Designing Responsive Interactive Applications by Emotion-Tracking and Pattern-Based Dynamic User Interface Adaptation

Christian Märtin(✉), Sanim Rashid, and Christian Herdin

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

Abstract. Model-based user interface development environments (MB-UIDEs) can be enhanced by pattern-based frameworks to allow for richer design capabilities and more flexible responsive behavior during the runtime of the implemented target application. In this paper an experimental system prototype for integrating facial analysis and eye-tracking into a pattern-based dynamic user interface adaptation process is discussed. The resulting system evaluates the emotional state of the system users to trigger the HCI-pattern-based adaptation of the user interface. By monitoring the emotional states of the users over longer time periods, while the system changes its behavior and its appearance, conclusions about the perceived quality dimensions of the interactive application can be drawn.

Keywords: Model-based development · MB-UIDE · Pattern-based development · HCI-patterns · Responsive design · Eye tracking · Facial analysis · Emotion tracking · Adaptive user interface

1 Introduction and Related Work

Model-based approaches have a rich history in HCI and can be fruitfully applied for building media-rich and flexible interactive systems [12]. In addition to classic model-based and model-driven approaches, in recent years the descriptive characteristics and powers of HCI patterns [11, 14], pattern languages [4, 9], and generic software pattern-based approaches, [10], were increasingly exploited for modeling structure, behavior, and presentational aspects of interactive systems.

The PaMGIS framework [5, 7] combines the model-based user interface development paradigm with a pattern-based modeling and design approach conforming to the CAMELEON reference model [2]. In PaMGIS the modeling and semi-automated construction of interactive applications is centered around a pattern repository that allows storage of and access to domain-independent and domain-dependent HCI pattern languages that are specified in the PPSL pattern specification language [6]. These patterns can be exploited for advanced user interface modeling and design purposes together with various other model types and modeling artifacts, e.g. task models [13], user models, context models, and environment models. So far, applications for various

M. Kurosu (Ed.): HCI 2016, Part III, LNCS 9733, pp. 28–36, 2016.
DOI: 10.1007/978-3-319-39513-5_3

application areas, e.g. web-shops, knowledge sharing systems and travel information systems have been constructed using the PaMGIS approach and tool environment. For these areas not only usable and functional application prototypes could be developed, but also the mapping of the same abstract and semi-abstract models to different target platforms and target devices was demonstrated.

In this paper we aim to go a step further in our view of the usefulness of the pattern-based modeling approach for interactive system design. In order to arrive at true context-aware software services, new software engineering approaches are required that couple standard or agile requirements engineering techniques with methods that monitor the users' behavior, emotions, and possibly their changing mental states at runtime in order to decipher their intentions during their interaction with software services in a sequence of situations starting with the desire to reach a certain goal and finishing with the goal-satisfying situation. For studying the requirements engineering process and the software engineering life-cycle for situation-aware software, the Situ framework was constructed [3]. Studies with the Situ framework involve the monitoring of humans-in-the-loop and could produce so called Situ_patterns and building blocks for runtime adaptation of the observed software services. The information gathered could later be used for situation-responsive design without the direct monitoring of a user's mental states.

In the more elementary approach, presented in this paper, we aim at demonstrating that a pattern-based approach generally is apt for arriving at adaptive interactive applications with highly responsive design and runtime-reconfiguration capabilities. For this purpose we use abstract and semi-abstract HCI-patterns from our repository as models and templates for runtime-adaptive user interfaces. Our experimental system controls the choice of layout and presentation characteristics by evaluating user behavior and user emotions during runtime. The feasibility of this new approach is demonstrated for a prototypical interactive application, where the PaMGIS pattern repository is coupled with highly interactive system components for behavior and emotion evaluation based on face reading and eye-tracking technology.

