Architecture for client-independent Web-based applications

Guido Menkhaus
Software Research Lab, University of Constance, D-78457 Constance, Germany
Email: Menkhaus@fmi.uni-konstanz.de

Abstract

The development effort for interactive Web applications is continuously increasing, because multiple clients with widely varying user interface (UI) capabilities have to be supported. In addition, personalization features render the task more complex. The MUSA (Multiple User Interfaces, Single Application) system addresses that issue by decoupling the application logic from the UI logic. The architecture of the MUSA system is based on an event-graph. The event-graph abstracts UI issues and personalization issues from the implementation of the application.

1 Introduction

The original idea of the World Wide Web was to create a 'universe of network accessible information' [4], i.e. a system offering information on static web pages for browsing only. Since the inception of the WWW and its original purpose, to help people to access and use information, it has evolved into an interactive medium, where more and more business operations rely on constantly changing information that is available on the Internet. A further tendency is the ubiquitous wireless Internet [15]. Users access the information whenever they need it and wherever they are through mobile gadgets with wireless connectivity built in.

The current situation shows, that there is a shift from the desktop PC as the principal device to access services and information on the Internet to consumer devices such as wireless phones, handheld computers, and a wide spectrum of Personal Digital Assistants (PDAs) [12]. As a consequence, the variety of devices, which access information on the Internet grows as the distribution of consumer devices and the mobile communication infrastructure is being put in place. This is why interactive Web applications have to become more and more flexible in order to adapt to the growing variety of mobile devices and UI capabilities. These range from graphical UIs on displays with varying quality and size, and Web-based interfaces using applets to automatic speech recognition and natural language understanding.

The basic problem is that most Web applications are designed to be accessed with a browser on a high-resolution monitor with sophisticated graphical capabilities. However, consumer devices with integrated Internet-access are becoming more and more popular, but the offer of services compatible with devices with low or non-visual capabilities falls short. The reason is that most systems are designed with a specific client device and UI in mind. Once the system is installed, it is difficult to extend the service's range of target devices without modifying the service itself.

Standard software architectures following the Model-View-Controller paradigm [2] or concepts from pattern-based architectures [6] have already considered the means offered by abstract interfaces to address the issue of multiple UIs. These architectures apply separation of the system's functionality to isolate UI concerns from the application logic. The decomposition eases the integration of multiple UIs, which share the same application logic. The abstraction and separation is used to ensure that changes to a component do not affect other components, as long as the interface of the component remains unchanged. General software architectures following the ideas of abstraction and separation do address issues regarding multiple UIs in only one

aspect. They have the potential to support multiple UIs, assuming that each guarantees to implement the application logic in a completely satisfactory way. Nevertheless nothing ensures that an existing application logic can use a new UI. This aspect is of significant importance, because of the fundamental differences in human machine interaction of the new consumer devices.

Personalizing an application gains more and more attention in human machine interaction. Having the possibility to personalize and organize the working space is important for a user accessing applications on small consumer devices. The UI on those devices has only limited capabilities. Focusing on the aspects of an application the user actually needs and works with is of significant advantage to the user.

The objective of the paper is to introduce the architecture of a system that employs an event driven dialog architecture, which allows the realization of Web applications decoupled of the UI. This is done under s pecific consideration of the fundamental differences in UI interaction of the new consumer devices. The paper presents an event graph for designing and implementing dynamic interactive Web applications. The event graph is a result of an analysis of the requirements of interactive Web application, which support a variety of different UIs. In section 2 we consider existing solutions. Section 3 presents the architecture of the MUSA system (Multiple User Interfaces, Single Application) . Section 4 introduces th e event graph and the set of events used. How to personalize a Web application, realized with the event graph, is discussed in section 5.

Problem-description: How is it possible to realize interactive Internet Services whose functionality and architecture is independent of the characteristics of the UI and whose UI is nonetheless consistent and coherent on every client device with the service logic. The architecture of a Web application should be able to easily implement an interactive service, enable the promotion of interchangeable UI to a single interactive service logic and data.

These issues have to be attacked from the very beginning of system design. Recent research has resulted in several Web application -modeling methods [3][8][10][13]. They are different in their approach and motivation, but they all agree upon a separation of content, hypertext/structure, presentation, similar to the Model/View/Controller paradigm [2], Interaction and Implementation of a Web application during the design process.

