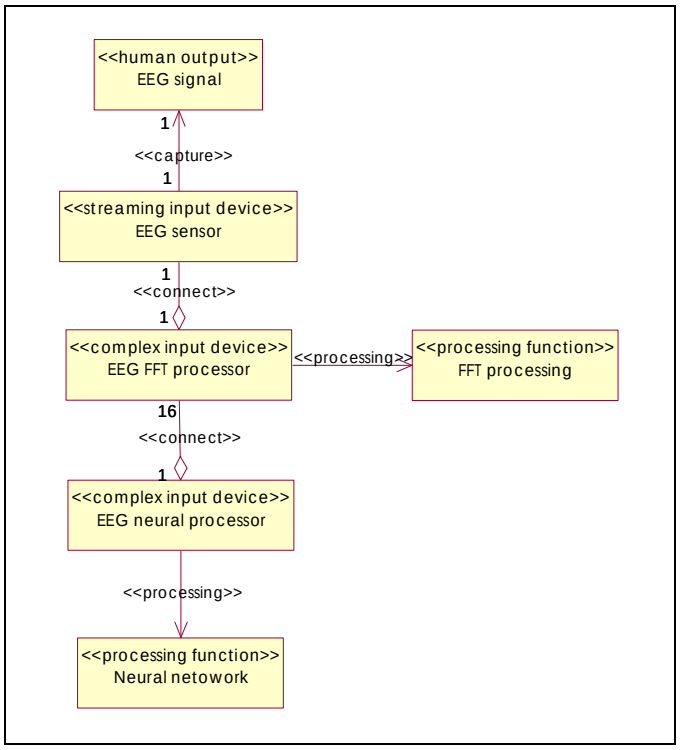

Figure 5. A model of an EEG device.

DISCUSSION: MODEL TRANSFORMATIONS AND ANALYSIS

Based on UML models, we can get a formal XML description of various aspects of a user interface devices. These descriptions can be transformed in any other form, using various tools, such as XSLT translators. UML supports both graphical and textual notations, as well as compilers for generating code for multiple platforms. In this way it is possible to create an architecture which can enhance communication between users of different notations for UML concepts by providing a standard, centralized store for these concepts. In this way, it facilitates the construction of formal model compilers, test-case generators, and consistency checkers. The UML provides basis for generating serialized formats, such as the XML Metadata Interchange, with file based interchange in XML Schema and application programming interfaces such as the Java Metadata Interface, which provides dynamic access to UML model storage from Java [11].

For example, based on class diagram descriptions of device, we can create and object diagram where we can describe concrete values of device properties. These class and object diagrams can then be transformed into XML form, and be used by tools that can analyse or transform content aimed at presentation on some device. In this way, it is possible to analyse models, do reverse engineering, or repurpose existing content. For example, we used this approach to create a model-driven framework for multimedia content repurposing [12].

Device properties such as resolution, size, refresh frequency or color depth, can be used as parameter for repurposing content among devices. When repurposing, it is often necessary to change the original presentation dimensions. For instance, when transforming a Windows bitmap into a wireless bitmap (WBMP) it is necessary to map 24-bit color space into 1-bit black and white color space of the WBMP. In addition, the transformation has to shrink the picture in order to fit it into the smaller wireless device presentation space. In this case we scale the original dimensions to fit the range of target dimensions of the same type. Alternatively, it is possible to change one presentation dimension with the dimension of another type. For example, if the presentation space is very small, then some big picture can be transformed into an animated or user-navigated picture. In this case, the space dimension is replaced with the time dimension. The Multimedia Metamodel can aid this process by providing formal models of the content's presentation dimensions and the presentation possibilities of target devices, as illustrated in the previous section, where we described the presentation device package.

CONCLUSIONS

In this note we have proposed a unique framework for modeling multimodal human-computer interaction. Our modeling framework does not define any specific interaction modality but instead defines a generic approach for modeling such modalities. In this way, it enables description of broader classes of user interface devices than existing solutions. Although our framework's general nature makes it independent of a specific modeling language, we applied it to UML. UML is a good choice for modeling multimodal systems for several reasons. It is a widely adopted standard that is familiar to many software practitioners, widely taught in undergraduate courses, and supported by many books, training courses, and tools from different vendors. And what is the most important there is a good conceptual match between the object paradigm of UML and user interface device concepts, what we have demonstrated on several examples. Our framework can be easily extended with additional elements using UML extension mechanisms.

By providing a standard means for representing multimodal interaction, we can seamlessly transfer UML models of user interface devices between design and specialized analysis tools. Standardization provides a significant driving force for further progress because it codifies best practices, enables and encourages reuse, and facilitates interworking between complementary tools. With UML, we can jump on the bandwagon of new software development technologies, such as model driven development. Our modeling framework feats neatly in the model driven development approach, and consequently, it will be able to make use of the tools that support it.

Proposed solutions can serve several purposes. The

metamodel of multimodal interaction can provide the context of multimodal concepts where we could perceive many relations that are not always obvious. Definition of semantic extensions of UML can be used for formal description of user interface devices on various levels of abstraction. These descriptions can be used as a meta-description of these devices, but by using automation, it is possible to create tools for analysis and transformations of content created for these devices. Using formal models it is also possible to develop tools for repurposing of existing user interfaces to other platforms.

In our future work, we plan to extend existing software development processes, such sa Rational Unified Process with primitives for better description of interactive systems, including used devices, and to integrate our solutions into existing CASE tools.

REFERENCES

1. Gurminder S., "Content Repurposing", IEEE Multimedia, Vol. 11, No. 1, January-March 2004, pp. 20-21.

2. Card S.K., Mackinlay J.D, Robertson G.G., "The Design Space of Input Devices", in M.M. Blattner and R.B. Dannenberg (Eds.), *Multimedia Interface Design*, ACM Press and Addison Wesley, Reading, Mass., 1992, pp. 217-232.

3. Blattner M.M., Glinter E.P., "Multimodal Integration", IEEE Multimedia, Winter 1996, pp. 14-24.

4. Allanson J., "Electrophysiologically Interactive Computer Systems", Computer, March 2002, pp. 60-65.

5. Ram A. et al., “PML: Adding Flexibility to Multimedia Presentations”, IEEE Multimedia, IEEE CS Press, April-June 1999, pp. 40-52.

6. Posnak E.J., Lavender R.G., and Vin H.M., “An Adaptive Framework for Developing Multimedia Software Components”, Comm. of the ACM, ACM Press, Vol. 40, No. 10, October 1997, pp. 43-47.

7. Selic B., "The Pragmatics of Model-Driven Development", IEEE Software, Vol. 20, No. 5, September / October 2003, pp. 19-25.

8. Edwards J., "New Interfaces: Making Computers More Accessible", IEEE Computer, December 1997, pp. 12-14;

9. Kraemer A., "Two Speakers Are Better Than 5.1", IEEE Spectrum, IEEE Press, May 2001, pp. 71-74.

10. Millán J.R., "Adaptive Brain Interfaces", Comm. of the ACM, Vol. 46, No. 3, March 2003, pp. 75-80.

11. Sendall S. and Kozaczynski W., "Model Transformation; The Heart and Soul of Model-Driven Software Development", IEEE Software, Vol. 20, No. 5, September / October 2003, pp. 42-45.

12. Obrenovic Z., Starcevic D., Selic B., "A Model Driven Approach to Content Repurposing", IEEE Multimedia, Vol. 11, No. 1, January-March 2004, pp. 62-71.