Evaluation of Model-Based User Interface Development Approaches

Jürgen Engel, Christian Herdin, and Christian Märtin

Augsburg University of Applied Sciences, Faculty of Computer Science,
An der Hochschule 1, 86161 Augsburg, Germany
{Juergen.Engel,Christian.Herdin,
Christian.Maertin}@hs-augsburg.de

Abstract. The PaMGIS framework was developed at Augsburg University of Applied Sciences and is aimed at supporting user interface designers without profound software development skills to specify the diverse models which allow for at least semi-automated generation of user interface source code. Currently these are task, dialog, interaction, and layout models as well as user, device, and environment models. The complexity of the model definitions is reduced by the application of patterns of various types and different abstraction levels. These patterns are specified by means of the PaMGIS Pattern Specification Language (PPSL) that is a further refinement of the Pattern Language Markup Language (PLML). Amongst other descriptive information PPSL specifications incorporate sophisticated pattern relationships and model fragments, which are deployed as soon as an individual pattern is applied. In this context we have evaluated existing model-based user interface development frameworks in order to elicit new ideas to improve the applicability of PaMGIS.

Keywords: Model-based user interface development, pattern-based development, user interface modeling, user interface generation, HCI patterns.

1 Introduction

In the scope of our research within the Automation in Usability Engineering group (AUE) at Augsburg University of Applied Sciences we develop an integrated approach for the design and semi-automated generation of user interfaces (UI) of interactive software applications named Pattern-based Modeling and Generation of Interactive Systems (PaMGIS) [9], [10]. It combines both, model-based and pattern-based development techniques and methods. We have identified room for improvement regarding the modeling of dynamic UI behavior and the modeling of UI layout aspects. In addition, we are interested in possibilities to influence the UI appearance at runtime.

In this context we have conducted a literature review of existing model-based UI development environments (MB-UIDE) in terms of their functionality, suitability, adequacy, conformance to the abstraction layers defined by the CAMELEON Reference Framework (CRF), i.e. Model, Abstract UI (AUI), Concrete UI (CUI), and

Final UI (FUI) [1], and their general availability. The results will be used to extend the potential of our PaMGIS framework and to overcome the mentioned deficiencies.

The rest of this paper is organized as follows: the review approach is described in Section 2, brief descriptions of the considered MB-UIDEs are provided in Section 3, the review results are summarized in tabular format in Section 4, and our lessons learned and decisions regarding PaMGIS are depicted in Section 5. Finally, Section 0 provides the list of literature being consulted during the review process.

2 Review Approach

Several MBUID reviews have already been carried out and the results are available through the Internet, e.g. [7], [29], [42]. However, two of the documents date from the 1990's [29], [42] and the most current from the year 2001 [7] and therefore they do not cover novel approaches. Nevertheless, these documents delivered valuable input for our updated evaluation, notably for defining the MB-UIDE characteristics to be investigated.

Subject of the literature review has been an assortment of existing MD-UIDEs. Within the current paper we focus on the environments listed in Table 1.

Table 1. MB-UIDEs considered in this paper

MB-UIDE	Originator	Literature reviewed
AME	Augsburg University of Applied Sciences, DE	[24],[25],[26]
ITS	IBM T.J. Watson Research Center, US	[1],[45],[46]
MARIAE	Istituto di Scienza e Tecnologie dell' Inform., IT	[20],[32],[33],[34],[35],[36]
MECANO	Stanford University, US	[38],[41]
MOBI-D	Stanford University, US	[37],[39],[40]
SUPPLE	University of Washington, US	[12],[13],[14],[15][16],[17],[18]
TERESA	Istituto di Scienza e Tecnologie dell' Inform., IT	[3],[4],[27],[28],[31]

Our literature review actually compassed several more MB-UIDEs, including ADEPT [23], FUSE [22], GENIUS [21], HUMANOID [43], JANUS [1], MASTER-MIND [44], TADEUS [8], TEALLACH [19], TRIDENT [5], and UIDE [11]. On the one hand, we could not retrieve any more recent documentation for these fairly old approaches and on the other hand, they have been already covered within the former evaluations [7], [29], [42]. Therefore, and due to space restrictions we picked the most current MB-UIDEs and such systems for that we assumed they would deliver the most promising results for our purposes.

