Abstract

This is a submission to the [W3C Model-Based UI Working Group](https://www.w3.org/2013/04/beacon/) and describes a metamodel and XML format for defining abstract dialog models for context-sensitive interactive systems.

1. Conformance

As well as sections marked as non-normative, all authoring guidelines, diagrams, examples, and notes in this specification are non-normative. Everything else in this specification is normative.

2. Introduction

This section is non-normative.

The design of context-sensitive interactive systems still poses many challenges. While many of today’s modeling approaches strongly focus on the automated generation of task-based user interfaces and do not sufficiently support the flexible description of the dialog aspects (e.g. navigations) of the generated user interface and their presentation (e.g. in form of layout aspects) [1][2][3]. To overcome these shortcomings, a modeling architecture has been proposed that addresses the flexible design of context-sensitive user interfaces for ambient intelligent production environments [4] (see Fig. 1).
The architecture consists of six core models that are embedded in a user-centered development process:

- **Task Model**: The task model is used as the initial, high level description of the tasks, which are performed by the user to achieve a certain goal. The task model is build up, using interactive backend functions that are provided by the production systems within the user's environment. The task model is specified using an adapted version of the Useware Modeling Language (useML) [5].

- **Context Model**: This model describes the user’s environment in form of interaction zones. Interaction zones specify, which tasks can be performed at a certain location or might be disabled by the system due to safety reasons.

- **Function Model**: The function model describes the interactive functions of the production environment, which can be viewed and manipulated by the user.

- **Dialog Model**: The dialog model represents the abstract user interface representation and groups all interactions that can be performed at one point in time in several dialogs. Furthermore, the dialog model is used to specify the interaction and navigation behavior of the user interface. The dialog model is specified using the Useware Dialog Modeling (useDM) Language that is presented in the next chapters.

- **Presentation Model**: The presentation model is used to describe the layout and style aspects of the elements, specified in the dialog model. The presentation model is specified using the User Interface Markup Language (UIML) [6].

- **Adaptation Model**: The adaptation model is used to specify the explicit adaptation rules that have to be performed on the dialog and presentation model, according to the current context situation.

While the task and context model are used to structure the functional requirements in the early analysis phase, the dialog, function, adaptation and presentation model represent the
intermediate design models that are manipulated by the developers and interpreted by a renderer.

3. useDM Language Metamodel

*This section is normative.*

The Useware Dialog Modeling (useDM) Language represents the core model for the abstract design of context-sensitive user interfaces. The dialog model represents the abstract user interface that is used to structure the user interface by grouping abstract, modality independent interaction objects. Furthermore, the dialog model specifies the navigational aspects of the UI that express how the user can “proceed” through the single dialogs. A dialog may contain a set of hierarchically structured abstract interaction objects, which enable the execution of an interactive task.

![Useware Dialog Modeling (useDM) Metamodel](image)

A useDM model consists of *meta information*, *templates*, and a *dialog* which is refined by further dialogs. A dialog is split up into *structure* and a *behavior*. The *structure* specifies the abstract interaction objects as well as their hierarchical structure. useDM specifies six heterogeneous abstract interaction objects that support the specification of a modality-independent information exchange between the user and the user interface:

- **Container**: A container is used to hierarchically structure the abstract interaction objects.
- **Input**: The input element is used to specify the data input from the user perspective. The data can be of any type, e.g. text or numbers.
- **Output**: The output element is used to specify the data output from the user perspective. The data can be of any type, e.g. text or numbers.
- **Change**: The change element is used to specify the increment/decrement of a numerical value within a predefined interval.
- **Select**: The select element is used to specify the inclusive/exclusive selection of a value from a set of values.
- **Trigger**: The trigger element is used to specify a command from the user perspective. This might be a function call or a navigation trigger.

The *behavior* section of a dialog is used to specify the user interface behavior. Core of the behavior description is a event-based transition model that supports the *call* of backend functions, the execution of *restructure* and *statement* expressions that have an effect on the interaction objects in the dialog’s structure section and absolute/relative *navigations*. While
Variables are used to store global and local data within the user interface, conditions serve as ‘guards’ to specify under which conditions a transition might be executed.

To bridge the gap between the abstract and concrete modeling phase, additional, semantic information are incorporated into useDM that support the machine-interpretable refinement of the abstract dialog model. An external, project-specific Interactor Schema is used to specify “useDM Presentation types” (see Fig. 3) that provide detailed semantics for each interactor of the dialog model. An output-interactor, for instance, can be augmented with the information that it is used to present a large text. This information can be used in the mapping process to map the abstract output element onto a label within the presentation model for instance.

Fig 3. Useware Dialog Modeling (useDM) Metamodell

4. useDM Language XML Schema

This section is normative.

The XML schema for the useDM Language as a basis for an XML serialization of abstract dialog models can be found in useDM_schema.xsd and useDM_schema_interactors.xsd.

A. References

A.1 Normative references

A.2 Informative references


