Making Distributed User Interfaces Interruption-Resistant: A Model-Based Approach

Félix Albertos, Víctor Penichet, José Gallud
Computer Systems Department
University of Castilla-La Mancha
Albacete, Spain
{victor.penichet,jose.gallud,felix.albertos}@uclm.es

Marco Winckler
ICS-IRIT team
Université Paul Sabatier
Toulouse, France
winckler@irit.fr

ABSTRACT
Distributed User Interfaces (DUIs) have gone beyond the fact that traditional user interfaces run on the same computing platform in the same environment. This new interaction paradigm affects the way these novel systems are designed and developed. New features need to be taken into account from the very beginning of the development process and new models and tools need to be considered for the correct development of interactive systems based on DUIs. The starting point of this paper is that DUI-based systems are susceptible of being interrupted in several ways as they are dependent on connectivity. In this proposal this issue is assessed from a conceptual point of view, asking the question of what new features should be considered and how should they be included within the development process? The model-based approach presented provides developers with means to make DUIs resilient or resistant to interruptions.

Author Keywords
Work interruption, caching modeling, model-based approach, distributed user interface.

ACM Classification Keywords
H.5.m. Information interfaces and presentation (e.g., HCI): Miscellaneous.

INTRODUCTION
Connectivity is one assumption that in the last years has been commonly assumed. But there are a lot of scenarios where this assumption may not be totally guarantee. Travelling by flight, natural disasters, areas where the connection is not available or limited, are only a small set of scenarios where the lack of connection has to be taken into account. When developing systems connected through the Internet, deal with disconnection problems is an open issue, and have to be included when modeling these systems. Some examples of systems susceptible of being interrupted are those based on the Distributed User Interfaces paradigm.

A DUI is a user interface whose components are distributed across one or more of the dimensions input, output, platform, space, and time [2]. The interface may be restricted to the same physical (and geographic) space, or can be distributed geographically. In the last scenario, interfaces may use the client-server paradigm to get connected. This paradigm is followed by the web-based distributed user interface [6], where Internet is used to connect the DUIs. Following the paradigm of the web, a connection to the server must exist in order to interact with the system. But, what happen if there is no connection to the server? In this scenario, an interruption due to the lost of connectivity to Internet arises this question: are users able to continue interacting with the system without being affected by the system interruptions? Our aim is to make DUIs resilient to interruptions, providing continuity of service for DUIs connected over the Internet, dealing with the problematic of interruptions, as shown in the Figure 1. The term resilient is often used to address systems that are able to recover from failures, but in the present context it is used to qualify systems that can prevent from the occurrence of interruptions, help users to resume from interrupted tasks, and/or ensure a minimum level of service for performing a task despite of the interruption [8] in DUIs environments.

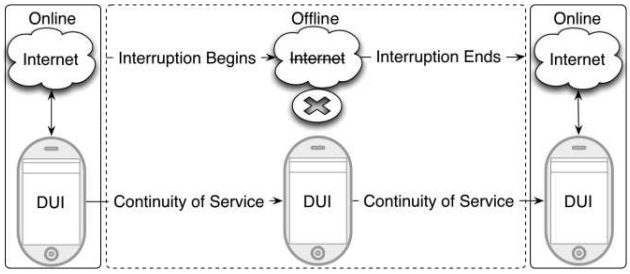

Figure 1: Providing Continuity of Service.

STATE OF ART
McFarlane [7] proposed a general, interdisciplinary, theory-based definition of human interruption, which says that human interruption is "the process of coordinating abrupt changes in people's activities." Interruptions occur when a

person is working on a primary task (usually long-lasting) and an alert for a secondary task occurs.

There are previous works to systematically detect and deal with interruptions when doing task on interactive systems [8]. They try to model and to predict the impact of interruptions on those systems.

There are some DUIs approaches susceptible of being interrupted due to the lack of connectivity. In [4] is presented DUIs to enhance decision making in disaster situations. One of its main goals is to be fault-tolerant and to take into consideration the requirements from all stakeholders. Therefore, a lot of devices and architectures must be supported, and the following of international standards may be welcomed to design the system that supports them, such us the HTML standards, widely adopted.

