Wireframe Mockups to ConcurTaskTrees

A WYSIWYG User Interface Modeling Approach

Miroslav Sili & Christopher Mayer
Health & Environment Department
AIT Austrian Institute of Technology GmbH
Vienna, Austria
e-mail: miroslav.sili@ait.ac.at, christopher.mayer@ait.ac.at

Daniel Pahr
Student at the TU Wien
Vienna, Austria
e-mail: e0906438@student.tuwien.ac.at

***Abstract*— Nowadays, we use a variety of devices to interact with local and cloud-based systems and services and are used to aesthetic and tailored user interfaces. This fact induces challenges for user interface designers regarding additional efforts for the development of multiple user interfaces for (all) available devices. Model-based user interface design tackles this challenge by creating abstract models for a transformation to various devices, but with the disadvantage of additional efforts for using and learning these techniques. Thus, we propose a transformation process deriving an abstract model on the basis of an integrated mockup and wireframe design tool. This allows combining the advantages of model-based user interface design with the world of the classical user interface design process. The engine transfers sketches to an abstract task model in ConcurTaskTrees notation. The model is the input for a separate model interpretation layer generating concrete user interfaces for various device types. The prototype delivers promising results and future research has to focus on extending its applicability by addressing structural constraints and limitations.**

Keywords*— *Wireframe; Mockup; Sketches; User Interface Design; WYSIWYG; ConcurTaskTrees; CTT; UI Model; Adaptivity; Transformation; XSLT

I. INTRODUCTION

The last two decades have seen a growing trend towards mobile and ubiquitous computing. This trend changed the way of Human Computer Interaction (HCI). Nowadays, we use a variety of devices to interact with local and cloud-based systems and services. We also become accustomed to use aesthetic and tailored user interfaces (UI) on different device types. This variety provides some advantages for end-users, but also some significant disadvantages for UI designers. Additional efforts are needed to design and develop multiple UIs for different device types. Considering the number of potential UIs, the classical design approach is not appropriate anymore. At this point, the model-based UI design approach can help to save design resources in terms of time and money. Besides its advantages, it has also some disadvantages. To name a few: the design process is not intuitive enough, additional resources during the learning and training phase are needed and the continuous testing routines require additional efforts. The proposed mixed approach aims to minimize these negative aspects. The idea is to integrate mockup and wireframe design tools, which are common in the classical UI design process, into the model-based UI design process. This What You See Is What You Get (WYSIWYG) - like design helps designers to manifest their vision of concrete UIs without the need to spend too much efforts on specific model-based techniques and language notations.

The remainder of this paper is structured as follows: In Section 2, one finds a short overview of related work. Section 3 provides an introduction to the model-based UI design process, the concept of task models in ConcurTaskTrees (CTT) notation, the evaluation of Wireframe and Sketch-based tools as well as an overview of the chosen tool. Section 4 describes in detail the transformation engine, which is able to transform wireframe mockup output data into abstract UI CTT models. In Section 5, the results are summarized and discussed.

II. RELATED WORK

In the past, similar approaches for the generation of abstract user interface models have been proposed. Those approaches share the employment of WYSIWYG-like editors with our proposed solution and use different abstract models for the representation of the interface.

In [18], the concept of model-driven development is examined with the goal to propose a solution for the automatic generation of user-interfaces. To achieve this, CTT models are introduced into the model driven approach to capture interaction requirements of user interfaces. The proposed tool allows developers to create user interfaces by using sketch-based drawings. However, designers need to provide additional context to UI elements in order to identify them distinctly. This process enables the creation of verifiable, explicit CTT models for a user interface. Using those components a model compiler could automatically create the code for a platform specific user interface.

Gummy [6] uses a WYSIWYG user interface editor to produce an abstract representation in User Interface Markup Language (UIML) format [19]. It offers a live representation of multiple different user interfaces for different platforms while editing. UIML specifications of user interface are very similar to concrete user interface specifications as opposed to the highly abstract nature of CTT used in our proposed solution.