For each of the MD-UIDEs we captured (1) the short name, (2) the full name, if any, (3) its originator, (4) date of first publication, (5) actuality in terms of current version or most current publication, (6) provided functionality in terms of supporting UI modeling, UI generation, UI runtime environment, (7) provided support of CRF abstraction levels, i.e. model, AUI, CUI, and FUI, (8) models actually supported, (9) utilized model notations resp. User Interface Description Languages (UIDL), (10) whether the MB-UIDE was mainly intended to support multi-device, multi- platform, multi-user, or multi-environment developments, (11) tool-support offered with the MB-UIDE, (12) supported target programming languages, (13) supported target devices respectively target platforms, (14) its availability in terms of whether there is a real implementation of the MB-UIDE, meaningful application examples are available and whether the framework can be freely downloaded from the Internet, and finally (15) type of available documentation.

Due to space limits it was not possible to present all details of the evaluated characteristics within this paper. Therefore, we decided to provide as much information as possible within the textual MB-UIDE descriptions (see Section 3) and in summarized tabular form (see Section 4).

3 Description of Considered MB-UIDE

3.1 AME

The *Application Modeling Environment* (AME) was developed at Augsburg University of Applied Sciences between 1992 and 1996 [24], [25], [26]. AME's goal was to tightly integrate the object-oriented software development process with user interface modeling and design starting already in the early phases of the software engineering life cycle and to accompany the software developer until the final implementation of the interactive system. AME used structural information, object-oriented relationships, and semantic knowledge about the application context to automatically generate prototypical MS Windows GUIs for business applications including the dynamic behavior of the UI and the binding to the business objects.

AME uses an object-oriented analysis (OOA) model as starting point. An OOA model defines the domain classes with their attributes and typically contains only abstract specifications of the domain class methods (i.e. method name, calling parameters, calling parameter types, return type) as well as the relationships between classes. OOA models define the domain space of an application. They are created by model editor tools and are transformed into AME's internal model representation. The resulting class models unify the modeling functionality both from Rumbaugh's OMT and Coad and Yourdon's OOA. Attribute names and data types, as well as relationships and their types (generalization/specialization, aggregation, association with semantic information) are parsed by the structure refinement tools to automatically create the window and dialog box structure of the application and for defining abstract interaction objects (AIOs). Domain classes may also include message links (i.e. dynamic relationships to other classes) that can be exploited by AME's behavior tools to automatically create dynamic user interface behavior. No task model is therefore needed. Thus, an OOD model that defines the basis for the solution space, including the user interface structure, can be generated automatically. However, it is also possible for developers to introduce their own OOD classes with AIOs added to the domain objects or to modify the OOD model.

A series of additional knowledge-based automated tools can then be applied to create a prototype of the user interface with concrete interaction objects (CIOs) in the UI builder of Intellicorp's KAPPA-PC development platform including dynamic behavior (i.e. interaction dynamics, navigation, function calls). At this stage, developers and designers can again interfere with the prototype to add their own styles or change the types of the generated CIOs. The UI CIOs are still directly linked to the internal OOD representation. In a final step, the detailed OOD representation is again parsed to finally generate C++ UI code for MS Windows.

3.2 ITS

The *Interactive Transaction System* (ITS) has been developed in the context of a scientific project at the *T.J. Watson Research Center* of the *International Business Machines Corporation* (IBM) in Yorktown Heights. The first publication dates from 1989. ITS provides a rule-based approach for the definition and generation of application and user interface models and incorporates a runtime environment for execution of these models [2]. Amongst others, the visitor information system of the world exposition EXPO 1992 in Sevilla (Spain) has been implemented using ITS [46], [45].

