Model-Based Approaches for Multi-Device User Interface Design and Development

Eman Saleh
Alzaytoonah University, Jordan
dreman@alzaytoonah.edu.jo

Raafah Khazem
Alzaytoonah University, Jordan
drrafaa@alzaytoonah.edu.jo

Amr Kamel
Cairo University, Egypt
a.Kamel@fci-cu.edu.eg

ABSTRACT

Model-based approaches for user interface (UI) design and development typically focus on creating mappings between concrete features and abstract features of the user interface design and development. This work presents a review of the history of Model-Based User Interface Design and Development focusing on the most recent approaches that had built an applicable solution which can allow the designers to design and develop multi-device user interfaces through a number of model-transformations.

Key Words: Model-Based User Interface Design, ConcurTaskTrees, Task Model, Domain Model, Presentation Model, Dialog Model, User Interface Description Languages, HCI .

1. Introduction

Multi-device user interface design and development has become an emerging topic due to the large and continuously increasing number of new interactive devises offered to the market.
Model-Based UI approaches give the researchers a new designing methodology that eases the creation of user interface and tackles the problems of producing a new design for every new device. Model Based systems was defined by Luyten[2] as "a piece of software that uses a set of models to support the design of user interfaces." Examples of model-based systems are Mobi-D, Teresa and Dygimes, the last two systems will be discussed in detail in the next sections. High level User-interface Description -Languages had been linked extensively to Model-Based user interface development because they offer multi-device UI creation, more specific the XML-BASED- High-Level User interface description Languages, because most of them have firmly focused on usability and scalability: making one design for many devices is the main goal, they succeed to achieve this goal for form-based interfaces, but its not the case for graphical multi-modal interfaces. Examples of these languages are the UIML[8], RIML[9], Teresa XML[6], useML[5], ISML[7]…, and there are many existing languages differ in their degree of abstraction, model coverage, Standardization and the availability for users. This paper is structured as follows: The next section gives an overview of Model-Based User Interface Development, then the Teresa tool and Dygimes environment are discussed and compared.

2. Model-Based User Interface Development

The increased interest of the academic and industrial HCI community in Model-Based User Interface Development (and High-

Level User Interface Description Languages is due to the applicability of this technique for multi-device creation. Clercks et al. [4] have discussed several advantages of MBUID like: (1) abstraction from implementation, (2) assist and/or automatically generate user interfaces from abstract models, (3) verify and validate user interfaces, and (4) obtaining consistency with older versions and other user interfaces. Figure 1 shows a common architecture for model-based systems and how the different models can be positioned inside Model-Based User Interface Development. This architecture supports user-centered design.

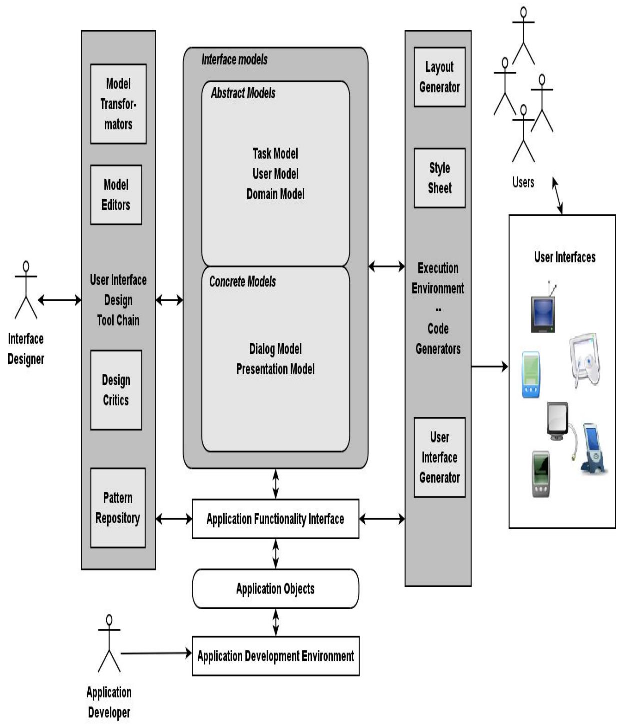

Figure 1. Model-Based User Interface Development environments

A model can be informally defined as a non-empty set with elements, with a set of relations specified between these elements. A model is an abstraction of a concept that the UI should reflect; the model gathers information about this concept.