SketchiXML [20] offers an editor more focused on drawing and gesture based interaction than graphical user interfaces. A user can draw its prototype on a canvas and assign a context to the different parts of the sketch to create a user interface model, which in turn, can be viewed on multiple fidelity levels. SketchyXML supports UIML as the main export format but an export to UsiXML [21] is also possible.

 ISBN: 978-1-61208-505-0

III. MODEL-BASED USER INTERFACE DESIGN PROCESS

In contrast to the classical UI design process, model-based design divides the UI generation process into at least two steps. The first step is related to the modeling of an abstract UI, followed by two or more steps related to the generation of concrete UIs. The abstract and declarative model of the first step is composed of components like a user-task model, a user model, a dialog model, a presentation model and a domain model. These models provide a formal representation of the UI design [1]. In the second step, this formal representation can be automatically transformed into concrete UIs by using a separate model interpretation layer, e.g., AALuis [2]-[5]. This (at least) two-step approach offers the opportunity to specify the UI only once, which facilitates the process of changing and editing [6].

Model-based user interface design had its origin in the mid-late 1990s [7]-[9]. Researchers and developers have investigated different model-based techniques in order to structure and automate the user interface design process since then. Some approaches rely, e.g., on State chart XML (SCXML) models [10][11], some on the Business Process Model and Notation (BPMN) models [12][13] and some on CTT models [14][15].

In general, the model-based UI design cannot be declared as an easy and intuitive process. Designers need time to get familiar with the indirect design concept and to learn these model-based techniques and language notations. Furthermore, they also require additional time to test their UI models continuously and to compare the intended output with the automatically generated output. Our proposed transformation engine aims to minimize some of these challenges. Instead of using abstract design elements to model abstract UIs, designers may use existing sketch-based tools to design concrete UI representations. The transformation engine transforms these representations, into abstract UI models. These UI models, in turn, are used in the second phase as input for an automatic transformation into concrete UIs. On the occasion of our current running research and development project YouDo [16][17], which uses abstract UI models in CTT notation, the developed transformation engine also focuses on CTT notation. Nevertheless, the underlying architecture generally allows also transformations into other notations like SCXML notation or BPMN.

A. Interaction Models in CTT notation

This paragraph helps to comprehend the main transformation steps by providing a briefly overview about different CTT tasks and temporal operators used in the CTT notation. The CTT notation distinguishes between four task categories, namely interaction, system, user and abstract tasks [14]. Interaction tasks are related to concrete user interactions. These tasks are represented in the final UI, e.g., as control elements or text input elements. System tasks are responsible to receive data from the system and to provide information to the user. User tasks represent internal cognitive or physical activities and abstract tasks are used for complex actions, which need sub-tasks of different categories [23]. Next to these categories, tasks are also classified into different types. Just to mention some, interaction tasks may be of type control, edit or selection. Regarding the proposed transformation engine especially these task types are particularly relevant. As an example, an edit interaction tasks specifies an object which can be manually edited by the user. Depending on the concrete transformation such an edit interaction task may be represented, e.g., by a graphical text field. Next to different task categories and task types the CTT notation specifies also eight temporal operators. Temporal operators are able to describe the relationship between single tasks. [24] provides a detail explanation of the eight temporal operators.

B. Evaluation of Wireframe and Sketch-based Tools

Based on a detailed evaluation of existing wireframe and sketch-based tools for user interfaces design, we decide to use the Balsamiq mockup tool [22]. Our evaluation included in total 11 tools and the following set of evaluation criteria: a) the price, b) supported output formats, c) required learning efforts and the tool usability, d) supported platforms, d) available feature set and finally d) the project/tool activity in terms of development and maintenance status. Table I summarizes the list of evaluated tools and some of the mentioned evaluation criteria.

