Automatic Form-Factor Transformations of Web Pages and Web Services UIs

by

Piotr Lubinski

LIU-IDA/LITH-EX-A--09/025--SE

2009-04-30
Final Thesis

Automatic Form-Factor Transformations of Web Pages and Web Services UIs

by

Piotr Lubinski

LIU-IDA/LITH-EX-A--09/025--SE

2009-04-30

Supervisor: Erik Berglund
Examiner: Erik Berglund
# Table of content

Abstract ................................................................................................................. 4  
1 Introduction ......................................................................................................... 4  
1.1 Motivation ....................................................................................................... 5  
1.2 Focus of the thesis ............................................................................................ 6  
1.2.1 Research questions ...................................................................................... 6  
1.2.2 Thesis outline ............................................................................................... 7  
1.2.3 Thesis contribution ......................................................................................... 7  
2 Split UI transformations: definitions and related work .......................................... 8  
2.1 Research domain ............................................................................................... 8  
2.2 Terminology used in this thesis ......................................................................... 9  
2.3 Web4CE standard definition ............................................................................ 9  
2.4 CE-HTML standard definition ......................................................................... 10  
2.5 Related work ................................................................................................... 12  
3 User interface and interaction approach ................................................................ 14  
3.1 User interaction approach ................................................................................ 14  
3.1.1 Content and experience sharing ................................................................... 14  
3.1.2 Exploring web content .................................................................................. 15  
3.2 User interface .................................................................................................. 15  
3.2.1 Control A/V playback while browsing the Web .......................................... 15  
3.2.2 Control TV playback in a lean-back position ............................................. 16  
3.3 Web page content summarization interface ..................................................... 17  
4 System architecture .............................................................................................. 20  
4.1 Split UI system for living room environment .................................................... 20  
4.2 Network node type ........................................................................................... 21  
4.2.1 Server/client architecture ............................................................................ 21  
4.2.2 Web proxy approach .................................................................................... 22  
4.3 Transforming node internal structure ................................................................ 23  
4.3.1 HTTP server ................................................................................................. 24  
4.3.2 Accept Type Detection ................................................................................ 25  
4.3.3 HTTP client ................................................................................................. 25  
4.3.4 Content-Type Detection ............................................................................. 26  
4.3.5 Document initial scan .................................................................................. 26  
4.3.6 Extended transformation setup ..................................................................... 28  
4.3.7 File and cascading stylesheet repository ..................................................... 28  
4.4 Communication scheme for split UI message exchange .................................... 28  
4.5 Conclusions ..................................................................................................... 29  
5 Split UI home system notification ......................................................................... 32  
5.1 Enabling for notification message exchange mechanism ................................... 32  
5.1.1 Notification object registration ................................................................... 33  
5.1.2 Capability exchange mechanism ................................................................. 34  
5.2 Notification messages distribution characteristics ............................................. 36  
5.3 In-session notification ....................................................................................... 37  
5.4 Video and audio media player .......................................................................... 37  
5.5 Conclusions ..................................................................................................... 38  
6 Conclusion .......................................................................................................... 39  
7 Discussion and future work .................................................................................. 40  
8 References .......................................................................................................... 44  
Appendix A ............................................................................................................ 46  
Appendix B ............................................................................................................ 48  
Appendix C ............................................................................................................ 50
Abstract

Browsing the Web on a distant screen may very often become user’s unpleasant experience. Such a situation may happen in a living room environment, where the user attempts to browse the Web on TV sitting at 10-foot distance from the screen. In a vast number of web sites that the user is able to request, many may contain, for instance, tiny text characters, which would become unreadable when watched from the distance. On the other hand, also navigation through the Web offerings can become troublesome. Especially, when the user is only equipped with the commonly used remote controller, interaction with the displayed content may prove inefficient. Therefore, methods to overcome content visibility and navigation problems become challenging.

In this thesis we perceive living room environment as a good scenario, in which web-enabled handheld devices could further be utilized to interact with Consumer Electronics products, such as TVs. This sort of interaction should make an advantage of publically available Web resources to have them displayed on TV and controlled from the distance by the user who is equipped with, for instance, a web-enabled mobile phone. We think that such an interaction approach would become beneficial in several ways, making the Web browsing routine an interactive, possibly a group activity.

Discussed in this thesis solutions are the first step of automatic transformations of web content into a form that can be rendered on Web4CE (Web for Consumer Electronics) products. A particularly important aspect of the proposed system is the concept of request/response HTTP message processing and HTML modification. According to this approach, living room browsing activity should be achievable without the need to re-author already existing web sites.
1 Introduction

Recently we have experienced a rapid expansion in the number and variety of the devices capable of utilizing the Internet resources. In the light of current developments, this expansion indicates that the Internet and its lately famous pillar, the Web 2.0 [24], are becoming more and more omnipresent nowadays. At the same time also Web 2.0 applications offer the users more opportunities than ever before. They allow for self-expression [18], entertainment and content sharing [19], social networking [20][21], on-line shopping [22][23] and many more. For this reason, we think that it would be an interesting alternative to have these applications used on various types of web-enabled products and in different usage scenarios. One such an interesting usage scenario has been found in a living room environment.

Consumer electronics (CE) devices are available in almost every living room establishment. TV set belongs to consumer electronics family and so, in terms of browsing the Web, it occurs to be an interesting representative of CE family [2]. Using TV as a web browsing platform has several advantages, mainly due to sharing Web content with a broad living room audience on a large screen. Depending on the content that is being viewed, such an interaction approach may benefit in several ways.

However, using internet applications on TV screen has some limitations. Unlike the classical PC paradigm, where the user usually remains in front of the display, browsing the Web at 10-foot distance may seriously impact web site content visibility. As a result, reading tiny text characters or watching small images on a TV screen may become the user's unpleasant experience. What is more, for full interaction with web content, the user is also expected to be equipped with a keyboard and/or a mouse pointer. In contrast to these devices, commonly used TV remote controllers are only capable of transmitting limited set of control signals towards the TV browser. Therefore, the remote controllers prove to be inconvenient navigation tools in this situation. In this case, finding an efficient method for distant interaction between the user and TV browser becomes challenging.

1.1 Motivation

In order to address the limitations of living room browsing, split in-browser web interface is further developed in this thesis. The idea behind the concept is to allow the user to browse the Internet resources on multi-device platform, which bundles functionality of web-enabled handheld device and TV.

In the context of distant screen browsing, such an interaction concept has several advantages. Firstly, splitting web interfaces does not require the user to browse the Internet on the distant TV screen only. Instead, being equipped with the handheld device, the user gains an opportunity to watch some of the web offerings on a close-up display. This solves out the 10-foot distance visibility problem. Secondly, due to web transformation operations being applied to every requested HTML document, the user also gets a chance to “redirect” some of requested web content from the handheld device onto a large TV screen. This sort of
functionality becomes possible since, in the investigated system, the handheld device is able to communicate with the TV browser via the wireless link. Thirdly, in the majority of cases, when navigating through the Web offerings on a distant TV screen, handheld device with the split interface gives the user more control over the interaction than the classical remote controller. Thus, for example, an attempt for text-based information input becomes easier and more intuitive in respect to 10-foot distance browsing. Also, the handheld device via its interface (e.g. touch screen) may offer the user more intuitive way to “click and share” some of the web site offerings (Section 3). This, in comparison to classical remote controllers, proves to be an important system advantage.

Splitting web site interfaces gives an alternative solution to the developers to overcome the problems concerning 10-foot distance browsing. Since, many advantages have also been found in living room browsing activity, automatic form-factor transformation of web pages and web services user interfaces remains an interesting thesis topic to investigate.

1.2 Focus of the thesis

In this thesis, the problem of automatic form-factor transformation of web pages and web services user interfaces is addressed. For this reason, various methods and tools to perform web document modifications are investigated.

1.2.1 Research questions

Splitting web sites user interfaces among TV and handheld device requires some principles of the design work. In this thesis, the design activity should result in a new type of web interface that will customize web offerings to display them on the proposed multi-device platforms. For this reason the first research question is addressed:

1. In which way should original web site user interface be transformed to allow for split UI browsing in the living room environment?

Since the system implementation largely relies on HTML document transformation and HTTP request/response time message processing (Section 4.4), efficient transformation methods and algorithms need to be investigated. These algorithms should become an integral part of software (middleware) that resides between the user, TV and the public Internet. This problem leads to another research question to be addressed in this thesis:

2. What sort of transformation middleware is required to allow for real-time transformation of the web documents into a form that would allow for split UI browsing interaction?

