A MULTI-LAYER APPROACH OF VIEW MODELS DESIGNING

Olena Chebanyuk, Olexandr Palahin

Abstract: *Interface prototyping is a progressive approach of requirement analysis. Interface models allow setting a dialog between stakeholders and customer. Involving this approach to model-driven development will get a chance to precise initial development information, namely CIM models, by means of tracing them to interface components. From the other hand, defining components events allows to get a set of initial information about architectural solutions designing.*

Paper proposes an approach of interface designing based on matching data flow parts to events of view model components. Paper contains case study.

Keywords*: BPMN diagram, Use Case, User Interface, MockFlow.*

ACM Classification Keywords*: D.2 Software Engineering; D.2.2 Design Tools and Techniques*

Introduction

One of the reasons of growing development costs is necessity to design the same interface for range of devices, technology platforms and communication channels.

This leads to the fact that front-end development is an expensive and inefficient process. Manual designing of user interface components with the aim of the reuse is difficult task.

In order to solve this task component-based engineering is involved. Central idea of component-based engineering is implementing reuse procedure in the level of component, scaling software system.

The idea of software components has been present since the very beginnings of software engineering. It was inspired by other areas of engineering, such as electronics, in which systems are assembled from pre-existing, standardized parts – components [Stefan P, 2014].

Component-based engineering brings scalability since systems are defined as compositions of components that may in turn be further composed to form hierarchical system descriptions. Furthermore, components can be reused across many systems to minimize development efforts, thereby reducing project schedules and budgets.

To correctly identify composite components for view, it is needed to make the prototype of program's interface and correctly identify the use case.

For this purpose, it will be most expedient to use Interaction Flow Modeling Language (IFML, 2016).

IFML is a standardized modeling language in the field of software engineering. IFML includes a set of graphic notations to create visual models of user interactions and front-end behavior in software systems (IFML, 2016).

The Interaction Flow Modeling Language was developed in 2012 and 2013 under the lead of WebRatio (Acerbis, R. et. al., 2004) and was inspired by the WebML (Moreno N. et. el., 2007) notation, as well as by a few other experiences in the Web modeling field.

It was adopted as a standard by the Object Management Group (OMG) in March 2013 (IFML, 2016).

IFML supports the platform independent description of graphical user interfaces for applications accessed or deployed on such systems as desktop computers, laptop computers, PDAs, mobile phones, and tablets. The focus of the description is on the structure and behavior of the application as perceived by the end user (Marco Brambilla and Pietro Fraternali, 2015).

Related papers

Paper [Lu, Xudong, and Jiancheng Wan., 2007] proposes a model driven development approach of complex user interface designing. The approach captures the process data in user interfaces. User interfaces depicted as objects, components and their cooperative relations in an Interaction Model. These objects are merged with the aim to compose a view model for common user interface. Such models are composed from the different types of UI templates.

There are many researches reflecting achievements in model-based user interface approaches. They are Drive (Mitchell K. et. al, 1996), MOBI-D (Angel Puerta and Jacob Eisenstein, 1999), Wisdom (Nunes, N.J. at. el., 2005), Teresa (Mori G. et. al., 2004), Teallach (Griffiths, Tonya, Barclay and Peter J, Teallach, 2001), JUST-UI (Pedro J. Molina et. al., 2002), SUPPLE (Krzysztof Gajos and Daniel S. Weld, 2004), etc. Many approaches for UI design and model-based user interface development environments (MB-UIDEs) have been proposed. Such approach do not consider totally complexity of UI controls as grid, graph and tree, sharing presentation space that can overlapping many present units so can show different content in different context, and their operational relations [Lu, Xudong, and Jiancheng Wan., 2007].

Paper (Karagkasidis A., 2008) proposes the pattern-based approach defining user-interface in modern architectural design patterns as MVC and HMVC (Hieratical Model-View-Controller) design patterns. Authors consider a problem of defining the best component for implementing the user scenario pointing that one task can be solved by different user interface components that have different relations to business logic and controller layers. They describe such problem by example: “Let’s consider a typical usage scenario of a graphical user interface (GUI) application: A user wants to accomplish some (business) task. He/she selects the corresponding menu item, and an input dialog is displayed. The user types some data in and submits the dialog. The data is processed by the application, and results are displayed to the user (Karagkasidis A., 2008).