2 Application Prototype

The aim of this project was to develop a prototype which is able to adapt a web page dynamically in order to control the user experience, so that the interests and satisfaction of the user persist as long as possible. The adaptation should happen on the basis of facial analysis and the gaze motion. For changing the appearance of the user interface the PaMGIS pattern repository can be accessed dynamically. To gain deep insights into consumer behavior, eye tracking was used as a sensor technology. For the facial analysis the FaceReader 6.1, developed by Noldus, was used. The resulting software system can be seen as a test bed for studying user emotions and mental states of users during interaction with web applications. It can also be used for evaluating the impact of runtime-adaptable user interfaces on the mood and mental states of users.

2.1 Technologies and Tools

As one of our goals was the rapid development of the target system with state-of-the art software technologies, the following tools and systems were used for the implementation of the prototype. The ASP.NET MVC web application framework, developed by Microsoft, implements the Model-View-Controller (MVC) pattern. The open-source JavaScript Framework AngularJS was used for running the resulting single page application. AngularJS basically does not implement the MVC in the traditional sense, but rather close to the MVVM (Model-view-view-model) pattern. With AngularJS, HTML DOM can be extended with additional attributes, to be more responsive to user actions. Furthermore it allows rapid prototyping by supporting data binding and dependency injection. All the relevant application parts are implemented within the browser. Therefore a complete client-side solution is created that is ideally fit for coupling it to any server technology [1] and eases all aspects of implementing user privacy. The Cascading Style Sheets (CSS) describe the design of the elements from the HTML page. The elements are derived from HCI patterns. They represent the platform-dependent final user interface and can be retrieved from the pattern repository at runtime.

The facial expression analysis software FaceReader 6.1 was employed for analyzing users' emotions. The software tracks six different basic facial expressions (happy, sad, surprised, disgusted, angry and scared) [8]. The gaze directions, head orientation and person characteristics (gender and age) can also be logged automatically [8]. An Application Programming Interface is also included in the FaceReader 6.1 software. The language binding assembly for the .NET Framework is the FaceReaderAPI.dll. This API serves as an interface between the different software programs and eases integration. The framework allows other software components to respond instantly to the emotional state of the participant. The API includes real-time export of detailed state log data, which enables the facial analysis, tagging and inference of cognitive affective mental states from facial video [8].

For the eye-tracking the Tobii eye-tracker X-2-60 was used. For the development of eye-tracking applications Tobii provides the Tobii Analytics SDK. This SDK includes an application programming interface, which is implemented as a core library, "tetio", and a set of language bindings which are built on top of the core. The language binding assembly for the .NET Framework is the Tobii.EyeTracking.IO.dll [17]. The Tobii eye-tracker presents itself on the LAN by using a technology called "zero configuration networking" (zeroconf). It automatically creates a usable computer network based on the Internet Protocols Suite (TCP/IP) when computers or network peripherals are interconnected [16]. No manual operator intervention or special configuration servers are required. An instance of the EyeTrackerBrowser class is created and started to consider the connected eye tracker in the network.

2.2 Software Architecture

The UXDataControlForm in Fig. 1 illustrates the Windows Form application, which has access to the data of the eye tracker and FaceReader in real-time due to the APIs [15]. When the application is connected to the FaceReader, it is possible to control the Face

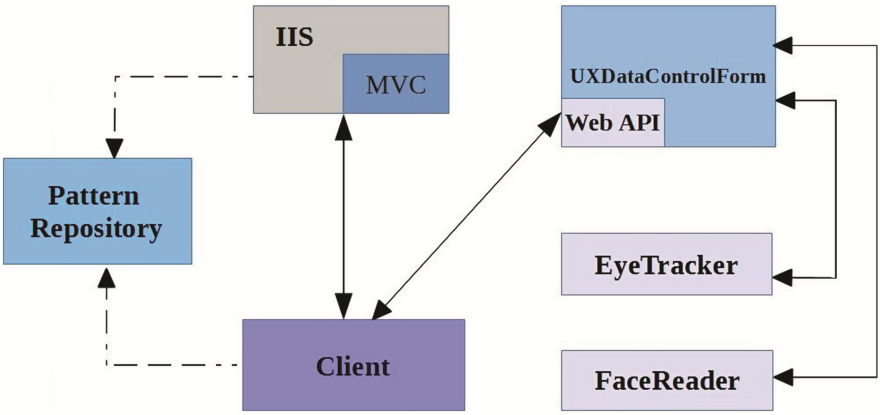

Fig. 1. Software architecture of the prototype. The dashed lines indicate that information in the pattern repository is interpreted to access the needed concrete implementation of patterns that may reside in the client or the server.

