Bridging User Context and Design Models to Build Adaptive User Interfaces*

Mladjan Jovanovic[1], Dusan Starcevic[2], and Zoran Jovanovic[2]

[1] DISI, University of Trento, Italy
[2] School of Electrical Engineering, University of Belgrade, Serbia
mladjan.jovanovic@unitn.it, starcev@fon.rs, zoran@rcub.bg.ac.rs

Abstract. With respect to modeling the context of interaction, two different research communities consider the context from different viewpoints. The user-centered view which prevails in the HCI and the device-centered view which is dominant in the mobile and ubiquitous computing. Despite existing advances, context modeling and user interface (UI) design methods are still poorly integrated, making it difficult to use the contextual elements directly in UI design. This paper focuses on bringing user-related aspects of the interaction context in UI design. We propose a model-driven framework for the development of adaptive user interfaces. The framework describes the interaction context by integrating contextual factors from different context perspectives in a unison manner. Then it provides formal semantic relations between contextual and UI elements. The framework has been used in the data visualization domain, particularly in the design of the software instrument table for UAV (Unmanned Aerial Vehicle) that takes into account user context, namely human perceptual abilities.

Keywords: user interface design, user interface models, user context, user abilities, model-driven engineering.

1 Introduction

Trend to build UIs that look and behave according to the context of interaction is emerging [1]. This requires appropriate methods and techniques for formal integration of interaction context elements into UIs development and adaptation processes accordingly. While there are numerous efforts to integrate contextual data into UI design, they still suffer from a semantic gap between contextual and UI elements.

In our research, we aim to provide seamless formal integration of interaction context information into the UI design. Our work is based on a phenomenological view of the context [2] which advocates that humans create and modify the context during interaction. Likewise, we believe that humans perceive context as a feature of

* Research described in this paper is derived from the first author's previous work with the University of Belgrade, Serbia. We acknowledge EU FP7-PEOPLE 607062 ESSENCE Project.

S. Sauer et al. (Eds.): HCSE 2014, LNCS 8742, pp. 36–56, 2014.

interaction, rather than of objects or other people. This is important because the description of interaction context between humans and computers requires a clear definition of context. Based on existing developments, we propose a unified generic framework for designing UIs that provides formal specification of the interaction context in a way that it can be integrated in UI design. The framework includes an ontological and an architectural foundation that structure the development of adaptive UIs in a uniform way. In particular, our research has been concerned with:

- The design of the user-centered interaction model which integrates knowledge from a different context perspectives into a single unified view;
- Integration of the proposed model with the framework for development of adaptive UIs;
- Application of the proposed framework in adaptive UI design.

Next section briefly describes the previous research work devoted to user-related aspects of interaction context in the adaptive UI design. From that work we propose a generic approach to articulating and combining the interaction context and UI models in the space of Model Driven Engineering (MDE). The approach is fully supported by mainstream MDE technologies. Finally, the case study illustrates the approach considering human perception in designing visual UIs.

2 User Context Considerations in UI Design

If we consider interaction context, the UI design process typically becomes more complex since the number of situations (contexts) in which the system will operate increases. Early efforts to use interaction context in adaptive UI design can be found in work by Thevenin and Coutaz [3]. They have introduced the notion of UI plasticity to describe the ability of UI to adapt, or to be adapted, to the context of use while preserving usability. This brought Calvary et al. [4] to propose the theoretical framework for development of adaptive UIs, namely the CAMELEON reference framework. The aim of the framework is to combine all possible situation-optimized designs into a single, uniform design.

Based on seminal work of Calvary et al. [4], researchers and developers have introduced many implementations of the reference framework [5, 6, 7]. These approaches are mostly declarative and model-based relying on a number of models in describing different aspects of a UI. In the last decade, they have evolved in parallel with the aim of coping with the different challenges raised by the design of adaptive UIs in continuously changing technological settings and usage scenarios [5]. This initiated a large body of UI design methods producing productive models that can be automatically processed by computers. A number of different languages and tools was proposed, each focusing in specific aspect(s) of UI design [8]. Some of them are task-based, such as CTTE [9, 10] and HAMSTERS [11], while others work with multiple kinds of UI models such as UsiXML [7], MARIA [12] and CAP [13]. There are also approaches addressing development of interactive systems for a specific domain. Well-known example is the Petshop tool, based on a Petri net formalism, for designing UIs for safety-critical systems [14].

Analyzing the existing work from a usability perspective, we can see that the adaptation they perform is mainly device-oriented. If we consider user context, we can notice that the user is mainly modeled with logical activities it performs during interaction with the systems, i.e. various task notations [9, 10, 11, 15]. Existing approaches lack in providing formal specification of other user features (such as various abilities and skills) in a way that can be directly used in UI adaptation either in design-time or run-time or fits into UI development process. There are some attempts to describe and integrate user abilities [16], however, they are brought down to optimization problems.

With respect to the formal semantic connections between contextual and UI elements, existing approaches mostly aim at the utilization of context information built into UI design models to support the generation of user interfaces at runtime. The interconnection between contextual and UI elements is implicitly hidden within the UI model description and the interpretation is not made explicit.

3 Connecting Interaction Context and UI Elements

The proposed approach focuses on design; its current implementation does not support the automatic detection or recognition of contextual data, but rather complements to them. In this regard, we focus on:

- Formal specification of interaction context elements and
- Their integration in designing adaptive UIs.

To cope with the heterogeneity of interaction context elements and the lack of a specification to uniformly access and integrate these elements, we introduce a user-centered context interaction model. Using the model, we derive a generic framework for adaptive UIs. We propose a front-end architecture that complements with back-end architectures that may focus on the context acquisition and abstraction process.

In this section, we outline the technological foundation for building the framework. Thereafter we describe the unified interaction model and the generic UI development framework, respectively.

3.1 Technological Ground

In following paragraphs, we give a brief overview of software technologies we have used to design the models and the framework.

Modeling Language

In describing modeling framework, we extend standard UML with elements specific for adaptive UIs. The UML is mainly being used to communicate about the design of a software system. This makes the framework accessible to a wide range of software engineers from aspects of development tools and processes. The fact that all information is expressed in UML makes it easier to integrate the UI specification with the specification of the application functional core. Although there are UML extensions

for UI modeling [17, 18, 19], they consider UI design process partially. UMLi [17] provides extensions for GUI design, while CUP [18] proposes UML-based task modeling notation using activities and states. Wisdom [19] is a UML proflile that describes high-level UI models at early stages of design.

UML Profile defines extensions to a reference UML metamodel with the purpose of adapting the metamodel to a specific platform or domain. The primary construct is the Stereotype, which is defined as part of a Profile. Stereotypes can be used to add constraints and properties (tagged values) to model elements. The profile provides UML constructs that describe particular domain. We have used the profile mechanism to create domain specific languages (DSL) for designing adaptive user interfaces.