In order to catch events of User Interface components the Observer pattern is used. The views and controllers register themselves with the model. Upon state changes, the model notifies all the registered components, which then retrieve the required data from the model. The input data is passed to the application logic objects and processed there. The results are then returned to the GUI part, and the visual state of the presentation objects is updated.

Library of components gathers user interface components in the widget-level allowing to select a widget, namely, a view is a visual component from the toolkit. More precisely, each toolkit widget in the GUI is considered as a view and controller levels in an MVC-based framework.

In a large application, with dozens of usage scenarios, the system design may contain hundreds of classes with intensive interaction among objects at runtime. The more features a GUI application provides, the tighter the relationships among objects are. This makes the developing of GUI applications a rather difficult task (Karagkasidis A., 2008).

There are researches devoted designing of user interface and matching functionality to its components from the other side, namely, paying attention to composition of user interface components. Interface itself is designed by using XML techniques of user screen representation, combining XML mark-up ((Lepreux, S. and Vanderdonckt, J., 2007), the ServFace project (Patern` O. F. et. al., 2011), Alias (Joffroy, C. et. al., 2011) and Transparent Interface (Ginzburg, J. et. al., 2007).) or reuse interface widgets from companies repositories (Compose (Gabillon Y. et. al., 2011), COTS-UI (Criado, J. et. al., 2010), CRUISe (Pietschmann, S. at. el., 2009), WinCuts (Tan, D. S. at. el., 2004) and on-the-fly mashup composition (Zhao, Q. at. el, 2008)). Functionality of user interface components is based on supporting collaboration of services using specific collaboration patterns.

To contribute to better support the composition process, it was propose a new composition model and a prototype of a component assembler, the so-called OntoCompo, which implements the model. The model describes applications in terms of Task, UI and software components. The prototype allows a composition mainly driven by the direct manipulation of UI elements, the other

components being hidden, but still being linked to the UI elements (Christian Brel et. al, 2014)

The approach is applied to the application composition driven by UI manipulation. Thus, starting from a selected part of the UI, corresponding software components are identified. The connections between models are exploited in a process of selection, composition by substitution and layout reorganization.

In the paper (Christian Brel et. al, 2014) were presented approach of application composition based on three application descriptions or models, namely the UI, Task and Software Components descriptions or models. OntoCompo is the prototype applying approach to an application composition driven by manipulation of UI graphical elements only.

The user testing it was performed with OntoCompo highlighted that this restricted manipulation was not enough to achieve an appropriate composition. The most realistic of our working hypotheses was the weak hypothesis, namely: to perform their composition task, developers need to manipulate the three kinds of models together, but to different degrees; developers must have the control of the three models; they need to visualize and manipulate these models when needed.

However, to achieve the composition, developers did not need to manipulate the code of the resulting application.

Proposed approach

1) Prepare detailed requirement specification.
2) Design a Use Case diagrams.
3) Transform them into the BPMN diagrams.
4) Match constituents of BPMN diagrams with mock-up of UI components.
5) Define events of UI components and match them for code.
6) Organize UI components into composite ones.
7) Trace containers and events on BPMN diagram.

Case study

It is proposed to design an interface with script libraries supporting functionalities of analyzing content obtained from different segments of search content. Requirement specification for the designed application is given below.

Req. code	Req. description
F1	Support search from different google regional sub-domains
F1.1	Support search from Bulgarian area
F 1.2	Support search in Dutch area
F 1.3	Support search from Espanol area
F2	Contain lists of references for content in different sub-domains that are pointing to different content
F3	Possibility to organize search history and calculate common numbers of references that are the same for searching by specific keywords in different sub-domains
F4	Allow users to choose a background

The result of specification analysis is BPMN diagram designed after Use Case diagram transformation. This diagram is represented in the Figure 1.

Activities of this diagram are matched to components of interface for creating user screen allowing solving task of analysis internet content in different sub-domains. The interface of the proposed specification is represented on the Figure 2. It is created by means of MockFlow online utility. Every component has a set of standard events according to MVC architectural style.