- The content level refers to pieces of information included in the application.
- The hypertext/structure level denotes the content's organization and the navigational design.
- The presentation level handles the representation of the hypertext level, i.e. th e visualization of information and the dynamic features, such as navigation and interaction.
- The implementation level abstracts the implementation of all levels, especially of the application logic.
- The interaction level links the application's dynamic f unctionalities with presentation elements [7].

The decoupling of these levels, especially the clean separation between the navigational and the abstract UI design, the service logic and instantiation of an actual UI impacts directly the flexibility of the service, eases the modification, extension and maintenance activity [1]. It allows building multiple UIs for the same navigational structure and the same Web application logic. It aims at the

1. Reduction of UI modifications, in the case of a modification ofhe application logic.
2. Reduction of modifications of the navigational structure, in the case of a modification of the service logic.
3. Easement of the removal, addition and modification of UIs.

2 Existing Solutions

To overcome the problem of supporting multiple UIs, there are mainly three approaches that try to bring Internet Services to a wide range of devices.

1. Adapting an existing service and its architecture in a way that it is apt to be accessed from multiple devices is a straightforward approach. The advantage of this approach is, that the service and its UI are optimized for each device. However, this approach requires the service to be rewritten several times to take into account the characteristics of each client device. This results in high redundancy and consistency checks of each version of the service. The administrative effort of this approach is prohibitive.
2. Most interactive Internet services were designed with a single UI in mind. If the service provider extends the service's range of client devices, the existing UI is adapted and the UI-elements are mapped to UI-elements of the new UI [11]. In this approach the client uses the same service, regardless of the UI and device that accesses the service. Though, only if the new client's set of UI elements is a subset of the original client's set, the existing UI can be mapped satisfactory to the new client's UI. In [9] for example the conversion of Web pages written in html to wml caused significant problems.
3. [1] introduces a reactive constraint graph to design and implement interactive services with multiple UI. The reactive constraint graph however, which incorporates the service logic and the content is static and cannot be modified, without translating and recompiling the system. The MUSA system is inspired by this approach. However, it is radically different in realizing the service logic and UI implementation.

3 Architecture

The MUSA system is an approach to designing architecture for interactive Web applications supporting multiple UIs. It attacks the described problems by separation of navigational structure, service interaction and implementation of the application. The introduction of an event-graph allows the rapid development of interactive services without addressing the UI issue. An event is the means to handle the communication between a Web application and the user. Different events account for different user interaction.

The MUSA system is designed in an event-driven architecture, which is commonly used in UI environments [14]. Web applications realized with the MUSA system support multiple UIs. This is the reason why the set of employed events is small. The difficulty consists of the support of a wide variety of possible UIs to interactive Web applications. For example the graphical UI of an application intended for a desktop computer may be quite different to an UI that is appropriate for a mobile telephone with a very small display. The employed set of events is a subset of the events a wide range of UIs can implement. We found that a set of four types of events, namely navigation event, action event, notification event, and request event is sufficient for a wide range of interactive services. This set of design elements allows the writing of Web applications having a UI, which is rich enough to guarantee a seamless communication between the UI and the Web application and to employ a wide range of devices with different UI.

The architecture is shown in Figure 1. The architecture consists of three layers. The request processor layer deals with a client's request processing. The application controller administrates the event graph and the associated event processing. The application layer consists of the actual implementation of the application logic.

The communication between the Web application and the UI passes through the request processor and the application controller. The request processor is a link between the application controller and the UI and converts client requests into events, which are used throughout the Web

application. This component allows for centralizing all client specific request handling. The request processor receives a set of events from the Web application through the Application Controller. This set of events is enabled within the current dialog communicatio n. The request processor transfers the set of events to the UI. The UI prompts the current events and collects requests resulting from the user interaction. The request processor converts client requests into events and dispatches them to the application c ontroller. The application controller handles the processing of the events of the event graph. The event graph implements the navigational design, the content organization and the interaction of the application. The event graph abstracts these aspects of t he Web application. The actual implementation of the application's functionality is separated from the design and implementation of the event graph. The application controller sends the set of input events that it has received from the request processor to the Web application for processing. The Web application processes the set of events and its associated computation. In response to the event processing the application controller transfers the next set of enabled events to the application controller, whic h forwards the set to the request processor. The application controller assigns to dialog events semantic objects that encapsulate the knowledge of the event handling.