Once the answer to this question is found, it is important to think of the notification “strategy” that would allow for communicating handheld device with a TV. Therefore, another research question arises:

3. What sort of functionality needs to be implemented inside the split UI system to allow for efficient communication between the handheld device and a TV?
1.2.2 Thesis outline

The rest of this thesis is structured as follows:

In 2nd Chapter, the domain for split UI research study is described. As the background knowledge, the two major frameworks concerning web technology adaptation onto CE devices are introduced. At the end of the second chapter, a few examples of work that already contributes to the area of web document transformation are shortly discussed. In 3rd Chapter, a few sample user interaction approaches for split UI system are given. This includes general concept description, close-up user interface design as well as introduction to web sites content summarization approach. In 4th Chapter, system architecture design for split UI transformation is proposed. Suggested middleware architecture works accordingly to processing pipeline. For this reason, a few subsections of the 4th chapter describe the internal modules of the architecture. At the same time 4th chapter also shows how each internal module depends from another in the processing pipeline. In 5th chapter, a notification system for exchanging messages between the user's handheld device and the TV browser is described. Chapter 6th summarizes this master thesis project, while chapter 7th shows several courses, in which the designed system can further be developed.

1.2.3 Thesis contribution

The main contributions to the research described in this thesis are:

- Analysis of methods for browsing the Web in a split user interface mode,
- System architecture design for real time HTML document transformation and content sharing control,
- Method for web audio and video playback control on a distant TV screen investigation and concept description.
2 Split UI transformations: definitions and related work

In order to classify split user interface web sites transformations into a certain research field, Web Engineering domain is introduced. For this purpose, Section 2.1 describes certain dependencies between Web Engineering and proposed in this thesis web transformation approaches. Section 2.2 presents the terminology that is used in this thesis. Section 2.3 focuses on Web for Consumer Electronics (Web4CE) – the framework that contributes to adapting web onto consumer electronics devices. Section 2.4 introduces CE-HTML standard. This extension to popular and widely used hypertext markup language suggests novel solutions and tools for web applications’ content adaptation onto CE products. As a conclusion to this chapter some more related work is highlighted in Section 2.5.

2.1 Research domain

Described in this thesis user interface transformation approaches require a multi-domain problem analysis. In the first place, the area of expertise contributes to web development. This, in principle, involves certain techniques commonly used for web site design and development. An important aspect, to which we will very often refer to, is DOM (Document Object Modeling) representation of web document structure. According to DOM definition [14] well represented HTML code (with a little support of for instance JavaScript programming language) can freely be “rearranged” in pieces to eventually lead us slowly towards the ultimate goal – the web sites’ transformations. In this thesis project equally important aspects of the design are also the cascading stylesheets (CSS) – mainly due to the fact that together with HTML code, they describe the layout of the web sites. Thus, at this point, the reader is expected to have a firm understanding of DOM, HTML, JavaScript and the cascading stylesheets.

Nonetheless, web development techniques used alone are not sufficient enough to fully and automatically transform web documents. In this thesis they are rather considered as supporting tools, which in cooperation with the transformation engine (Section 4) allow for ultimate user interface modifications. For the purpose of the transformation engine implementation, any programming language that strongly supports networking techniques\(^1\) can be used. It is clear by now that the topic of automatic transformations of web pages and web services UIs will also be related to certain aspects of software development.

Since solutions suggested in this thesis work require studies of at least two fundamental computer and information science domains, it is essential to define a common scientific field for them. Web Engineering is relatively new area of studies that “(...) actively promotes systematic, disciplined and quantifiable approaches towards successful development of high-quality, ubiquitously usable Web-based systems and applications (...)” [24]. In particular, Web Engineering is [4]:

\(^1\) http server-client architecture, multithread applications, HTML parsing, TCP support
“A multidisciplinary field, with encompassing inputs from diverse areas such as human-computer interaction, user interface, systems analysis and design, software engineering, requirements engineering, hypermedia engineering, information structures, testing, modeling and simulation and project management, as well as social sciences, arts and graphic design.”

Taking all the above given guidelines into consideration, in this thesis, split UIs transformation approaches are considered to be parts of Web Engineering research domain.

2.2 Terminology used in this thesis

- **Content (and experience) sharing living room activity**: A living room browsing activity that involves more than one person in watching requested World Wide Web offerings on a TV set screen,
- **Split user interfaces (split UI)**: an interface that facilitates content and experience sharing activity,
- **User interface transformations**: a set of operations performed upon web document to enable split UI browsing,
- **Transforming node**: A “gateway” to the public network that controls web document exchange and allows for web document transformations,
- **Transformation engine**: an integral part of transforming node that performs core operations upon web documents in request/response processing time.

2.3 Web4CE standard definition

Adapting web technologies onto consumer electronics devices remains a fundamental research field of Web4CE framework. This standard “(…) defines necessary mechanisms to allow user interface to be remotely displayed on and controlled by devices or control points other than the one hosting the logic(…)”[1]. Web4CE framework proposes certain techniques for adjusting existing web services, protocols, mark-up languages and related technologies to serve in different usage scenarios on consumer electronics products. The main goals of Web4CE specification are based on the following principles [1]:

A. Provide a mechanism that allows remote presentation and control of user interfaces directed to consumer devices.
B. Handle remote UI content that may reside either:
   - on UPnP (Universal Plug and Play) UI server devices or,
   - on Internet-based services.
C. Use existing web standards for UI content (e.g. xHTML)
   - Define extensions only when necessary.
D. Support a variety of different consumer client devices
   - STBs (set-top boxes), TVs, mobile phones,
E. Allow for dynamic interaction between Remote UI Clients and Servers
• Provide timely, partial UI updates from a Remote UI Server,
  F. Allow clients to receive important UI notifications from server devices at any time (if permitted by the user),
  G. Allow for A/V (audio/video) functionality to be part of the UI experience.

Proposed in this thesis solutions reflect Web4CE framework’s requirements and goals. Firstly, due to the fact that web content is presented on distant TV screen and controlled from the handheld device, principle (A) is met. Secondly, taking into consideration that proposed transformation engine interacts with the resources available publically on the Internet, principle (B) is also met in the design. Thirdly, web transformation techniques being applied to HTML documents use well known web development standards and/or limited set of extensions, therefore, meeting principle (C). Fourthly, because web transformation involves at least handheld device and TV in interaction, principle (D) is also satisfied. Fifthly, since dynamic interaction between client (TV set, user's web-enabled device) and server (an integral part of transforming node) is required in the design and eventually lead to UI notification updates sending between interested parties, principles (E) and (F) are satisfied also. And finally, meeting principle (G) is in fact the major focus of this thesis.

2.4 CE-HTML standard definition

An extension to XHTML mark-up language that directly addresses web sites’ adaptation onto consumer electronics devices is known as CE-HTML (Consumer Electronics – HTML). Features and mechanisms for CE-HTML include [5]:

• Mechanisms for dealing with a wide variety of CE-devices, such as capability exchange and removing possible ambiguities between different browsers,
• Mechanisms to deal with Remote Control / TV-based interaction,
• Mechanisms to deal with two-way communication and dynamic page updates, which are important for many of the use cases that were considered during the development of CE-HTML,
• Mechanisms to be able to integrate A/V as part of the UI, for example to control the size and play-speed of a video and to deal with overlaying (parts of) the UI on top of a video,
• Mechanism for migrating a UI session from one device to another.

CE-HTML framework can also be perceived as a collection of different existing standards, such as XHTML, CSS TV profile and DOM level 2, together with various extensions. In respect to CE-HTML, this thesis suggests certain ways in which standard extensions could be utilized for adapting web UI onto specific Web4CE products. Further extensions and tools that take advantage of CE-HTML standard are also considered. Figure 2.1 shows CE-HTML place inside the realm of web standards:
Except technologies commonly used by W3C and WWW specs, particularly important definitions are included in CE-HTML standard and remains an important aspect of this thesis. This includes 3 fundamental terms of CE-HTML bubble (Figure 2.1):

- **Notifsockets:**
  - “A method for in-session notifications uses a Persistent TCP Connection for both the home domain and the Internet domain. In this method, a Remote UI Server MAY send messages (e.g. text or text-encoded binary messages) to an Remote UI Client at anytime for event notification” [1],
  - “A polling based method for in-session notifications using XMLHttpRequest for the Internet domain” [1],

- **Capability exchange mechanisms for in-session cross-device notification,**

- **A/V object:** “Rendering and controlling A/V as part of a web page is usually done through plugins. CE-HTML can support such plugins and indicate their existence inside the UI capabilities it send to the server” [5].
2.5 Related work

Web content transformation and using web services beyond PC paradigm have already been addressed in numerous papers. Contributed work has been studied and as a result several concepts have been used in this thesis research.

