IRIS STUDIO – IDE PROTOTYPE FOR MODELING
ADAPTIVE USER INTERFACES FOR BUSINESS
INFORMATION SYSTEMS

Margarita Atanasova¹ §, Anna Malinova²
¹,²University of Plovdiv Paisii Hilendarski
Plovdiv 4000, BULGARIA

Abstract: Business information systems available today are very complex in regard to their functionalities and platform requirements. Adding an additional layer of context-awareness makes the development of such systems very expensive, complicated and time-consuming process. Therefore, we developed a web-based tool for modeling adaptive user interfaces which provides a library of predefined tasks according to the functional areas in the operation of businesses. For developing the prototype, we use a model-driven user interface development approach. The process of creation of the UI using our environment includes four major steps – building a task tree model, defining abstract and concrete user interface models and retrieving the final code. All of these models have a bidirectional connection so that they are easily updated. In the final step of the UI creation process we generate platform-independent XML specification of the UI as well as HTML and CSS downloadable code for web and mobile regarding to the specified models. In this code we also include the designer’s input on the corporate identity of the developed system, including logo, colors, font-family and more. In this paper we also examine the different situations in which our software prototype can be used. We provide examples how designers can define tasks according to the users’ culture, level of experience with the system, physical environment, hardware, software, level of disability, role within the system and more.

AMS Subject Classification: 68N19, 68U35
Key Words: business information systems, adaptive user interfaces, context of use, user interface modeling, code generation

Received: 2017-07-09
Revised: 2017-10-28
Published: November 19, 2017

§Correspondence author
1. Introduction

With the rapidly evolving technologies and the need of low-cost support of multi-device and multi-context software applications, model-driven user interface development becomes an important topic not only in scientific research papers but also in many business conferences. At the threshold of the Web 4.0 era where all devices will be connected in the real and virtual world, and personal intelligent assistants will help us in our everyday lives, the creation of context-aware user interfaces will be a mandatory step in the software development process.

There are many research topics discussing the advantages and the limitation of the model-driven user interface development [1, 2, 3]. As a whole it provides a structured approach where modality and platform independent models can be specified and used in the real UI development. All stakeholders can take part in this process including clients, users, product managers, software architects, designers and developers. Using a proper tool supporting this process reduces the cost of adaptive UI creation. But still this approach is not widely adopted by the majority of software companies. One of the limitations of MBUID is the steep learning curve. Modeling context-aware user interfaces introduces additional level of complexity which is hard to understand and maintain. Another limitation is the lack of an easy way to update the existing models when changes on the final product are made. Going back and forth between the real prototype and the models increases the time of development which is inefficient when the product follows a certain go-to-market strategy.

To address the described challenges, we have developed “Iris Studio”, a web-based integrated development environment facilitating the user interface development. The contributions of this paper include our vision on enhancing this process especially for building UIs for business information systems, which users often find difficult to use according to different studies [4, 5, 6]. Therefore, our contributions cover the aspects described below.

Firstly, we support both the design and implementation phases by providing an integrated tool for maintaining bidirectional connection between all four levels of abstraction, as suggested by the conceptual Cameleon Reference Framework (project by Information Society Technologies): Task and Domain, Abstract User Interface, Concrete User Interface, and Final User Interface [7]. For the first step we have partially adopted the ConcurTaskTrees notation [8, 9].

Additionally, the selection of business functional areas of operation in the first step of UI creation process, helps us determine what predefined task models to propose to the developer/designer. We have built-in library for task models
that can be reused to speed up the prototyping process. Moreover, the users
can create and save their own task tree models to extend the library.

In “Iris Studio” the designer can add multiple contexts of use to different
tasks on any of the development cycle steps. That way the designer can create
and preview different combinations of elements in the user interface, simulating
real world situations.

Finally, the underlying business logic model enables us to easily extend the
IDE to generate and run code written in any modern front-end framework for
web applications. We also provide XML code generation. As an example of
specific web technology code generation we transform the concrete model into
HTML and CSS pages, using Bootstrap.