Model-based user interface (UI) design involves the creation of formal, declarative models that describe both the look and feel of the UI, and the tasks, domain, and users that it is intended to support.

Puerta [1] defined model based user interface development system as a system that uses a limited number of selected models and does not define how these models are organized in the interface design cycle. While model based user interface development environment must support three parts:

- Design time tools: Tools that allow us to create and relate the different models.
- Run Time System: A system that allow to execute and combine different models, resulting in a concrete user interface.
- Run Time Tools: Tools that allow us to manipulate and transform the models while executing.

There are a wide range of different models that can be used in Model-Based User Interface Development: *Task model* (a model that describes the goals that the user hopes to accomplish, and the actions that must be taken to accomplish them), *Data* or *Domain model* (a model that describes the objects and data that the user will be concerned with), *Application model*, *dialog model* (a model that describes the mechanics of how the user is to interact with the UI. It specifies the navigational structure of the UI, and the used interaction techniques), *presentation model* (a model that describes the visual appearance of the user interface. It specifies which widgets have been selected, and where they are placed, among other things) and *user model* (that describes properties of the users themselves, such as their level of expertise, or their security clearance model). The data, domain and application model can be situated at the end of the application logic of the system. They define the type of objects and the operations on objects that can be used or needed to be supported by the interactive system. The task and user model are closest to the user and specify the tasks the user executes and the user or user group profile(s) respectively. The dialog model and presentation model are closest to the final user interface. An emerging new kind of model is the *Context model*: a model that can describe the context-of-use for an interactive system. E.g. a context model could specify a set of external parameters that can influence the appearance, usage, … of an interactive system. This model is the least explored, but becomes increasingly important as modern interactive systems are no longer bound to a single place and situation. Some approaches considered data model or domain model as the first models to be used in user interface design, while other approaches started with the task model. Abstract Models include: *task, domain* and *user* models and Concrete models include: *presentation* and *dialog* models [3].

2.1 Relations and Mappings between Models

MBUID environment is made up of set of models, where one model is related to another model that is, inter-model relationships must be defined. Inter-model relationships are considered as variations of the mapping problem [10] where a mapping can be expressed as a function that maps one model to another. Five mechanisms are identified to solve the mapping problem:

- Model Derivation: information of existing model is used to construct another model. E.g. Use information from task model to derive a dialog model [11, 12, 13, 14].
- Partial Model Derivation: Adding elements and the relationships between elements of a model to another model. E.g. adding transitions between states of a dialog model while examining a task model [14].
- Model Linking: Linking models to each other. E.g.: linking presentation units to unit tasks of a task model [15].
- Model Manipulation: Applying user's changes to the model manually [1].
- Model Updating: Updating the system according to the changes done by the user. E.g. updating the task model when parts of the presentation model are changed [17].

There are five goals that can be considered in MBUID, the first four were defined by Szekely [16] while the fifth was added by Kris Luytn [2]:
Challenge 1: Task-Centered Interfaces
Challenge 2: Multi-Platform Support
Challenge 3: Interface Tailoring
Challenge 4: Multi-Modal Interfaces
Challenge 5: Context-Sensetive Interfaces

3. Previous work

If we take the definition of MBUID as a set of models, *Mastermind* [19]; is one of the first projects to generate a user interface by combining different models; it used the presentation, application and dialog models to automatically generate the user interface [18, 20].
Trident (Tools foR an Interactive Development EnvironmeNT) is a model-based system to create an interactive system. [21, 22]. It was one of the first design tools that recognized the importance of a clear separation between an abstract representation of the presentation model and a concrete representation thus supporting a multitude of interaction style alternatives for the same functional core. It also integrated task analysis as an important component to create a usable interface. Together with DON; which is an earlier tool supporting the domain model and integrates the presentation model in its design methodology; Trident can be considered to be one of the first "complete" Model-Based User Interface Development Environments that where available.
Tadeus (Task Analysis/Design/End User Systems) is a Model-Based User Interface Development environment that focuses on a user model, a task model, a domain model, a dialog model and later an interaction model was added [23].
Mobi-D is a model-based integrated development environment that combines several declarative models and assists the user interface designers with the creation of these models and with the decisions they will have to make during the design of the user interface [1]. Mobi-D offers a

complete design cycle with a set of tools, and supports iterative refinements in the design of the user interface. Mobi-D works task driven.