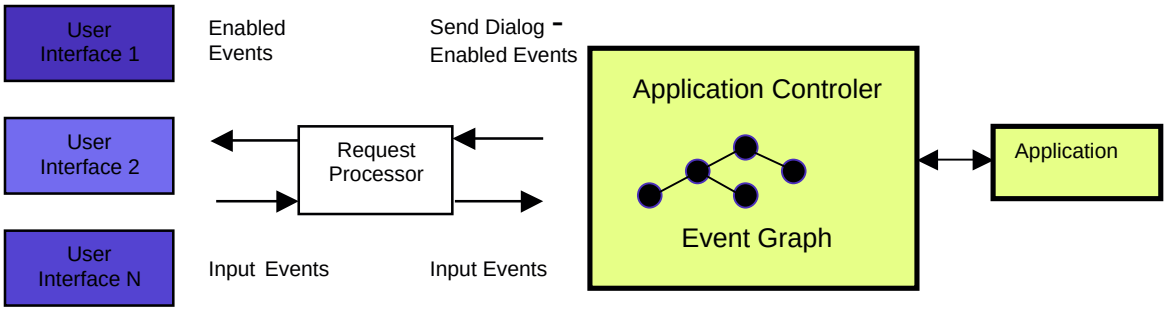

Figure 1: The MUSA Architecture.

4 The Event Graph

The int roduction of a domain specific language allows rapid development of Web applications independent of the UI. It was designed to ease the writing of Web applications. It is operationalized as an and/or graph, which concentrates on the separation of navigatio nal structure, content design, and application interaction. The introduction of the different types of events reflects the separation of these concerns. The traversal of the graph is driven by the reception of events from the UI. The graph traverses its no des and processes the events. The graph consists of

1. Dialog-nodes and
2. Sets of events. Each dialog contains a set of events.

The principal element of an interactive Web application is a dialog. The user navigates from dialog-node to dialog -node while communicating with the application. Typically a dialog spans over request, action and response. The Web application requests information from the user, the user submits a set of events, the Web application processes the events and returns a response. A dialog-node itself consists of a set of events. The set of events constitutes an element of a virtual UI. The virtual UI is finally mapped to an actual UI. The dialog -node contains a set of combinations of the following four events:

Navigation Event: The set of navigation events abstracts the navigational structure of the Web application. It indicates to the application, that the user requested to go to a specific dialog in the navigational structure of the application logic. This event navigates the user to a dialog that is associated with a specific part of the Web application, indicated by the event context. A Web application is realized as a sequence of dialogs and processes. In [1] the navigation sequence of the user is linear. The user can reach from a dialog a specific dialog of a set of child dialogs, the next sibling dialog or the parent dialog. This unnecessarily restricts the navigation. If there are no application-conditioned restrictions, the navigation event allows the navigation to every dialog within the navigation structure. The navigation event can be processed in a visible and a non-visible mode.

Visible Navigation: The visible navigation event is user-driven and part of the events which the UI prompts for user interaction. The user has to explicitly select the event to navigate to the next dialog.

Non-visible Navigation: The non-visible navigation event is application-driven and navigates the UI to the target dialog without user intervention.

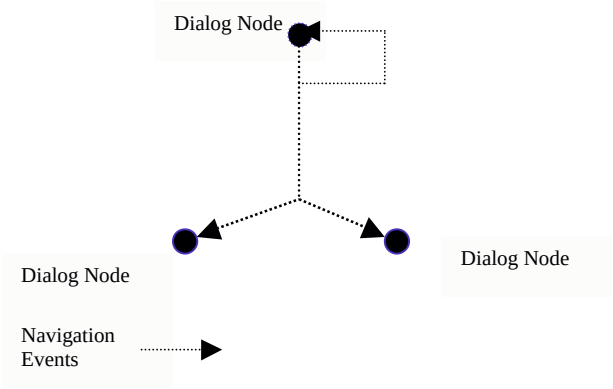

Figure 2: Event graph traversal with navigation events.