The next sections of this paper are organized as follows: Section 2 presents
related work and an overview of the existing tools for model-driven UI devel-
opment. In Section 3 we present our system – the underlying model, the archi-
tecture and the transformations we made on the input data. We also describe
what the steps of generating final UI are. Section 4 shows different real-world
situations where UI adaptations are needed and how our system handles them.
Section 5 concludes the paper and gives an outlook on future work.

2. Related Work Overview

In this section we will make a brief overview of similar model-driven tools for UI
generation. The strengths and the shortcomings of many early model-based UI
development systems are already a focus of discussion of other scientific papers
[2, 10].

The model-driven approach is used to define high-level models, which are
used for the creation of a detailed specification and client’s requirements anal-
yses from a semantic point of view without a focus on the implementation.
There are different tools supporting this process. Each one of them proposes
different solutions to their users and puts an emphasis on different artifacts of
the MBUID.

Some of them like OpenXava [11] and Mira framework [12] require very
good programming skills for creating the description of the models. Their
target group of users are rather programmers than designers. The final code is
HTML and CSS code. Mira uses Bootstrap so that the result is responsive as
well as modern.

CTTE [13] and Responsive CTT [14] support partially the process of model-
driven development by providing tools for task tree modeling using the Con-
curTaskTrees notation. They produce an XML specification.

SketchiXML [15] supports the creation of concrete UI models using a tablet and a stylus. The final specification can be obtained as a usiXML code. There are also several other similar applications that support sketching a user interface with a tablet and resulting in code generation: SILK [16], JavaSketchIt [17], Freeform [18].

The users in IdealXML [19] can create task models and abstract UIs. The end result here is also a user interface specification in usiXML.

Supple [20] is another interesting project aiming at the creation of JavaScript library, which analyses the skill levels of the users and alters the system according to the results. It mainly gathers input to adapt the system for people with motor and vision impairments. The authors in the Supple project have developed ARNAULD, a system which changes its graphic interface according to the user’s behavior [21].

CanonSketch [22] is another tool for defining abstract user interface models. This tool uses Wisdom notation, which is proposed as a UML extension. An additional view is also implemented, which generates a possible specific implementation of the user interface in the form of HTML.

GrafiXML [23] focuses on the generation of multi-platform context-sensitive user interfaces. It is based on usiXML. The final code can be generated in (X)HTML or Java. One of the outlined perspectives in the paper is to make the UI adaptations during the runtime while now they are in the design time.

Another system that puts its emphasis on the context-aware user interface generation is MASP [24]. It uses model-driven approach, maintaining all levels in the Cameleon Reference Framework to support the creation of UIs for smart homes. The adaptations happen in the runtime. The final code generation supports XML, WML, HTML, VoiceXML.

Gummy [25] is a visual editor for developing multi-platform user interfaces facilitating the design process. It is aimed to be user-friendly for designers, hiding details of the abstract model, but providing functionality of editing the concrete UI model. At the end Gummy “translates” the concrete elements into UIML (User Interface Markup Language) and XML.

Damask [26] is a tool for creating user interface prototypes for different devices implementing partially the model-driven approach. It supports VoiceXML commands. Damask does not aim to create final UI or to add any layer of context-awareness, but focuses on the multi-platform UI modeling.

CIAT-GUI [27] and Cedar Studio [28] also adopt the Cameleon Reference Framework supporting the four levels of abstraction. The generated final UI in CIAT-GUI is in XAML, while CedarStudio generates XML specification with
added attributes to the elements for different contexts of use.

Approaches like GrafiXML, Masp, Supple, Cedar Studio and MIRA present tools and techniques for supporting the development of adaptive UIs. The designer input in these approaches is very small to none. The final generated user interface in nearly all of the reviewed systems is not ready to be used directly in a real project without any changes. It lacks a way to make visual changes to the elements or apply corporate identity. Also these tools require additional knowledge and are hard to understand from a beginner. Our aim is to propose a solution to these problems by creating an environment, which supports the designer/developer of adaptive UIs through the whole cycle in the UI development. The environment has to be self-explanatory and produce final user interface according to modern business requirements. We maintain all steps from the clients’ requirements analysis to the downloadable code, which can be directly implemented into the product. That is why except the XML generation we implemented one of the most popular front-end libraries Bootstrap. In the next section we also explain how we can add new technologies for the final code generation.