Humanoid [25] interprets its models and generates a user interface from these models. On the other hand, *FUSE* [26] generates C++ code that can be compiled into a user interface.

The most recent MDUID are Teresa (Transformation Environment for interactive Systems representAtions) [3, 13] and Dygimes (DYnamically Generating Interfaces for Mobile and Embedded Systems) [2],

3. ConcurrentTaskTrees Notation

CTT notation is the most usable and modern specification notation used for task modeling. It provides a graphical syntax, an hierarchical structure and a notation to specify the temporal relation between tasks, an example of CTT task model is shown in figure 2. With this notation, tasks can be classified into four categories: abstract tasks ,interaction tasks ,user tasks and application tasks . Tasks at the same level can be can be connected by temporal operators like choice ([]), independent concurrency (|||), concurrency with information exchange (|[]|), disabling ([>) , enabling (>>), enabling with information exchange ([]>>), suspend/resume (|>) and order independence (|=|). The precedence of these operators from highest to lowest are : [] > {|||, |[]|}> {[>,|>} > {>>,[]>>} [28].

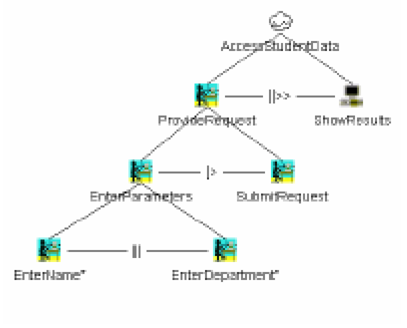

Figure 2.Simple Example of CTT Task Model

4. Teresa Tool

Teresa is a transformation-based environment, with multimodal interfaces. It provides a semi-automatic environment that supports a number of transformations to build and analyze the design at different abstraction levels and consequently generate the user interface for a specific type of platform. The steps Teresa followed for Model-Based Design [3, 27] are:

- *High level task modeling of Multi-context application*, In this phase the logical activities and the relationships among them must be specified, that is, designer must develop one model to address possible contexts of use and the domain model to identify the objects to perform tasks and the relationships among objects. This was done using the ConcurTaskTrees (CTT) notation. By this tool, designers indicate the platforms according to the performance of each task. There are many possibilities to describe how tasks are performed on different platforms: The same tasks are presented in the same way on different platforms(links in application are presented in the same way on different platforms), the same tasks are presented using *different User interface objects on different platforms*(a map that can be appeared on desktop may be replaced with links to every position on the map), the same tasks are presented using *different domain objects on* different platforms (CTT tool enable the designer to specify what information can be presented according to the type of platform), the same tasks are presented using *different task decomposition* on different platforms (tasks can be decomposed into different hierarchy according to the limitations of platforms), the same tasks are presented using *different temporal constraints* on different platforms(information that may be entered in parallel way on desktop must be entered in sequential way on mobile), or Tasks performed on different platforms are related to each other(with desktop an airline ticket can be reserved while the reservation number can be received at mobile)

- *Filtering & Refining the Task Model for different platforms* where the task model is filtered and refined according to the desired platform. This implies adding/removing tasks to/from task model depending on whether the tasks are supported or not by the target platform.
- *Generating abstract user interface from task model*, where an abstract description of the user interface; that is composed of a set of abstract presentations; is obtained from analyzing the task relationships and structured by interactors. Temporal relationships in task specification (CTT task model) control the transitions among the user interface presentations, for example, tasks with the same parent are logically related to each other on the other hand concurrent tasks that exchange information can be merged.
- *Generating the User Interface*: Here the specific properties of the target device have to be considered in order to generate the UI, that is, generated UI should take into consideration the interactor capabilities available in the target device such as the browser or available soft-keys.