Figure 2 outlines the concept of the graph traversal. The navigation takes place as the user triggers a navigation event. The traversal occurs from a dialog node to another dialog node, or back to the same dialog node. The navigation can be automatic. In this case, the user is not aware that the application has passed an intermediate dialog, since the navigation is immediate.

Notification Event: The notification event models the content level of the Web application. It indicates to an instance of a UI, what notification or message the service requests to communicate to the user. This event transfers a message to the user. The notification event can have an action event as its child-event. The action event is embedded in the notification event to retrieve information from the Web application.

Action Event: The action event comprises the behavioral aspects of the Web application. It links the UI with an action of the Web application. It communicates with the application logic and the application's data. It changes the status of the application and manipulates the data. If the application's action needs to be parameterized, information is requested from the user by means of request events. The request events are child events and need to be evaluated in the scope of the action event. The request events represent a constraint to their parent events and require to be successfully evaluated before a parent event is processed. If the application action returns data,

this data is associated to the context of the event, just as the request event constitutes part of the context of the action event. The action event can be processed in two different modes.
Visible Action Event: In the visible mode, the action event forms part of the UI and requests user interaction to be processed.
Non-visible Action Event: The non-visible action event is automatically processed without user interaction. It is triggered and processed by the Web application. If the action event requires parameterization and request events are part of the event sub-graph, the event cannot be processed in a non-visible mode.

Request Event: The request event represents the event that prompts the user for input, which the Web application demands for processing. The request event requests information from the user and verifies this data. The data is either accepted, or rejected. If the data is rejected the request event evaluates to false and has evaluated unsuccessfully. Typically the request event is a child-event of the action event. In general, information is requested in order to parameterize a process. The process is parameterized with user provided information.

Event Evaluation: The navigational structure of a Web application is represented by the set of navigation events, which navigate between dialogs. The navigation between Dialogs is called inter-dialog navigation. The inter-dialog navigation is either user-driven, in the case of a visible navigation event, or application-driven, if the navigation is non-visible. However, a dialog comprises request, user interaction and response and the intra-dialog navigation accounts for this part of the interactive communication between Web application and user. The intra-dialog communication within a specific dialog between request, action and response is done in response to event processing. An event consists of three sections, which are successively processed. Each section itself consists of a set of events.

- **Try-Section**: In this section the current event executes its associated action and its child-events, which constitute a constraint to the parent event. The Try-section is successfully processed if every child event within this section evaluates successfully. If a Try-section contains child events, the Try-section is successful, if the Try-section of each child event has been executed successfully. If one Try-section has been evaluates to false, the Try-section of the parent event has evaluated unsuccessfully.
- **Satisfied-Section**: The Satisfied-section is being executed, after the Try-section of the current event has executed successfully.
- **Violated-Section**: If the Try-section or the Satisfied-section of the current event has executed unsuccessfully, the Violated-section is processed.

After the typical cycle of intra-dialog navigation between request, user interaction and response has been accomplished, the inter-dialog navigation proceeds to the next dialog.

The concept of an event reflects the intra-dialog navigation structure. For example the action event contains a Try, Satisfied and Violated section. The Try section on the one hand and the Satisfied and Violated section on the other hand correspond to the request and the response part of a dialog respectively. If the user navigates to a dialog node containing the action event, the UI mapping of this event prompts for user interaction. If the Try section contains any request events, they prompt for information from the user as well. If the user submits the action event to the application controller, the event is processed. If the Try section evaluates successfully, the Satisfied section is processed, otherwise the Violated section. Figure 3 shows the structure of a dialog node and the set of events it contains. In this example, the dialog's set of events consists of a notification event, a navigation event and an action event. The action event is presented in more detail. It contains a Try, Satisfied and Violated section. The Try section contains as child event a request event. The request event constraints the action event and is prior evaluated