A major principle is the strict separation of the actual content of a software application respectively a user interface from its presentation [2]. The ITS architecture is subdivided into four layers: (1) The *Action Layer* implements the necessary functions of the application's business logic independent of any dialog control matters. (2) The *Dialog Layer* defines the content of the user interface without considering its presentation. Dialogs are specified by means of logical frames and the control flow among them. (3) The *Style Rule Layer* defines the presentation and behavior of the user interface. Based on modifiable rules the system decides automatically by which concrete interaction object every single abstract interaction object will be replaced. Style rules are executed at compile time. (4) Finally the *Style Program Layer* takes care for the mapping of toolkit primitives according to the settings defined within the Style Rule Layer. These decisions are made during runtime, i.e., the final layout is determined not until a frame is displayed on the screen [45].

In the initial step of the ITS development process an expert of the problem domain specifies the data types being exchanged between the UI and the application as well as the dialogs. An application programmer implements the functions of the business logic [45]. From the data type and dialog definitions the *Dialog Compiler* generates a parse tree. This tree is subsequently passed to the *Style Compiler* that assigns the appropriate interaction objects to the tree's nodes by exploiting the *Style Rules*. The resulting full-featured parse tree is processed by the runtime environment that is responsible for calling the application functions and displaying the user interface [2].

3.3 TERESA

The *Transformation Environment for Interactive Systems Representations* (TERESA) has been developed by the Human-Computer Interaction (HCI) group of the *Istituto di Scienza e Tecnologie dell' Informazione* which is an Institute of the National Research Council of Italy (CNR) [4]. This approach supports the design and development of multi-device user interfaces. The work on TERESA started in 2003 [20].

The first step of the TERESA development methodology envisages the creation of a high-level task model which includes not only the actual relevant tasks and sub-tasks, but also information related to contexts of use and involved roles. Additionally, it uses a domain model which describes all interaction objects that have to be manipulated during the task execution as well as the relationships between these objects. From this task model so-called platform-specific system task models are derived by carrying out filtering and optional refinement actions. This forms the basis for generating abstract user interfaces (AUI) consisting of a set of abstract presentations which arise from the analysis of the interrelations of the sub-tasks. The presentations can be understood as compositions of abstract interaction objects resulting from the application of various composition operators, including grouping, ordering, hierarchy, and relation. From the AUI source a platform-dependent UI description is generated considering any specifics of the target device and target operating system. In this stage each abstract interaction object is replaced by a concrete one [28]. From this concrete description (CUI), in turn, the final user interface (FUI), is generated, e.g. in terms of XHTML or Java code [27]. TERESA supports the UI designer by applying different strategies with regard to the fact that it is not necessarily reasonable to implement all the tasks and sub-tasks in a similar way on each intended target platform [4], [27]. Furthermore the construction of multi-modal user interfaces is supported [31].

3.4 MARIAE

The *Model-based Language for Interactive Applications* (MARIA) *Authoring Environment* (MARIAE) is also developed by the HCI group of ISTI-CNR [32]. The initial version has been designed on the basis of the expertise and experiences collected with the predecessor environment TERESA. The first publications date from 2009 [34], [35], [36]. MARIAE is still under development; the current version 1.5.6 can be downloaded from the Internet[1].

MARIAE supports the design and development of Web Service-based interactive applications for multiple target platforms. Usually, Web Services are not constructed in the course of the development of the interactive application, but in fact already existing ones are being accessed. This matter of fact is factored within the MARIAE development process that combines both top-down and bottom-up approaches [36]. Additionally, on the one hand, MARIAE is fully compliant to the CRF degrees of abstraction and hence implements the model level as well as AUI, CUI, and FUI [32]. This implies a top-down development procedure. On the other hand, the analysis and planning regarding the utilization of Web Services demands a bottom-up approach [36].

In a first step the task model of the interactive application is elaborated using the ConcurTaskTrees (CTT) notation [30]. Subsequently, the relations between the task model and the chosen Web Services are established. They are described by means of

[1] See `http://giove.isti.cnr.it/tools/MARIAE/download`

the Web Service Description Language (WDSL)[2] and optionally possess *Annotations* which contain information regarding their later appearance in the user interface. On the basis of the so-called enriched task model the AUI can be generated [32]. The transformation process mainly exploits the hierarchic structure of the task model, the task types, the temporal relationships between the tasks, and the mentioned Web service Annotations. In this stage Presentation Task Sets are identified which consist of tasks being active at the same period of time [33]. In the next step, the AUI is transformed into the platform-specific CUI. This process can be regarded as refinement of the abstract model where essentially the abstract interaction objects are replaced by selected concrete ones. These conversions are specified via *Extensible Stylesheet Language Transformation* (XSLT)[3] [33]. AUI and CUI are described my means of MARIA XML [34]. Finally, the CUI is transformed by means of XSLT into a target language [33], e.g., *Extensible Hypertext Markup Language* (XHTML)[4].