3. Iris Studio Architecture

We have developed “Iris Studio”, an IDE prototype for building adaptive user interfaces, by extending the Cameleon Reference Framework’s four levels of UI development cycle. The aim is to support and accelerate the creation of user interfaces for business information systems, thereby we propose a modification to the CRF on the implementation level by adding an additional step for defining business functional areas of operations. That way “Iris Studio” extracts predefined task models depending on the selected industry, therefore speeding up the task analysis step and respectively the development process. The functional areas of operations and the task modeling in our IDE prototype as well as the transformations made between the task model and the abstract user interface model are described in another paper [29].

The workflow of creating user interface in Iris Studio is as follows: First, the user creates a project and selects functional areas of operations. Than models the user tasks (See Fig. 1), generates abstract UI model (Fig. 2), concrete UI model (Fig. 3) and final UI (Fig. 4). On the last step the user applies corporate identity – logo, colors, fonts, repositions the elements if needed and downloads the final code.

There is no standard notation for defining concrete UI model. Depending
Figure 1: Iris IDE – first step of the UI modeling process – task modeling

Figure 2: Iris IDE – second step of the UI modeling process – generating Abstract UI model

on the abstract interactive object type (input, output, control or navigation) and node type (parent or leaf), the tasks can be transformed into different
concrete objects. This choice is made by the user as this cannot be automatically selected. For example, if we have abstract interactive object from type “output” in the concrete UI model generation in Iris Studio the user can select from multiple types of output concrete objects like: table, pie chart, bar chart, line chart, paragraph, list, etc. We always transform the root element from the abstract UI model into a Window concrete element in the concrete UI model. From the user input we generate the concrete objects and present them in the screen in a structured way similar to a rapid prototyping tool. We provide the users a toolbox on the top from which they can add different concrete objects in the model. They can also put them in a grid, defined by rows and columns. With the implemented bidirectional connections between the models, every added object in the concrete model automatically updates and is added to the abstract model and the task trees. For example, if the user adds a text area in the concrete model, it appears in the abstract model as an input facet and as an interaction task type in the task model.

The final user interface (Fig. 4) is automatically generated and loaded into
an iframe in the client’s browser. We use a master template as a container of the tasks. This template contains header, top navigation with placeholder for logo, search bar, left navigation, main container and footer. The modeled tasks are placed in the main container section. The designer/developer can hide and show different contexts, thereby seeing different combinations of UI elements on the screen. Data attributes are added to those elements in the final code so that they can be managed accordingly by the back-end logic in the real project. This is also the step where the designer can upload logo, add Google fonts, change the font color and background colors to the design. In “Iris Studio” one project can have multiple user interface files. That is why all corporate identity parameters are stored in the project’s database and are applied to all files in the same project so that the designer does not need to apply them every time. We use Google Charts as a charting library for the diagrams representation in the final UI.

Figure 4: Iris IDE – fourth step of the UI modeling process – generating Final UI
All user interface files in a single project can be downloaded as a .zip archive, containing HTML, CSS and JavaScript files with linked Bootstrap library from its’ CDN. The final user interface can also be downloaded as an XML file. We provide a detailed explanation in the studio’s help section how this file can be transformed into executable code.

The business logic behind abovementioned steps is shown in Fig. 5. We maintain different users with multiple projects each. Each project can have many user interface files. The UI file Class is responsible for model transformations, data validation and code generation. Every UI file can be added to a personal library of predefined tasks and reused in different projects. The UI nodes store all attributes of the task tree model, abstract and concrete models. They are connected to the contexts of use. Each node can have none, one or more contexts to which it belongs. The UI nodes are linked between each other so that the workflow between the tasks can be specified during the requirements analysis phase. The corporate identity is linked to the project so that all files in a single project have the same logo, font style and colors. Every project has a functional area so that the user has access to predefined selection of ready-to-use task models. The Technology Parser Class is responsible for the code generation. It transforms the concrete objects into code depending on the technology. Currently we have implemented parsers for XML and web (HTML, CSS).