TABLE I. LISTING OF EVALUATED WIREFRAME AND SKETCH-BASED TOOLS AND SOME OF THE EVALUATION CRITERIA

Tool	Price[a]	Supported output formats	Supported platforms	Feature set[b]	Activity[b]
Gummy [26]	/	UIML	Windows	-	-
Glade Interface Designer [27]	/	Libglade, GtkBuilder	Linux, Windows, Mac	+	+
softandGUI [28]	£99	PNG, Word, HTML, XML	Windows	+	+
WireframeSketcher [29]	$99	PNG, PDF, HTML	Linux, Windows, Mac, Eclipse plugin	+	+
iPLOTZ [30]	$99/yr.	JPG, PNG, PDF, iPotz File, XML	Windows & Mac, Web	+	+
Evolus Pencil [31]	/	PNG, HTML, PDF, SVG, ODT	Linux, Windows, Mac, Web	+	+
Mockup-designer [32]	n/a	JSON, PNG	Web	-	n/a
Balsamiq Mockups [22]	$89	BMML (XML-Bas)	Windows, Mac, Web	+	+
Maqetta [33]	/	HTML	Linux, Windows, Mac,	+	-
draw.io [34]	/	XML	Web	+	+
Moqups [35]	19€/mo.	PDF, PNG	Web	+	+

a. Price for the single- user license. The slash symbol </> represents GPL, GPL2 or open source licenses. b. The plus symbol <+> represents positive evaluated criteria whereas the minus symbol <-> represents the opposite.

Balsamiq's easy to use export format Balsamiq Mockups Markup Language (BMML), the cross platform application, or the comprehensive feature list are just some of the positive criteria which influenced our decision. Balsamiq features a very simple, yet powerful, drag and drop-based mockup editor.

 ISBN: 978-1-61208-505-0

A navigation bar presents the user with multiple common user interface elements which can simply be dragged to the canvas. One can then arrange those elements to create a full sketch of a user interface and by using very few basic functions, even create linked user interface prototypes. Such prototypes normally serve to present the look and feel of a finished software product to test it on a specific group of users, or simply to try out the design for developers.

Balsamiq's BMMF file features an XML format. As such, a whole mockup can be viewed like a tree structure with a root node, which has child nodes representing the components of the mockups. Multiple attributes and property nodes can be assigned to the components to mirror the design of the mockup. Furthermore, a grouping function is available, which actually serves to create groups of interface elements and enables the user to move them as a whole. However, in the scope of transformation this function will be more of use to assign identifying objects to UI elements that are inherently without a concrete description, such as text boxes or check boxes. The linking of mockups is also resembled in a simple property node that can be attached to UI elements like buttons.

IV. WIREFRAME TO CTT TRANSFORMATION ENGINE

With Balsamiq mockup files, serving as the input for the transformation engine, the next step was to design mapping rules and to create a concrete transformations from Balsamiq' export file format BMML into the CTT format. Due to the fact that both file formats use an XML structure the Extensible Stylesheet Language Transformation (XSLT) [25] was used to carry out the transformation. XSLT requires a stylesheet template to establish transformation rules which are able to convert a document from one specific XML structure into another XML structure. The transformation itself is carried out by an XSLT processor. For this purpose, we implemented a small Java application.

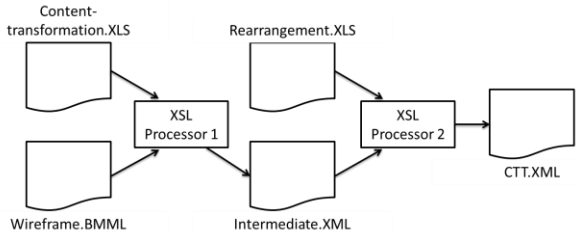

Fig. 1. Illustration of the two-step transformation from Balsamiqs' specific BMML file format into the UI model in CTT notation.