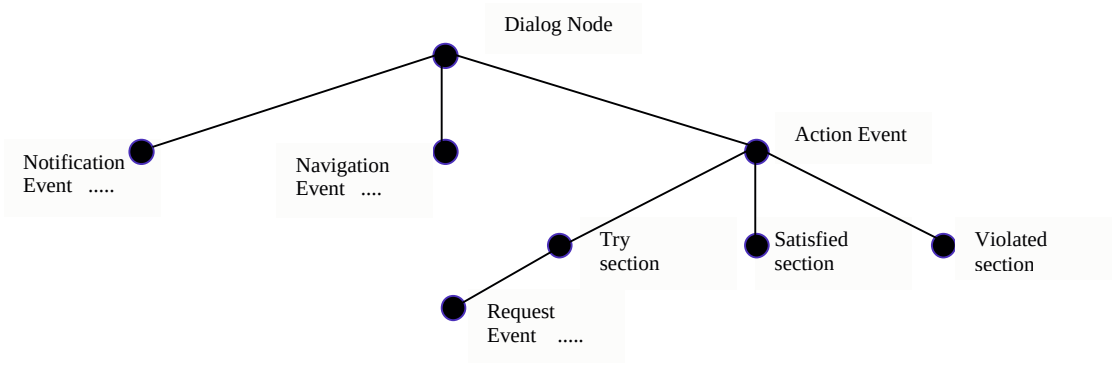

Figure 3: Dialog node with three events. Action event is shown with Try, Satisfied and Violated section.

5 Personalizing an Event-Graph Based Web Application

The World Wide Web has established itself as a new type of information and appli cation space [5]. But there is a lack of systems, which can adapt or can be adapted to individual user preferences. The accessibility of information and Web applications is completely impersonal. Every user sees the same Web application and has to adapt to the impersonal, default application space. The impersonal organization of applications has as negative consequences a reduction of productivity for the user. The user has rarely the possibility to

- Organize his personal working space, by structuring an application
- Increase productivity by submitting personal information, which he is forced to fill in very often, like his address, email address, or his name….
- To personalize specific action.

Instead the user should have the possibility to adapt the applica tion to its personal needs. The requirements of personalizing a Web application vary along two dimensions. On the one hand, the user needs to have a personalization facility for each device, with which he accesses a Web application, on the other hand for each role, in which he uses the device.

1. The user accesses a Web application on different devices. According to the computing power and the visualization capacity of the device, the user will adjust how he will use a specific application. The user will use a mail application different on a desktop computer than on a PDA without keyboard. He will probably use the PDA to only read mail, but not to write lengthy messages. On the desktop computer he will use the complete functionality of the mail application.
2. The user might use the mail application for business and for private purpose. The menu, principle address book will change subject to who is using the mail application, the user as the businessman or as a private person.

A Web application should provide the potential to support different user profiles for different UIs and multiple profiles for the same UI.
Figure 4 shows the three-dimensional personalization space. The same application can be used on different devices. For each device, there are different u ser profiles, accounting for the different usage of the application on different devices. For the same device there are different user profiles, accounting for the different role, in which a user accesses an application.

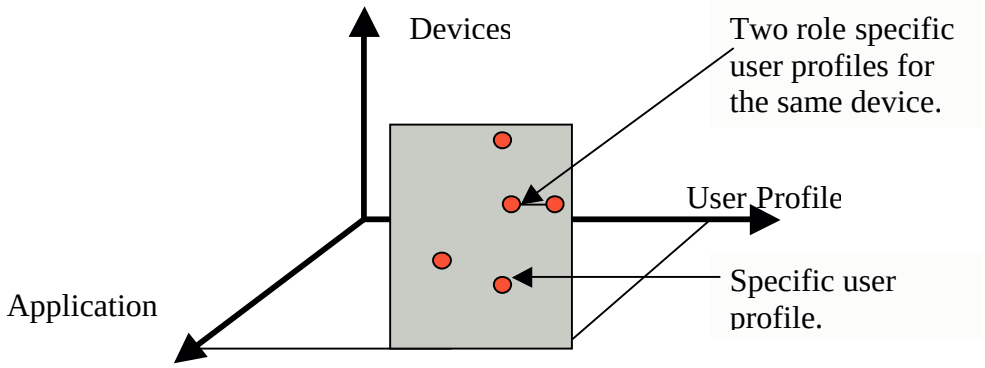

Figure 4: Three-dimensional personalization space. The same application can be used on different devices. For each device, there are different user profiles, accounting for the different usage of the application on different devices. For the same device there are different user profiles, accounting for the different role, in which a user accesses an application.