4. User Tasks Modeling for Different Contexts of Use in Iris Studio

In this section we present various examples of task models from the perspective of different contexts of use that are created using “Iris Studio”. Moreover, this helps designers focus on the user tasks rather than the implementation.

Task modeling helps user experience designers understand what people want and how to design the business requirements into a usable, friendly product. Task analysis is an effective process where UX designers decompose complex tasks into simpler ones. In this process stakeholders and clients might also be involved as no coding is required for the tasks’ description.

Designing such task structures is not always easy, especially when different user profiles and contexts of use should be defined at the same time. The developed software system enables its users to define each of the tasks to which context of use is applied. One task might be applicable to several contexts or to none. The behavior of the final user interface depends on the specific parameters sent from the back-end during the run-time.
In the following sections we will try to give as many as possible live examples for a variety of situations where UI adaptations are needed.
4.1. Cultural Adaptivity

Systems with culturally adaptive user interface can adapt according to the cultural characteristics of their users [30]. Applications that take into account the location and the country of their users are considered to be much more efficient and convenient. A simple example is the user interface of Google. While the clean design of the search engine is preferred in Europe, the leading search engine in Russia is Yandex, in China is Baidu and in South Korea is the colourful Naver [31]. Naver offers to its users many images, advertisements and different search categories – common technique for websites in South Korea.

An example diagram with described user tasks and their contexts of use, representing a similar case, is presented below (See Fig. 6). Using our prototype, designers can describe user tasks using ConcurTaskTrees notation which means that we have four main types of tasks (Abstraction, Interaction, System and User) and different kind of operators between them. In addition to this we implemented a mechanism for adding contexts to the nodes in the tree. Every node can be part of different contexts. The result is a combination of variety of contexts. Contexts might be switched on and off so that the designer might preview the different combinations in the next steps of the UI modeling process which are the Abstract User Interface and the Concrete User Interface. In Fig. 6 we are showing how easy it is to determine that these particular tasks (Banner and News) are related only to the context of use named “South Korea”.

Figure 6: Task model of user task “Search”. Context of use named “South Korea” for users located in this country is applied.
4.2. User Roles

Most of the web sites and business information systems have different user roles and permissions. Carefully planning the visibility of different parts of the application in early stage of user experience design is essential as the system grows. Different aspects of this issue are shown in [40, 41] concerning the Distributed Platform for E-learning (DisPeL). In Fig. 7, by using our software prototype, we’ve managed to describe a dashboard task model with different UI components, visible for different user groups. The executive director can see all reports, while the HR specialist can see only the hiring and headcount report. The administrator can view Settings and Users links. If the user of the system does not apply to any of these contexts, they can only see the News and Search.

All the other roles can see these two tasks (News and Search) as well as their own. The final user interface is once again generated run-time according to the parameters that the back-end sends to the front-end. In this example, we might have 14 different combinations of user interfaces for the dashboard depending on what user roles are assigned to the person. This will help the UX designer easily test their prototype with representatives of their personas\(^1\).

\(^1\)“Persona” is a descriptive model of a certain type of user based on data from a customer survey [32]
4.3. Environment

Nowadays information technologies have the resources and the capability to create Rich Internet Applications, to personalize the experience, to analyze real time data while gathering information from different sources (website, sensors in the mobile devices and many other). That way designers have unlimited possibilities to prototype adaptive user interfaces in order to improve the usability of the system.

One of the first examples of worldwide popular adaptive systems is the GPS navigation. Using the location of the user, the system draws the shortest route to the end point. When the sun goes down or the car enters a tunnel, the system interface directly switches to night mode in order not to dazzle the driver. This is a good example of a user interface that adapts to environmental changes (one of the four components of the context of use).

Other change of the environment might be the different Internet connection speed. A context for low Internet speed might be defined where the images might use lazy-loading mechanism or to not load at all so that the whole website loads faster.