Content and experience sharing on Web4CE devices are introduced in [2]. According to Lawet and Shrubsole, numerous advantages are found in content and experience sharing activity. They stress the important meaning of terms “Presence and instant messaging”, “Video chat”, “Picture apart together”, “Co-browsing” and “Any application apart together” and find these activities very suitable for browsing on Web4CE devices in relaxing living room lean-back position. To facilitate content and experience sharing, they suggest several techniques that are necessary to achieve such interactive activities. These include major technologies that are fundamental for Web4CE and CE-HTML frameworks (XHTML, DOM level 2, CSS TV profile [13], AJAX [15]). Even thought solutions proposed in [2] are dedicated to interaction between distant homes locations, certain fields of Lawet and Shrubsole research field are taken into consideration and treated as guidelines in this research work.

Accessing web resources is also a research topic of [3]. Discussed solutions are related to certain techniques for exchanging control messages between consumer devices and web services. The most important among these techniques are: service discovery, capability exchange, UI and A/V content, In-session notification, out-of-session notification, remote control / TV-based interaction and migrating sessions. In this thesis work important aspects inspired by Dees and Shrubsole are mechanisms for service discovery, capabilities exchange, A/V playback control and in-session notification methods. Even though the major focus of this thesis is an interaction approach between web-enabled handheld device and the TV in a living room environment, certain methods introduced in [3] for browsing the web with limited navigation capabilities are considered as interesting system alternative in this thesis and they will be further discussed in one of the following chapters.

A few concepts concerning web document transformations have also been discussed both in [7] and [6]. Proposed web adaptation approaches suggest transcoding proxies for intermediate web sites content adaptation.

[7] sets a focus on real-time HTML code transformation into a form that can be rendered on wireless devices, like the mobile phones. Suggested in [7] solutions require XHTML to XML document conversion. In a later process special tags identification takes place and representation of a web site based on retrieved information WML (Wireless Markup Language) is created. Therefore, a processing pipeline inside the transcoding proxy is involved in three stage original document modification.

In [6] further extensions to suggested transcoding proxy approach are discussed. These include interesting solutions for making the transcoders not only statically, but also dynamically adaptive. In the lights of Fialal’s and Houben’s studies, the user’s feedback has an important meaning in dynamic web sites transformations. This feedback is stored locally in a context data repository and includes tracking history of the user’s behavior during the Internet browsing activity. This approach is known in [6] as GAC (Generic Adaptation...
Component). Therefore, using GAC concept for split UI transformation remains an alternative for project future development.

Transcoder approach for making web sites adaptive is an important approach for split UI transformations research study. Transcoding proxies, together with various extensions of Web4CE and CE-HTML are recognized in this thesis as theoretical background for further project work.
3 User interface and interaction approach

In this chapter we will consider several use cases, in which split UI browsing can be applied. In addition, we will also present some custom created user interfaces that are designed specifically for the purpose of split UI browsing. For this reason, in Section 3.1, the user interaction approach with split UI system is described. In Section 3.2 a close-up interface design is introduced. In Section 3.3, an approach for web site content summarization is discussed. As it turns out, this approach for web site visual modification provides numerous benefits to existing concepts of large screen 10-foot distance interaction.

3.1 User interaction approach

3.1.1 Content and experience sharing

In this scenario living room browsing becomes a group activity. One person is using a web navigation device (such as a web-enabled wireless tablet or a mobile phone) to select and share certain Web content. Triggering split user interface events\(^2\) is assumed to be intuitive and straightforward task.

In **content and experience sharing activity**, one person is always responsible for interaction with the handheld device. In a meantime living room audience observes TV screen offerings from the distance (Figure 3.1).

---

\(^2\) exporting and controlling the content remotely on a TV
What can be perceived as an interesting advantage of this use case is “telling the stories” to the broader living room audience during meetings or parties. The user involved in “story telling” is required to request the proper multimedia content uploaded beforehand to one of the Web applications (e.g. [19] or [21]). Than, sharing the content with friends or family becomes a matter of browsing the Web and exporting media to the TV screen when needed.

3.1.2 Exploring web content

Split user interface does not necessary mean content sharing activity only. An interesting browsing experience may also be achieved in a living room during regular web sites “surfing”. In this situation, not only TV screen is used for interaction with the user, but also other interconnected consumer electronics devices (sound system, lights, speakers, microphones, cameras) may cooperate with each other to create much different web browsing experience (Figure 3.2).

3.2 User interface

3.2.1 Control A/V playback while browsing the Web

One of the proposed system features is to enable the user to watch web A/V tracks on a distant TV screen. In such a case every handheld device attempt to request A/V content is intercepted by split screen transformation engine and processed accordingly to the engine predefined “intelligence”. As the result the user not only get the requested web resources displayed on the handheld device screen, but also discovers (embedded in the original HTML document A/V) control panel (Figure 3.3). For the purpose of handheld device and TV
interaction, A/V playback control panel allows for several actions, such as: play/pause, stop, next track, previous track or volume control.

![Handheld device screen (NOKIA N800) displays embedded control panel in the web site](image)

**Figure 3.3:** Handheld device screen (NOKIA N800) displays embedded control panel in the web site [19]

An interesting feature proposed to the control panel design is the ability to navigate through the visited multimedia content even after reloading current or requesting for another web document. This sort of functionality is possible, since the transformation engine saves the Media Resource Locators (MRLs) that the user has already visited. Thus, whenever some MRLs are stored in the system, every requested web document will get the control panel re-embedded in its HTML code. Therefore, even if the original document does not hold any multimedia content the control panel will be activated to give the user a chance to see the history of previously visited audios and videos.

### 3.2.2 Control TV playback in a lean-back position

When a reasonable amount of multimedia content is already available in the system, the user interface can be extended into a form that facilitates more advance control over the playback (Figure 3.4). Since tracking visited multimedia content has already been enabled and discussed in Section 3.2.1, there are no constraints to extend this functionality and enable the system to store MRLs locally on transforming node storage permanently. In this case, the users will be given an opportunity to save the list of favorite content and return to it even after switching off the system.

Moreover, except tracking the history of visited multimedia content or storing it locally on a hard drive, the extended user interface offers a functionality to arrange requested MRLs into the playlists. In this way, a long term scheme for web video and audio playback can easily be achieved. This gives the user a chance to playback video or audio content in a given order and, in the meantime, for browsing the Web.
Nonetheless, in order to allow the user for basic control over the playback, basic functionality must remain available for this interface design as well. Since, control buttons are now present on the sides of the handheld device screen (Figure 3.4) it is likely that the user will easily locate them with the touch (thumb), but without a need to switch eyesight between TV and a control screen. In this way watching and controlling the playback in a lean-back position is achievable in a more comfortable manner.

3.3 Web page content summarization interface

It has already been mentioned that one of the biggest problems concerning 10-foot distance browsing is the content visibility. Suggesting split user interface for web sites navigation solves many problems in this area, mainly because some of the content is close-up to the user. However, living room browsing primary objective is to allow for content and experience sharing activity on a TV screen. For this reason, web site content summarization interface is proposed and assumed to be more straightforward and system-specific solution to present the Web offerings. This approach is illustrated in Figure 3.5, where the handheld device screen is divided into two parts. Left side bar presents web site content summarization menu, while the right side is used for displaying different pieces of originally request document. Then, it is up to the user which piece of web site is displayed on the right side part (and eventually TV screen) by pressing the proper icon (available on the left side bar).
Several benefits are recognized in the web site content summarization approach:

- From the perspective of TV browsing, only an interesting web content can be extracted from the original HTML document,
- Unnecessary information is hidden from the user,
- Content sharing activity becomes a matter of choosing a proper icon and exporting content to a TV screen (Figure 3.6),
- Limited amount of content can more easily be rendered on a large TV screen. In respect to 10-foot distance browsing, zoomed in piece of web site solves the visibility problem, on both TV and handheld device screens,
We also think that web page summarization approach, efficiently performed, may even enable for browsing the Web without the need to deploy split screens interactions. Instead of that, based on icons on the left side of the screen, remote controller can be used to switch between different pieces of the Web site. Since more space for displaying web content is provided to the TV, 10-foot distance scalability problem might be solved in this way. Further development of remote control navigation is, however, beyond the scope of this thesis. More research studies about web sites content summarization can be found in [8], [9] and [10].
4 System architecture

The previous chapter discussed user interaction approaches for browsing the Web in a split UI mode. This chapter proposes certain ways in which split UI system can be designed. Section 4.1 discusses a generic system architecture sketch. In Section 4.2 different types of transforming nodes are considered. Section 4.3 sets a focus on transformation engine internal structure and functional dependency between the local components. Since, the split screen algorithms largely relies on HTTP protocol and HTTP message exchange mechanism, a communication scheme for split UI documents exchange is introduced in Section 4.4. Conclusion to this chapter outlines the limitations of the design and possibilities for future extensions.

