# A Theoretical Survey of User Interface Description Languages: Preliminary Results

*Josefina Guerrero-García*
*Juan Manuel González-Calleros*
*Jean Vanderdonckt*
Université catholique de Louvain
Place des Doyens, 1 – B-1348
Louvain-la-Neuve, Belgium {josefina.guerrero, juan.m.gonzalez, jean.vanderdonckt}@uclouvain.be

*Jaime Muñoz-Arteaga*
Sistemas de Información
Universidad Autónoma de Aguascalientes
Av. Universidad No. 940, Col. Bosques, 20100
Aguascalientes, Aguascalientes (México)
jmunozar@correo.uaa.mx
CENIDET, Internado Palmira S/N, Col. Palmira, C.P. 62490, Cuernavaca, Morelos. México

***Abstract*— A user interface description language (UIDL) consists of a specification language that describes various aspects of a user interface under development. A comparative review of some selected user interface description languages is produced in order to analyze how they support the various stages of user interface development life cycle and development goals, such as support for multi-platform, device-independence, modality independence, and content delivery. There has been a long history and tradition to attempt capturing the essence of user interfaces at various levels of abstraction for different purposes, including those of development. The recent return of this effort today gains more attraction, along with the dissemination of XML markup languages, and gives birth to many proposals for various user interface description languages. Consequently, an in-depth analysis of the salient features that make these languages different from each other is desired in order to identify when and where they are appropriate for a specific purpose. The review is conducted based on a systematic analysis grid and some user interfaces implemented with these languages.**

***Keywords-component;* User interfaces, User Interface Description Language, Extensible Markup Language, User Interface extensible Markup Language.**

## I. INTRODUCTION

For many years, Human-Computer Interaction (HCI) witnessed a perennial race for the ultimate User Interface (UI) Description Language that would ideally capture the essence of what a UI could be or should be. A UI Description Language (UIDL) consists of a high-level computer language for describing characteristics of interest of a UI with respect to the rest of an interactive application in order to be used during some stages of the UI development life cycle. Such a language involves defining a syntax (i.e. how these characteristics can be expressed in terms of the language) and semantics (i.e., what do these characteristics mean in the real world). It can be considered as a common way to specify a UI independently of any target language (e.g., programming or markup) that would serve to implement this UI.

The issue of UIDL was first raised when it was required to develop a UI like a module of an interactive application rather than merely a series of lines codes. In a second time, the issue was reinforced when the desire appears to model a UI by a set of specifications so as to communicate these specifications and to share them across stakeholders, or to (semi-)automatically generate the code of the UI, as desired in model-based approaches for developing UIs. When a UI was required to be run simultaneously on different computing platforms, this need took shape in some language that would be exchanged from one platform to another without any changes to avoid any extraneous effort.

For some years, the race progressively slept. The wide availability of markup languages and the capability of introducing any language based on XML meta-language, along with the multiplicity of today's available platforms (e.g., mobile phone, smart phone, pocket PC, handheld PC, Tiqit PC, tablet PC, laptop, traditional PC, and even wall screens) have awaken this race and have exacerbated it to a point where today more than a dozen of UIDLs exist that focus on some of the desired characteristics. To shed light on this proliferation of UIDLs, we conducted a systematic comparison based on an analysis grid and UIs. The paper focuses only on XML-based languages, because XML is a well established standard that is easily extensible and that could work with yet-to-be-invented appliances without many changes. Furthermore, it is declarative and can be use by non-programmers or occasional users.

For the purpose of this survey, we gathered and analyzed as much literature as possible on each UIDL. Then, depending on available tools, we systematically developed a multi-platform or multi-context UI for a simple dictionary so as to identify the capabilities of the UIDL and the ability of this UIDL to be supported by editing, critiquing, analysis tools, and, of course, tools for producing executable UIs, both by compilation/execution and by interpretation.

The remainder of this paper is structured as follows: Section 2 provides an overview of existing UIDLs, including some of the UIDLs that have not been considered for the comparison for different reasons (e.g., their accessibility). Section 3 respectively describes each selected UIDL in the comparison and identifies the main goals pursued by each one. Section 4 defines the comparison

978-0-7695-3856-3/09 $26.00 © 2009 IEEE
DOI 10.1109/LA-WEB.2009.40

criteria used in the comparison analysis and provides the final analysis grid. Section 5 presents the conclusion.

## II. Some User Interface Description Languages

In this section, relevant contributions of XML-compliant languages for the definition of UIs are analyzed, based on the available literature and tools. These languages were not considered for a detailed comparison for one or more of the following characteristics:

- *Specificity*. The specificity of a language differentiates from general purpose or generic languages from very specific ones. For instance, XISL [11] is a very interesting approach but is just specific to multi-modal user interfaces.
- *Accessibility*. The accessibility of the language refers to the available information to analyze it. When the information is totally confidential is impossible to have access to the semantics of the language, this is the case with most of the software vendors (Microsoft, IBM) languages.
- *Relatedness*. The relatedness characteristic is use to differentiate whether the language is a User Interface Description language (UIDL).
- *Standard*. This property indicates if the language has been object of a standardization process. Even that these language are very important and relevant in some cases they have one of the previous listed properties. Consequently, they were not considered for the comparison.