Reader software and its actions like start and stop of a facial analysis, enable detailed log and state log. The detailed log contains all the classifications enabled in the logging settings [8]. The state log shows the dominant expressions of the participant. If the eye tracker is connected to the local area network, the application automatically detects the tracker and a connection can be established. The application allows the user to start/stop tracking and run the eye-tracker calibration. Figure 2 shows a snapshot of the UXDataControlForm [15]. The Windows Form Application is not really the main part of the project. It only offers the possibility to access the hardware and to retrieve user experience data (emotion, gaze points).

The central part of our prototype architecture is the Client, the browser that accesses the real-time data. To enable data access, the browser needs a connection to the UXDataControlForm. For this purpose there were no separate Internet Information Services (IIS) required. The connection was implemented using a self-host Web API inside the Windows Form application.

The application listens to the http://localhost:8080/ address. The Web API Controller "UXDataController", which is defined in the application, uses the GET action and returns the FaceReader and Eye Tracker data via an interface.

The prototype of the web application, which handles the user experience state, is basically implemented as a client-side application in AngularJS. AngularJS allows dynamic changes in the page without having to load the whole page again. Via the API Angular requests the data and evaluates it in the directive. A directive is essentially a function that executes, when the Angular compiler finds it in the DOM (Document Object Model) [1].

The structure of the web application is kept simple. The web application implements a railway link map system where railway link maps according to the user´s desired destination are shown. Figure 3 shows a snapshot of the web application, where a timetable information has to be entered in order to get a link map. The link map data are

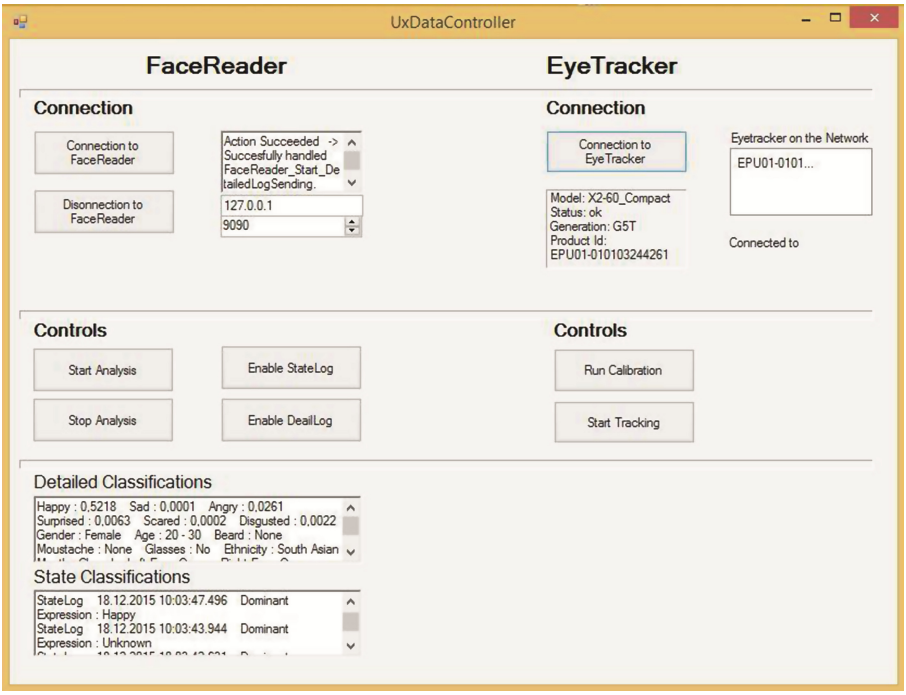

Fig. 2. WinForms Application as a Web API host application

requested and retrieved from a database. Every web page of the application has a controller and a directive in AngularJS. Main functions are defined in services that are accessed in directives. The service, e.g., fetches the eye-tracker and Face Reader data.