3.5 MECANO and MOBI-D

MECANO was developed at the Department of Medicine and Computer Science at the Stanford University in the context of the Mecano project [41]. The development started in 1995 [38] and the latest publication has been published at the CADUI-Conference in 1996 [41]. MECANO is a model-based interface development environment that enhances the concept of generating interface specifications from data models. It utilizes domain and interface models. The domain model is employed to generate the layout and the relationships inside the model to determine the dynamic behavior of user interfaces [41].

The MECANO Interface Model (MIM) is described by means of the purpose-built MECANO interface modeling language named MIMIC. MIMIC is an object-oriented language that supports modeling at a meta-level and assigns specific roles to each interface element. The grammar of MIMIC is written in Backus-Naur-Form (BNF) [38]. The development environment that supports the MECANO framework in terms of MIMIC and its associated MIMs is called MOBI-D [37].

The Model-based Interface Designer (MOBI-D) is the successor of MECANO [37]. The development started in January 1997 [37] and the last publication dates from 1999 [40]. Like MECANO, MOBI-D supports model-based design of user interfaces. It uses five models to reach this goal: user, task and domain model as abstract models, dialog and presentation model as concrete models [39]. MOBI-D has no support for automatic transformation between the models, but the user can do this conversion manually. All models and mappings are specified with the Mecano Interface Model (MIM) textual notation. The Mecano interface modeling languages (MIMIC) are used to define the components, structure, the elements and relations within interface models [37].

[2] See `http://www.w3.org/TR/wsdl`
[3] See `http://www.w3c.org/TR/xslt`
[4] See `http://www.w3.org/TR/xhtml1/`

MOBI-D uses a textual task description of an end user as starting point. The MOBI-D tool U-TEL translates this description into a structured user-task description. The UI developer uses this description to build the user-task and domain models with the help of MOBI-D's model editing tools. MOBI-D uses the user-task and domain models to display suggestions for the presentation and interaction techniques. The developer can select one of these suggestions for the programming of concrete end user interfaces. Finally, the end user conducts a test of the new user interface [39].

3.6 SUPPLE

The SUPPLE system has been developed at the University of Washington in Seattle. The first publication dates from 2004 [14], the most current document we discovered in the Internet is from 2010 [15]. This approach utilizes functional interface specifications as well as device and user models. User interface generation and adaptation are treated as decision-theoretic optimization problems. SUPPLE searches for optimal renditions considering any relevant device constraints and minimizing the user's effort required to carry out the necessary UI actions [14]. In addition, SUPPLE is capable to adapt the user interface to the user's individual work style [13] as well as to personal preferences [16]. UI generation and adaptation is executed during runtime [13]. SUPPLE++ is a variant of the SUPPLE system and supports automatic creation and modification of user interfaces for users with motor and/or visual impairments. The initial publication regarding SUPPLE++ is from 2007 [15].

Within SUPPLE a functional interface specification is defined as a set of abstract interface elements and a set of interface constraints. The elements are specified in terms of their data types that can be either primitive or complex. The constraints are expressed as functions mapping renderings to a Boolean value and allow, for instance, map certain elements to the selfsame widget. The device model comprises of the available widgets, device-related constraints, and two device-specific functions for evaluating the adequacy of the widgets to be used. One function measures the appropriateness of the widgets for interacting with the variables of the given types while the other calculates the user's effort required for navigating through the UI. The user model is defined by means of user traces, which are a type of logs of user actions, recorded at runtime. Supple is aimed at finding the most appropriate rendering for each individual abstract interface element. This is achieved by means of a branch-and-bound algorithm for minimizing a cost function, which is composed of the previously mentioned functions and information from the device and user models [14]. The cost function consists of more than 40 concerted parameters and cannot easily be determined manually. Therefore, a tool named ARNAULD has been developed in order to facilitate this process [16]. SUPPLE++ primarily utilizes even more complex cost functions in order to consider the motor and visual impairments of handicapped users. In analogy to ARNAULD SUPPLE++ is supported by another tool named *Activity Modeler* [12].