Transformation Language

Model transformations represent the central operation for handling models in MDE. In our approach, we have opted for ATLAS Transformation Language [20] as the technology for UI model transformations based on the following arguments: an open-source software, the large user community, a solid developer support, a rich knowledge base of model transformation examples and projects, and a mature tool support. In addition, the technology provides dedicated support for UML model transformations. We have used ATL to design transformations between UI models on different abstraction levels that can be parameterized with various contextual factors contained in the interaction model.

3.2 The User-Centered Context Interaction Model

With respect to context representation, we adopted the canonical view of context that includes three components – the users, the devices and the environment [4]. However, the view contains elements at too high abstraction level to include in UI optimization at either design time or runtime. For this reason, we introduced the interaction model that serves as binding component and provides primitives that bridge semantic gap between UI elements and contextual elements (Figure 1). The interaction context enables us to describe context elements relevant to a particular user's interactions.

The generic interaction model describes common characteristics of contextual factors regardless of their specific manifestations.

Human model combines human abilities and preferences. Human abilities include various sensory, perceptual, cognitive and motor abilities. In describing human abilities we rely on the model that structures levels of human information processing. At the very basic level is sensation - a combination of biochemical and neurological events triggered by the stimulus of sensory organs. These sensory stimuli lead up to the level of perception where the signals are organized, analyzed and interpreted. Cognition is the eventual accretion of perceptual interpretations and past experiences leading up to intelligence and human actions. Preferences present subjective human characteristics that are tied to a particular domain, for example, preference to use specific communication channels (such as visual, voice or multimodal).

Devices are described with physical and logical properties, both connected to various stimuli they detect or produce, such as sound, light or pressure. Stimuli

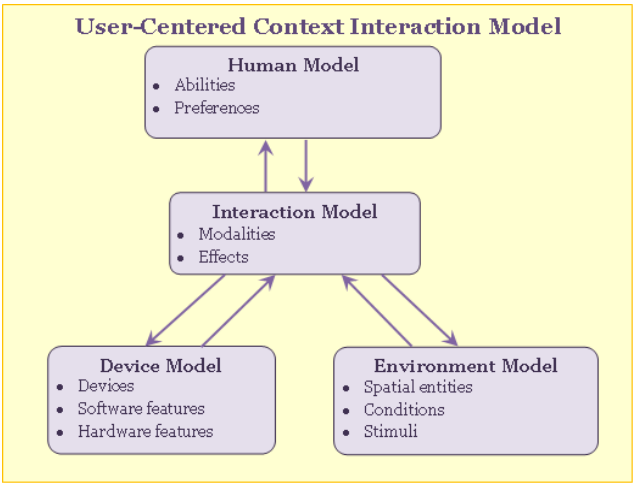

Fig. 1. High-level view on unified user-centered interaction model

are defined as physical signals in the environment with sensible properties, such as amplitude or frequency. In addition, environment model assumes various kinds of spaces in which interaction occurs (for example, room, vehicle or public space), as well as the existence of physical conditions that may influence the interaction (for example, sound and light conditions).

Interaction model is described using existing generic framework for modeling multimodal HCI [21]. Multimodal UIs combine natural human input modalities — such as speech, pen, touch, hand gestures, eye gaze, and head and body movements — in a coordinated manner with multimedia system output. In describing multimodal HCI, we consider UIs through modalities they employ. Modality is seen as a form of interaction designed to engage a number of human abilities. In this sense, it produces effects on humans or processes effects produced by humans. The idea is to describe UIs in terms of modalities that produce effects. Modalities can be simple or complex. Complex modality integrates other modalities to create simultaneous use of them, whereas simple modality presents a primitive form of interaction. Simple modalities can be input or output. An input modality transfers human output into a form suitable for device processing, while an output modality presents data to the user. Effects are abstract concepts and are closely related to human abilities. They are organized in five main categories: sensory, perceptual, linguistic, motor, and cognitive effects. Sensory effects describe processing of stimuli performed by human sensory apparatus. Perceptual effects are more complex effects that the human perceptual system obtains by analyzing data received from sensors, such as shape recognition, grouping, or highlighting. Motor effects describe human mechanical actions, such as hand movement. Linguistic effects are associated with human speech, listening, reading, and writing. Cognitive effects appear at a higher level of human information processing, such as memory processes or attention. More elaborate description of the UML multimodal HCI framework can be found elsewhere [21].

Following subsections describe the unified interaction model. In describing the model we focus on interaction model and user modeling.

Common Ground – Interaction Model

Interaction model is described using multimodal HCI design issues. In this way, we consider UIs through modalities they employ. Modality is seen as a form of interaction designed to engage a number of user capabilities. In other words, it produces effects on users or processes effects produced by users.

The basic idea behind the proposal is to model the UIs with modalities they use and associated effects. Next step is to connect these models to the user, device and environment descriptions [22]. The connections between contextual factors and UI elements are created on modality-specific (CUI) level. While we describe UIs with multimodal concepts, we introduce the term of contextual factor to describe concepts related to interaction semantics (Figure 2). UI element uses interaction modalities to engage specific human abilities. As the result of these engagements, various effects on users are generated. On the other side, human, environment and device entities specify contextual factors that act as filters for produced effects.

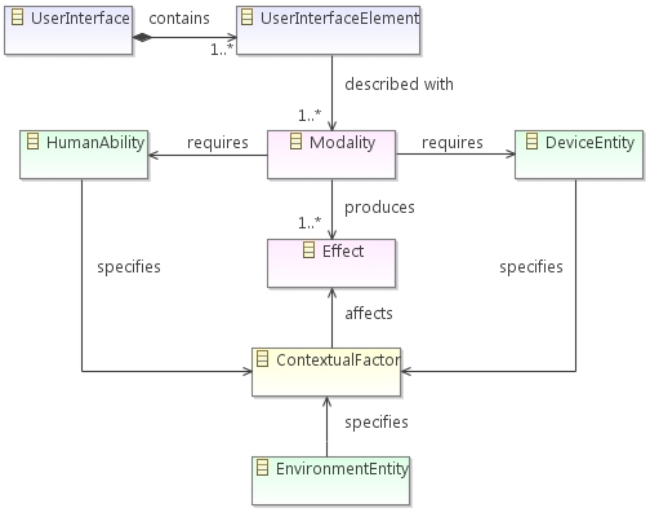

Fig. 2. Semantic relations between contextual and UI elements

Contextual factors can reduce or enhance usage of some effects from a user, device and environment perspectives (Figure 3).

In other words, contextual factors are associated with effects which they influence. This relation is described with the level in which some effects are available for a given contextual factor in terms of associated rating scale. The concept of a rating scale allows different types of evaluation. This way, we can precisely express the way contextual factors affect produced effects. Some relations can be described with a qualitative scale (for example, low, medium, high), while others can use quantitative scale from 0.0 to 1.0, or from 0 to 100 % with resolution of, for instance, 1%.