Every single page is segmented in panels (Fig. 3) and has a specifically defined directive, which is given to each panel. For each panel there is a defined update-method in the directive.

With the help of the eye tracker, which is adjusted to the screen size, the eye motion positions of the user can be easily retrieved. To get the most exact positions, a calibration procedure can be executed by the user in the UXDataController (see Fig. 2).

If the user looks at a specific panel (in Fig. 3), the directive automatically calls the appropriate update-method of the panel. In the update-method the current emotion state of the user is retrieved from the Face Reader and analyzed. The analysis algorithm examines, how often the user has looked at a given panel and stores the result in an array together with the inherent emotion state at that time. The algorithm can be extended at any time for more sophisticated evaluations. The final dominant expression (over a specific time period), which is figured out by a statistical calculation, can be assigned to the panel. Once the dominant expression is assigned to a panel, the system designer can decide what the update-method should do. Dependent on the dominant expression and the situation (click behavior of the user) the page can change its color dynamically or some extra features can be added to give the user a better user experience. In addition

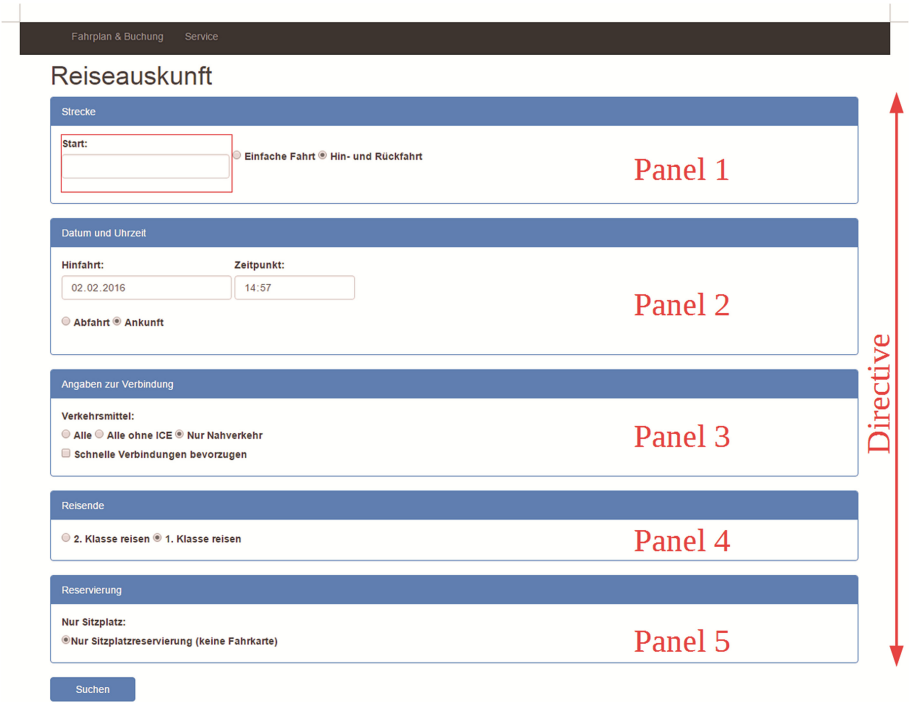

Fig. 3. Screenshot of the web application [15]

the user characteristics are exploited and the page can be adjusted even more precisely. For example the font size can be increased, if the user is aged [15].