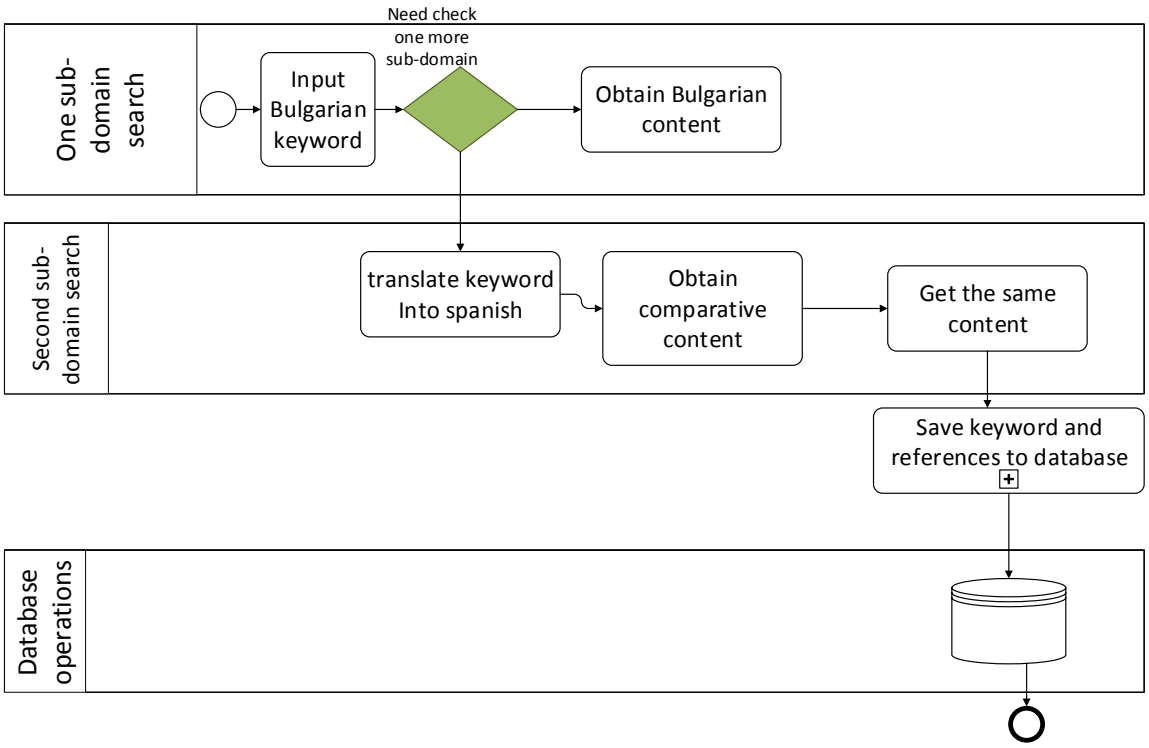

Figure 1. BPMN diagram of the process “Analysis of sub-domain content”