4 Summary of Review Results

General MB-UIDE characteristics are depicted in Table 2 in a condensed format.

Table 2. General MB-UIDE characteristics

MB-UIDE	First Publ.	Current Vers. / Latest Publ.	Functionality	CRF Abstraction Levels	Target (Multi~)
AME	1993 [26]	1998 [24]	Model, Generat., Runtime [24][26]	Model, AUI, CUI, FUI [26]	n/a
ITS	1989 [2]	1990 [45],[46]	Model, Runtime [46]	Model, AUI, CUI [2]	Platform, User [46]
TERESA	2003 [28]	Version 3.4 2008 [31]	Model, Generation [27]	Model, AUI, CUI, FUI [27],[28],[31]	Platform [28],[31] Modal [31]
MARIAE	2009 [34],[35]	Version 1.5.6	Model, Generation [34], [35]	Model, AUI, CUI, FUI [34]	Platform [34]
MECANO	1995 [38]	1996 [38]	Model, Generat., Runtime [41]	Model, AUI, CUI [38][41]	n/a
MOBI-D	1997 [37]	1999 [40]	Model, Generat., Runtime [39]	Model, AUI, CUI [39]	n/a
SUPPLE	2004 [14]	2010 [15]	Model, Generation [14],[12], Runtime [15][13]	Model, AUI, CUI [14]	Device [14] User [14],[17]

Details on utilized models and supported target platforms and program languages are provided in Table 3.

Table 3. MB-UIDE models and supported platforms and languages

MB-UIDE	Models	Model notations	Target Platforms	Target Languages
AME	Application, Domain, Kappa PC UI Prototype [26]	OOA, OOD [26]	desktop [26]	C++, KAL [26]
ITS	Domain [46], Dialog [2]	Proprietary [2],[45]	desktop [2],[45],[46]	ITS runtime environment [45]
TERESA	Task, Domain, System Task [28], Interaction [31]	Task: CTT [28], AUI, CUI: TERESA XML [31]	graphical desktop, vocal, cellphone, graphical & vocal, graphical & gestural, digital TV [31]	XForms [27,] XHTML MP, VoiceXML, X+V, SVG, Xlet, Gesture Library for MS [31]

Table 3. (*continued*)

MARIAE	Data, Event, Dialog, Task [34],[35] Transformation [36]	Data: XSD, Task: CTT [35], Transformation: XSLT [34], AUI, CUI: MARIA XML [33]	graphical form-based, graphical mobile form-based, vocal, digital TV, graphical direct manipulation, multimodal desktop / mobile, advanced mobile [34]	XHTML, Java [36]
MECANO	Domain, Interface [38][41]	MIM, MIMIC [38][41]	desktop [38][41]	MECANO runtime environment [41]
MOBI-D	User, User-task, Domain, Presentation, Dialog [40]	MIM, MIMIC [37]	desktop [37][40]	MOBI-D runtime environment [37]
SUPPLE	Interface, Device, User, Data [14], Cost [12], Preference, Ability [18]	Proprietary [14],[12]	mobile phone, touch screen devices [14], desktop computer [12]	SUPPLE runtime environment [14],[12]

Information on comprised tools, availability of the MB-UIDEs, and application examples are summarized in Table 4.