*eXtensible Interaction Scenario Language* (XISL) [11] holds an interest for our work as it is the only Web-based language that is supported by a tool enabling the development of multi-modal UIs based on interaction scenarios between the user and the system manipulating compliant devices: (i.e., PCs, mobile phones, PDAs).

The *eXtensible mark-up language for MultiModal interaction with Virtual Reality worlds* (XMMVR) [18] is used to specify multi modal (Voice and graphical interaction) with virtual reality presentation.

*X3D* is an open recommendation from Web3d (www.web3d.org) for 3D content delivery. Surprisingly VRML and X3D are not a programming libraries in the strict sense of their definition.

The *Device Independent Authoring Language* (DIAL) [29] is a markup language for the filtering and presentation of Web page content available across different delivery contexts. The delivery context is a set of attributes that characterizes the capabilities of the access mechanism, the preferences of the user, and other aspects of the context into which a web page is to be delivered. In particular, the capabilities of the device (including the modalities and representations) it supports, the characteristics of the network over which delivery occurs, and the preferences of the user will all potentially affect the user experience provided [27]. The delivery context [30] is composed of the following characteristics: Device, Network, User preferences, Dynamic characteristics, and Context.

The *Extensible MultiModal Annotation Markup Language* (EMMA) [32] is used to contain and annotate information automatically extracted from the input of users which manipulate multi-modal UIs. The language is capable to convey meaning for different types of single input, i.e., text, speech, handwriting and combinations of any previous modalities. These combinations are compliant with the W3C Interaction Framework, which includes, among other, the languages: InkML [28] an XML data format for representing digital ink data that is input with an electronic pen or stylus as part of a multimodal system; VoiceXML [26] for web development and content delivery to voice applications.

*XForms* [31] separates the presentation from the data, keeping the principle of separation of concepts, allowing reuse and device independence. XForms is not a free-standing document type, but is intended to be integrated into other markup languages, such as XHTML or SVG [31]. XForms, while designed to be integrated into XHTML, is no longer restricted only to be a part of that language, but may be integrated into any suitable markup language.

Software Vendors, not included in this review, also provide solution for commercial tools, some examples are:

- MXML (Adobe) [1] is used to describe UI layout and behaviors, and Action Script for the Flex Framework.
- Open Laszlo (Laszlo) [14] is a XML-based language for rich Internet applications.
- SISL (Lucent Technologies) [15] is a XML-based language service logic that is shared across many different user interfaces, including speech-based natural language interfaces.
- XAML (Microsoft) [17] is a markup language for declarative application programming for the Windows Presentation Foundation.
- XUL (Mozilla) [33] is used to build feature-rich cross platform applications that can run connected or disconnected from the Internet.

## III. A Review of XML-Compliant User Interface Description Languages

In the previous section we described some existing UIDL that were not considered for this survey; in this section we present an overview of UIDLs that have been considered for different reasons: they are available for testing, they have been used in some development cases, they are widely used.

*Dialog and Interface Specification Language* (DISL) [22] is a user interface markup language (UIML) subset that extends the language in order to enable generic and modality independent dialog descriptions. Modifications to UIML mainly concerned the description of generic widgets and improvements to the behavioral aspects. Generic widgets are introduced in order to separate the presentation from the structure and behavior, i.e., mainly to separate user- and

device-specific properties and modalities from a modality-independent presentation. The use of generic widget attribute enables to assign each widget to a particular type of functionality it ensures (e.g., command, variable field, text field, etc.). Further, a DISL rendering engine can use this information to create interface components appropriated to the interaction modality (i.e., graphical, vocal) in which the widget will operate. The global DISL structure consists of an optional head element for Meta information and a collection of templates and interfaces from which one interface is considered to be active at one time. Interfaces are used to describe the dialog structure, style, and behavior, whereas templates only describe structure and style in order to be reusable by other dialog components.

The *Generalized Interface Markup Language* (GIML) is used for the generalized Interface Toolkit (GITK) [13]. GIML is used in this context as an interface descriptor. Following the OMG principles of separation of concerns GIML splits functionality and presentation. While the functionality is preserved in GIML the UI is derived from XSL files, which come from user and system profiles. This information is merged with the functional descriptions by using XSLT to form a final interface description. The profile data could come directly from a file−system or from a remote profile server. GIML avoids the use of concepts such as "push-button", "scrollbar", whereas GIML uses terms such as "action", "data-entry/value- choice/single/limited". The goal is to use interface patterns in the future. These media neutral identifiers are the foundation for an interface object hierarchy.