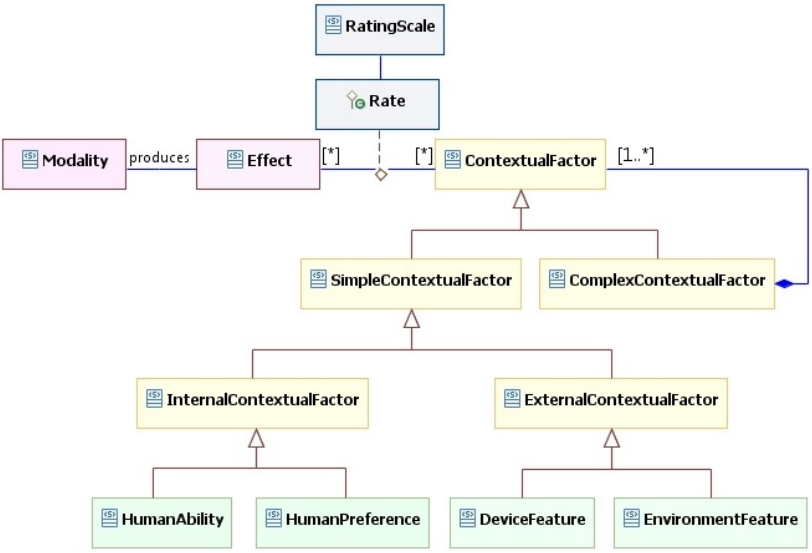

Fig. 3. Proposed approach to flexible definition of contextual factors described with UML Class Stereotypes

Contextual factors are classified into two categories – simple and complex. Simple contextual factors can be internal or external looking from the user viewpoint. Internal factors are further classified as human model elements, whereas external factors are organized as device and environment features. Internal factors describe the usage of effects from a human's perspective. External factors describe the way interaction environment and device characteristics influence the effects.

Proposed classification encapsulates information about a single piece of interaction context such as user ability or environment condition and provides a uniform interface for components that use the context. Moreover, it provides a flexible mechanism to design contextual factors of different complexities. In general, resulting factor can be a combination of user abilities, environment conditions and device properties.

Modeling User Abilities

With respect to formal specification of user-related aspects [23], the following principles are important for UI design:

- Architecture level to describe the user – We may focus on presentation details, or on the dialogue level expressing the order of user actions and responses;
- Level of abstraction to describe a user - We may have a concrete user model for a particular system design, or we may opt for a more generic model;
- Purpose of describing the user - We may create a formal description to be part of a running or system under design, or we may use the formal description to perform automated or hand analysis.

With respect to the principles above, appropriate user specification technique requires underlying ontology of human factors. The International Classification of

Functioning, Disability and Health (ICF) [24] proposed by the World Health Organization provides a comprehensive overview of many important functions of humans. It covers functioning from both an individual (such as various physiological and psychological functions) and social (such as communication issues and interpersonal relations) point of view. The ICF is a good candidate for describing all important user functionalities since it provides a detailed description of human functioning. Categories are the units of classification and are arranged hierarchically (e.g., the domain of Sensory Functions and Pain has nested categories such as Seeing Functions, Quality of Vision, and Light Sensitivity). The classification introduces generic qualifiers for measuring the extent or the magnitude of the functioning or disability for each class of human functions. In this respect, it has been used in the domain of personalized e-health systems [25]. The ICF is designed in a form of hierarchical textual ontology accessible through an online browser [26]. We have adapted and formalized the ICF ontology in a form of the UML profile.

Figure 4 shows basic ICF anatomy and function categories designed as UML Class stereotypes.

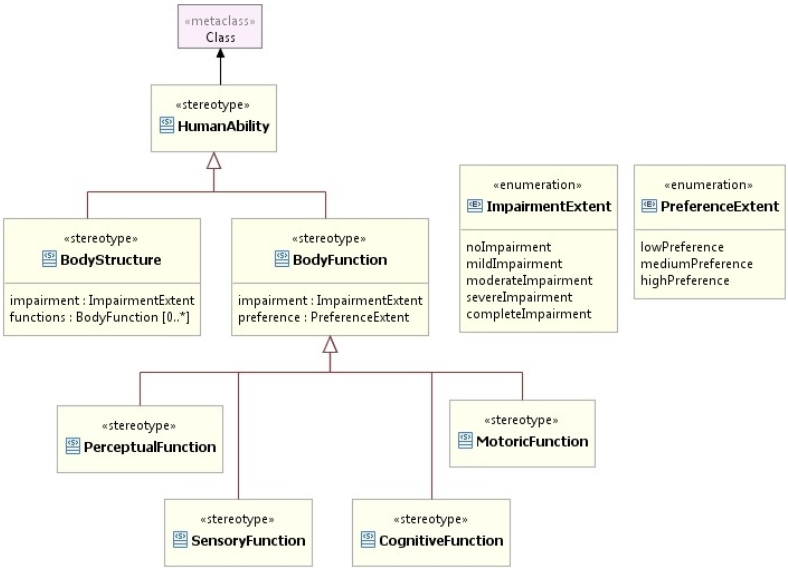

Fig. 4. UML Stereotypes for modeling human abilities

Specific categories are derived from these root elements. The measure of performance in human functioning is described with tagged value (ImpairmentExtent). These values are quantitative, expressed as a collection of distinct values in accordance with ICF. In order to express user individual preference for using particular human function, we have introduced additional attribute (tagged value) of enumerable type (PreferenceExtent). List of possible values of this type includes: *lowPreference*, *mediumPreference* and *highPreference*.

Generic concepts of modalities and effects are naturally connected to human functions described in ICF. This gives us formal mechanisms to describe humans' multimodal communication channels in a clear and flexible way.

4 The Framework

In this section we describe the UI development framework together with modeling extensions for designing UI models on different abstraction levels.

The community working on design of interactive systems has reached a general consensus in identifying several levels of abstraction by CAMELEON reference framework [4]. Connections between models from different abstraction levels are established through a sequence of model transformations.

Task model is provided for the end user's tasks. Modality independent model (Abstract UI) describes UI independently of any interaction modality and implementation. Modality specific model (Concrete UI) describes a potential UI after a particular interaction modality has been selected (for example visual, speech or multimodal). In existing framework, UI code (Final UI) is generated from modality specific level. At this point, we extend standard framework with additional abstraction level – platform specific UI model adapted to implementation technology as intermediate level between Concrete UI and Final UI. From this model the Final UI (the code) is generated. We notice that for given modality in Concrete UI (for example, List element in visual UI) a number of different implementation exists in different languages (including declarative, imperative or hybrid). On one hand, the model that captures the features of concrete technology allows us to better preserve UI interaction semantics contained in modality-specific model when decide to implement it. On the other hand, UI code generation from platform-specific model is easier and straightforward. This is especially important in case of more complex UIs such as ours.

In Figure 5 we describe the development space for the framework in which we connect interaction models with UI design models.