Table 4. MB-UIDE tools, availability, and application examples

MB-UIDE	Tools	Availability	Application Examples
AME	TRANTOOL, OODevelopTool, ODE Editor [26]	Prototype [24]	Small office applications for evaluation purposes
ITS	Dialog Compiler, Style Compiler [2]	Existing framework [45],[46]	Visitor Information System EXPO 1992 [45],[46]
TERESA	CTTE, Editors and Generators [31]	Version 3.4 available at http://giove.isti.cnr.it/tools/TERESA/download	Museum Application [27]
MARIAE	Transformation Editor, Tasks-Services Binding Editor, UI Editor (AUI, CUI), FUI Preview [36]	Version 1.5.6 available at http://giove.ist.cnr.it/tools/MARIAE/download	Pac-Man game [34], Home control application [35], Sales order management [33], DVD management application [32]
MECANO	MOBI-D [37]	Exist. framework [41]	Ship protection system [38]
MOBI-D	TIMM [40], U-TEL [39]	Exist. framework [40]	Logistic example [40]
SUPPLE	ARNAULD [16][18] [13], Activity Modeler [16][18]	Exist. framework [14]	FTP client, Classroom equipment controller [14], Email client, Amazon Web Service interface [12]

5 Conclusion

All the MB-UIDEs considered in our detailed literature review can be regarded as valuable contributions to model-based user interface design and development. However, none of the approaches makes use of a combination of model-based and pattern-based development methods comparable to PaMGIS.

AME integrates an object-oriented software development process with user interface modeling and design and employs OOA and OOD models without requiring an explicit task model. AME aims at desktop computers as target platform.

Compared to PaMGIS, MECANO and MOBI-D use a similar set of models as basis for UI generation. Like AME, the target platform is desktop computer.

SUPPLE and SUPPLE++ start with a data model and treat UI generation as decision-theoretic optimization problem. On the whole we regard this as a very interesting approach, but too different to the current PaMGIS proceeding.

With regard to the further development of our PaMGIS framework, we intend to inspect ITS and MARIAE in more detail. On one hand this decision is based on the fact that these two MB-UIDEs provide solutions for the features we are looking for. On the other hand the pattern-based part of PaMGIS strongly resembles MARIAE in terms of its accordance with the CRF abstraction levels, types of utilized models, and model exploitation. In addition, MARIAE development is still ongoing and the current version is even available on the Internet and allows for practical exertion.

References

1. Balzert, H., et al.: The JANUS Application Development Environment - Generating More than the User Interface. In: Computer-Aided Design of User Interfaces, pp. 183–206. Namur University Press (1996)
2. Bennett, W.E., et al.: Transformations on a Dialog Tree: Rule-Based Mapping of Content to Style. In: Proceedings of the ACM SIGGRAPH Symposium on User Interface Software and Technology, Williamsburg, Virginia, USA (1989)
3. Berti, S., et al.: TERESA: A Transformation-based Environment for Designing and Developing Multi-Device Interfaces. In:Proceedings of ACM CHI 2004 (Vienna, April 2004), vol. II, pp. 793–794. ACM Press (2004)
4. Berti, S., Mori, G., Paternò, F., Santoro, C.: TERESA: An Environment for Designing Multi-Device Interactive Services (2005), http://giove.isti.cnr.it/attachments/publications/2005-A2-80.pdf (last Website call on January 25, 2014)
5. Bodart, F., et al.: Towards a Systematic Building of Software Architectures: the TRIDENT Methodological Guide. In: Design, Specification and Verification of Interactive Systems, pp. 262–278. Springer (1995)
6. Calvary, G., et al.: The CAMELEON Reference Framework. Document D1.1 of the CAMELEON R&D Project IST-2000-30104 (2002)
7. da Silva, P.P.: User interface declarative models and development environments: A survey. In: Palanque, P., Paternó, F. (eds.) DSV-IS 2000. LNCS, vol. 1946, pp. 207–226. Springer, Heidelberg (2001)