*Interface Specification Meta-Language* (ISML) [5] was developed with the intention that methaphors (shared concepts between the user and the computer) be made explicit in design. ISML de-couples that metaphor model from any particular implementation, and express mappings between the concepts shared between the user and the system. It provides a framework that supports mappings between both user-oriented models (such a task descriptions) and software architecture concerns (interactor definitions). The ISML framework composites these concepts within five layers (devices, components, meta-objects, metaphor, interactors), using a variety of mappings to link them together.

*Renderer-Independent Markup Language* (RIML) [6] is a markup language based on W3C standards that allows document authoring in a device independent fashion. RIML is based on standards such as: XHMTL 2.0 and XFORMS. Special row and column structures are used in RIML to specify content adaptation. Their semantics is enhanced to cover pagination and layout directives in case pagination needs to be done. Due to the use of XForms, RIML is device independent and can be mapped into a XHTML specification according to the target device. RIML semantics is enhanced to cover pagination and layout directives in case pagination needs to be done, in this sense it was possible to specify how to display a sequence of elements of the UI.

*Software Engineering for Embedded Systems using a Component-Oriented Approach* (SeescoaXML) [16] consists of a suite of models and a mechanism to automatically produce different final UIs at runtime for different computing platforms, possibly equipped with different input/output devices offering various modalities (e.g. a joystick). This system is context-sensitive as it is expressed first in a modality-independent way, and then connected to a specialization for each specific platform. The context-sensitivity of the UI is here focusing on computing platforms variations. An abstract UI is maintained that contains specifications for the different rendering mechanisms (presentation aspects) and their related behavior (dialog aspects). These specifications are written in a XML-compliant UIDL that is then transformed into platform-specific specifications using XSLT transformations. These specifications are then connected to a high-level description of input/output devices. The entry point of this forward engineering approach is therefore located at the level of Abstract UIs.

*Simple Unified Natural Markup Language* (SunML) [20] is an XML language to specify concrete user interfaces that can be mapped to different devices (PC, PDA, voice). The innovation of this language is the capacity to specify dynamically components. In SunML it is also possible to encapsulate the style and the content of each widget independent of the others. Two different files are used for that purpose. Another interesting feature offered in SunML is widget composition. Some operators have been defined for that purpose: union (semantically-common widgets), intersection, subtraction, substitution, inclusion. Widgets Merging Language (WML) is the extension used for that purpose. SunML presents a reduced set of elements that seems to be not enough, but the composition of widgets is used to specify more complex widgets.

*TeresaXML* [19] is a UIDL for producing multiple final UIs for multiple computing platforms at design time. They suggest starting with the task model of the system, then identifying the abstract UI specifications in terms of its static structure (the presentation model) and dynamic behavior (the dialog model): such abstract specifications are exploited to drive the implementation. This time, the translation from one context of use to another is operated at the highest level: task and concepts. This allows maximal flexibility, to later support multiple variations of the task depending on constraints imposed by the context of use. Here again, the context of use is limited to computing platforms only. The whole process is defined for design time and not for run-time. For instance, there is no embarked model that will be used during the execution of the interactive system, contrarily to the SEESCOA approach [16]. At the AUI level, the tool provides designers with some assistance in refining the specifications for the different computing platforms considered. The AUI is described in terms of interactors that are in turn transformed into Concrete Interaction Objects (CIOs) once a specific target has been selected.

*MariaXML* (http://ftp.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-407/paper15.pdf) is the successor of TeresaXML in order to support dynamic behaviors, events, rich internet applications, multi-target user interfaces, in particular those based on web services. In this way, it is possible to have a UI specified in MariaXML attached to a

web service. MariaXML is also compatible with the Cameleon Reference Framework [4].

*User Interface Markup Language* (UIML) [2] is an XML-based language that provides: (1) a device-independent method to describe a UI, (2) a modality-independent method to specify a UI. UIML allows describing the appearance, the interaction and the connection of the UI with the application logic. The following concepts underlie UIML:

1. UIML is a meta-language: UIML defines a small set of tags (e.g., used to describe a part of a UI) that are modality-independent, target platform-independent (e.g., PC, phone) and target language-independent (e.g., Java, VoiceXML). The specification of a UI is done through a toolkit vocabulary that specifies a set of classes of parts and properties of the classes. Different groups of people can define different vocabularies: one group might define a vocabulary whose classes have a 1-to-1 correspondence to UI widgets in a particular language (e.g., Java Swing API), whereas another group might define a vocabulary whose classes match abstractions used by a UI designer
2. UIML separates the elements of a UI and identifies: (a) which parts are composing the UI and the presentation style, (b) the content of each part (e.g., text, sounds, images) and binding of content to external resources, (c) the behavior of parts expressed as a set of rules with conditions and actions and (d) the definition of the vocabulary of part classes.
3. UIML groups logically the UI in a tree of UI parts that changes over the lifetime of the interface. During the lifetime of a UI the initial tree of parts may dynamically change shape by adding or deleting parts. UIML provides elements to describe the initial tree structure and to dynamically modify the structure.
4. UIML allows UI parts and part-trees to be packaged in templates: these templates may then be reused in various interface designs.