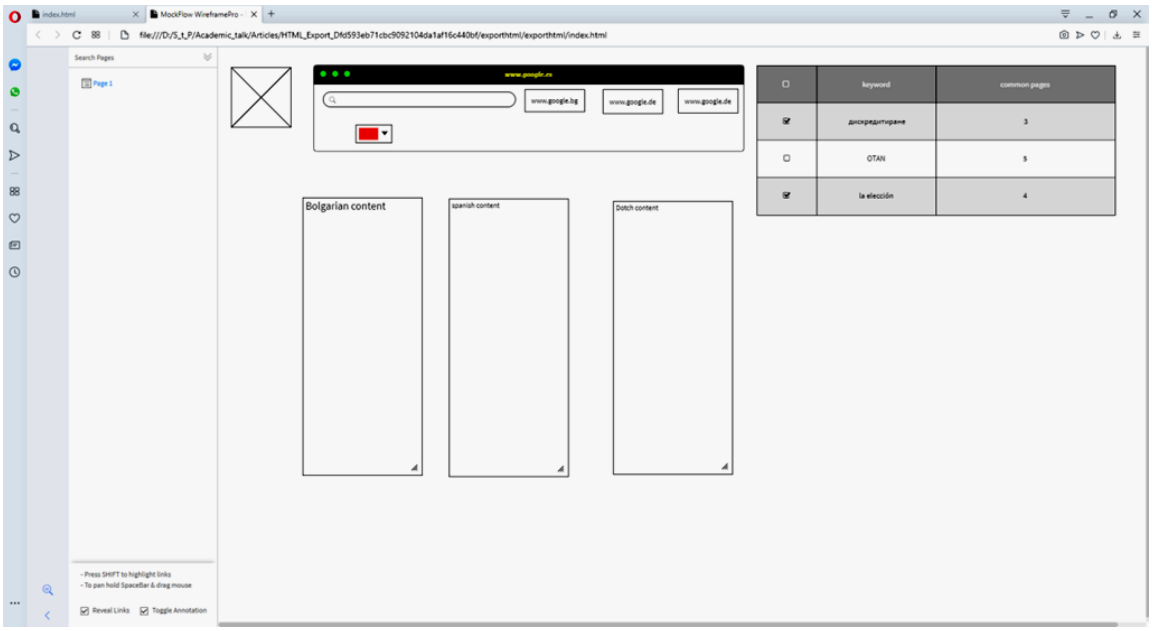

Figure 2. Designed interface on the basic of BPMN diagram

Designed *.html of the created interface is represented below.

```
<!DOCTYPE html>
	<head>

	<base href="">

	<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
	<meta name="google" content="notranslate" />
	<title>MockFlow WireframePro - Viewer</title>

<!-- external css files -->
<link
href="https://wireframepro.mockflow.com/css/external/bootstrap3/bootstrap-editor-min.css" rel="stylesheet" type="text/css" />
		<link
href="https://wireframepro.mockflow.com/css/external/bootstrap3/bootstrap-theme.css" rel="stylesheet" type="text/css" />

		<link href="https://wireframepro.mockflow.com/css/external/jquery-ui-1.11.4.custom.min.css" rel="stylesheet" type="text/css" />
		<link
href="https://wireframepro.mockflow.com/css/external/styles/kendo.common.min.css" rel="stylesheet" type="text/css" />
		<link
href="https://wireframepro.mockflow.com/css/external/styles/kendo.custom.metro.css" rel="stylesheet" type="text/css" />
		<link
href="https://wireframepro.mockflow.com/css/external/bootstrap-toggle-buttons.css" rel="stylesheet" type="text/css" />
		<link href="css/external/fontawesome/css/font-awesome.css"
rel="stylesheet" type="text/css" />

		<link
href="https://fonts.googleapis.com/css?family=Source+Sans+Pro:300,300italic,regular,italic,600,700,700italic" rel="stylesheet">

<link href="css/internal/icons/style.css" rel="stylesheet" type="text/css" />
		 <!-- internal css files -->
		<link href="css/internal/editor-min.css?v=2019_12_20_16_22_10"
rel="stylesheet" type="text/css">
```

```
		<!-- external js files -->
		<!-- mfexternal-min.js -->

		<script type="text/javascript" src="data/data.js"></script>
		<script type="text/javascript"
src="https://d20hhedk3h2l88.cloudfront.net/apps/wireframepro/external/js/webfont.js"></script>

<script
src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.4/jquery.min.js"></script>
<!-- <script type="text/javascript" src="js/external/jquery-2.1.4.min.js"></script>-->
		<script type="text/javascript"
src="https://d20hhedk3h2l88.cloudfront.net/apps/wireframepro/external/js/jquery-ui-1.11.4.custom.min.js"></script>
		<script type="text/javascript"
src="https://d20hhedk3h2l88.cloudfront.net/apps/wireframepro/external/js/jquery-migrate-1.2.1.min.js"></script>
		<script type="text/javascript"
src="https://d20hhedk3h2l88.cloudfront.net/apps/wireframepro/external/js/kendo.custom.min.js"></script>
		<script type="text/javascript"
src="https://d20hhedk3h2l88.cloudfront.net/apps/wireframepro/external/js/bootstrap3.min.js"></script>
		<script type="text/javascript"
src="https://d20hhedk3h2l88.cloudfront.net/apps/external/js/jquery.textchange.min.js"></script>
		<script type="text/javascript"
src="https://d20hhedk3h2l88.cloudfront.net/apps/external/js/jquery.event.drag-2.2.min.js"></script>
		<script type="text/javascript"
src="https://d20hhedk3h2l88.cloudfront.net/apps/external/js/arraycollection.min.js"></script>
		<script type="text/javascript"
src="https://d20hhedk3h2l88.cloudfront.net/apps/wireframepro/external/js/spin.min.js"></script>
		<script type="text/javascript"
src="https://d20hhedk3h2l88.cloudfront.net/apps/wireframepro/external/js/multiline.min.js"></script>
		<script type="text/javascript"
src="https://d20hhedk3h2l88.cloudfront.net/apps/wireframepro/external/js/jquery.address-1.5.min.js"></script>

		<!-- internal js files -->
		<script type="text/javascript" src="js/internal/mf_wireframepro-min.js?v=2019_12_20_16_22_10"></script>
```