HTML5 is the latest version of the HTML standard. One of the new ideas present in the Web development on these days is a set of specifications known as "Web Applications" [9]. The goal of this specification is to enable improved client-side application development on the Web. Due to the variety of scenarios where Web Applications can be used, new ways are needed to support the development of applications. One of the aspects to take into account is the network connection availability or the interruptions while using Web Applications. In order to enable users to continue interacting with Web Applications and documents even when their network connection is unavailable authors can provide a manifest that lists the files needed for the Web Application to work offline and which causes the user's browser to keep a copy of the files for use Offline Web Applications [10]. Other set of technologies allows Web Applications to store locally information. Keeping a copy of the files used in Web Applications is not always enough to make them interruptions resilient. Webstorage introduces two related mechanisms for storing name-value pairs on the client side. However, it does not provide in-order retrieval of keys, efficient searching over values, or storage of duplicate values for a key.

Earlier works has identified the problem of disconnection in Web environments. They proposed models to deal with this issue, such us [1, 3, 5]. The main drawbacks on these works are that most of them don't use standard technologies, restricting the target platforms to use, and the difficulties or the impossibility to be adapted to existent Web Applications. Also, the main functionalities are focused in only catching the information locally, not to deal with the consequences of the interruption. Finally, most of them are hard to implement and involve the programmatically implementation of the provided features.

MAKING WEB APPLICATIONS RESILIENT TO INTERRUPTIONS

When designing Web Applications there are several model-based approaches, as aforementioned. But our proposal deals with an issue that has been neglected when modeling Web Applications: how the application behaves when it is interrupted and how to recover from interrupted work. The proposed conceptual model-based approach combines explicit representation of end-user navigation and local information storage. It provides users with information about which Web site's contents are available when they are interrupted and how they can get easy access to local cache content.

The proposed model represents the *static* properties of a Web site, as well as it *behavior*. The *static* properties define the structure of the Web site. The *behavior* defines how the system will react to the events and how it will change over the time. Web pages are the basic elements in the World Wide Web. They are also the basic elements in the offline model. They are defined as *nodes*. A *node* is an element in the model that may be connected to other *nodes*. They are connected to each other to conform what is known as Web projects. A *Web project* is a superset of *nodes* and it is the upper level of abstraction in the model. We can refer Web projects as Web sites.

The model shows both the *static* properties as well as the *behavior* of the Web site. To represent these properties, nodes have associated properties, represented as *node states*. We propose three state levels for *nodes*: *static*, *navigational* and *data*. *Static* state is defined according to the requirements of the site and the site structure. Static states are *normal*, *precacheable*, *nocacheable*, *initial* and *external*. *Navigational* state changes over the time when users use the Web site. *Navigational* states are *novisited* and *visited*. *Data* states are the result of the combination of the two previous states. Possible values for data states are *cached* and *nocached*. ***Static*** states are defined to answer the following questions. The first question is: is the node internal or external? External nodes are set to *external* state. This state excludes other static states for the node. The second question is: is the node initial? Initial node is set to *initial* state. It will be always accessible in offline mode and *precacheable*. *Initial* nodes are unique within the Web project. The third question is: will be the node cached when the Web site is visited? *Precacheable* nodes will be always cached when the Web site is visited, but *nocacheable* nodes will not be cached ever. ***Navigational*** state is set when users navigate through the site. It changes when the site is used. It is a dynamic state. Have been defined two states: *novisited* and *visited*. The question answered with this state is: has been the node visited? When a node has been visited, it is set to *visited*. Meanwhile, the node is set to *novisited*. ***Data*** state is the result of the combination of the two previous states. As a result, a node can be *cached* or *nocached*. When a node is *cached*, it will be available when the site is interrupted. When a node is *nocached*, it wouldn't be available when the site is interrupted.

The model introduces means to deal with interruptions due to offline navigation. One of the mechanisms used to

support offline navigation is the transformation of the elements of Web pages. These transformations may act removing or altering the content of Web pages. Available transformations are *element disabling* and *alternative link destinations*.

When using Web pages, some of the elements may not be available for users in offline mode. This restriction may be due to several reasons. One of the reasons could be that part of the Web page requires a connection with some external resource. Since the fact that there is no connection to the server, the element wouldn't work. An example of this scenario is when a form is used to send information to a remote server. Other scenario is when linking to an external resource. Since the site is in offline mode, the action couldn't be performed. Another reason for disabling an element is when it shows information retrieved from an external server. An example of this scenario is when using Web pages to show online maps or *Facebook* walls. To overcome these situations, the model allows *element disabling*. Through this technique, any element in the Web page could be disabled, belonging it to our server or to an external server, preventing it to be presented in the Web page when it is in offline mode. Since Web pages are described in HTML and most elements can be nested, when disabling an element, all the elements enclosed within this element will be disabled too. Another available transformation is the *alternative link destination*. When using the Web in offline mode, some destination will not be reachable due to lack of connectivity or for design constraints. To prevent the problems associated with the lack of connectivity and to support the design constraints, the model allows giving an alternative destination to any link in Web pages. As a result, when in offline mode, alternative links will work instead of the original.