*USer Interface eXtensible Markup Language* (UsiXML) [25] is structured according to different levels of abstraction defined by the Cameleon reference framework [4]. The framework represents a reference for classifying UIs supporting a target platform and a context of use, and enables to structure the development life cycle into four levels of abstraction: task and concepts, abstract UI (AUI), concrete UI (CUI) and final UI (FUI). Thus, the Task and Concepts level is computational-independent, the AUI level is modality-independent (In the cockpit it can be several physical, Vocal, GUI, Tactile) and the CUI level is toolkit-independent. UsiXML relies on a transformational approach that progressively moves among levels to the FUI. The transformational methodology of UsiXML allows the modification of the development sub-steps, thus ensuring various alternatives for the existing sub-steps to be explored and/or expanded with new sub-steps. UsiXML has a unique underlying abstract formalism represented under the form of a graph-based syntax.

*Web Service eXperience Language* (WSXL) [3] [11] is designed to represent data, presentation and control. WSXL relies on existing standards; in particular, XML based standards such as XPath, XML Events, DOM, XForms and XLink as well as Web Services standards such as SOAP, WSDL and WSFL. WSXL includes an extensible Adaptation Description Language where explicit locations of adaptation points, the permissible operations on adaptation points (e.g. insert, delete, modify), and the constraints on the contents of adaptation (e.g. via an XML Schema) can be specified. The Adaptation Description Language can be used during a post-processing step where the output of a WSXL component can be adapted independently without invoking the component. Finally, A WSXL collection provides an execution and management environment for WSXL components. It calls the lifecycle operations on WSXL components it instantiates, and implements a set of interfaces and a processing model for use by WSXL components and objects external to the collection. An object implementing the WSXL collection interface need not be a WSXL component. The developer can create new and more abstract UI components.

The *eXtensible user-Interface Markup Language* (XICL) [9] is an easy way to develop User Interface Components to Browser-based software. New UI components are created from HTML components and others XICL components. The XICL description is translated into DHTML code. An XICL documents is composed by a UI description composed by HTML or XICL elements and several components (Structure, Properties, Events and Methods. XICL is a language to UI development by specifying its structure and behavior in an abstract level than using only DHTML. It also promotes reuse and extensibility of user interface components.

The *eXtensible Interface Markup Language* (XIML) [7] [8], is a language developed by Redwhale Software, derived from XML and able to store the models developed in MIMIC [21]. MIMIC is a meta-language that structures and organizes interface models. It divides the interface into model components: user-task, presentation, domain, dialog, user, and design models. The design model contains all the mappings between elements belonging to the other models. The XIML is thus the updated XML version of this previous language. The XIML language is mainly composed of four types of components: models, elements, attributes, and relations between the elements. The presentation model is composed of several embedded elements, which correspond to the widgets of the UI, and attributes of these elements representing their characteristics (color, size…). The relations at the presentation level are mainly the links between labels and the widgets that these labels describe. XIML supports design, operation, organization, and evaluation functions; it is able to relate the abstract and concrete data elements of an interface; and it enables

knowledge-based systems to exploit the captured data.

## IV. USER INTERFACE DESCRIPTION LANGUAGES COMPARISON

Many UILDs have been introduced in the literature and are widely used in practice. With this comes need to understand their scopes and their differences. The purpose of this section is to make a general comparison of the cited languages in section III. The protocol selected was first used on a previous review on UIDLs [23], this works updates and extends our previous survey considering the latest results (Table 1) along the following dimensions:

- *Specificity* indicates if the UIDL could be used in one or multi platforms or devices.
- *Publicly available*: depending on the availability of the language deep analysis can be done. This category was used to discard many languages that lack on documentation or that is confidential. The possible values are: 0 = no information available, 1 = not available, 2 = poorly available, 3 = moderately available, 4 = completely available and 5 = completely available with meta-models.
- *Type criterion* informs whether the UIDL is a research or industry work.
- *Weight of the organization* behind denotes the organization to which the UIDL belongs. Efforts from Universities are significant, particularly, those where more than one university has adopted the use of the UIDL. Those UIDL coming from the industry might have more impact and this is reflected in its level of usage.
- *Level of usage*: depending on the usage of the language we create the following categories: 0 = unknown, 1 = one person, 2 = two or more persons, 3 = one organization, 4 = two or more organizations and 5 = wide usage.