4.1 Split UI system for living room environment

Proposed in this thesis solutions for splitting web sites' user interfaces between the CE devices require an additional network node to be involved in interaction with the handheld device and TV. This additional network component plays a core function in processing and transforming HTML documents. For this reason, some storage capacity and computation power need to be provided to this device. Also, an important aspect of the design is the logical position of transforming node in the user's home network hierarchy. Figure 4.1 depicts this concept.

Figure 4.1 Split screen UI network logical hierarchy
In a real life scenario, split UI system could be realized in various ways. Firstly, the TV set may have all the necessary transforming node components built inside it. In this case, physical connection between the TV and the transforming node is no longer required. Secondly, transforming node can also be considered as a separated network node, either built-inside another Web4CE device or as a dedicated standalone system-specific solution (possibly a new CE product proposal). Despite, which approach is chosen, system functionality always remains the same.

Two different usage scenarios are also recognized. In the first case, the user is able to establish an indirect connection with the Internet. From that moment all users’ outgoing requests are forwarded towards the transforming node and processed accordingly to transformation engine's algorithms. In the process the resulting web site become TV and handheld device customized and browsed in the split UI mode. In the second case, if the user is no longer interested in content sharing activity, it is possible for him/her to browse the Web in a traditional, non-transformed way. Hence, switching system back to classical UI mode results in transformation engine deactivation and forces split UI system infrastructure to act as a classical wireless access point.

4.2 Network node type

Single UI mode does not put any additional restrictions on accessing the Internet from the home local area network. The situation changes when the split UI mode is used. In such a case, transforming node is no longer responsible for routing IP packages only, but also for processing web document content. To achieve split UI browsing, the user is required to address transformation engine in every HTTP request (transformation engine is in this context perceived as a system middleware). Two different approaches are recognized: server/client architecture and web proxy.

4.2.1 Server/client architecture

In order to use the server/client architecture, the user requests web resources directly from the transforming node. Therefore, Uniform Resource Locator (URL) [16] always points to the location of transforming node explicitly. Each time transforming node receives a given input from the user, the transformation process begins.
Since, in the server/client approach, URL address is always set to the transforming node location, the transforming node itself, in order to pass the message along for further distribution, needs to extract some data from the original HTTP header. Based on the extracted information, a new request towards the origin server is performed. Before that however, several attributes carried inside the original HTTP request needs to be modifies. This process is known in this thesis as header translation. Figure 4.2 depicts this concept.

Transforming node also fetches the HTTP responses. For this reason, it can somehow be perceived as acting on behalf of user's agent. However, before forwarding the response to the receiving-end, transforming node may decide whether to perform content modification of the original HTTP response, or to pass the message intact. Despite undertaken transformation strategy, user’s agent always accepts the modified document and parses the content accordingly.

4.2.2 Web proxy approach

Transforming node can also be implemented as a proxy server (Figure 4.3). This approach is just a slight modification of server/client architecture. The major difference is found in the way, in which client performs web resource request. Since, HTTP protocol supports proxy server functionality [16], it is no longer necessary for the home network client to address transforming node explicitly in every URL address. The only action user must take in advance is to enable proxy connections for every outgoing web request. By definition, proxy server is [16]:

“An intermediary program, which for the purpose of making requests on behalf of other clients, acts as both a server and a client. Requests are serviced internally or by passing them on, with possible translation, to other servers. A proxy MUST implement both the client and server requirements of this specification”

Proxy connections are usually specified by both: host name (or IP address) and/or port number. Therefore, in order to allow the transforming node for successful reception of user's data, a preconfigured TCP socket connection needs to be opened on the transforming node side before establishing any communication.
In the case of proxy server, the *transforming node* does not need to modify URL address specified in the original HTTP request. Since it is already accurate, forwarding the message will enable the same functionality as in the case of server/client architecture. Benefits behind this concept are discussed in more details in the last section of this chapter.

Formal definition of proxy introduces two major types of servers [16]:

- **transparent proxy** - “is a proxy that does not modify the request or response beyond what is required for proxy authentication and identification” [16],
- **non-transparent proxy** - “is a proxy that modifies the request or response in order to provide some added service to the user agent, such as group annotation services, media type transformation, protocol reduction, or anonymity filtering” [16].

A natural choice for the split screen user interface transforming node is a proxy similar with its functionality to *non-transparent*. Its formal and generic definition, however, does not say anything about content transformation of inbound origin server response. Therefore, for the purpose of this research, using transcoding proxy [6][7][8] functionality is proposed. Formal definition of the transcoding proxy can be found in the following definition [11]:

> “Proxy servers can modify the body format of content before delivering it to clients. This transparent translation between data representations is called transcoding”

Therefore, since this moment, *transcoding proxy* is considered as a network component used for the purpose of split UI transformations.

### 4.3 Transforming node internal structure

Depending on web document type and structure, different transformation abstraction levels are considered:

- **Visual transformation** – HTML layout is changed in the transformation process,
- **Functional transformation** – transformation engine affects original HTML code with JavaScript that enables for split UI interaction,
- **Hybrid transformation** – a visual and functional hybrid approach.
The internal structure of the *transforming node* (Figure 4.4) should allow for visual, functional and hybrid transformations.

Figure 4.4: Transforming node internal structure

All the internal modules of *transforming node* are functionally depended from each other and are organized into a processing pipeline. For this reason, it is mandatory for every internal element to produce well-formulated input for another. If this condition is fulfilled, total interoperability is preserved and centrally located *transforming node* supports different network node types (Section 4.2). While processing HTTP requests and responses each component is activated at a time.

### 4.3.1 HTTP server

HTTP server directly communicates with the handheld device. Besides providing standard server functionality, it is also responsible for translating HTTP requests into a form that is understandable to other components (which eventually will make use of the information carried inside the headers). Header reconstruction interface, on behalf of HTTP server, extracts at least the following key information out of the original user request:

- HTTP Method [16]: token that indicates the method to be performed on the resource identified by the Request-URI,
• Request-URI [16]: identifies the resource upon which to apply the request,
• Accept [16]: used to indicate that the request is specifically limited to a small set of desired types,
• Content-Length: used when HTTP body is present
• Message body: Optionally available in (for example) POST HTTP message.

At this moment, the rest of HTTP message does not have an important meaning to \textit{transforming node} and for further processing should be passed along intact.

4.3.2 Accept Type Detection

Another element in the \textit{transforming node} processing pipeline is the Accept Type Detection unit. Transforming node requires this component for HTTP message distribution purpose. Accept Type Detection helps the \textit{transforming node} to determine whether the user message is the origin server request or home notification attempt (Section 5). To distinguish home notification attempt from classical HTTP request, new MIME subtype is proposed as a CE-HTML standard extension:

\textbf{Definition 4.1 (x-home-notification)}: application/x-home-notification is a non-standard file type that specifies the user's home notification attempt. If the network client agent uses x-home-notification MIME subtype in HTTP request (Accept header field), a message carried inside the body should be passed towards CE device. This message should not be transmitted beyond the home network.

4.3.3 HTTP client

Since the \textit{transforming node} operates on a specific collection of HTTP arguments, in cooperation with Header Reconstruction Unit and on behalf of the user’s agent, HTTP client must formulate a new version HTTP request. Once such an HTTP message is created, it is sent towards the origin server URL location. At this moment HTTP client waits for the response. When the response arrives, Header Reconstruction module is yet again utilized. It extracts at least a few mandatory parameters before passing the message along to the next component:

• Content-Type [16]: indicates the media type of the entity-body sent to the recipient or, in the case of the HEAD method, the media type that would have been sent had the request been a GET,
• Character-Encoding [16]: optionally specified in Content-Type header field. Carry an information about character encoding that should be applied for read/write operation on the recipient side for HTTP entity-body,
• Content-Encoding [16]: its value indicates what additional content encodings have been applied to the entity-body, and thus what decoding mechanisms must be applied in order to obtain the media-type referenced by the Content-Type header field,
• Content-Length [16]: indicates the size of the entity-body, in decimal number of Octets, sent to the recipient or, in the case of the HEAD method, the size of the entity-body that would have been sent had the request been a GET.