INTERRUPTION-RESILIENT DUI APPROACH

In the Web Application's Offline Model the properties of the system define the behavior of Web Applications. The main elements are the Web pages that form the Web site and the HTML elements that describe those Web pages. They are independent of each other. Links are the only connection between Web pages. But within DUIs, there is an important issue to be addressed. When an interruption occurs on DUIs environment, it is not only affected the actual interface being displayed, a DUI, but also is affected the overall DUI system. For example, it is not the same a DUI that only show information than other DUI that is used to input important data within the Web Application. The first one may not be critical for the overall system; meanwhile the second may be vital for the overall system task. Therefore, new mechanisms have to be introduced to deal with this kind of scenarios.

To make DUIs resilient to interruptions we have included in the Web Applications Offline Model the DUI Offline Model. In this Model, DUI-based systems have two kinds of elements, *soft* and *strong*. A *soft element* in a DUI system is a non-critical element of the UI (called *soft DUI*). A *strong element* in a DUI system is an element that host critical information (input or output) and interruptions can affect the system (called *strong DUI*). Each type of DUI behaves depending of it connection status: online (non-interrupted) or offline (interrupted), and specific mechanisms are provided to deal with interruptions. These mechanisms, allow users to interact with interrupted DUIs and to synchronize offline work. Mainly these mechanisms provide caching features, content removal and transformation and change link destinations.

The first type of DUIs is the *Soft DUI*. These DUIs do not have a special behavior in the model. From the point of view of the overall system, when they are not connected is assumed that they have been disconnected from the system. From the point of view of these kinds of DUIs, they do not provide any special mechanism to deal with the interruption. These DUIs are represented in the model with dotted lines, as depicted in the Figure 2.

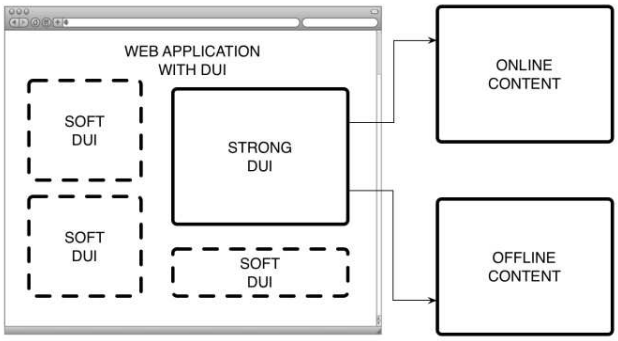

Figure 2: Content transformation on *strong DUIs*

The second type of DUIs is the *strong* DUI. These DUIs are provided with mechanism for offline operation when they are interrupted. They are represented in the model with solid lines, as depicted in the Figure 2. From the point of view of the overall system, the connection interruption with a *strong* DUI doesn't means that the DUI has been disconnected from the system. To disconnect the DUI there must be an explicit disconnection operation. As a consequence, within the status of this type of DUIs, the connection status must be represented in the model. When the *strong* DUI is created, it is represented within the system with two possible statuses: *online* or *offline*. *Online* indicates that the DUI has not been interrupted meanwhile the *offline* status shows that the DUI is interrupted, but has not been an explicit disconnection from the overall system. From the point of view of *strong* DUIs, they are always cached for offline operation. The mechanisms involved in this task have been described previously in the Offline Model for Web Applications. When an interruption arises, end users can use the DUI and the mechanisms for offline work have to be defined, as follows. During the interruption, the content is susceptible of been transformed for its operation, as depicted in the Figure 2. Available