One of the key features of the MUSA system is the fact that the event graph abstracts the implementation of the application. An additional feature is, that it is interpreted, instead of being compiled as in [1]. This allows the Web application, which is realized in the event graph, to be highly dynamic.

1. The event graph can be modified without affecting the implementation of the Web application. The navigational structure and the content can be adapted to personal needs via a dedicated UI. The application logic does not need to undergo any modification. The essential point is, that a Web application designed with the event graph automatically incorporates the personalization feature. An application designer does not have to consider this aspect, but can concentrate on developing the application.
2. The event graph is interpreted and offers therefore the possibility to be modified during run-time. It can reflect immediately any changes made to him and reacts not only to modifications triggered by the user, but also to application-driven modifications. For example, the event graph changes automatically the navigational structure of the application in response to the preferences of the user.

6 Concluding Remarks

This paper presents the architecture of the MUSA system that decouples the UI issue from the Application Logic. The architecture is based on an event driven approach. Four events have been identified, which are considered sufficient for the communication between a wide range of UIs and interactive Web applications. An event-graph, incorporating the four types of event, within a navigational dialog structure has been introduced for the design and the implementation of interactive applications. Personalizing an application is becoming more and more essential, especially on consumer devices with low visualization capacities. This paper shows how it is possible to personalize an application realized as an event-graph, without incorporating the feature explicitly in the application implementation.

7 References

[1] Ball, T., Colby, C., Danielsen, P., Jagadeesan, L. J., Jagadeesan, R., Läufer, K., Mataga, P., Rehor, K., *Sisl: Several Interfaces, Single Logic , International Journal of Speech Technology* , Volume 3, Issue 2, June 2000, pp. 93-108.

[2] Bass, L., Clements, P., Kazman, R., *Software Architecture in Practice* , The SEI Series in Software Engineering, 1998.

[3] Baumeister, H., Koch, N., Mandel, L., *Towards a UML extension for hypermedia design, UML'99*. The Unified Modeling Language – Beyond the Standard, LNCS 1723, Fort Collins, USA, Springer, October 1999.

[4] Berners-Lee, T., *The World Wide Web – Past , Present and Future* . Journal of Digital Information 1, http://journals.ecs.soton.ac.uk/jodi/Artivles/timbl.html

[5] Cole, B.C. *The Emergence of Net-Centric Computing* : Network Computers, Internet Appliances, and Connected PCs, Prentice Hall, January 1999.

[6] Gamma,E., Helm, R., Johnson, R., Vlissides, J.: " *Design Patterns: Elements of reusable object-oriented software"*. Addison Wesley, 1995.

[7] Garzotto, F., Mainetti, L., Paolini, P., *Hypermedia design, analysis and evaluation issues* ; Commun. ACM *38, 8* (Aug. 1995), Pages 74 – 86.

[8] Garzotto, F., Paolini, P., Schwabe, D., HDM – *A Model-Based Approach to Hypertext Application Design*, ACM Transactions on Information Systems, Vol 11, No.1, January 1995

[9] Kaasinen, E., Aaltonen, M., Kolari, J., Melakoski, S., Laakko, T., *Two Approaches to Bringing Internet Services to WAP Devices* , The Ninth International World Wide Web Conference, 2000, http://www.www9.org/w9cdrom/228/228.html

[10] Nanard, J., Nanard, M., *Hypertext Design Environment and Hypertext Design Process* . CACM, Vol. 38, No.8, August 1995.

[11] *Oracle Portal-to-go*, An Oracle Business White Paper, 1/2000, http://www.oracle.com/ip/build/portal-to-go/P2Gsupercomm.pdf

[12] Papadoupolus, G., *The Death of Wire Protocols* , Keynote, The Eight International World Wide Web Conference, 1999

[13]Rossi, G., Schwabe, D., Garrido, A., *Designing Computational Hypermedia Applications* , Journal of Digital Information (JODI), 1(4), February 1999.

[14] Wang, K. *An event driven model for dialogue systems* . In Proceedings of the International Conference on Spoken Language Processing, volume 2, p. 393-396, Sydney, Australia. Australian Speech Science and Technology Association, Incorporated.

[15] UbiNet: *"The Ubiquitous Internet Will be Wireless"*, Computer, 10/99, pp. 128, 126-127.