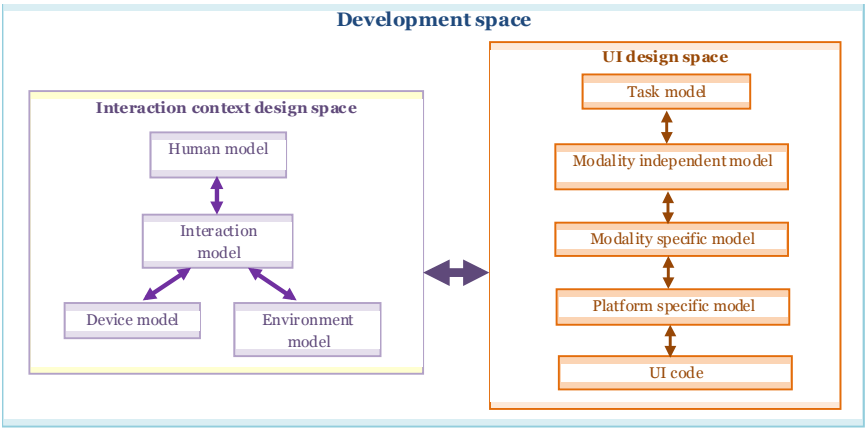

Fig. 5. Combining UI and interaction design spaces

In order to provide framework's tool support, we have developed customized extensions of Eclipse Modeling Framework (EMF), one of the most widely used UML tools, based on the described approach. UI designers can use proposed UML's semantic extensions to provide formal descriptions of interaction models and UI models. Developed tools incorporate descriptions of standard UI controls in terms of

modalities and effects. The tools also automate UI design process since they include ATL libraries that perform transformations between UI models at different abstraction levels. These transformations are parameterized by interaction context models (taking and analysing them as additional input models). In addition, EMF provides mechanisms for describing object-oriented languages with UML models. We have used these mechanisms to design a platform-specific UML models adapted to a specific language, such as Java. Currently, the tools work with Java implementations of UIs.

4.1 Task Modeling

Figure 6 shows the basic set of UML stereotypes for task modeling.

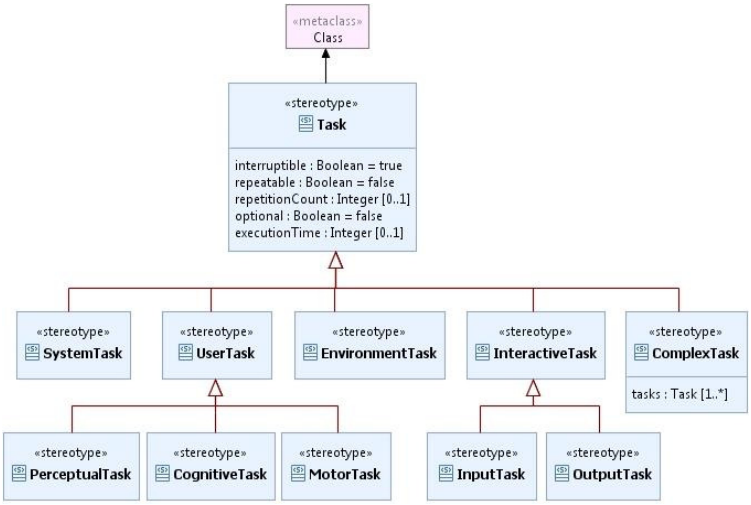

Fig. 6. Basic UML Stereotypes for task design

We consider the concept of a task in a broad sense. In this way, task can be system (connected to computer processing), user, environmental, interactive (input or output taking the computer as a reference point) and complex (integrates simpler ones). Depending on the level of information processing, user task can be perceptual, cognitive or motoric. In addition, each task is described with attributes saying whether it is interruptible, repeatable and optional, or containing information about execution time. We have introduced the ComplexTask type that enables designing composite tasks. In analogy with CTTE [9], we have also introduced stereotypes for modeling task relations as extensions of UML Association.

4.2 Modality-Independent UI Modeling

Figure 7 describes modeling extensions for description of abstract UI. AbstractUIComponent can be simple or grouping (serves as a container component). InteractiveComponent realizes presentation aspects of the interface, as well as the interaction with the user. In this respect, it can be input or output. FunctionalComponent describes UI functionalities with the set of UIOperations and is associated with corresponding interactive component.

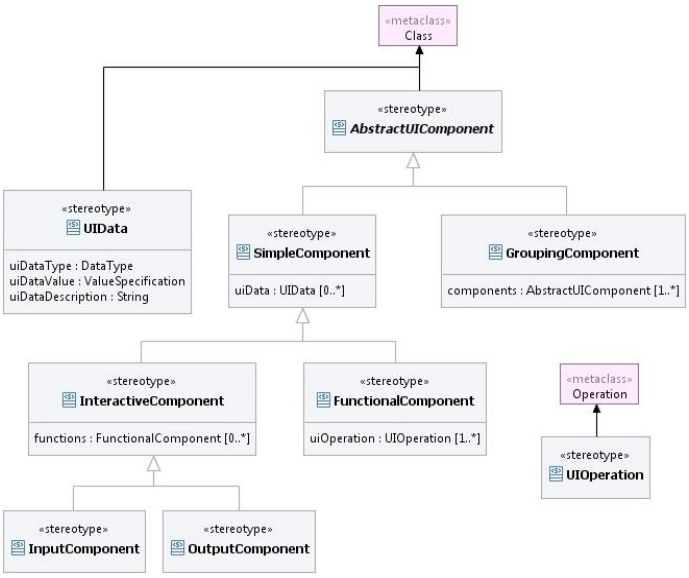

Fig. 7. Basic UML Stereotypes for AUI design

4.3 Modality-Specific UI Modeling

Figure 8 gives simplified view on UML extensions for visual UI design. Depending on complexity, visual UI component can be simple or complex (GroupingVisual-Component). The VisualComponentType type determines whether the component is an input, output or input-output (type tagged value). Simple component can be interactive (describes the interaction with the user) or functional (defines functionalities that enable the interaction).