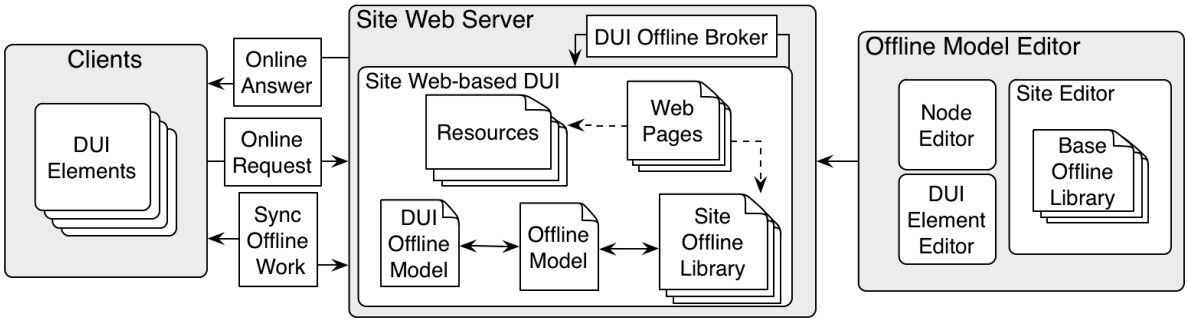

Figure 3: Overall architecture of the offline model approach for DUI Web Applications.

transformations are *element disabling* and *alternative link destinations*. These transformations have been described within the Offline Model for Web Applications.

The architecture of the offline model approach for DUI Web Applications is depicted in the Figure 3. The architecture includes the DUI Offline Model. It contains information about the DUIs elements, such us the type of DUIs that conform the system and the content transformation policies.

CONCLUSION AND FUTURE WORK

In this proposal, interruptions of DUIs in Web Applications are assessed since a modelling perspective, which constitutes an extension of the aforementioned Offline Model. The main goal is to provide with means to make DUIs resilient to interruptions. The provided conceptual mechanism includes the definition of two types of DUI elements, *soft* and *strong* for dealing with interruptions. Also, useful information is presented to describe the behavior of DUIs in Web Applications. As future work, we are working on the creation of a proxy server to annotate web applications on the fly. With this feature, the Offline Model will be included without modifying the original Web Application, simplifying the Offline Model management.

ACKNOWLEDGMENTS

This research has been partially supported by the CICYT TIN2011-27767-C02-01 project and the regional project with reference PPII10-0300-4174.

REFERENCES

1. Bret Cannon. 2011. Minimizing Resource Access and Management Disparities Between Desktop and Web Applications. PhD thesis, January 2011.
2. Elmqvist, N. Distributed User Interfaces: State of the Art. Workshop on Distributed User Interfaces 2011 (DUI) at the 29th ACM CHI Conference on Human Factors in Computing Systems 2011, ISBN: 978-84-693-9829-6, Vancouver, Canada, May 7--12, 2011.
3. E. Goncalves and A. M. Leitao, “Implementing offline work in web applications for rich domains,” in Proc. of the 11th Int’l Symposium on Web Systems Evolution (WSE). IEEE, 2009, pp. 79–82.
4. M. Heupel, M. Bourimi, D. Kesdogan, T. Barth, P. Schwarte and P. G. Villanueva. Enhancing security and usability of DUI based collaboration with proof based credential systems. Proceedings of the Distributed User Interfaces 2012 CHI Workshop, held in conjunction with 2012 CHI conference, 23-26, ISBN-10: 84-695-3318-5.
5. Yung-Wei Kao, ChiaFeng Lin, Kuei-An Yang, and Shyan-Ming Yuan. 2012. A Web-based, Offline-able, and Personalized Runtime Environment for executing applications on mobile devices. Comput. Stand. Interfaces 34, 1 (January 2012), 212-224. DOI=10.1016/j.csi.2011.08.006 http://dx.doi.org/10.1016/j.csi.2011.08.006.
6. C. Lee, F. A. Musyaffa, Y.-M. Kwon, "Collaborative Social Authoring Technology Using Web-based Distributed User Interface (DUI)," Int'l Conf. CHI 2012, Workshop on DUI (Distributed User Interfaces), May 2012.
7. McFarlane, D. C. (1997). Interruption of people in human-computer interaction: A general unifying definition of human interruption and taxonomy (NRL Formal Report NRL/FR/5510–97–9870): Naval Research Laboratory, Washington, DC.
8. Philippe Palanque, Marco Winckler, Jean-François Ladry, Maurice H. ter Beek, Giorgio Faconti, and Mieke Massink. 2009. A formal approach supporting the comparative predictive assessment of the interruption-tolerance of interactive systems. In Proceedings of the 1st ACM SIGCHI symposium on Engineering interactive computing systems (EICS '09). ACM, New York, NY, USA, 211-220. DOI=10.1145/1570433.1570473 http://doi.acm.org/10.1145/1570433.1570473
9. W3C. Web Applications Working Group Charter. http://www.w3.org/2012/webapps/charter/ (Last access Apr. 2013).
10. WHATWG. Offline Web applications. http://www.whatwg.org/specs/web-apps/current-work/#offline (Last access Apr. 2013).