For business information systems there could also be different changes according to the environment. For example, systems, which are responsible for the manufacturing and production process, can offer different UI functionalities depending on the user’s location – in the factory or outside the factory. Information about the user’s location is typically gathered from global navigation satellite systems. In Fig. 8 we describe a simple task for a production process in a web based industrial application.
4.4. Level of Experience

No matter what the target group of an application is there are always three types of users: beginners, intermediates or experts. Usually UX designers manage to combine different UI elements so that the user interface is friendly for most of the users [42]. A good design approach is adding tasks for beginners when the user logs into the system for the first time. A quick guide, video tutorial or different notification messages might lead the users through the system so that they learn quickly the functionality without making unnecessary errors.

Completing a new task for the first time is difficult for the average user of business information systems. Most of the business applications do not even support adaptive user interface for users with different skills such as beginners or experts [33]. On Fig. 9 we describe a simple task model for monitoring a networking device. When a user tries to add a new device in the monitoring software a choice for how-to tutorial or video appears. Here the UX designer might add additional “Do not show this tutorial again” checkbox or there might be different parameters indicating that the user is no longer in the beginners group. The other two subtasks below the “Add new device for monitoring” are for entering new device and monitoring meta data information. They are visible for the default context – for all users that are not beginners.

4.5. Hardware and Software

Mobile and user devices faced significant changes during the last two decades. They differ not only by screen size and resolution, but also by RAM memory, battery time, Internet connection capabilities, operating system and many
more. Designing a software for many different devices requires good planning in order to deliver the specific functionality which can be supported by these devices. Usually the focus is one or several device groups that have similar hardware parameters [34].

Using our software tool for prototyping adaptive user interfaces we propose a sample model for task “Track a Training”. The device target of the app are devices, that might track the distance and the duration of the training. This includes smartphones and smart watches. The default user interface for mobile phones includes all the functionality, as shown in Fig. 10: track the time and the distance, enter pulse, show tracking chart, take a photo, show burnt calories, play music during the training and manage a training with start, pause and stop buttons. We added context of use for smart watches which includes only the functionality that can be supported by that particular device – track the time and the distance, enter pulse, show burnt calories and manage a training.

4.6. People with Disabilities

Adaptive user interfaces for people with disabilities are a common research topic of many projects as the percentage of this group of people is quite high – 15% of the world population lives with some form of disability [35]. The users might differ in traits such as tactile perception, memory, vision, hearing, ability to navigate in unfamiliar environments and others. The contexts of use for people with disabilities should be very carefully examined by the UX designers before any prototyping. Also, whenever possible, accessibility specialist(s) should be included in the design process. There are different guidelines for the different disabilities that should be followed in order to create a user-friendly software...
system. For example, the UK Department of Health proposed a methodology for creation of documents for people with learning disabilities [36].

People with Autism Spectrum Disorders have very different personal preferences and needs. That is why personalization for such users is a key element for the success of the project [37]. In Fig. 11 we describe a “Read a Tale” task with added context for people with ASD. Depending on the design guidelines for such people [38, 39], the text for them should be simplified and should consist of more images. Also a nice to have feature is the system to be able to read the text to the users with ASD.

5. Conclusion and Future Work

The creation of adaptive user interfaces for different contexts of use benefits a lot from the model-driven development, especially when the designers have a proper tool facilitating the process. It this paper we describe “Iris Studio” the authors’ software prototype that supports the adaptive user interface modeling process. We also describe different situations that require adaptations according to contexts of use. There are many possibilities for defining adaptive behavior: hardware and software, social and physical environment, user skills level, users’ culture, disabilities, roles and other. In this paper we make an overview of other systems facilitating the model-driven user interface design process. As a future work we are going to implement a functionality of generating multiple pages
from a single task model. We also intend adding more master page templates for the final UI. In addition, an evaluation study with real users has to be conducted so that we analyze the usability of “Iris Studio” and the overall development time of a final user interface.

6. Acknowledgement

The work described in this paper has been partially supported by the project FP17-FMI-008 of the Scientific Fund of the University of Plovdiv “Paisii Hilendarski”, Bulgaria.

References