The two underlying XML structures involved in this transformation are fundamentally different. BMML represents the concrete graphic user interface, whereas CTT describes a task model representing a succession of interactions that need to be performed in order to achieve a certain goal. In order to extract sufficient semantic information from a user interface mockup and to limit the number of possible CTT interpretations, the following structural limitations have been introduced:

- Each mockup contains exactly one window, which represents a single static user interface.
- The transformation supports just user interfaces, which can be represented as a set of linked windows.
- The interface must have a designated start point, symbolized by the first window that is presented to the user.
- Each window expresses a so called Presentation Task Set (PTS). A PTS is a set of user interactions, respectively tasks, needed to perform one distinct action. Example: to perform a login action the user needs to fill in the username, the password and to operate the login button within a single window.
- The user input order within a single window is irrelevant.
- One window has at least one control button, which ends users' interaction on that window and forces the system to start the processing of the entered data.
- The system outcome, in turn, is represented by a single window.

Furthermore, due to its complexity the transformation itself takes place in two separate steps: (Step 1) UI content transformation into CTT tasks and their grouping and (Step 2) rearrangement of tasks and the generation of the intended interaction flow using CTTs temporal operators. Fig. 1 illustrates the two-step transformation approach.

1) Step 1 - UI Content Transformation: The UI content transformation consists of two substeps: (substep 1.1) the mapping of Balsamiq UI elements into CTT tasks and (substep 1.2) the grouping of CTT tasks into subtrees, which represent single PTS.

a) Substep 1.1: In this substep, UI elements are translated into semantically corresponding CTT tasks. Table II illustrates the basic set of implemented mapping rules. This approach is also applicable for more complex UI elements like tables, menu groups or street maps. The CTT notation offers the possibility to define interaction tasks of custom type. Thus, one can define an interaction tak e.g., of type table within the XLS transformation. Once implemented, every Balsamiq's table UI element will result in a corresponding table interaction task. Like for any other interaction tasks the CTT interpretation layer is responsible to render these custom interaction tasks accordingly. Additionally due to the possibility to link multiple mockups, a hierarchical structure can be extracted. In this step, single window elements are transformed into CTT subtrees and each UI element nested within the UI window is transformed into a single CTT task.

b) Substep 1.2: In this substep, CTT tasks are grouped into subtrees representing separate PTS. Additionally, each subtree is supplemented by an initial system task (required by the model interpretation layer). The following design patterns are used during the grouping process:

- Each subtree is supplemented by an initial system task. The initial system task is required by the model interpretation layer. The system task causes the layer to pull concrete data values from the backend system. These data values, if present, are used by the layer to perform an auto fill in on the rendered UI elements.
- Each subtree contains one or more interaction tasks of a type unlike control. These interaction tasks are gained from the substep 1.1 using the window UI element and its child UI elements.

 ISBN: 978-1-61208-505-0

- Each subtree contains one or more abstract tasks gained from substep 1.1 using the button UI element. As described in Table II, a button UI element is transformed into a composition of the following three CTT tasks:
  - A single interaction task of type control. This interaction task causes the model interpretation layer to render the concrete final UI element, e.g., a graphical button.
  - A single system task. Like the initial system task, this task is required by the model interpretation layer. In contrast to the initial system task, which pulls data values, this task pushes user input values towards the backend system.
  - If present, an abstract task containing the next PTS in form of a separate subtree. This step applies the design patterns recursively.

2) Step 2 - Rearrangement of PTS tasks and generation of the interaction flow: As already mentioned, after the first step of the transformation, a semi-complete CTT model is produced. The second step consists as well of two substeps in order to complete the CTT model: (substep 2.1) the rearrangement of PTS tasks and (substep 2.2) the generation of the intended interaction flow using CTTs temporal operators.