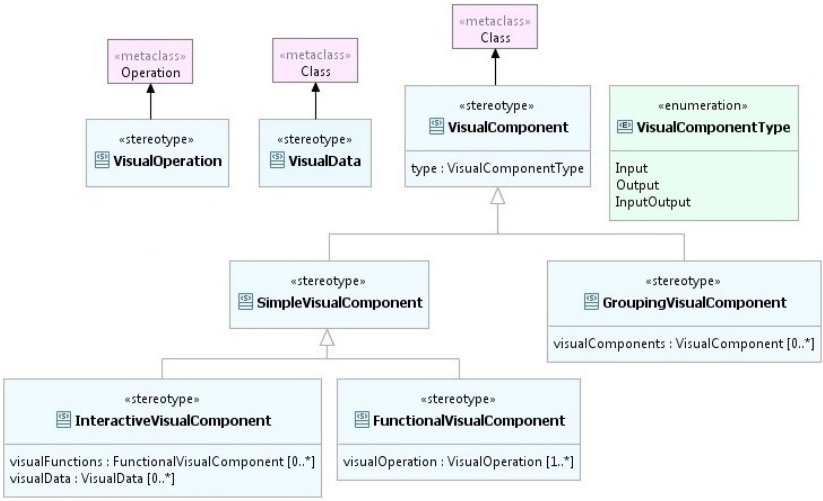

Fig. 8. Basic UML Stereotypes for visual UI design

4.4 Platform-Specific UI Modeling

Creating the platform specific model means to create UML model of a particular technology. Such a model can be built using a number of methods. EMF has so-called model importers that are able to construct UML models from XMI, XML Schema, or even Java annotated source code. We have used the model importer to describe Java Swing/AWT controls in UML. Figure 9 shows UML descriptions of common Java GUI controls.

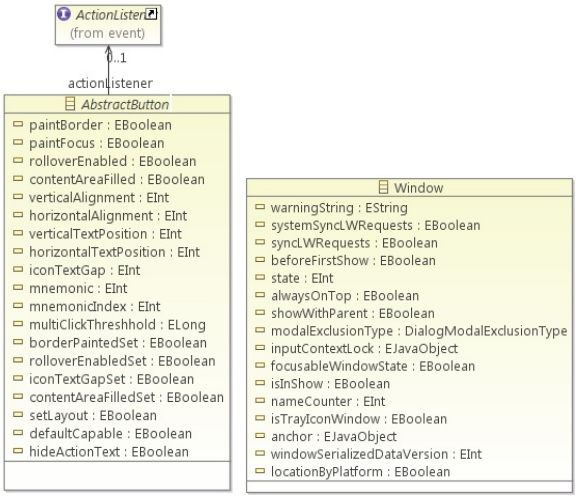

Fig. 9. Java GUI controls as UML classes

This model serves as the basis for UI code generation. We have developed code generators that transform platform-specific models into Java source code.

5 Case Study – Design for User Context

To improve usability of the UAV (Unmanned Aerial Vehicle) data visualization software tool [27, 28], we have used the framework for the design of adaptive software instrument tables that takes into account human perception. The table serves for tracking UAV flight parameters, as well as propulsion parameters. First, we describe user model used in adaptation, then we go through UI models on different abstraction levels.

5.1 User Context – Model of Human Abilities

The user is modeled with sensory, perceptual, cognitive and motoric abilities (Figure 10). *UAVOperatorProfile* is designed as InternalContextualFactor stereotype. It contains relevant human physiological and psychological functions expressed as corresponding ICF categories. Each function is designed as corresponding UML stereotype. Specific functions are organized as subcategories (nested classes) of the

containing function. For example, specific eyesight functions, such as distant vision, near vision, visual field and light sensitivity, are contained in the *SeeingFunctions* class that is modeled as the *SeeingFunction* stereotype.

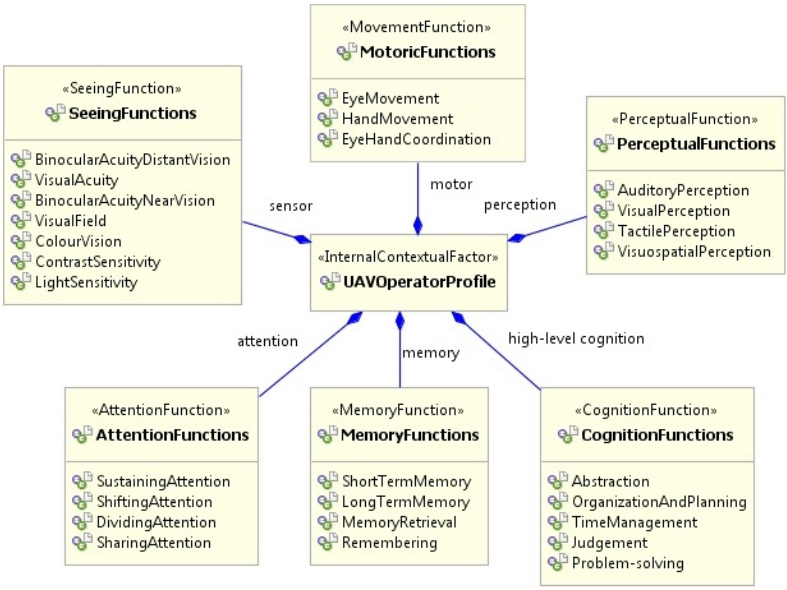

Fig. 10. Model of human abilities

Performance measure of each function is described with impairment extent and preference level designed as tagged values (attributes). On the other hand, these measures determine exploitation of the effects dependent on them. For the purpose of this paper, on the example of a specific instrument type (horizon instrument), we demonstrate instrument table design based on the analysis of impairment extent and preference level of using specific human function - visuospatial perception. Visuospatial perception is human perceptual function involved in distinction of the relative position of objects in the environment or in relation to oneself. The ability is important for the domain of interactive visualization dealing with a multidimensional set of data, such as the UAV data visualization platform.

5.2 Task Level

Analyzing existing types of tasks of a user responsible for UAV manning, we came to a common structure of a task concerned with aircraft flight parameter maintenance (Figure 11).

The task of UAV parameter maintenance is described as ComplexTask stereotype. It comprises two interactive tasks, an OutputTask stereotype of reading parameter's value and an InputTask stereotype of assigning parameter's value. The output task includes human tasks of parameter perception and parameter cognition, whereas the

input task assumes the execution of human motoric action (such as voice or hand movement). Based on this generalized model, we have derived task models for maintaining various kinds of parameters, such as air speed and horizon. The task of reading the parameter's value is carried out using a specific instrument. On the other hand, the assignment of values is executed using manual steering console.

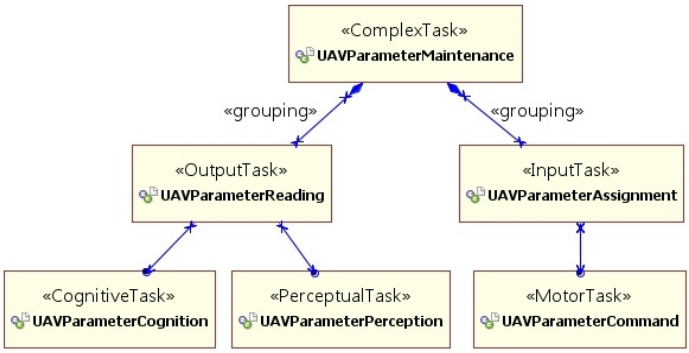

Fig. 11. Generalized task structure describing UAV parameter maintenance

5.3 Modality-Independent Level

Task model is transformed into an abstract UI model of instrument table, where each task is mapped to the corresponding abstract instrument. Figure 12 shows the model of abstract instrument that is independent of interaction modality.