Represented listing of an *.html shows that user Interface components functionality is provided by third party resources https://d20hhedk3h2l88.cloudfront.net/apps/wireframepro/external/js/

supporting JQuery libraries.

Conclusion

Modern user interfaces are highly dynamic and interactive. They often compose in various ways user interface components. Thus, there is a need to understand what can be composed in user interfaces and how.

The technologies for the development of UI evolve very rapidly like any other relatable field in web development. The way technologies are evolving in the front end frameworks has been changing the logic and methodologies used in the development process.

There are various numbers of technologies, environments and approaches for composite component designing. Each of these methods has advantages and disadvantages.

Implementation of the proposed approach will increase the development speed by means of shrinking time for testing and development. Other advantage of the proposed approach is the next: matching BPMN diagrams from previous and current project, one can reuse interface components that are combined into containers, and even whole user screens can be prepared for reuse.

Bibliography

(Acerbis, R. et. al., 2004) Acerbis, R., Bongio, A., Butti, S., Ceri, S., Ciapessoni, F., Conserva, C., ... & Carughi, G. T. Webratio, an innovative technology for web application development. In *International Conference on Web Engineering* (pp. 613-614). Springer, Berlin, Heidelberg. 2004, July.

(Criado, J. et. al., 2010) Criado, J., Padilla, N., Iribarne, L. and Asensio, J.-A., User interface composition with cots-ui and trading approaches: Application for web-based environmental information systems, in 'Knowledge Management, Information Systems, E-Learning, and Sustainability Research', Springer, pp. 259–266, 2010.

(Christian Brel et. al, 2014) Christian Brel, Philippe Renevier, Alain Giboin, Anne-Marie Dery, Michel Riveill. Reusing and Combining UI, Task and Software Component Models to Compose New Applications. BCS HCI, Sep 2014, Southport, United Kingdom. Electronic Workshops in Computing (eWiC) / British Computer Society, BCS-HCI'14 Proceedings of the 28th Annual BCS Interaction Specialist Group Conference on People and Computers.

(IFML, 2016) [electronic resource] / Wikipedia – Mode of access https://en.wikipedia.org/wiki/Interaction_Flow_Modeling_Language

(Ginzburg, J. et. al., 2007) Ginzburg, J., Rossi, G., Urbieta, M. and Distante, D., Transparent interface composition in web applications, in 'Proceedings of the 7th international conference on Web engineering', Springer-Verlag, pp. 152–166, 2007.

(Griffiths, Tonya, Barclay and Peter J, Teallach, 2001) Griffiths, Tonya, Barclay and Peter J, Teallach: A model-based user interface development environment for object databases, Interacting with Computers,vol.1, 2001,pp.31-68.

(Gabillon Y. et. al., 2011) Gabillon, Y., Petit, M., Calvary, G., Fiorino, H., Automated planning for user interface composition, in 'Proceeding of the 2nd SEMAIS workshop of the IUI 2011 conference', 2011.

(Joffroy, C. et. al., 2011) Joffroy, C., Caramel, B., Dery-Pinna, A.-M. and Riveill, M., When the functional composition drives the user interfaces composition: process and formalization, in 'Proceedings of the 3rd ACM SIGCHI symposium on Engineering interactive computing systems', EICS '11, ACM, New York, NY, USA, pp. 207–216, 2011.

(Karagkasidis A., 2008) Karagkasidis, A. Developing GUI Applications: Architectural Patterns Revisited. In *EuroPLoP*, 2008

(Krzysztof Gajos and Daniel S. Weld, 2004) Krzysztof Gajos and Daniel S. Weld, SUPPLE: Automatically Generating User Interfaces, in Proceedings of IUI'04, Funchal, Portugal,2004, pp.83-100.