TABLE II. MAPPING BETWEEN BASIC INTERACTION ELEMENTS IN BALSAMIQ AND CTT TASKS

Balsamiq	CTT	Note
Text field	Interaction task of type edit	
Combo box or Radio button	Interaction task of type single choice	Radio buttons belonging to the same choice need to be grouped together
Check box	Interaction task of type Multiple Choice	Check boxes belonging to the same choice need to be grouped together
Label	Interaction task without explicit task type	
Button	Subgroup of one interaction task of type control, one system task and if present, one abstract task containing the next PTS	Buttons represent the navigations through different PTS
Window	Abstract task containing the current PTS	The title attribute is used to name the abstract task

a) Substep 2.1: Using the complete set of grouped CTT tasks produced in substep 1.2, this substep is responsible to rearrange all tasks within every subtree. The following three ordering rules are used during this process (see for an example Fig. 3):

- Firstly, the initial system task is the first task in the subtree.
- Secondly, all interaction tasks with a type different than "control" follow the initial system task.
- Thirdly, all interaction task of type "control" are arranged lastly in the subtree.

b) Substep 2.2: This is the last processing step performed by the transformation engine. This substep generates the intended interaction flow using CTTs temporal operators. The following rules are applied (see for an example Fig. 3):

- The initial system task is connected to the first interaction tasks with the sequential enabling information processing operator.
- Due to the fact that user inputs do not rely on a specific input order (compare to limitations introduced above), all following interaction tasks with a type different than "control" are connected to the order independence operator.
- The last interaction task with a type different than "control" is connected to the first abstract task (resulting from the UI button element) to the disabling operator.
- Possible following abstract tasks (resulting from the UI button element) are connected to the choice operator.

V. RESULTS

Fig. 2 illustrates an example for a sequence of two simple mockups. The link between the two windows is symbolized by the arrow, leading from the Login-button to the Welcome window.

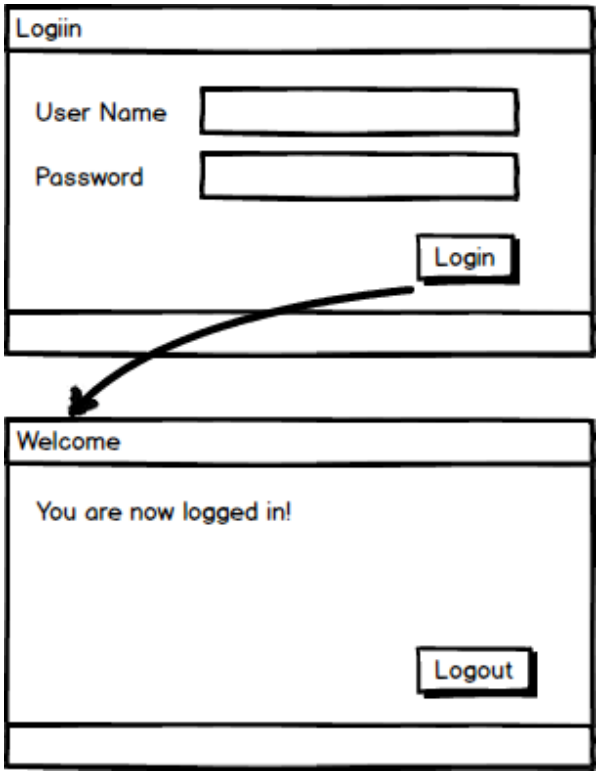

Fig. 2. Illustration of sequence of two simple mockups designed by means of the Balsamiq Mockup Tool

In Fig. 3, one can see the output of the transformation engine when processing the mockups from Fig. 2. Moreover, Fig. 3 provides a comparison of the intermediate semi-complete tree, obtained from the first transformation step and the final task tree, obtained from the second transformation

 ISBN: 978-1-61208-505-0

step. The intermediate tree features no temporal operators and the order of the components is not correct. The mockup UI windows “Login” and “Welcome” have been transformed into the corresponding abstract tasks named “Login” and “Welcome”. The UI input elements “User Name” and “Password” have been translated to corresponding interaction tasks of type edit. The UI button “Login” has been transformed into an abstract task named “Control_Group_Login”. This, in turn, is composed by an interaction task of type control named “Button_Login”, a system task named “Finalization_Login” and finally the already transformed abstract task named “Welcome”, which represents the second mockup window. In the final tree, the CTT is complete and all temporal operators have been placed between tasks.