To demonstrate the importance of an optimized user experience in the experimental setting some errors were installed into the web application. In Fig. 3, for example, there is no destination field, so it is not possible to enter data. This is an attempt to upset the user and change his or her emotional state to “angry”. If the emotional state is “angry”, the background color will be changed to an enjoyable color and the layout will be adjusted by the CSS-based UI implementation of a user experience pattern to mollify the user and manipulate his mind and mood. The prototype also includes several external HTML pages, modeled after HCI-patterns that can offer help services, if needed. They are dynamically placed on the site, to help the user in a specific situation. In this web application example the destination field is required, to reach the intended goal. The display of an external HTML template is realized by using features of AngularJS directives. Directives are very powerful concepts in Angular. HTML itself does not support embedding HTML pages within HTML pages. To achieve this functionality, the directives in AngularJS can be exploited. The directives allows to load dynamic templates into the web page [1]. In this example an external HTML page, which contains the option to select a destination is added.

The display of the external HTML page is also done in the current directive and update-method of the panel. Here the system designer has to decide, which template has

to be given to the directive. The parameter passing (path of the template) is possible with scope. Scope is an object that refers to the application model. It is a link between the application controller and the view [1]. The scope object is controller-specific. If some model data in the view are changed, the controller automatically gets the modified value. AngularJS also allows the access of a parent scope in a directive. So it is possible to render the updated value to the DOM. In the prototype the update-method calls the parent scope and tells it, which template has to be loaded. The ng-include directive, which is defined in the main HTML page, contains the scope object (the path of the template). By getting a value, the ng-include fetches the external HTML resource, compiles it and finally includes it in the HTML page. The ng-include directive creates a new scope, but it can also refer to the parent scope with $parent.scope.

2.3 PaMGIS Pattern Repository

The integration of HCI-patterns for adaptive interactive systems based on emotion tracking and the mentioned technologies and their implementation for the final user interface is accomplished by using CSS style sheets and features of AngularJS.

The PaMGIS pattern repository uses the PPSL pattern specification language for describing HCI patterns on all abstraction levels [7]. The CSS style sheets and HTML pages used in the prototypical Web application presented in this paper are modeled after structural and presentation HCI patterns. In order to link the actual pattern implementation code to the patterns from which they are derived, the specification language offers the *Deployment* element. This specification element contains powerful specification attributes to describe the more implementation-related aspects like concrete code and model fragments which can also be used for automated user interface generation. In the context of the discussed prototype they also contain the links to the runtime environment that interprets the CSS stylesheet or HTML code for each pattern accessed by the update methods of the applications.

The prototypical application in Fig. 3 implements user interface adaptation by constructing the displayed HTML page from two interpreted patterns. The first pattern is responsible for the structure of the first panel of the input form. The second is responsible for the color experience of the entire form.

Figure 3 shows the initial page setup. For each panel a pattern is available in the repository. For the pattern *Panel1* the initial pattern implementation is *Panel1:TravelStart.* For the pattern *TimeTableInformationColor*, the initial setup uses its implementation *TimeTableInformationColor:Neutral*. If the mood of the user changes to “angry”, the structure will be adapted by executing the HTML code of the pattern implementation *Panel1:TravelStartDestination* and changing the color of the complete page by accessing the pattern implementation *TimeTableInformationColor:Mollifying.* For each pattern a PPSL specification is available in the repository. The PPSL top level element <*Deployment*> [7] contains a list of all implementations of a given pattern. The update method accesses the selected implementation by entering an identifier. As a result, a link to the relevant HTML code is returned. An alternative would be the direct storage of the implementation code fragment within the repository, which is also supported by PPSL.

In order to allow for the most flexible architectures for all types of resulting interactive applications, the PaMGIS pattern repository can be accessed either by the client side or the server side.

3 Conclusion

In this paper an architecture for building adaptive interactive systems by introducing emotion tracking, flexible client technologies for dynamic and responsive application design and integrating a pattern-repository into the design environment was demonstrated. This architecture will serve as an experimental platform for testing the PaMGIS framework as a basis for modeling and constructing a wide spectrum of domain-dependent and independent adaptive applications for various target platforms, contexts, and devices.