8. Elwert, T., Schlungbaum, E.: Modelling and Generation of Graphical User Interfaces in the TADEUS Approach. In: Designing, Specification and Verification of Interactive Systems, pp. 193–208. Springer (1995)
9. Engel, J., Märtin, C.: PaMGIS: A Framework for Pattern-Based Modeling and Generation of Interactive Systems. In: Jacko, J.A. (ed.) Human-Computer Interaction, Part I, HCII 2013. LNCS, vol. 5610, pp. 826–835. Springer, Heidelberg (2009)
10. Engel, J., Märtin, C., Herdin, C., Forbrig, P.: Formal Pattern Specifications to Facilitate Semi-automated User Interface Generation. In: Kurosu, M. (ed.) HCII/HCI 2013, Part I. LNCS, vol. 8004, pp. 300–309. Springer, Heidelberg (2013)
11. Foley, J., et al.: The User Interface Design Environment - A Computer Aided Software Engineering Tool for the User Computer Interface. IEEE Software 6, 25–32 (1989)
12. Gajos, K., Christianson, D., Hoffmann, R., Shaked, T., Henning, K., Long, J.J., Weld, D.S.: Fast and Robust Interface Generation for Ubiquitous Applications. In: Beigl, M., Intille, S.S., Rekimoto, J., Tokuda, H. (eds.) UbiComp 2005. LNCS, vol. 3660, pp. 37–55. Springer, Heidelberg (2005)
13. Gajos, K., Weld, D.: Preference Elicitation for Interface Optimization. In: UIST 2005: Proceedings of the 18th Annual ACM Symposium on User Interface Software and Technology, New York, USA (2005)
14. Gajos, K., Weld, D.S.: SUPPLE: Automatically Generating User Interfaces. In: Proceedings of the 9th International Conference on Intelligent User Interfaces, pp. 93–100 (2004)
15. Gajos, K., Weld, D., Wobbrock, J.: Automatically Generating Personalized User Interfaces with SUPPLE. Artificial Intelligence 174, 910–950 (2010)
16. Gajos, K., Weld, S., Wobbrock, J.: Decision-Theoretic User Interface Generation. In: AAAI 2008, pp. 1532–1536. AAAI Press (2008)
17. Gajos, K., Wobbrock, J., Weld, D.: Automatically Generating User Interfaces Adapted to Users' Motor and Vision Capabilities. In: UIST 2007: Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology, New Port, Rhode Island, USA (2007)
18. Gajos, K., Wobbrock, J., Weld, D.: Improving the Performance of Motor-Impaired Users with Automaticalls-generated, Ability-Based Interfaces. In: CHI 2008: Proceeding of the 26th Annual SIGCHI Conference on Human Factors in Computing Systems, New York, USA (2008)
19. Griffiths, T., et al.: Teallach: A Model-Based User Interface Development Environment for Object Databases. In: Proceedings of UIDIS 1999, pp. 86–96. IEEE Press (1999)
20. ISTI: MARIA Fact Sheet (2011), `http://giove.isti.cnr.it/tools/MARIA/MARIA%20Fact%20Sheet.pdf` (last Website call on January 25, 2014]
21. Janssen, C., Weisbecker, A., Ziegler, J.: Generating User Interfaces from Data Models and Dialogue Net Specifications. In: Proceedings of Inter CHI 1993, pp. 418–423. ACM Press (1993)
22. Lonczewski, F., Schreiber, S.: The FUSE-System: an Integrated User Interface Desgin Environment. In: Computer-Aided Design of User Interfaces, pp. 37–56. Namur University Press (1996)
23. Markopoulos, P., Pycock, J., Wilson, S., Johnson, P.: Adept - A Task Based Design Environment. In: Proceedings of the 25th Hawaii International Conference on System Sciences, pp. 587–596. IEEE Computer Society Press (1992)
24. Märtin, C.: Model-Based Software Engineering for Interactive Systems. In: Systems: Theory and Practice. Advances in Computing Science Series, pp. 187–211. Springer, Heidelberg (1998)