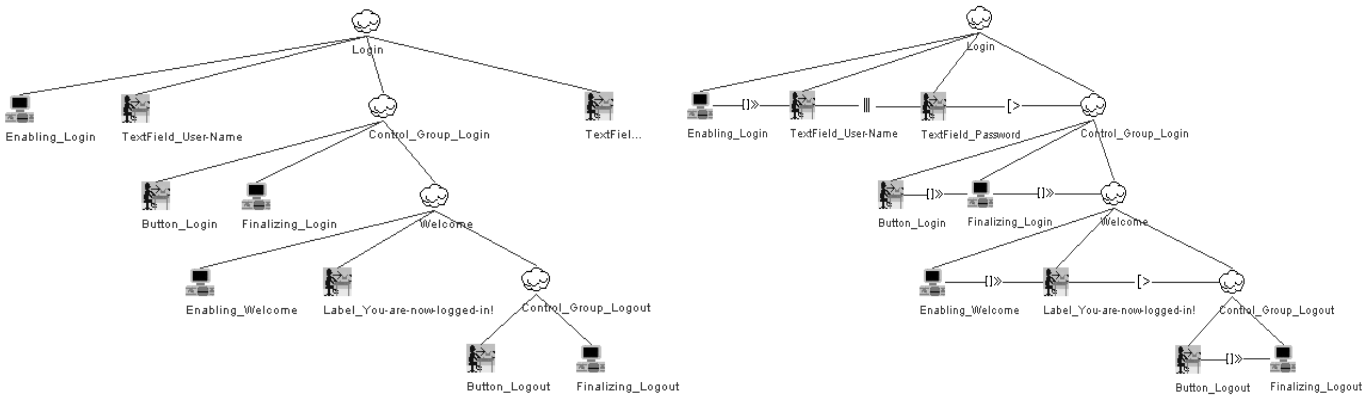

Fig. 3. Illustration of the intermediate semi-complete CTT tree (left) and the complete CTT (right).

The prototype for the proposed transformation tool delivers promising results for the use in combination with an appropriate CTT model interpretation layer, e.g., the AALuis layer. The transformation can be applied to any sequence of mockups following the specified conventions specified. In the moment however, the application of the tool is limited to sequential links that do not contain loops. Such a loop is created when a mockup links to another that has previously appeared on the same path. In future work, the transformations have to be modified to support such interfaces. An idea for further research is the development of a direct presentation of the final user interface, when developing the mockups, thus not only displaying the CTT model as the abstract user interface representation, but as well the transformed user interface.

VI. LIMITATIONS

As mentioned initially, in the model-based UI design process developers are requested to investigate some efforts and practice in order to benefit from model-based generated UIs. The aim of this work is to illustrate one potential solution to minimize these efforts and thus to increase the acceptance of model-based UI development. In our literature review we have been focused on similar practicable and easy to use state-of-the-art solutions and approaches. However, the scope of this work is not on an exhausting listing and detailed comparison of these solutions and approaches, but rather on the description of the proposed solution from a technical and methodological point of view. Moreover, this work tackles the generation of interaction models in CTT notation but not the concrete interpretation of these models. Interpretations concepts have been published previously [2]-[5]. The proposed solution is still under development and requires a detailed validation and evaluation in the scope of performance, acceptance and usability.

ACKNOWLEDGMENT

The project YouDo is co-funded by the AAL Joint Programme (REF. AAL-2012-5-155) and the following National Authorities and R&D programs in Austria, Germany and Switzerland: BMVIT, program benefit, FFG (AT), BMBF (DE) and SERI (CH).