(Lepreux, S. and Vanderdonckt, J., 2007) Lepreux, S. and Vanderdonckt, J., Towards a support of user interface design by composition rules, in 'Computer-Aided Design of User Interfaces V', Springer, pp. 231–244, 2007.

(Lu, Xudong, and Jiancheng Wan., 2007) Lu, Xudong, and Jiancheng Wan. "Model Driven Development of Complex User Interface." *MDDAUI*. 2007.

(Marco Brambilla, Pietro Fraternali, 2015) Marco Brambilla, Pietro Fraternali. Interaction Flow Modeling Language. Model-Driven UI Engineering of Web and Mobile Apps with IFML, 2015.

(Mitchell K. et. al, 1996) K. Mitchell, J. Kennedy, P. Barclay, A Framework for User Interfaces to Databases (DRIVE), in Proceeding of AVI, ACM Press, 1996.

(Mori G. et. al., 2004) Giulio Mori, Fabio Paterno and Carmen Santoro, Design and Development of Multidevice User Interfaces through Multiple Logical Descriptions, IEEE Transactions on Software Engineering, 30(8), 2004, pp.1-14.

(Moreno N. et. el., 2007) MORENO, Nathalie; FRATERNALI, Piero; VALLECILLO, Antonio. WebML modelling in UML. *IET software*, 2007, vol. 1, no 3, p. 67-80.

(Nunes, N.J. at. el., 2005) Nunes, N.J. and J.F.e. Cunha, Wisdom – A UML based architecture for interactive systems, in Proceeding of DSV-IS, Limerick, Ireland, 2000, pp.191-205.

(Puerta A. and Eisenstein J., 1999) Angel Puerta and Jacob Eisenstein, Towards a General Computational Framework for Model-Based Interface Development Systems (MOBI-D), in Proceeding of ACMIUI 1999, Redondo Beach CA USA

(Pedro J. Molina et. al., 2002) Pedro J. Molina, Santiago Meliá and Oscar Pastor, JUST-UI: A User Interface Specification Mode, in Proceedings of CADUI 2002, Valenciennes, France, 2002, pp.63-74.

(Patern` O. F. et. al., 2011) Patern` O. F., Santoro, C. and Spano, L. D., 'Engineering the authoring of usable service front ends', J. Syst. Softw. 84(10), 1806–1822, 2011.

(Pietschmann, S. at. el., 2009) Pietschmann, S., Voigt, M., R¨umpel, A. and Meibner, K., Cruise: Composition of rich user interface services, in 'Web Engineering', Springer, pp. 473–476, 2009.

(Stefan P, 2014) Petr Štepán. STEPAN, Petr. *Interaction Patterns as Composite Connectors in Component-Based Software Development*. 2014. Tesis Doctoral. The University of Manchester (United Kingdom).

(Tan, D. S. at. el., 2004) Tan, D. S., Meyers, B. and Czerwinski, M., Wincuts: manipulating arbitrary window regions for more effective use of screen space, in 'CHI'04 extended abstracts on Human factors in computing systems', ACM, pp. 1525–1528, 2004.

(Zhao, Q. at. el, 2008) Zhao, Q., Huang, G., Huang, J., Liu, X. and Mei, H., A web-based mashup environment for on-the-fly service composition, in 'Service- Oriented System Engineering, 2008. SOSE'08. IEEE International Symposium on', IEEE, pp. 32– 37, 2008.

Authors' Information

Elena Chebanyuk – *Software Engineering Department, National Aviation University, Kyiv, Ukraine,*

Major Fields of Scientific Research: *Model-Driven Architecture, Model-Driven Development, Software architecture, Software development.*

e-mail: chebanyuk.elena@ithea.org

Oleksandr Palagin – *Depute-director of V.M.Glushkov Institute of Cybernetics of National Academy of Sciences of Ukraine, Academician of National Academy of Sciences of Ukraine, Doctor of technical sciences, professor; Prospect Akademika Glushkova 40, Kiev, 03187, Ukraine; e-mail: palagin_a@ukr.net*