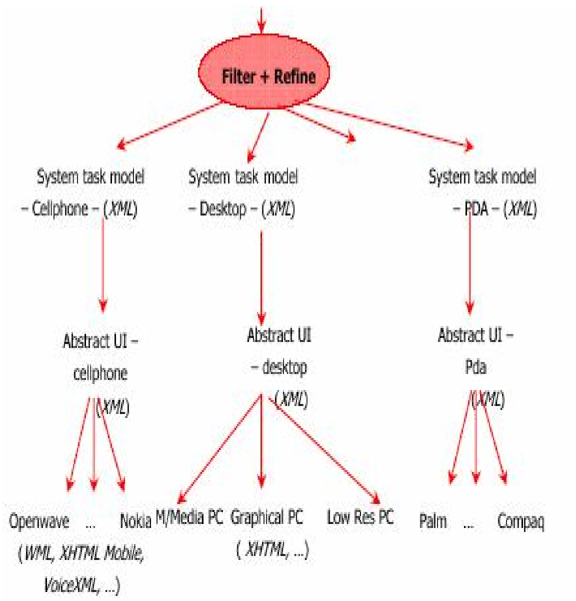
Filter + Refine
System task model - Cellphone - (XML)
System task model - Desktop - (XML)
System task model - PDA - (XML)
Abstract UI - cellphone (XML)
Abstract UI - desktop (XML)
Abstract UI - Pda (XML)
Openwave ... Nokia
(WML, XHTML Mobile, VoiceXML, ...)
M/Media PC
Graphical PC
Low Res PC
(XHTML, ...)
Palm ... Compaq

Figure 3. Transformations Supported in Teresa

5. Dygimes Environment

Dygimes is a model-based run time environment, Dygimes dynamically generates the user interface for mobiles and embedded systems, it is a task-centered approach. The environment can generate a concrete user interface from the models without code generation.

Dygimes focused on three models: the task model, the dialog model and the presentation model. Application and domain models were not considered intensively. The steps Dygimes followed for Model-Based Design are[2,29]:

- Providing the task specification using the ConcurTaskTrees notation.
- Annotating leaves in task notation with abstract UI description (user interface building blocks), then a graphical tool (the ConcurTaskTrees annotation tool] is used to attach these UI building blocks to the leaves of the tree.
- Providing one “annotated” task specification by combining task specification (XML document) and UI abstract description (XML document) in a single XML document that can be processed by the system.
- Transforming the task tree generated in the first step into a priority tree according to the precedence of temporal operators.
- Computing the Enabled Task Sets: "Tasks that can be active at the same time and presented to the user all at once."
- Creating the dialog model depending on temporal relations between tasks and the enabled task sets computed in the third step. The State Transition Network (STN)[30], was used to specify the activity chain: A chain that represents a path that the dialog will follow to reach a certain goal. Each state represents an enabled task set and connected to other enabled task set, this directed connection indicates the transition between dialogs.
- Generating abstract user interface description from enabled task sets and the STN were the enabled task sets give the content of a dialog and STN gives that transition between the dialogs.
- Generating the actual user interface.
- Testing the user interface: Here the designer feedback can change grouping or splitting the enabled task sets according to

heuristics rules presented by Paterno et al [3]