4.3.4 Content-Type Detection

Not every HTTP response is meant for transformation. For this reason, decision making factor is implemented as a functional unit of the transforming node. Based on the data provided to this component, system can automatically decide if requested document should be transformed, stored for local notification purpose\(^3\) or sent to the client intact. Decisions rules are based on information carried inside the Content-Type header field and are analyzed by the Content-Type Detection unit. Three general cases are recognized:

• Content-Type: text/html – document transformation must be performed upon HTTP response,
• Content-Type: <media-type>/<supported multimedia sub-type>\(^4\) - HTTP response must be stored for local notification purpose,
• Every other case – entity is forwarded to the client.

4.3.5 Document initial scan

If the document Content-Type is recognized as text/html, document is passed to another pipeline unit - Document Initial Scan. The primary function of this component is to analyze HTTP document without parsing HTML code and creating its DOM representation internally. During this stage of transformation, based on simple text detection methods, different types of information can be recognized in the HTML code (e.g. document type, character encoding, extra tags defined for the layout/visual transformation purpose\(^5\)). Thus, during document initial scan tracking well known HTML sections becomes possible. This particular feature facilitates an important transformation technique – affecting HTML document with the Javascript functions. For this purpose, HTML document scanning window algorithm is proposed:

**Definition 4.1 (HTML scanning window):** HTML scanning window (Figure 4.5) is used for detecting key phrases embedded inside HTML message. Three mandatory attributes are passed to the algorithm as an input: message body (upon which search operation is performed), searched phrase and swap phrase. For algorithm ease-of-use, searched string is kept as an array of characters.

In the every algorithmic iteration, window size parameter is used to indicate which subsequent character should be picked from the search phrase array and to be compared against the following character from the input stream.

Scanning window algorithm bases on the following block diagram (Figure 4.5):

---

\(^3\) Extended discussion about notification mechanism can be found in 5th chapter

\(^4\) Supported media: audio, video; supported multimedia sub-type is VLC media player dependent

\(^5\) More information about this approach can be found in 7th chapters
If the algorithm recognizes the entire key phrase, it triggers an event and replaces the searched phrase with a predefined swap phrase. For instance, in order to insert some JavaScript functionality to the requested web page, the Document Initial Scan unit shall search for "</head>" in the HTML code. Once such a phrase is recognized by the scanning window algorithm, it can easily be replaced with a predefined string. Such an operation ensures that the code is always affected in the proper position and will be considered when the handheld device browser will process the HTTP response.

--

"</head>" phrase in requested HTML code is replaced with "&lt;script type="text/javascript" src=[script_location]="/</head>"
4.3.6 Extended transformation setup

For additional support in HTML document transformation, extended transformation setup is proposed. So far, in the world of software development numerous techniques for HTML document parsing had been invented. In the Java programming language at least several parsers that strongly support HTML language exist. One such parsing example has been found in Jericho library [30].

DOM parser unit creates local transformation engine’s Document Object Model (DOM) representation of the web site. Because DOM supports navigation in any direction (parent, child and sibling relationships) and allows for arbitrary modification to the document structure, more advanced web document transformation techniques become possible.

4.3.7 File and cascading stylesheet repository

It is also recommended to give the transformation system an ability to keep additional files locally on the transforming node local storage. For this purpose two additional supporting modules are added to the system. These are the file repository and cascading stylesheets repository. Using these two components is crucial whenever we are interested in extending the scope of required web site resource beyond those that are originally embedded inside the HTML document.

4.4 Communication scheme for split UI message exchange

Split UI system architecture requires at least two network devices to be involved in interaction. To facilitate split user interface browsing, not only HTML documents are modified, but also the way in which HTTP messages are exchanged. Two fundamental terms are introduced in this section:

- global network message exchange,
- home system notification.
Global network message exchange (solid black arrows in Figure 4.6) together with web document transformation approaches were already discussed in previous sections. One important feature that has not yet been described is the aspect of sharing and controlling the content on the distant TV screen. For this reason home system notification is introduced (dashed grey arrows in Figure 4.6).

Unlike global network message exchange, home system notification does not send any data outside user's home network. In more details home system notification is described in the next chapter.

4.5 Conclusions

This chapter discussed system architecture for split UI web browsing. Two major approaches for transforming node design were proposed: server/client architecture and web proxy. Both designs were implemented and tested in the laboratory scenario. Both approaches provided similar results of web document transformation. However, server/client architecture setup proved to be more sophisticated in achieving our primary objective.

In the server/client architecture every HTTP request is addressed to the transforming node explicitly. Therefore, server responsibility becomes to process user data and forward the
message towards the origin server. For this reason header reconstruction was required (Figure 4.2). In this approach, a problem occurs in a situation when relative URL addresses are used for external resource addressing (JavaScript, stylesheets, images, etc...). In principle, this problem derives directly from the following definition [17]:

"The term relative URL implies that there exists some absolute base URL against which the relative reference is applied. Indeed, the base URL is necessary to define the semantics of any embedded relative URLs; without it, a relative reference is meaningless. In order for relative URLs to be usable within a document, the base URL of that document must be known to the parser".

Unless default base-URL is defined inside the document base code [17], end-client browser will create base-URL either from the Encapsulation Entity [17] or, in majority of cases, from the Retrieval Entity [17]. One solution to this problem is to insert additional HTML code with the default base-URL definition [17]. Such an approach would be possible during HTML code transformation procedure. Base-URL tag can be formulated as follows:

```
Base-header = "Base" ";" ";URL:" absoluteURL ">
```

Due to relative addressing problem, using server/client architecture proves to be an inconvenient choice for automatic web sites transformations. However, with certain modifications to split UI concept, the server/client architecture may turn to be an important system design feature. Since modern web applications offer programming interfaces for accessing their internal resources [25] [19] [26], new possibilities for creating filial web sites appear. From the split UI development perspective, using APIs for accessing external web services means almost unlimited opportunities for implementing 10-foot distance specific web applications. In respect to splitting user interfaces, these services would become accessible via the transforming node HTTP client (using for instance RPC – Remote Procedure Call, RMI – Remote Method Invocation).

Automatic transformations of web pages and web services UIs remains possible when transcoding proxy is available. There is, however, a disadvantage of this approach and it emerges when the user is obliged to pre-configure web browser before establishing any connection with the transforming node. Depending on usage scenario, this particular feature may cause user general dissatisfaction. Certain modifications to network infrastructure can, however, successfully solve this problem [11]:

“There are several techniques where the network infrastructure intercepts and steers web traffic into a proxy, without the client’s knowledge or participation. This interception typically relies on switching and routing devices that watch for HTTP traffic, intercept it, and shunt the traffic into a proxy, without the client’s knowledge. This is called an intercepting proxy”.

---

In some cases, base URL default location can already be specified in original document source code. For this reason, transforming node should scan the document against base-header presence in a first place.

```
Therefore, using *transcoding proxy* with a proper network support for intercepting requests and responses are considered as ultimate solutions for automatic split UI transformations.
5 Split UI home system notification

The previous chapter discussed system architecture for enabling split UI transformations. Major focus was set on design approaches and transforming node internal structure. Transforming node pipelining introduced the concept of request/response mechanism for global network message exchange. However, the idea of requesting resources from the Internet and performing transformations upon the responses do not conclude split UI browsing problem definitely. What remains undescribed for the moment is the notification system for home network message exchange. This chapter, in the few steps, provides an analysis of this topic.

Section Error! Reference source not found., based on the example of split screen A/V playback control, discusses methods for enabling home notification message exchange. Section 5.2, in more details, introduces split UI message distribution system. As it occurs to be, mechanisms used for end-to-end message delivery, cannot fully rely on HTTP message exchange strategy any longer. For this reason, an extension to the TV set browser is proposed in Section 5.3. TV set media player that enables major A/V interactions, is briefly discussed in Section 5.4. Performance discussion summarizes this chapter.

5.1 Enabling for notification message exchange mechanism

Web transformation engine intercepts and modifies inbound HTML data. As the result, every requested web site gains additional split UI functionality. Since split user interface exists on both, handheld device and the TV, methods for capability exchange and communication between both devices are required.

Automatic content transformations applied to the web sites can either modify or add certain content to the requested HTML document. This feature is possible due to the document scanning window algorithm that was introduces in section 4.3.5. An important algorithm feature was to insert JavaScript code to the original HTML document before requested web site is delivered to the user.

JavaScript allows for arbitrary DOM structure modifications [29]. Moreover, it also provides necessary mechanisms that can be used for communication between handheld device and TV. These mechanisms include Asynchronous JavaScript and XML (AJAX) methods. According to AJAX specification [15], sending HTTP requests (XMLHttpRequest) to the server remains possible even after the web site is fully loaded into the browser. However, before AJAX “dialog” between the user’s handheld device and the TV is opened, several conditions must be met. In the following subsections, necessary operations that are required for home notification system are described.
5.1.1 Notification object registration

In order to enable split UI content sharing activity, every requested web site must have its split UI interface preconfigured. Therefore, during HTML parsing run time, for every requested web site, a notification object must be created locally in a browser. For this purpose, inserted JavaScript code is used. In case of video and audio playback control, notification object must hold numerous attributes, such as:

A. Top window domain name associated with a web site (information purpose),
B. Object registration flag (a boolean value holding an information about current object registration status),
C. For the example of A/V playback control, a collection of already visited, registered A/V resource identifiers,
D. For the example of A/V playback control, a collection of user favorite registered audio/video resource identifiers,
E. For the example of A/V playback control, a collection of selected video/audio resources organized into a playlist.

Figure 5.1: Relationship between frames [12]

The major problem with the object registration procedure appears when multiple frames are embedded inside HTML document main body. Because every sub-frame requires an additional HTML message request, it is highly probable that for a single web site, multiple notification objects will be created. Hence, another weakness of the transformation engine emerges. It is very difficult to distinguish whether a given HTTP request represents web site's
main body or embedded sub-frame’s HTML code (Figure 5.1). Every such a case is treated equally and so every HTML document is transformed in the same manner. Unless multiple objects registration is prevented, notification message exchange procedure (Section 5.1.2) will always produce system confusion (for a single web site, multiple attempts to communicate with the transforming node would be made, which in practice would be an unwanted feature). For this reason, following JavaScript condition is used locally in a browser to distinguish the main body element from the embedded sub-frame:

```javascript
if (window.self==window.top) {
  //
  // create JavaScript notification object;
  // start notification message exchange
  //
}
```

If the above condition is met, notification object will be created only once per requested web site and will always belong to the top window HTML element. Thus, only one notification object will participate in notification message exchange procedure with the transforming node and will not produce performance demotion.

Forbidding multiple notification object registration gains its importance when we take into consideration the example of A/V playback control. In this sample usage scenario the user expects to have the in-browser A/V playback control panel activated on the handheld device whenever some A/V content is requested via HTTP (Section 3.2). For every requested web site, playback control panel is initially embedded inside every HTML document body, however, it remains hidden from the user until first video or audio notification message (Section 5.1.2) arrives. Once this happens, object registration flag (B) is set true and playback control panel appears on the screen. From this moment, A/V control panel will wait for multimedia content updates until current page is reloaded. This process is repeated for every requested web site. Thus, if multiple notification objects would be created per requested web site it would be highly probable that multiple A/V control panels would be presented to the user. This, of course, would be an unwanted outcome of the transformation operation.

### 5.1.2 Capability exchange mechanism

**Capability exchange mechanism** is a split UI system strategy for keeping handheld device up to date with for instance new multimedia content. Since, in the first place, multimedia content is intercepted and registered locally by the transformation engine, it is essential to deploy efficient methods for informing client’s notification object about currently available playback capabilities. For this purpose message exchange pooling strategy is proposed. According to this approach, client’s interface constantly remains in one of the transition graph states depicted in Figure 5.2.
Split UI notification begins with the *notification request* (Appendix A, Listing 1). Since this moment, client’s interface expects a response from the transforming node. Therefore, it stays in *wait* state until it arrives. While client interface remains in a *wait* state, *notification message* is processed by the transformation engine. Several scenarios are possible:

I. A/V control panel has not yet been shown: This means that new web page has been loaded to handheld device browser and now user waits for multimedia offerings update and:
   A. If multimedia content is already available at the transformation engine, immediate update is sent,
   B. Otherwise system sets a local timeout for this request and does not response until:
      i. Timeout reaches the end,
      ii. New multimedia content arrives,
      iii. User requests new URL.

Despite which scenario is considered, system *notification response* is always formulated accordingly to DTD scheme, which is available in Appendix A, Listing 2.

II. A/V control panel already exists: This means that the handheld device has already registered some multimedia content and now waits additional updates. Again, system sets local timeout and does not response until:
   i. Timeout reaches the end,
   ii. New multimedia content arrives,
   iii. User requests new URL.

In this case *notification response* is this time formulated accordingly to DTD scheme presented in Appendix A, Listing 3.
Within the requests and responses sent between transformation engine and the handheld device also A/V playback control operations are transmitted. They are available in Appendix, listing 4-6.

5.2 Notification messages distribution characteristics

Transforming node, located centrally between the user, TV and a public Network, controls message exchange distribution for split UI interaction (Figure 5.3). Decision rules are made internally in transformation engine's components (Section 4.3). Based on this logic, following message exchange paths are considered for end-to-end split UI message exchange system:

1. Handheld device – public network
2. Handheld device – TV
3. Public network – TV

Taking end-to-end message exchange system into consideration, an important technology problem appears. In principle, this issue concerns the nature of system notification strategy. As it was described in section 5.1.1, client, in order to stay up to date with new resources available at transformation engine, sends periodic XMLHttpRequests. Unfortunately, in order to set a communication link between the transforming node and the TV browser, message pooling strategy is no longer accurate. Because the transforming node is responsible for triggering events on the TV browser interface, pooling mechanism will likely cause uncontrolled delays and as a result may cause system performance demotion. For this reason, using in-session notification mechanism is recommended.

Figure 5.3: End-to-end split UI message exchange system
5.3 In-session notification

In-session notification mechanism uses a persistent TCP socket connection to establish communication between web server and any remote web browser. Under certain circumstances, such a mechanism can be perceived as a reverse HTTP request/response protocol. According to notifsockets specification [1], server side application gains an opportunity to notify TV browser without being requested beforehand. In a normal situation, such a scenario would cause a serious security threat to a browser. However, in a strictly controlled manner, this disadvantage turns to be an important system feature. Since transforming node does not need to wait for client periodic requests any longer, it can remotely trigger TV browser event immediately after receiving the user's agent notification messages. In this way exporting web content and controlling TV screen behavior is possible during split UI interaction.

In order to allow for in-session notification, an extension to Mozilla Firefox® [27] browser is proposed. To enable this extension, it is mandatory to embed a proper extension object entry in a TV browser application HTML code. This object entry has the following form:

```html
<object id="notifsocket" type="application/notifsocket"/>
```

Registering entry for in-session notification allows TV application to open a persistent socket connection and listen to inbound web notification traffic. In such a scenario, it is also required that the transforming node also opens socket connection on a given IP address and specific port number. This procedure can successfully be accomplished during system start-up. Once connection between transforming node and the TV has been established, it remains open for the entire split UI session. In this way, transforming node, via notifsocket entry object⁸, gains access to unlimited amount of Javascript functions that can be defined for the TV browser application.

5.4 Video and audio media player

Once notifsocket entry to the TV browser has been created, triggering remote events on TV screen becomes possible. In this way, TV browser Javascript APIs can be easily be created and can serve for a split screen content and experience sharing activity. One such an example that is discussed in this thesis is video and audio playback. For the purpose of remote controlling the playback, another extension - VLC® [28] - to Mozilla Firefox ® browser [27] is proposed. This extension allows for:

- uploading audio/video tracks into the media player,
- uploading audio/video list of items (playlist),
- toggling play/pause,
- toggling stop,
- controlling volume,
- and many more.

More information about VLC Media Player® can be found in [28].

⁸ Extended set of properties and methods for in-session notification interface are available in Appendix B
5.5 Conclusions

Message distribution methods for home system notification are important aspect of split UI browsing. They allow for end-to-end XML message delivery in strictly controlled and effective way.

Pooling system for exchanging control information between the user’s handheld device and the transforming node is used (Section 5.1.2). This mechanism works accordingly to state machine illustrated in Figure 5.2. This approach has an important benefit in comparison to notifsocket persistent connections, which are used for communication between transforming node and TV. In principle, this concerns availability of XMLHttpRequest mechanism in the majority of available for mobile devices web browsers. Since, most of them support AJAX functionality, vast number of handhelds will be supported by the system without the need to create dedicated numerous notifsockets extensions for every such possible case.

On the other hand, from the perspective of TV set and transforming node interaction, using notifsockets has also several advantages over the pooling mechanism. In a first place, this concerns notification methods that are used for triggering remote events on the TV interface. Since, transforming node is responsible for message distribution in split UI system; delivering urgent control information to the TV requires immediate notification mechanism. Therefore, opened socket connection between these two communication end-points will satisfy this requirement. In a second place, it is also important to notice that TV requirement for in-browsers extensions are not as strict as it is a case in respect to handheld device interface. Since TV can operate on one dedicated TV application, there is not further need for producing numerous extensions for various TV browsers.
6 Conclusion

What was discussed in this master thesis is the proposal of living room system for browsing the Web in a split user interface mode. The primary objective was to take the Internet and its resources beyond classical PC paradigm. One obligatory aspect of the proposed solution was to involve one person in browsing the Web on a handheld device and to share certain web content among broad living room audience. An important goal of this thesis was achieved, since different perspective of web browsing routine was clearly introduced.

In this thesis also interface design and system implementation were tested in a lab scenario. For the test purpose, Nokia N800 Internet Tablet and a large LCD display were used. The transformation engine was implemented on PC class machine connected directly to wireless access point and public network. Such a test setup allowed us to perform web content transformations in HTTP request/response time and watch the outcome of TV and handheld device interaction.

Important design issues appeared in the numerous test activities. In the first place, a generic user interface design was investigated. That process allowed us to highlight the major design requirements in respect to 10-foot distance browsing. Once requirements were set, trial system architecture sketch was created. Accordingly to the initial requirements an attempt has been made to create a generic middleware transforming node. Eventually, what we came up with was a transcoding proxy tool for intercepting HTTP requests and responses and modifying HTML body content. Accordingly to investigated and implemented functionality web transformations were achieved. Functional dependency between browsing the Web on a handheld device and exporting the content onto a large TV screen was also observed. As the result the user was able to request web documents and multimedia content as it is normally done in a classical PC browsing scenario. However, because the transforming node was involved in processing HTML code, original web pages had either its layout redesigned or gained some additional functionality.

Nonetheless, user interface design, system architecture and communication scheme between interconnected devices do not fully conclude the undertaken topic of transforming web documents to browse them in a living room. Results achieved during the research study and system implementation create a skeleton for further concept development. Also described in this thesis example of audio and video notification certainly does not put a limit to what is possible to achieve with the suggested system. Other use cases can also be found easily. Therefore the A/V example is treated here as a proof of concept to this thesis work. Thus, an automatic form-factor transformation of web pages and web service UIs remains an open topic for future investigation. Possibilities for further extensions to the idea are discussed in the next chapter.
7 Discussion and future work

Before the discussion and future work proposal to this master thesis project will be provided, several already discussed key definitions will be summarized in a slightly different context:

- Transforming node: is a network component inside user’s living room establishment. It allows for ultimate transformations of the web sites in order to display them in a split screen UI mode. This means that before the requested document is presented to the user, in transformation process, it gains some additional functionality. Until this moment, the scope of the additional functionality was limited only to video and audio playback. Because the possibilities of commonly used web development techniques (like Javascript/AJAX, cascading style sheets) transcend the limitations of simple web site alteration, we may expect that system capabilities may also go far beyond the scope of video and audio playback notification and control. In this chapter a few more proposals for the system functionality will be provided.

- Notification system: is an ultimate way to allow for message exchange and therefore interaction between the TV and the handheld device. The ability for exchanging messages is a result of web document transformation. Yet again, in this thesis, we have put an emphasis on video and audio control. This chapter will determine more examples of how the notification system can further be developed in order to enable more advanced options of split screen browsing,

- The parser: is used as an integral part of the transformation engine. Inside transformation engine, it allows for creating local representation of the web document. Once the document is internally presented as a tree structure DOM hierarchy, it gives more flexibility in understanding the web site layout and therefore it allows for customizing the Web for consumer electronics products further.

All above listed components comprise into the key system features. Based on the suggested techniques used for web site’s transformations, numerous modifications to the web site’s layout and functionality can be performed. Not all of them, however, can be achieved in a fully automatic transformation process (web sites structures are too heavily human dependent). For that reason, for the transformation engine, we suggest a human factor to be also involved in creating additional logics as an additional system input. Therefore, for the split screen system, not only the requested document is used for the transformation input, but also predefined metadata. The only question that needs to be answered on the way to successfully design that sort of system is where to get the additional human origin metadata factor?

Suggested approach is to involve the web developers in split screen UI web site transformation activity. In that sense, based on internally implemented transformation engine tools, the developers would have an opportunity to work with original HTML documents to easily extract the metadata out of the web sites content and eventually save the effort on a
common storage. Below a sample scenario of how the system could work in the “development mode” is presented:

1) The developer requests a web site using classical PC browser.
2) The system checks the HTTP request body and header. If specified field carries the information that the system is now working in the “development mode” it will normally request the origin server for the resources, fetch them back and transform the response. Here, however, since the “development mode” is switched on, the web site will be transformed in a manner that will enable performing the split screen development tasks.
3) Whenever the developer wishes to save the current status of transformation, a notification should be sent to the transformation engine with the update.
4) Once the development process is over upon the requested resources, the developer is obliged to inform the system about the outcome. This can be achieved by sending “development over” message to the transformation engine.
5) Once the transformation engine receives “development over” message, it will either save the resulting metadata internally in its storage or export the metadata to the global repository (directory, database⁹).

Once the “development task” is performed upon the requested web site, the extracted metadata will become available for the split screen applications used in “non-development mode”. One such an application was introduced in 3rd chapter of this thesis, where the web site summarization interface has been presented. Sample development process, in which web site summarization can be achieved will now be considered.

![YouTube](https://www.youtube.com/watch?v=example_video_id)

**Figure 7.1:** Predefined Javascript allows for detecting mouse events. Dashed outline over the DIV element is a result of onMoveOver event [19]

⁹ Storing the metadata will be determined in further development process
In the first place, the split screen engine needs to insert some Javascript, cascading style sheets or additional tags into the originally requested HTML documents. Now using the inserted scripts the developers would get a chance to:

- select different DOM components using mouse events (e.g. select a piece of the web site [19] – Figure 7.1).
- based on selected DOM component, perform actions upon the item (extract the selected node and its children node(s) from the original web page).
- see the visualized representation of the operation in the custom prepared HTML container (Figure 7.2).
- inside the container, add some extra input for the selected component (for instance an icon – Figure 7.2).
- send the outcome of the operation towards the transformation engine, when the development task upon the requested web page is complete.

![Figure 7.2: Exporting HTML component to the custom prepared embedded container [19]](image)

Whenever the developer has extracted the complete and required set of information from the web site, he should inform the transformation engine about the outcome. From this moment the engine is obliged to save the metadata together with the web page URL for future user requests. Thus, if the same URL address will be requested for the split screen UI interaction the metadata will be in place to process the HTML code accordingly to the newly created rules. This scenario can be depicted as follows (Figure 7.3):
The “developer task” in its early stage of development has purposely been introduced in this thesis to show that customizing HTML is possible and does not require any additional and sophisticated development knowledge. Moreover, we expect that some additional and already existing tools may serve very well to create the metadata for more advanced transformation purposes. This for instance can be found in the so called web splitter applications.
8 References


Appendix A

DTD listings used in the thesis:

```xml
<?xml version="1.0"?>
<!ELEMENT NOTIFICATION (REFERRER)>  
<!ATTLIST NOTIFICATION TYPE 
{request|play|pause|stop|volume_up|volume_down|mute|playlist|favorite} "request">  
<!ELEMENT REFERRER (#PCDATA)>  

Listing 1: Notification message initiating message exchange

<?xml version="1.0"?>
<!ELEMENT NOTIFICATION (VISITED?,FAVORITE?,UPDATE?) >  
<!ELEMENT VISITED (MEDIA+)>  
<!ATTLIST MEDIA TYPE (video|audio) "video">  
<!ELEMENT TITLE (#PCDATA)>  
<!ELEMENT REFERENCE (#PCDATA)>  
<!ELEMENT FAVORITE (MEDIA+)>  
<!ATTLIST MEDIA TYPE (video|audio) "video">  
<!ELEMENT TITLE (#PCDATA)>  
<!ELEMENT REFERENCE (#PCDATA)>  
<!ELEMENT UPDATE (MEDIA?)>  
<!ATTLIST MEDIA TYPE (video|audio) "video">  
<!ELEMENT TITLE (#PCDATA)>  
<!ELEMENT REFERENCE (#PCDATA)>  

Listing 2: Notification response for a new URL

<?xml version="1.0"?>
<!ELEMENT NOTIFICATION (UPDATE)>  
<!ELEMENT UPDATE (MEDIA?)>  
<!ELEMENT MEDIA (TITLE?,REFERENCE)>  
<!ATTLIST MEDIA TYPE (video|audio) "video">  
<!ELEMENT TITLE (#PCDATA)>  
<!ELEMENT REFERENCE (#PCDATA)>  

Listing 3: Notification update for already registered URL

<?xml version="1.0"?>
<!ELEMENT NOTIFICATION (ITEM+)>
<!ATTLIST NOTIFICATION TYPE 
{request|play|pause|stop|volume_up|volume_down|mute|playlist|favorite} "playlist">  
<!ELEMENT ITEM (#PCDATA)>  