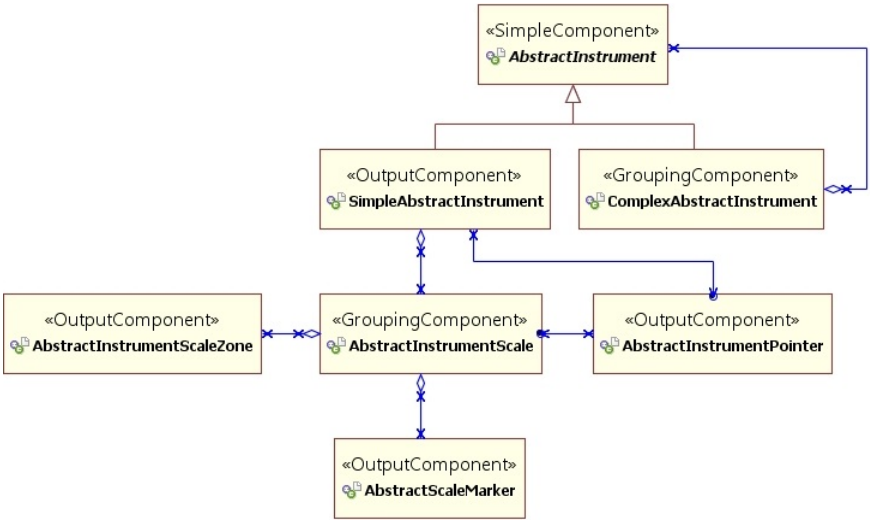

Fig. 12. Simplified abstract model of an aircraft instrument

An abstract instrument can be simple (designed as OutputComponent stereotype) or complex (designed as GroupingComponent stereotype). The complex type allows for definition of instruments that contain simpler ones. Simple instrument contains an instrument scale (designed as GroupingComponent stereotype) and scale pointer (designed as OutputComponent). The scale consists zones and markers. Zones enable presentation of critical value ranges for distinct parameters. Proposed approach to modeling instruments enables construction of generic instrument types with different structures and complexities.

5.4 Modality-Specific Level

This level combines different spaces for UI description, the UI space (Figure 13) and interaction context (Figure 14). Similar to abstract, visual instrument can be simple (designed as InteractiveVisualComponent stereotype) or complex (designed as GroupingVisualComponent stereotype). Simple instrument is further classified into common types of instruments that can be found in practice, such as counter instrument, angle counter instrument, gauge instrument and horizon instrument.

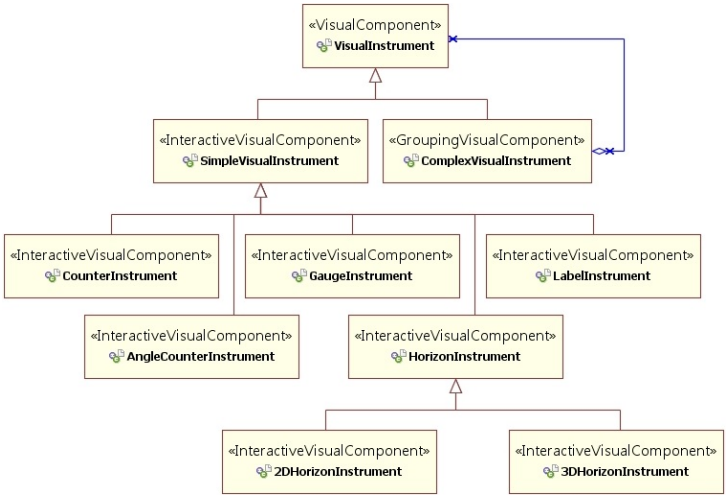

Fig. 13. Hierarchy of common visual instrument types described in terms of the UML Stereotypes

Figure 14 shows a simplified UML class diagram describing visual instrument as a complex modality together with associated effects using proposed UML modeling extensions [21]. The usage of these effects depends on related human functions contained in the model of human abilities (Figure 11). DynamicInstrumentPointer is modeled as a dynamic output modality consisting of StaticInstrumentPointers. Instrument pointer generates several perceptual effects: it is highlighted by its shape; orientation and motion. In addition, it employs slight eye movements.

InstrumentScale is designed as a complex modality consisted of scale zones and markers (both designed as static output modalites). *ScaleMarkers* add perceptual effects of highlighting by shape, size and color, whereas ScaleZones employ perceptual

effects of highlighting by color and shape, and grouping by proximity. The VisualInstrument itself produces cognitive effects of sustained attention and short-term memory. In a similar way, we have designed derived types of instruments (from Figure 13). Some of them employ specific perceptual effects. For example, HorizonInstrument produces spatial perceptual effects such as *HighlightingByDepth*.

Table 1 illustrates concrete entities and their attributes considered when creating particular type of instrument on platform-specific level.

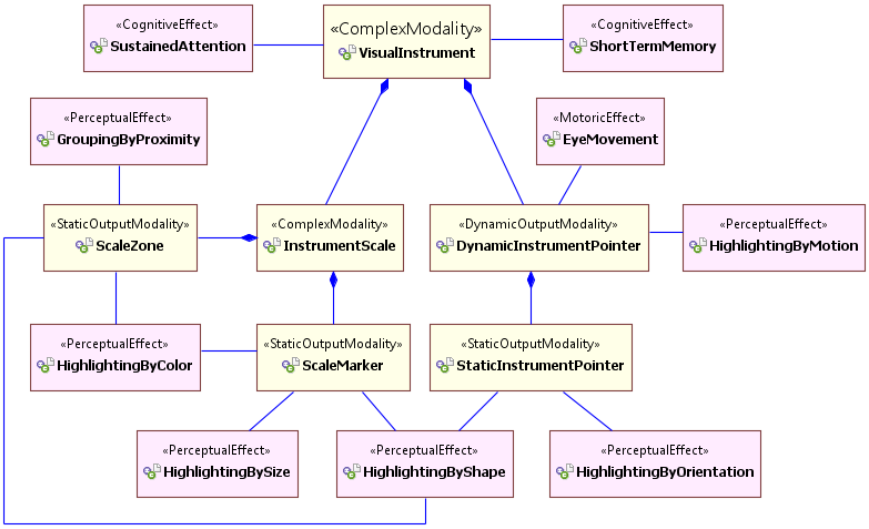

Fig. 14. Visual instrument described in terms of interaction concepts (modalities and effects)

Table 1. Elements of human perception and associated effects considered by transformation rule producing concrete instrument type

User context space		**UI design space**
Human Ability		**Perceptual Effects**
VisualPerception **VisuospatialPerception**		HighlightingByShape HighlightingBySize HighlightingByColor HighlightingByOrientation HighlightingByMotion GroupingByProximity **HighlightingByDepth**
Performance	**Preference**	**Rate**
noImpairment mildImpairment moderateImpairment severeImpairment completeImpairment	highPreference mediumPreference lowPreference	high medim low

The transformation rule takes two input elements – visual instrument (horizon) described in terms of modalities and effects (as shown in Figure 14), and model of human abilites (Figure 10). First, it performs mapping of the perceptual effects (*HighlightingByDepth*) to corresponding human functions (*VisuospatialPerception*). Analyzing performance and preference attributes for the function, it determines exploitation measure (rate) of related effects. Based on calculated ratings, it generates suitable horizon instrument model adapted to specific UI implementation technology (Figures 15).