Due to its number of concepts, UsiXML has been intentionally removed from Table 2 and it is used to illustrate the comparison protocol (Figure 1). On the left a series of developments steps compliant with the Cameleon reference framework [4], to the right the supported concepts and the transformations applied to UsiXML. Details on this comparison can be found in the model based incubator group [33] where this work has been reported. Table 2 compares the properties of the different UIDLs according the eight criteria:

- *Component models*: this criterion gives the aspects of the UI that can be specified in the description of the UIs. The *task model* is a description of the task to be accomplished by the user; the *domain model* is a description of the objects the user manipulates, accesses, or visualizes through the UIs; the *presentation model* contains the static representation of the UI, and the *dialog model* holds the conversational aspect of the UI.
- *Methodology*: different approaches to specify and model UIs exist: 1) Specification of a UI description for each of the different contexts of use. As a starting point, a UI specification for the context of use considered as representative of most case, the one valid for the context of use considered as the least constrained or finally the one valid for the context of use considered as the most comprehensive is specified. From this starting UI specification, corrective or factoring out decorations [23], (e.g., to add, remove, or modify any UI description) are applied so that UI specifications can be derived for the different contexts of use. 2) Specification of a generic (or abstract) UI description valid for all the different contexts of use. This generic UI description is then refined to meet the requirements of the different contexts of use.
- *Tools*: some of the languages are supported by a tool that helps designer and renders the specification to a specific language and/or platform.
- *Supported languages*: specify the programming languages to which the XML-based language can be translated.
- *Supported platforms*: specify the computing platform on which the language can be rendered by execution, interpretation or both.
- *Abstraction level*: each UIDL may exhibit the capability to express a runnable UI (instance level), one or many models involved in the development of this UI (model level), how these models are built (meta-model level), and what are the fundamental concepts on which this operation is based (meta-meta-model level).
- *Amount of tags*: to reach the above level of abstraction, each UIDL manipulates a certain amount of tags, which is also highly depending on the coverage of the concepts.
- *Coverage of concepts*: depending on the level of abstraction, each UIDL may introduce some specific vs. generic concepts (e.g., a given presentation model vs. any model, each custom-defined), their properties (e.g., to what extent can a concrete presentation be specified), and their relations.

## V. CONCLUSION

Six years from now, a first review of UIDLs was conducted [23]. That work were reviewed and updated accordingly to the progress of those UIDLs, while some works have continue, there were works with not reported update since then. In addition, to that review, new UIDLs that have been reported in the literature and are commercially available were added to this review. For space reason we did not include the complete set of UIDLs but selected those that seems more robust accordingly to the parameters that we evaluate.

There is a plethora of user interface description languages that are widely used, with different goals and different strengths. On one hand we have software vendors UIDLs and, on the other hand, there are free license UIDLs to use; also some of them can support just one platform and others

are multiplatform. Some of them (as WSXL or SunML) need a few tags while others (as UsiXML) have a significant amount. Also, some of them are the result of a research project, while some other born in an industry. Considering all those characteristics it might seems hard to pick one from the list. We believe that this choice is more dictated by the goals of the project and the particular needs, even the budget available should be considered as commercial UIDLs are not available for free.

The goal of this work is aimed to help authors to decide what UIDL to use for their projects. We hope this analysis helps in understanding and comparing the components of different UIDLs in a systematic way –their strengths, limitations, and appropriateness for use. There is currently such a large number of UIDLs available that choosing among them can be time consuming and difficult to do, this comparison can assist UI designers in choosing a language suited to their purposes.

## ACKNOWLEDGMENT

We gratefully acknowledge the support of the CONACYT program (www.conacyt.mx) supported by the Mexican government, the Human European project (Model-based Analysis of Human Errors during Aircraft Cockpit System Design, project funded by FP7-AAT-2007-RTD-1/CP-FP-211988 from European Commission), the ITEA2 Call 3 UsiXML project under reference 20080026, and the PROMEP net Project under Contract UAA-CA-48.

## REFERENCES

[1] Adobe (2009), Flex overview, Adobe Systems Incorporated, Available online: http://www.adobe.com/products/flex/overview/

[2] Abrams, M., Phanouriou, C., Batongbacal, A.L., Williams, S. & Shuster, J. (1999), UIML: An Appliance-Independent XML User Interface Language. In A. Mendelzon, editor, *Proceedings of 8th International World-Wide Web Conference WWW'8* (Toronto, May 11-14, 1999), Amsterdam, 1999. Elsevier Science Publishers.

[3] Arsanjani, A., Chamberlain, D. and et al. (2002), (WSXL) web service experience language version, 2002. Retrieved from: http://www-106.ibm.com/developerworks/library/ws-wsxl2/.

[4] Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L., Vanderdonckt, J.: A Unifying Reference Framework for Multi-Target User Interfaces. Interacting with Computers, Vol. 15, No. 3 (June 2003) 289–308.

[5] Crowle, S., Hole, L., ISML: An Interface Specification Meta-Language. DSV-IS 2003, Funchal, Madeira Island, Portugal. June 11-13, 2003. Lecture Notes I Computer Science 2844 Springer 2003, ISBN 3-540-20159-9.