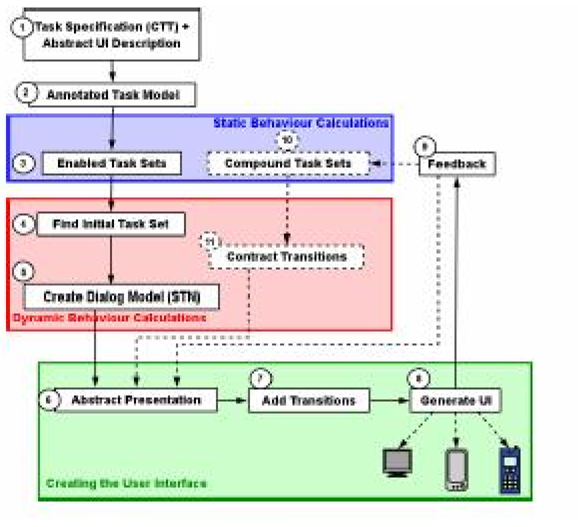

Figure 4.The Dygimes User Interface design and generation process

6. Comparison between Teresa and Dygimes

- *Task Specification*: Both Dygimes and Teresa are built around the ConcurrentTaskTrees notation but while Teresa extended ConcurTaskTrees Environment (CTTE) to generate platform-dependent task specification by adding the attributes of specific platforms (mobile, PDA, desktop) in each task specification and filtering a task model from the parts that are not supported by a given platform, Dygimes generated platform-independent task specification annotated with user interface building unit.
- *Presentation Model*: Both approaches need to calculate Enabled Task Sets using algorithm (Dygimes) or heuristics (Teresa) to generate presentation model, but in Dygimes, we need to transform original task tree into priority trees in order to compute the Enabled Task Sets for a task specification and then derive a dialog model that is expressed in STN.
- *Design Approach*: Dygimes used a bottom-up approach (starting with a concrete XML-based User Interface Description Language targeted towards embedded systems. Using task modeling, constraint based layout management, dialog modeling and context-sensitive models; support for user interface design for embedded systems is added. Teresa used a top-down approach (Designers first have to create more logical descriptions and then move on to more concrete representations until they reach the final system.
- *Tool Supporting*: Dygimes is considered as a run time environment with limited tool support while Teresa is considered as a design environment focused on tool support.
- *XML-Based High-Level Description Language*: Dygimes support User Interface Markup Language (UIML) for rendering because this language supports presentation and domain models not a task model which is supported by XML-Based notation (CTTE). UIML implements the four aspects of the presentation model: structure of the user interface, rendering hints, Widget mappings and layout description. Teresa supports Teresa XML which combines XML-Based High-Level User Interface Description Language with CTT XML-Based Task Model Language and it integrates task and presentation model and implements two of four presentation aspects: Structure of user interface and the widget mapping according to the output platform (mobile, desktop, PDA).

7. Summary

In this work we reviewed Model-Based Muti-Device User Interface Development supported by using User Interface Description-Languages UIDLs. Current tools and frameworks have been reviewed; intensively; the Teresa tool and the Dygimes environment. Although these systems got closer to the solution of the problem of developing Muti-Device User Interfaces they still suffer from acceptability by industry for many reasons such as: They can not influence the organization of the final user interface presentation, and they did not give a full coverage of *modalities* also they have poor *usability* since the resulting UI may not confirm the requirements of the user.

The use of UIDLs causes two problems the first is that they cannot take full advantage of the target widget set since most of UIDLs have a predefines abstractions that can be used to describe the user interface, the second drawback is the *complexity* of the process to transform the abstract user interface description into a concrete working UI for a specific platform.

Finally there is still lack of user-centered design to narrow the gap between the actual tasks that the user wants to perform and the user interface exposed by Muli-Device user interface to support these tasks.

References

[1] Puerta A. ,"A model-based interface development environment," IEEE Software., pp. 40–47, 1997.

[2] Coninx, K., Luyten, K., Vandervelpen, C., Van den Bergh, J.and Creemers, "Dygimes: Dynamically Generating Interfaces for Mobile Computing Devices and Embedded Systems,“ Mobile HCI, volume 2795 of Lecture Notes in Computer Science, pp. 256–270, Springer, 2003.

[3] Mori G., Paterno F. and Santoro C., "Design and Development of Multidevice User Interfaces through Multiple Logical Descriptions," IEE Transactions on Software Engineering, Vol.30, No. 8, 2004.

[4] Clerckx *T.* Winters F. and Coninx K., "Tool Support for Designing Context Sensitive User Interfaces using a Model Based Approach," Proceedings of the 4th international workshop on Task models and diagrams TAMODIA '05, ACM Press, 2005.

[5] World Wide Web consortium, Useware Markup Language http://www.uni-kl.de/pak/useML/

[6] Luyten K., Abrams M., Limbourg Q. and Vanderdonckt J., "Developing User Interfaces with XML: Advances on User Interface Description Languages," Sattelite workshop of Advanced Visual Interfaces (AVI) 2004, Expertise Centre for Digital Media, 2004.

[7] World Wide Web consortium, Interface Specification Meta Language, http://decweb.bournemouth.ac.uk/staff/scrowle/ISML/

[8] World Wide Web consortium, User Interface Markup Language, http://uiml.org

[9] World Wide Web consortium, Renderer Independent Markup Language, http://www.consinsus-online.org

[10] Puerta A. and Eisenstein J.,"Towards a General Computational Framework for Model-Based Interface Development Systems,"In IUI 1999 International Conference on Intelligent UserInterfaces, pp. 171–178, 1999.