In the prototype a relatively simple railway timetable information system was implemented as an exemplary interactive application. In the future any type of web-based application, e.g. web shops, games, apps for personal communication, etc., can serve as a target for pattern-based dynamic adaptation triggered by the monitoring of user emotions. The monitoring interface is also open for additional sensor-based gathering of emotional and mental states of the user. Wearable devices could provide bio-signals in addition to the already integrated facial analysis and eye-movement data.

In addition, experiments with more intelligent algorithms for drawing conclusions about the hidden mental states from the observed emotional, eye movement and biological data will be carried out.

One of the next steps towards a reliable system will be a thorough measurement and usability-lab-based evaluation of the mood-changing effects of emotion-state-triggered dynamically adapted user interfaces on diverse users and over longer time periods, when interacting with an extended version of the prototypical railway timetable application. Measurements of this kind will allow both, a reliable identification of diverse quality characteristics of the application in use, and, at the same time open the way to find out more about the hidden mental states of the users in different situations and contexts of use.

References

1. AngularJS Documentation. https://docs.angularjs.org. Accessed 21 Dec 2015
2. Calvary, G., Coutaz, J., Bouillon, L. et al.: The CAMELEON Reference Framework (2002). http://giove.isti.cnr.it/projects/cameleon/pdf/CAMELEON%20D1.1RefFramework.pdf. Accessed 15 April 2015
3. Chang, C.K.: Situation analytics: a foundation for a new software engineering paradigm. Computer **49**, 24–33 (2016)
4. Deng, J., Kemp, E., Todd, E.G.: Managing UI pattern collections. In: Proceedings of the 6th ACM SIGCHI New Zealand Chapter's International Conference on Computer-Human Interaction: Making CHI Natural, pp. 31–38. ACM (2005)
5. Engel, J., Märtin, C.: PaMGIS: Framework for pattern-based modeling and generation of interactive systems. In: Proceedings of HCI International 2009, San Diego, U.S.A., pp. 826–835 (2009)

6. Engel, J., Herdin, C., Märtin, C.: Exploiting HCI pattern collections for user interface generation. In: Proceedings of PATTERNS 2012, pp. 36–44 (2012)
7. Engel, J., Märtin, C., Forbrig, P.: A concerted model-driven and pattern-based framework for developing user interfaces of interactive ubiquitous applications. In: Proceedings of First International Workshop on Large-scale and Model-Based Interactive Systems, Duisburg, pp. 35–41 (2015)
8. FaceReader 6 Application Programming Interface. Technical Note
9. Fincher, S., Finlay, J.: Perspectives on HCI patterns: concepts and tools (introducing PLML). Interfaces **56**, 26–28 (2003)
10. Gamma, E., et al.: Design Patterns Elements of Reusable Object-Oriented Software. Addison-Wesley, Reading (1995)
11. Kruschitz, C., Hitz, M.: Human-computer interaction design patterns: structure, methods, and tools. Int. J. Adv. Softw. **3**(1 & 2) (2010)
12. Meixner, G., Calvary, G., Coutaz, J.: Introduction to model-based user interfaces. W3C Working Group Note 07 January 2014. http://www.w3.org/TR/mbui-intero/. Accessed 27 May 2015
13. Paternò, F.: The ConcurTaskTrees notation. In: Model-Based Design and Evaluation of Interactive Applications, pp. 39–66. Applied Computing. Springer, Berlin (2000)
14. Seffah, A.: The evolution of design patterns in HCI: from pattern langauges to pattern-oriented design. In: Proceedings of the 1st Interational Workshop on Pattern-Driven Engineering of Interactive Computing Systems (PEICS 2010), pp. 4–9 (2010)
15. Rashid S.: Entwicklung eines Prototypen zur User Experience Optimierung auf der Basis von Emotions- und Blickanalyse im Bereich E-Commerce. M.Sc. Thesis, Augsburg University of Applied Sciences (2016)
16. Zero Configuration Networking (ZeroConf). http://www.zeroconf.org/. Accessed 18 Dec 2015
17. Tobii Studio SDK. Developer Guide, 8 May 2013