5.5 Platform-Specific Level

This model is produced from modality-specific model and serves for UI code generation (Figure 15).

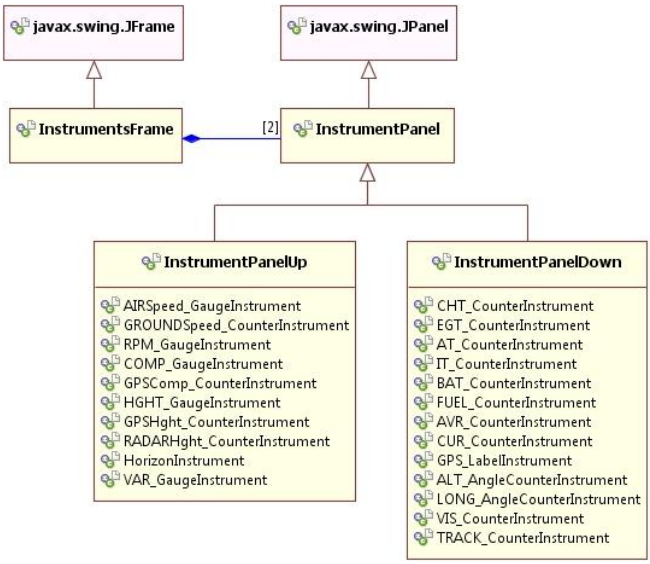

Fig. 15. Platform-specific software instrument table model

The platform-specific instrument table model comprises various types of visual instruments. The frame and panel component inherit corresponding Java Swing UML abstractions. Instruments are grouped according to their functions. The upper panel (*InstrumentPanelUp*) contains instruments for presentation of flight parameters, while the lower panel (*InstrumentPanelDown*) includes instruments for propulsion parameters.

Figure 16 shows the ATL rule that transforms UML visual horizon instrument from modality-specific level to platform-specific level (in this case adapted to Java platform).

```
rule Horizon2SwingHorizon extends VisualUIStereotypedClass {
 from s : UML2!"uml::Class" in IN
 (s.isHorizonComponent() and human.visuoSpatialPerception.isMediumToHigh())
    do {
       create3DSwingHorizon(s);
    }
}
```

Fig. 16. Simplified example of the ATL rule that produces platform-specific instrument

5.6 Implementation Level

Finally, we come to an executable UI level. Figure 17 shows different instances of software instrument table.

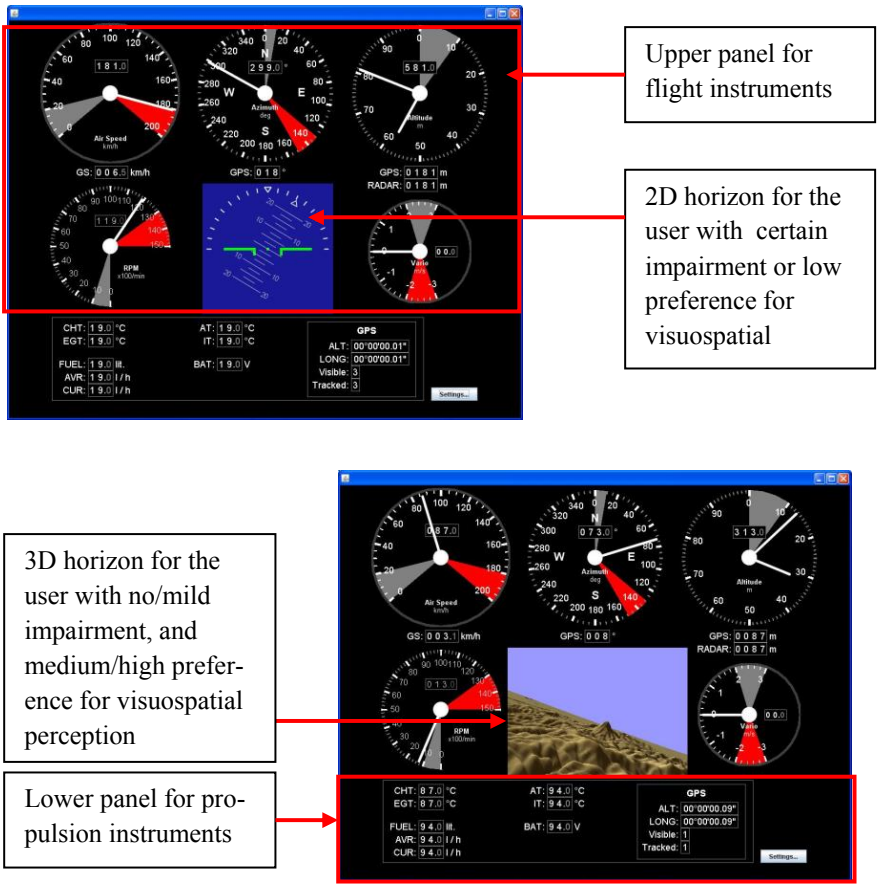

Fig. 17. Running prototypes of the software instrument table generated by analyzing human visuospatial perception

For clarity, we demonstrate the instrument table adaptation considering single instrument type – horizon instrument. The prototypes are produced from previous levels taking into account impairment extent and preference level of visuospatial perception for the concrete user. Accordingly, availability of corresponding perceptual effects is determined. Based on these, specific type of horizon instrument is created. In the first example, we have the user with certain impairment or low preference to use *VisuospatialPerception*. In this case, the availability of *HighlightingByDepth* effect is low and two-dimensional horizon instrument is created. The second example presents a three-dimensional horizon instrument for the user with no or mild impairment, and medium or high preference to engage *VisuospatialPerception*. In this case, the effect of *HighlightingByDepth* is exploited at medium or high level.

Figure 18. shows the ATL transformation rule that generates UI code from classes found in platform-specific model.

```
helper context UML2!"uml::Class" def : toCode() : String =
      self.visibilityStr +
      self.abstractStr +
      self.finalStr +
      'class ' + self.name + self.extendsClause() + self.implementsClause() +
      ' {\n' +
      self.nestedClassifier->iterate(e; acc : String = '' | acc + e.toCode()) +
      self.ownedAttribute->iterate(e; acc : String = '' | acc + e.toCode()) +
      self.ownedOperation->iterate(e; acc : String = '' | acc + e.toCode()) +
      '}\n\n';
```

Fig. 18. Fragment of the ATL code generation function

6 Conclusion

In this paper, we have proposed the model-driven framework for designing adaptive UIs. The main contribution of the approach is in defining formal mechanism for semantic integration of UI elements and interaction context elements. The mechanism is supported by the "Interaction Model" that brings together the human, device and environment contexts in terms of contextual factors. In turn, they can be used to adapt UI elements appropriately at multiple levels of abstraction.