REFERENCES

[1] A. R. Puerta, “A model-based interface development environment,” IEEE Software 14.4, pp. 40-47, 1997, doi: 10.1109/52.595902

[2] C. Mayer, M. Morandell, M. Gira, M. Sili, M. Petzold, S. Fagel, S. Schmehl, “User interfaces for older adults,” Universal Access in Human-Computer Interaction User and Context Diversity, vol. 8010, Springer Berlin Heidelberg, Jul. 2013, pp. 142-150, doi: 10.1007/978-3-642-39191-0_16

[3] M. Sili, C. Mayer, M. Morandell, M. Gira, M. Petzold, “A Practical Solution for the Automatic Generation of User Interfaces–What Are the Benefits of a Practical Solution for the Automatic Generation of User Interfaces?” Human-Computer Interaction. Theories, Methods, and Tools, vol. 8510, pp. 445-456, Jun. 2014, doi: 10.1007/978-3-319-07233-3_41

[4] C. Mayer, G. Zimmermann, A. Grguric, J. Alexandersson, M. Sili, C. Strobbe, “A comparative study of systems for the design of flexible user interfaces,” Journal of Ambient Intelligence and Smart Environments, vol. 8, pp. 125-148, Mar. 2016, doi: 10.3233/AIS-160370

[5] C. Mayer, M. Morandell, M. Gira, K. Hackbarth, M. Petzold, S. Fagel, “AALuis, a User Inter-face Layer That Brings Device Independence to Users of AAL Systems," Computers Helping People with Special Needs, vol. 7382, pp. 650-657, Jul. 2012, doi: 10.1007/978-3-642-31522-0_98

[6] J. Meskens, J. Vermeulen, K. Luyten, K. Coninx,”Gummy for multi-platform user interface designs: shape me, multiply me, fix me, use me,” Proceedings of the working conference on Advanced visual interfaces, pp. 233-240, 2008, doi: 10.1145/1385569.1385607

[7] C. Janssen, A. Weisbecker, J. Ziegler, “Generating user interfaces from data models and dialogue net specifications,” Proceedings of the INTERACT'93 and CHI'93 conference on human factors in computing systems, pp. 418-423, 1993, doi: 10.1145/169059.169335

 ISBN: 978-1-61208-505-0

[8] A. R. Puerta, H. Eriksson, J. H. Gennari, M. A. Musen, "Model-based automated generation of user interfaces, " AAAI, pp. 471-477, Okt. 1994,

[9] A. R. Puerta, J. Eisenstein, "Towards a general computational framework for model-based interface development systems," Knowledge-Based Systems, vol. 12, pp. 433-442, 1999

[10] J. Barnett, et al. "State chart XML (SCXML): State machine notation for control abstraction." W3C Recommendation, [Online] Available from: https://www.w3.org/TR/scxml retrieved: Jul. 2016

[11] A. Nuno, S. Samuel, T. Antonio, "Multimodal Multi-Device Application Supported by an SCXML State Chart Machine," Proceedings of EICS Workshop on Engineering Interactive Systems with SCXML, 2014

[12] M., Zur Muehlen, J. Recker, "How much language is enough? Theoretical and practical use of the business process modeling notation," InInternational Conference on Advanced Information Systems Engineering, pp. 465-479, Jun. 2008, doi: 10.1007/978-3-540-69534-9_35

[13] H. Trætteberg, "UI design without a task modeling language–using BPMN and Diamodl for task modeling and dialog design," Engineering Interactive Systems, pp. 110-117, 2008, doi: 10.1007/978-3-540-85992-5_9

[14] F. Paternò, C. Mancini, S. Meniconi, "ConcurTaskTrees: A diagrammatic notation for specifying task models," Human-Computer Interaction INTERACT'97, pp. 362-369, 1997, doi: 10.1007/978-0-387-35175-9_58