[11] Limbourg Q., Vanderdonckt J. and Souchon N., "The Task-Dialog and Task-Presentation Mapping Problem: Some Preliminary Results," In Palanque and Patern`o [PP00], pp. 227–246.

[12] Vanderdonckt J., Limbourg Q. and Florins M., "Deriving the Navigational Structure of a User Interface," In M. Rauterberg and J. Wesson, editors, Proceedings of 9th IFIP Conf. on Human-Computer Interaction Interact'2003, pp. 455–462, 2003.

[13] Mori G., Patern`o F. and Santoro C., " Tool Support for Designing Nomadic Applications," In Proceedings of the 2003 international conference on Intelligent user interfaces, pp. 141–148, Miami, Florida, USA, January 12–15 2003.

[14] Luyten K., Clerckx T., Coninx K. and Vanderdonckt J., “Derivation of a Dialog Model for a Task Model by Activity Chain Extraction,” DSV-IS 2003, Springer.

[15] Coninx K., Luyten K., Vandervelpen C. ,Bergh J.and Creemers B. ,”Dygimes: Dynamically Generating Interfaces for Mobile Computing Devices and Embedded Systems,” In Luca Chittaro, editor, Mobile HCI, volume 2795 of Lecture Notes in Computer Science, pp. 256–270, Springer, 2003.

[16] Vanderdonckt J,”Computer-Aided Design of User Interfaces II,” volume 2. Kluwer Academic, 1996

[17] Vanderdonckt J. and Puerta A., “Computer-Aided Design of User Interfaces III,” volume 3. Kluwer Academic, 1999.

[18] Stirewalt K., ”Automatic Generation of Interactive Systems from Declarative Models,” PhD thesis, Georgia Institute of Technology, 1997.

[19] Szekely A. P., Sukaviriya N. P., Castells P.,Muthukumarasamy J. Salcher E., “Declarative Interface Models for User Interface Construction Tools: The MASTERMIND Approach,” In EHCI, pp. 120–150, 1995.

[20] Stirewalt K. and Rugaber S. ,“Automating User-Interface Generation by Model Composition,” In Proceedings of the IEEE International Conference on Automated Software Engineering , 1998.

[21] Francois B., Hennebert A., Leheureux J. and Vanderdonckt J, “ Towards a Dynamic Strategy for Computer-Aided Visual Placement,” In Workshop on Advanced Visual Interfaces, pp. 78–87. ACM press, 1994.

[22] Vanderdonckt J. and Bodart F., ”Encapsulating Knowledge for Intelligent Automatic Interaction Objects Selection. In ACM Conference on Human Aspects in Computing Systems InterCHI’ 93, pp. 424–429. Addison Wesley, 1993.

[23] Forbrig P. and Stary C., “From Task to Dialog: How Many and What Kind of Models do Developers Need,” CHI’98 workshop, 1998.

[24] Szekely P., Luo P. and Neches R. ,” Facilitating the Exploration of Interface Design Alternatives: the HUMANOID model of interface design,” In CHI, pp. 507–515, 1992.

[25] Lonczewski F. and Schreiber S. ,”The FUSE-System: an Integrated User Interface DEsign Environment,” In Vanderdonckt [Van96], pp. 37–56.

[26] Patern`o F. ,Model-Based Design and Evaluation of Interactive Applications, Springer, 2000.

[27]Limbourg Q.,Jacob R. and Vanderdonckt J., “Computer-Aided Design of User Interfaces,” IV, volume 4. Kluwer Academic, 2004.

[28] World Wide Web consortium, ConcurrentTaskTrees http://giove.cnuce.cnr.it/ctte.html

[29] Luyten K., Laerhoven V. T., Coninx K. and Reeth V. F., ”Runtime Transformations for Modal Independent User Interface Migration. Interacting with Computers,” 15(3):329–347, 2003.

[30] Wasserman A., “Extending State Transition Diagrams for the Specification of Human-Computer Interaction,” IEEE Transactions on Software Engineering, 11:699–713, 1985