[6] Demler, G., Wasmund, M., Grassel, G., Spriestersbach, A. & Ziegert, T. (2003), Flexible pagination and layouting for device independent authoring, *WWW2003 Emerging Applications for Wireless and Mobile access Workshop*.

[7] Eisenstein J., Vanderdonckt J., Puerta A. (2000), Adapting to Mobile Contexts with User-Interface Modeling, *Proceedings of 3rd IEEE Workshop on Mobile Computing Systems and Applications WMCSA'2000* (Monterey, 7-8 December 2000), IEEE Press, Los Alamitos, 2000, pp. 83-92.

[8] Eisenstein J., Vanderdonckt J., Puerta A. (2001), Model-Based User-Interface Development Techniques for Mobile Computing, *Proceedings of 5th ACM Int. Conf. on Intelligent User Interfaces IUI'2001* (Santa Fe, 14-17 January 2001), Lester, J. (Ed.), ACM Press, New York, 2001, pp. 69-76.

[9] Gomes de Sousa, L. & Leite, J. C. (2003), XICL: a language for the user's interfaces development and its components. *Proceedings of the Latin American conference on Human-computer interaction (Rio de Janeiro, Brazil,* August 17 - 20, 2003)*,* ACM Press pages, New York, pp. 191-200.

[10] Helms, J., Schaefer, R., Luyten, K., Vermeulen, J., Abrams, M., Coyette, A., Vanderdonckt, J., Human-Centered Engineering with the User Interface Markup Language, in Seffah, A., Vander-donckt, J., Desmarais, M. (eds.), "Human-Centered Software Engineering", Chapter 7, HCI Series, Springer, London, 2009, pp. 141-173.

[11] IBM (2002), WSXL specification, April 2002, retrieved on Janury 2nd 2009.

[12] Katsurada, K., Nakamura, Y., Yamada, H., Nitta, T. (2003), XISL :A Language for Describing Multimodal Interaction Scenarios, *Proceedings of the 5th International Conference on Multimodal Interfaces ICMI'03* (Vancouver, Canada.

[13] Kost, S. (2004), Dynamically generated multi-modal application interfaces. Ph.D. Thesis, *Technical University of Dresden and Leipzig University of Applied Sciences*, Germany.

[14] Laszlo Systems Inc. (2008), OpenLaszlo Application Developer's Guide, Available online: Available online: http://www.openlaszlo.org/lps4.2/docs/developers/architecture.html

[15] Lucent (2000), Sisl: Several Interfaces, Single Logic, Lucent Technologies, Available online: http://www.bell-labs.com/user/lalita/sisl-external.html

[16] Luyten, K., Abrams, M., Vanderdonckt, J. & Limbourg, Q. (2004) Developing User Interfaces with XML: Advances on User Interface Description Languages, *Sattelite workshop of Advanced Visual Interfaces 2004*, Gallipoli, Italy.

[17] Microsoft (2009), XAML, *Microsoft Corporation*, Available online: http://msdn.microsoft.com/en-us/library/ms747122.aspx

[18] Olmedo,H., Escudero,D., & Cardenoso,V (2008).: A Framework for the Development of Applications Allowing Multimodal Interaction with Virtual Reality Worlds. *Communications Proceedings 16th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision'2008 WSCG'2008* (Plzen - Bory, Czech Republic, February 4-7), University of West Bohemia Press, pp. 79-86.

[19] Paternò, F. & Santoro, C. (2003), A Unified Method for Designing Interactive Systems Adaptable to Mobile and Stationary Platforms, *Interacting with Computers*, Elsevier, 15, pp. 349-366.

[20] Picard, E., Fierstone, J., Pinna-Dery, A-M. & M. Riveill (2003). Atelier de composition d'ihm et évaluation du modèle de composants. *Livrable l3, RNTL ASPECT*, Laboratoire I3S, mai.

[21] Puerta A.R. (1996), The Mecano Project: Comprehensive and Integrated Support for Model-Based Interface Development, *Proceedings of 2nd Int. Workshop on Computer-Aided Design of User Interfaces CADUI'96* (Namur, 5-7 June 1996), Presses Universitaires de Namur, 1996, pp. 19-35.

[22] Schaefer, R., Steffen, B., Wolfgang, M., Task Models and Diagrams for User Interface Design, *Proceedings of 5th International Workshop, TAMODIA'2006* (Hasselt, Belgium, October 2006), Lecture Notes in Computer Science, Vol. 4385, Springer Verlag Berlin, 2006, pp. 39-53.

[23] Souchon, N., Vanderdonckt, J., A Review of XML-Compliant User Interface Description Languages, Proc. of 10th Int. Conf. on Design, Specification, and Verification of Interactive Systems DSV-IS'2003 (Madeira, 4-6 June 2003), Jorge, J., Nunes, N.J., Falcao e Cunha, J. (Eds.), Lecture Notes in Computer Science, Vol. 2844, Springer-Verlag, Berlin, 2003, pp. 377-391.