[15] F. Paternò, C. Mancini, "Developing task models from informal scenarios," CHI'99 Extended Abstracts on Human Factors in Computing Systems, pp. 228-229, 1999, doi: 10.1145/632716.632858

[16] M. Sili, D. Bolliger, J. Morak, M. Gira, K. Wessig, D. Brunmeir, H. Tellioğlu, "YouDo-we help! - An Open Information and Training Platform for Informal Caregivers," Studies in health technology and informatics, vol. 217, pp. 873-877, 2014, doi: 10.3233/978-1-61499-566-1-873

[17] YouDo – we help! [Online] Available from: http://youdoproject.eu Retrieved: Jul. 2016

[18] O. Pastor, S. España, J. I. Panach, N. Aquino, "Model-driven development," Informatik-Spektrum, vol. 31(5), pp. 394-407, 2008, doi: 10.1007/s00287-008-0275-8

[19] M. Abrams, C. Phanouriou, A. L. Batongbacal, S. M. Williams, J. E. Shuster, "UIML: an appliance-independent XML user interface language," Computer Networks, vol. 31(11), pp. 1695-1708, 1999, doi: 10.1016/S1389-1286(99)00044-4

[20] A. Coyette, S. Kieffer, J. Vanderdonckt, "Multi-fidelity prototyping of user interfaces," IFIP Conference on Human-Computer Interaction, pp. 150-164, 2007, doi: 10.1007/978-3-540-74796-3_16

[21] Q. Limbourg, J. Vanderdonckt, B. Michotte, L. Bouillon, V. López-Jaquero, "USIXML: a language supporting multi-path development of user interfaces," International Workshop on Design, Specification, and Verification of Interactive Systems, pp. 200-220, Jul. 2007, doi: 10.1007/11431879_12

[22] Balsmaiq. Rapid, effective and fun wireframing software. | Balsamiq [Online] Available from: https://balsamiq.com Retrieved: Jul. 2016

[23] M. Sili, C. Mayer, M. Morandell, M. Petzold, "A framework for the automatic adaptation of user interfaces," Assistive Technology Research Series, vol. 33, pp. 1298-1303, 2013, doi: 10.3233/978-1-61499-304-9-1298

[24] Concur Task Trees (CTT), W3C Working Group Submission 2 February [Online] Available from http://www.w3.org/2012/02/ctt Retrieved: Jul. 2016

[25] XSL Transformations (XSLT) Version 1.0, W3C Recommendation 16 November 1999, [Online] Available from: http://www.w3.org/TR/xslt Retrieved: Jul. 2016

[26] Gummy-live [Online] Available from: http://research.edm.uhasselt.be/~gummy Retrieved: Aug. 2016

[27] Glade – A User interface Designer [Online] Available from: https://glade.gnome.org Retrieved: Aug. 2016

[28] Wireframing Tools, Application Prototyping, softandGUI - UXToolbox [Online] Available from: http://www.softandgui.co.uk Retrieved: Aug. 2016

[29] Wireframing Tool for Professionals – WireframeSketcher [Online] Available from: http://wireframesketcher.com Retrieved: Aug. 2016

[30] iPotz: wireframing, mockups and prototyping for websites and applications [Online] Available from: http://pencil.evolus.vn Retrieved: Aug. 2016

[31] Home – Pencil Project [Online] Available from: http://pencil.evolus.vn Retrieved: Aug. 2016

[32] Mockup Designer [Online] Available from: http://fatiherikli.github.io/mockup-designer Retrieved: Aug. 2016

[33] Maquetta [Online] Available from: http://maqetta.org Retrieved: Aug. 2016

[34] Flowchart Maker & Online Diagram Software [Online] Available from: https://www.draw.io Retrieved: Aug. 2016

[35] Onine Mockup, Wireframe & UI Prototyping Tool – Moqups [Online] Available from: https://app.moqups.com Retrieved: Aug. 2016

 ISBN: 978-1-61208-505-0