To this end, we have designed a generic framework that provides developers means for designing adaptive UIs, as well as UML-compliant language that describes the framework. Our framework relies on standard adaptive UI architectures and lets developers use object-oriented design principles to construct and communicate domain models. Furthermore, it provides the set of abstractions - in the form of contextual factors - to formalize the description of communication between the user and the application. With our framework, developers can work with generic descriptions of UIs in terms of interaction elements, providing more flexible and more reusable solutions, suitable for a broader range of domains. High-level formal UI specifications

could serve as a common ground for investigating both design and implementation concerns by design participants from different disciplines.

The framework was used to build adaptive user interfaces for data visualization domain in the context of human perceptual abilities. In order to evaluate the approach more thoroughly and to refine it, we plan to build different applications and tools based on the described framework including context perspectives other than human. The tools should facilitate the work of developers by providing appropriate visual development means.

References

1. Paternò, F.: User Interface Design Adaptation. In: Soegaard, M., Dam, R.F. (eds.) The Encyclopedia of Human-Computer Interaction, 2nd edn. The Interaction Design Foundation, Aarhus (2013)
2. Dourish, P.: What We Talk About When We Talk About Context. Personal and Ubiquitous Computing 8(1), 19–30 (2004)
3. Thevenin, D., Coutaz, J.: Plasticity of user interfaces: Framework and research agenda. In: INTERACT, pp. 110–117 (1999)
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 15(3), 289–308 (2003)
5. Coutaz, J.: User interface plasticity: model driven engineering to the limit! In: Proc. EICS 2010, pp. 1–8. ACM Press (2010)
6. Sottet, J.S., Calvary, G., Coutaz, J., Favre, J.M.: A model-driven engineering approach for the usability of plastic user interfaces. In: Gulliksen, J., Harning, M.B., van der Veer, G.C., Wesson, J. (eds.) EIS 2007. LNCS, vol. 4940, pp. 140–157. Springer, Heidelberg (2008)
7. Limbourg, Q., Vanderdonckt, J., Michotte, B., Bouillon, L., López-Jaquero, V.: USIXML: A language supporting multi-path development of user interfaces. In: Feige, U., Roth, J. (eds.) DSV-IS 2004 and EHCI 2004. LNCS, vol. 3425, pp. 200–220. Springer, Heidelberg (2005)
8. Jovanović, M., Starčević, D., Jovanović, Z.: Languages for model-driven development of user interfaces: The state of the art. The Yugoslav Journal of Operations Research 23(3), 327–341 (2013)
9. Mori, G., Paternò, F., Santoro, C.: CTTE: support for developing and analyzing task models for interactive system design. IEEE Trans. Soft. Eng. 28(8), 797–813 (2002)
10. Mori, G., Paternò, F., Santoro, C.: Design and Development of Multidevice User Interfaces through Multiple Logical Description. IEEE Trans. Soft. 30(8), 507–520 (2004)
11. Barboni, E., Ladry, J.F., Navarre, D., Palanque, P., Winckler, M.: Beyond modelling: an integrated environment supporting co-execution of tasks and systems models. In: Proc. EICS 2010, pp. 165–174. ACM Press (2010)
12. Paterno, F., Santoro, C., Spano, L.D.: MARIA: A universal, declarative, multiple abstraction-level language for service-oriented applications in ubiquitous environments. ACM TOCHI 16(4), paper no. 19 (2009)
13. Constantine, L.L.: Canonical abstract prototypes for abstract visual and interaction design. In: Jorge, J.A., Jardim Nunes, N., Falcão e Cunha, J. (eds.) DSV-IS 2003. LNCS, vol. 2844, pp. 1–15. Springer, Heidelberg (2003)

14. Navarre, D., Palanque, P., Ladry, J.F., Barboni, E.: A model-based user interface description technique dedicated to interactive systems addressing usability, reliability and scalability. ACM TOCHI 16(4), paper no. 18 (2009)
15. Paternò, F.: Model-Based Design of Interactive Applications. ACM Intelligence 11(4), 26–38 (2000)
16. Gajos, K., Weld, D., Wobbrock, J.: Automatically generating personalized user interfaces with Supple. Artificial Intelligence 174(12), 910–950 (2010)
17. Da Silva, P.P., Paton, N.W.: User interface modeling in UMLi. IEEE Software 20(4), 62–69 (2003)
18. Van den Bergh, J., Coninx, K.: Towards Modeling Context-Sensitive Interactive Applications: The Context-Sensitive User Interface Profile (CUP). In: Proc. ACM Soft. Vis. 2005, pp. 87–94. ACM Press (2005)
19. Nunes, N.J., Cunha, J.F.E.: Wisdom - A UML Based Architecture for Interactive Systems. In: Proc. Seventh Int. Conf. Design, Specification, and Verification of Interactive Systems, pp. 191–205 (2000)
20. Jouault, F., Allilaire, F., Bézivin, J., Kurtev, I.A.: A model transformation tool. Science of Computer Programming 72(1), 31–39 (2008)
21. Obrenovic, Z., Starcevic, D.: Modeling multimodal human-computer interaction. IEEE Computer 37(9), 65–72 (2004)
22. Jovanovic, M., Starcevic, D., Jovanovic, Z.: Formal specification of usability measures in model-driven development of context-sensitive user interfaces. In: Proceedings of the International Working Conference on Advanced Visual Interfaces, pp. 749–752. ACM (2012)
23. Jovanovic, M., Starcevic, D., Minovic, M., Stavljanin, V.: Motivation and multimodal interaction in model-driven educational game design. IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans 41(4), 817–824 (2011)
24. Bruyere, S.M., VanLooy, S., Peterson, D.: The International Classification of Functioning, Disability and Health: Contemporary Literature Overview. Rehabilitation Psychology 50(2), 113–121 (2005)
25. Chittaro, L., Carchietti, E., De Marco, L., Zampa, A.: Personalized emergency medical assistance for disabled people. User Modeling and User-Adapted Interaction 21(4-5), 407–440 (2011)
26. World Health Organization (WHO): International Classification of Functioning, Disability and Health (ICF), http://www.who.int/classifications/icf/en/
27. Jovanovic, M., Starcevic, D.: Software Architecture for Ground Control Station for Unmanned Aerial Vehicle. In: Proc. Int. Conf on Computer Modeling and Simulation, pp. 284–288. IEEE Press (2008)
28. Jovanović, M., Starčević, D., Jovanović, Z.: Reusable Design of Data Visualization Software Architecture for Unmanned Aerial Vehicles. Journal of Aerospace Information Systems 11(6), 359–371 (2014)