[24] Thevenin, D. (2001), Adaptation En Interaction Homme-Machine : Le Cas de la Plascticité. PhD thesis, *Université Joseph Fourier*, 21 December 2001.

[25] Vanderdonckt, J. A MDA-Compliant Environment for De-veloping User Interfaces of Information Systems. In Proc. of 17th Conf. on Advanced Information Systems Engineering CAiSE'05 (Porto, 13-17 June 2005), LNCS, Vol. 3520, Springer-Verlag, Berlin, 2005, pp. 16-

31.

[26] W3C (2004), Voice Extensible Markup Language (VoiceXML) Version 2.0, W3C recommendation, 16 March 2004, *W3C Consortium*. Available online: http://www.w3.org/TR/voicexml20.

[27] W3C (2003), Device Independence Principles, W3C Working Group Note, 01 September 2003. *W3C Consortium*. Available online: http://www.w3.org/TR/2003/NOTE-di-princ-20030901/

[28] W3C (2006), W3C InkML: Digital Ink Markup Language, W3C recommendation, 24 October 2006, *W3C consortium*. Available online: http://www.w3.org/2002/mmi/ink

[29] W3C (2007a), Dial: Device Independent Authoring Language, W3C Working Draft, *W3C consortium*. Available online: http://www.w3.org/TR/dial/

[30] W3C (2007b), Content Selection Primer 1.0, W3C Working Draft, *W3C consortium*. Available online: http://www.w3.org/TR/cselection-primer/

[31] W3C (2007c), XForms 1.0 (Third Edition), W3C recommendation, 29 October 2007, *W3C consortium*. Available online: http://www.w3.org/TR/2007/REC-xforms-20071029/

[32] W3C (2008), EMMA: Extensible MultiModal Annotation markup language, W3C Proposed Recommendation, *W3C consortium*. Available online: http://www.w3.org/TR/emma.

[33] W3C Model-based User Interfaces Incubator Group http://www.w3.org/2005/Incubator/model-based-ui/wiki/Task_Meta_ModelsXML User Interface Language (XUL) 1.0.