25. Märtin, C.: Software Life Cycle Automation for Interactive Applications: The AME Design Environment. In: Computer-Aided Design of User Interfaces, pp. 57–74. Namur University Press (1996)
26. Märtin, C., Winterhalder, C.: Integrating CASE and UIMS for Automatic Software Construction. In: Proceedings of the 5th Int. Conference on Human-Computer Interaction - HCI International 1993, pp. 291–296. Elsevier (1993)
27. Mori, G., Paternò, F., Santoro, C.: Design and Development of Multidevice User Interfaces through Multiple Logical Descriptions. Journal IEEE Transactions on Software Engineering 30(8), 507–520 (2004)
28. Mori, G., Paternò, F., Santoro, C.: Tool Support for Designing Nomadic Applications. In: IUI 2003 Proceedings of the 8th International Conference on Intelligent User Interfaces, pp. 141–148. ACM (2003)
29. Myers, B.A.: State of the Art in User Interface Software Tools. In: Advances in Human-Computer Interaction, vol. 4. Ablex Publishing (1992)
30. Paternò, F.: The ConcurTaskTrees Notation. In: Model-Based Design and Evaluation of Interactive Applications, pp. 39–66. Springer, Heidelberg (2000)
31. Paternò, F., et al.: Authoring Pervasive Multimodal user Interfaces. International Jounal of Web Engineering and Technology 4(2), 235–261 (2008)
32. Paternò, F., Santoro, C., Spano, L.D.: Engineering the Authoring of Usable Service Front Ends. The Journal of Systems and Software 84, 1806–1822 (2011)
33. Paternò, F., Santoro, C., Spano, L.D.: Exploiting Web Service Annotations in Model-based User Interface Development. In: Proceedings of EICS 2010 - 2nd ACM SIGCHI Symposium on Engineering Interactive Computing Systems, pp. 219–224. ACM (2010)
34. Paternò, F., Santoro, C., Spano, L.D.: MARIA: A Universal, Declarative, Multiple Abstraction-Level Language for Service-Oriented Applications in Ubiquitous Environments. ACM Transactions on Human-Computer Interaction (2009)
35. Paternò, F., Santoro, C., Spano, L.D.: Model-Based Design of Multi-device Interactive Applications Based on Web Services. In: Gross, T., Gulliksen, J., Kotzé, P., Oestreicher, L., Palanque, P., Prates, R.O., Winckler, M. (eds.) INTERACT 2009 Part I. LNCS, vol. 5726, pp. 892–905. Springer, Heidelberg (2009)
36. Paternò, F., Santoro, C., Spano, L.D.: Support for Authoring Service Front-Ends. In: Proceedings of EICS 2009 - 1st ACM SIGCHI Symposium on Engineering Interactive Computing Systems, Pittsburgh, PA, USA (2009)
37. Puerta, A., Maulsby, D.: Management of Interface Design Knowledge with MODI-D. In: Proceedings of IUI 1997, Orlando, FL, pp. 249–252 (1997)
38. Puerta, A.: The Mecano Project: Comprehensive and Integrated Support for Model-Based Interface Development. In: Computer-Aided Design of User Interfaces, pp. 19–36. Namur University Press (1996)
39. Puerta, A., Eisenstein, J.: Interactively Mapping Task Models to Interfaces in MOBI-D. In: Proc. Eurographics Workshop on Design, Specification and Validation of Interactive Systems (DSV-IS 1998), pp. 261–273 (1998)
40. Puerta, A., Eisenstein, J.: Towards a general computational framework for model-based interface development systems. In: IUI 1999 Proceedings of the 4th International Conference on Intelligent User Interfaces, pp. 171–178. ACM, New York (1999)
41. Puerta, A., Eriksson, H., Gennari, J., Musen, M.: Beyond Data Models for Automated User Interface Generation. In: Proc. British HCI 1994, pp. 353–366. University Press (1994)
42. Schlungbaum, E.: Model-based User Interface Software Tools - Current State of Declarative Models. Graphics, Visualization and Usability Centre, Georgia Institute of Technology, GVU Tech Report (1996)

43. Szekely, P., Luo, P., Neches, R.: Facilitating the Exploration of Interface Design Alternatives: The HUMANOID Model of Interface Design. In: Proceedings of SIGCHI 1992, vol. 1992, pp. 507–515 (1992)
44. Szekely, P., et al.: Declarative Interface Models for User Interface Construction Tools: the MASTERMIND Approach. In: Engineering for Human-Computer Interaction, pp. 120–150. Chapman & Hall (1996)
45. Wiecha, C., et al.: ITS: A Tool for Rapidly Developing Interactive Applications. ACM Transactions on Information Systems 8(3), 204–236 (1990)
46. Wiecha, C., Boies, S.: Generating User Interfaces: Principles and Use of ITS Style Rules. In: Proceedings of UIST 1990 (1990)