Listing 4: Playlist notification

<?xml version="1.0"?>
<!ELEMENT NOTIFICATION (REFERENCE)>  
<!ATTLIST NOTIFICATION TYPE 
{request|play|pause|stop|volume_up|volume_down|mute|playlist|favorite} "favorite">  
<!ELEMENT REFERENCE (#PCDATA)>  
<!ATTLIST REFERENCE TASK (add|delete) "add">  

Listing 5: Favorite multimedia ‘add’/ ‘delete’ operations
```
<table>
<thead>
<tr>
<th>Operation</th>
<th>XML Scheme</th>
</tr>
</thead>
</table>
| Play      | `<!ELEMENT NOTIFICATION (REFERENCE)>`  
            `<!ATTLIST NOTIFICATION TYPE (request|play|pause|stop|volume_up|volume_down|mute|playlist|favorite) "play">`  
            `<!ELEMENT REFERENCE (#PCDATA)>` |
| Pause     | `<!ELEMENT NOTIFICATION>`  
            `<!ATTLIST NOTIFICATION TYPE (request|play|pause|stop|volume_up|volume_down|mute|playlist|favorite) "pause">` |
| Stop      | `<!ELEMENT NOTIFICATION>`  
            `<!ATTLIST NOTIFICATION TYPE (request|play|pause|stop|volume_up|volume_down|mute|playlist|favorite) "stop">` |
| Volume up | `<!ELEMENT NOTIFICATION>`  
            `<!ATTLIST NOTIFICATION TYPE (request|play|pause|stop|volume_up|volume_down|mute|playlist|favorite) "volume_up">` |
| Volume down | `<!ELEMENT NOTIFICATION>`  
             `<!ATTLIST NOTIFICATION TYPE (request|play|pause|stop|volume_up|volume_down|mute|playlist|favorite) "volume_down">` |
| Mute      | `<!ELEMENT NOTIFICATION>`  
            `<!ATTLIST NOTIFICATION TYPE (request|play|pause|stop|volume_up|volume_down|mute|playlist|favorite) "mute">` |

**Listing 6:** DTD schemes for basic audio/video operations
Appendix B

NotifSocket Scripting Object

Every Remote UI Client SHALL support the embedding of a NotifSocket plugin object within a CE-HTML page through the use of <object>-elements, with application/notifsocket” as value for attribute “type”, i.e.:

<object id="NotifSocket" type="application/notifsocket"/>

Embedded NotifSocket plugin objects SHALL be made accessible through the DOM with the interface as defined below. (Note: a Remote UI Client MAY provide native support for supporting these <object>-elements, and is not required to support downloading of a plug-in to support these <object>-elements):

a) Properties:
   - **readonly Number tcpStatus** - read-only property that indicates the status of the underlying TCP connection:
     - 0 = disconnected or aborted (i.e. unplugged cable, too much interference of wireless network), whereby the underlying NotifSocket implementation has stopped retrying
     - 1 = connected to RUIS with TCP
     - 2 = succeeded to send to RUIS
     - 3 = failed to send to RUIS (but still connected and the underlying NotifSocket implementation stopped retrying).
   - **function() onStatusChange** – read-write property that specifies the function that SHALL be called when the tcpStatus changes. The function may test the value of tcpStatus and direct the further execution accordingly. For example, if the tcpStatus changes to 0, JavaScript MAY handle the connection-error by calling openPersistentConnection again, by reloading the RUIS page, or by indicating an error. If the tcpStatus changes to 3, JavaScript MAY resend the message or indicate an error.
   - **function() onDataArrival** – read-write property that specifies the function that SHALL be called when a message has been received through the TCP connection.

b) Methods:
   - **openPersistentConnection(String address, Number portNr, Boolean secure)** - Instructs the Remote UI Client to create a TCP/IP connection (using SSL/TLS security if attribute “secure” has value true) to the given address and port number of a host in the network. The domain of this host SHALL be the same as the domain of the server that served the script that calls this method. Only in cases whereby the client is able to trust the server that served the script that calls this method, this restriction does not apply. If opening the TCP/IP connection is successful, the tcpStatus will change to 1. If unsuccessful, an onStatusChange SHALL be triggered with value 0 (even if the tcpStatus already had value 0).
   - **getData(): String** – Returns the message up to the next EOF character that was received via the TCP/IP connection (which was opened through method
The EOF character SHALL be the Unicode character \u0004 (i.e. single byte with value 4 for UTF-8): “End of Transmission”. Note: this character SHALL be removed from the return value of this method. Note: The Remote UI Client is responsible for buffering incoming events to prevent incoming events from getting lost. Note: The EOF character needs to be inserted by the server as a character into the stream, and is not the same as the end-of-file condition that occurs when the connection is closed. The connection needs to remain open in order to send/receive the next message.

- **sendData(String data): Boolean** – To send the given data (concatenated with an EOF character) over the TCP/IP connection that has been opened through method `openPersistentConnection()` to the Server. This method SHALL automatically concatenate the value of attribute “data” with an EOF character before sending it. The EOF character SHALL be the Unicode character \u0004 (i.e. single byte with value 4 for UTF-8). This method SHALL not block the execution of subsequent methods, and SHALL return “true” if it has accepted the data to be sent, and SHALL return “false” if the method has not successfully sent the previous message yet. If the tcpStatus was 2 or 3 while calling this method, the tcpStatus will be set to 1.

- **close()** - closes the current TCP/IP connection, discards all send and receive queues and sets tcpStatus to 0.

Any Remote UI Server that wants to send In-Session Notifications through `NotifSocket` SHALL provide an event notification handling mechanism for communicating with `NotifSocket` on the Remote UI Client.
Appendix C

Philips Research

High Tech Campus 54 (WB-41), 5656 AE Eindhoven, The Netherlands

Andrew Tokmakoff
Information Processing
Architectures
Tel.: +31 (0)40 27 49547
Andrew.Tokmakoff@philips.com

Freddy Snijder
Storage Systems & Applications
Tel.: +31 (0)40 27 47994
Freddy.Snijder@philips.com

Student assignment

Date: 11/05/2007

Automatic form-factor transformation of web-pages and web-service UIs

The web is providing consumers new ways to socialize, be entertained, be productive and be creative. This is why Philips Research is looking for new ways for users to 'consume' web-services in the home that are easy and fun.

We are looking for a student who likes to do his or her final MSc. work here at Philips Research. The assignment is to investigate methods that either automatically or via authoring tools transform existing web-pages and in-browser web-service UIs to new UI form factors. These new form factors provide users new ways to use web-pages and web-services in a more fun and easy way in the home.

The student's work should result in new methods as described above, a working demonstrator and a master-thesis describing the work.

Preferably a student applicant has knowledge on, or is eager to learn about, web development, Style-sheets and XSLT.

Expected duration : ~9 months
Preferred start : Beginning of August 2007

If you are interested to apply, please send a resume and your motivation in an e-mail to Andrew.Tokmakoff@philips.com, CC Freddy.Snijder@philips.com.

Andrew Tokmakoff
Freddy Snijder
Automatic form-factor transformation of web pages and web service UIs

Piotr Lubinski

Browsing the Web on a distant screen may very often become user’s unpleasant experience. Such a situation may happen in a living room environment, where the user attempts to browse the Web on TV sitting at 10-foot distance from the screen. In a vast number of web sites that the user is able to request, many may contain, for instance, tiny text characters, which would become unreadable when watched from the distance. On the other hand, also navigation through the Web offerings can become troublesome. Especially, when the user is only equipped with the commonly used remote controller, interaction with the displayed content may prove inefficient. Therefore, methods to overcome content visibility and navigation problems become challenging.

In this thesis we perceive living room environment as a good scenario, in which web-enabled handheld devices could further be utilized to interact with Consumer Electronics products, such as TVs. This sort of interaction should make an advantage of publically available Web resources to have them displayed on TV and controlled from the distance by the user who is equipped with, for instance, a web-enabled mobile phone. We think that such an interaction approach would become beneficial in several ways, making the Web browsing routine an interactive, possibly a group activity.

Discussed in this thesis solutions are the first step of automatic transformations of web content into a form that can be rendered on Web4CE (Web for Consumer Electronics) products. A particularly important aspect of the proposed system is the concept of request/response HTTP message processing and HTML modification. According to this approach, living room browsing activity should be achievable without the need to re-author already existing web sites.

Keywords
Split User Interface, transformation, modification, notification, HTTP, HTML, processing, server, client, proxy, transcoding proxy