[34] http://www.mozilla.org/projects/xul/xul.htm

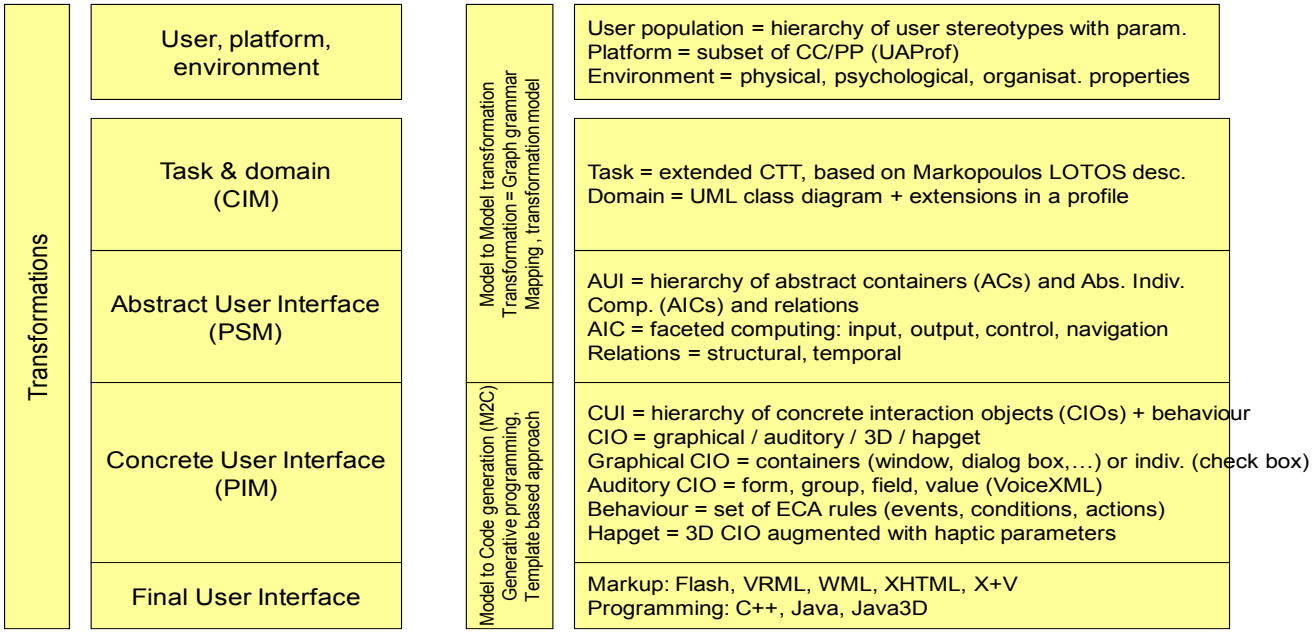


**Figure 1 Comparison protocol exemplified with UsiXML.**

**Table 1 General features of UIDLs**

| UIL | Specificity | Publicly available | **Type** | Weight of the organization behind | Level of usage |
|---|---|---|---|---|---|
| DISL | Multimodal UIs for mobile devices | 2 | Research | Paderborn University | 3 |
| GIML | Multimodal | 3 | Research | Technical University of Dresden and Leipzig University of Applied Sciences | 2 |
| ISML | GUI, multiplatform, multidevice | 2 | Research | Bournemouth University | 1 |
| RIML | Mobile devices | 0 | Industry | Industry: SAP Research, IBM Germany, and Nokia Research Center along with CURE, UbiCall, and Fuijitsu Invia | 3 |
| SeescoaXML | Multiplatform, multidevice, dynamic generation UI | 2 | Research | Expertise Centre for Digital Media Limburgs Universitair Centrum | 3 |
| SunML | Multiplatform | 4 | Research | Rainbow team, Nice University | 3 |
| TeresaXML | Multiplatform, multidevice, | 4 | Research | HCI Group of ISTI-C.N.R. | 3 |
| UIML | Multiplatform | 4 | Industry | Harmonia, Virginia Tech Corporate Research (OASIS) | 3 |
| UsiXML | Multiplatform | 5 | Research | UCL | 3 |
| WSXL | multiplatform, multidevice | 4 | Industry | IBM | 3 |
| XICL | Multiplatform | 3 | Research | Federal University of Rio Grande do Norte, Brazil | 3 |
| XIML | multiplatform, multidevice | 4 | Research | Redwhale Software | 3 |

**Table 2 Properties Comparison of UIDLs**

| UIL | Models | Methodology | Tools | Supported languages | Supported platforms | Level | Tags | Concepts |
|---|---|---|---|---|---|---|---|---|
| DISL | Presentation, dialog and control | Specification of a generic, platform-independent multimodal UI | Rendering engine | VoiceXML, Java MIDP, Java Swing, Visual C++ | Mobile and limited devices | Model level | Not specified | Head element, interface classes (structure, style, behavior), state, generic widgets |
| GIML | Presentation, dialog, and domain | Specification of a generic interface description. | GITK (Generalized Interface Toolkit) | C++, Java, Perl | Not specified | Meta-model | 15 tags | Interface, dialog, widget, objects |
| ISML | Presentation, task, dialog, domain | Specification of a generic UI description | Under construction | Java, Microsoft foundation class, Java swing classes | Desktop PC, 3D screen | Model level | Not specified | Mappings and constrains, action events, meta-objects, display parts, controller parts, interaction definition |
| RIML | There is no information | Specification of a generic UI description | There is no information | XHTML, XFORMS, XEvents, WML | Smart phone, pda, Mobile, Desktop Pc | Model level | There is no information | Dialog, Adaptation, layout, element |
| Seesco aXML | Task, Presentation, dialog | Specification of a generic UI description | CCOM (BetaVersion 1.0 2002) PacoSuite MSC Editor | Java AWT, Swing, HTML, java.microedition, applet, VoxML, WML Juggler | Mobile, desktop PC, Palm III | Model level | Not specified | Component, port, connector, contract, participant, blueprint, instance, scenario, pltform, user, device |
| SunML | Presentation, dialog, domain | Specification of a generic UI description | SunML Compiler | Java Swing, voiceXML, HTML, UIML, | Desktop Pc, pda | Model level | 14 tags | Element, list, link, dialog, interface, generic events, synchronization |
| Teresa XML | Presentation, task, dialog | Specification of a generic UI description | CTTE Tool for task Models Teresa | Markup: Digital TV, VoiceXML, XHTML/SVG, X+V Programming: C# | DigitalTV, Mobile, Desktop PC, | Model level | 19 tags | Mappings, models, , platform, task, input, output |
| UIML | Presentation, dialog, domain | Specification of a generic UI description | UIML.net, VoiceXML renderer, WML renderer, VB2UMIL | HTML, Java, C++, VoiceXML, QT, CORBA, and WML | desktop PC, a handheld device, tv, mobile | Model level | 50 tags | interconnection of the user interface to business logic, services |
| WSXL | Presentation,dialog, domain | Specification of a generic UI description | Not specified | HTML | PC, Mobile phone, | Model level | 12 tags | CUI=XForms, WSDL, Mapping=XLang Workflow=WSFL, Logic=XML event |
| XICL | Presentation,dialog, | Specification of a generic UI description | XICL STUDIO | HTML, ECMAScript, CSS e DOM. | desktop PC | Model level | Not specified | Component, structure, script, events, properties, interface |
| XIML | Presentation, task, dialog, domain | Specification of a generic UI description | XIML Schema | HTML, java swing, WLM | Mobile, desktop PC, PDA | Model level | 32 tags | Mappings, models, sub models, elements, attributes and relations between the elements |