# WS://IM: A Software Framework for Multimodal Web Interaction Management

Christopher Stephen Williams
Department of Computer Science
Virginia Tech, Blacksburg, VA 24061

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE
in
Computer Science and Applications

**Examining Committee:**

Naren Ramakrishnan, Chair
Edward A. Fox
Scott McCrickard

May 3, 2004
Blacksburg, Virginia

**Keywords:** browsing technique, out-of-turn interaction, rich dialogs, world wide web, staging transformations, multimodal, and interaction management.

# WS://IM: A Software Framework for Multimodal Web Interaction Management

Christopher Stephen Williams

**Abstract**

The rise of ubiquitous computing devices has provided the catalyst for the next generation World Wide Web, one that shifts the focus from the desktop computer to mobile devices such as cell phones and PDAs, in an ever increasing range of modalities. Web interaction management in this setting must contend with a plethora of interaction interfaces and a diverse range of content types in addition to helping realize the full potential of multimodality (i.e., supporting flexible and personalized interactions between humans and sites). This thesis presents WS://IM, a new software framework for web interaction management that is capable of supporting multimodal interactions. In addition to presenting a loosely bundled, factorized architecture that supports hyperlink interaction, WS://IM has the unique facilitation for out-of-turn interaction. Out-of-turn interaction is a novel technique that helps realize mixed-initiative interactions between humans and Web sites. Design methodology, implementation details, and exposition through three implemented case studies are provided.

*For*
*My parents and grandparents. Your guidance, support,*
*and encouragement have made all the difference in the world.*

# Acknowledgements

*"I don't understand all the numbers, but my faith is in the math." -John Mayer*

I would like to thank Dr. Naren Ramakrishnan for his guidance throughout my academic career. Without your encouragement, I would most likely not have pursued my Masters degree and missed out on this amazing opportunity. Also, I would like to thank him for all of the hours he has spent reviewing my designs, plans, and writings. I would like to thank Dr. Edward A. Fox and Dr. Scott McCrickard for their time and assistance throughout my thesis review and academic career. I would like to thank Saverio Perugini for the numerous hours we spent developing, testing, and redeveloping both the WS://IM implementation and his staging transformer; they were some of the best times I have had in graduate school.

I would like to acknowledge my family for their encouragement to continue my education and to keep going even when it seemed like too much. My family has provided me with the strength I have drawn from to accomplish this, the example I attempt to follow in life, and are the ones I turn to when I need help. Thank you does not say enough.

I would like to thank Laura MacKichan for her continual support and dedication to me during the many stressful days of labor that it took for me to get to this point. I would also like to thank Kenneth Henderson, Christopher Keller, Christopher Moore, Colette Zai, and the rest of my friends who have been there for me when I needed them and made me realize that life is more than just work. I would like to thank Kristen Hansen for all of the time she took to review my thesis.

I would like to thank John Garber, Graham Welling, and everyone at Cryptek, Inc. for providing me with the opportunity to attend Graduate School.

This work is supported in part by US National Science Foundation SGER grant IIS-0136182. The development of the CITIDEL case study was supported in part by the Virginia Tech CITIDEL research group funded in part by US National Science Foundation grant DUE-0121679. Section 3.5 of this document as developed through collaboration with the Virginia Tech 5S research group.

# Contents

# List of Figures

# Chapter 1

# Introduction

The Web is arguably the most complex and ubiquitous content distribution medium ever invented. It has evolved from its humble origins as a simple hypertext navigation system on desktop computers to support pervasive computing on mobile devices, such as 3G cell phones and personal digital assistants (PDAs), using several, and ever-increasing, modalities. This multi-device, multimodal context fosters a rich suite of user interaction paradigms, such as browsing [22], personalization [62], location-aware computing [42], and social networking [45].

## 1.1 Web Interaction Management

The focus of this thesis is *Web interaction management*–the problem of mediating communication between users and content providers. This term has traditionally involved a broad range of issues, such as automated delivery of statically, as well as dynamically, generated pages, sessioning, accommodating simultaneous users, concurrency control, stateful implementation of client-side functionality such as cloning windows and pursuing back buttons, and domain-specific language (DSL) support for targeted applications such as form-field interaction and database-centered services. The stateless nature of the HTTP protocol, on which interaction is realized, makes Web interaction management difficult, and thus, a legitimate topic of WWW research. In addition, the advent of the multimodal Web has expanded this scope further and necessitated an interaction management framework that combines diverse interaction media, in order to support flexible, contextual, and personalized interactions.

## 1.2 In This Thesis

The primary Web interaction management research issues studied in this thesis are:

1. Maintaining user state in an interaction (HTTP is stateless),
2. Distinguishing a given user from other users (session management),

3. Supporting rich, personalized dialogs (mixed-initiative interaction), and
4. Uniformly supporting interfaces using different I/O modalities and presentation styles.

The central thesis of my research is that *rich and contextual multimodal Web interactions can be supported through a factorized software framework using distributed components and centralized management of interaction*. The primary context of my work arises from the *staging transformations for multimodal Web interaction* project at Virginia Tech [55]. When this project was initiated, it provided a transformation-based approach to supporting user-site dialogs but lacked session management, modality application, and support for multiple sites. The research presented in this document fulfills these requirements by developing a web interaction management framework called `WS://IM` (Web Service for Interaction Management) and instantiating it for several important and relevant case studies. The case studies help assess the feasibility and scalability of the approach through different interaction interfaces and presentation styles.

The primary software artifacts described in this thesis are:

1. A fully-functional Web interaction management software framework for the staging transformations project. At a basic level, this framework resolves concurrency issues and provides session maintenance for the staging transformation engine.
2. New interaction interfaces using different modalities and presentation styles, exposing the concept of cascading modality application as a means to provide a tailored user experience.

The contributions of this work are:

1. For developers: Rapid development time due to reduced requirements for content provision.
2. For users: Flexible, contextual, and personalized dialogs facilitated through the multimodality and staging transformation capabilities inherent in the system design.

These contributions build on the current research in interaction management software frameworks as surveyed in this thesis. Outside the scope of this thesis are issues such as usability assessment of dialog-based web interaction management and the theory of transformations (see [59] for a discussion of these aspects).

## 1.3 Outline of Thesis Document

1. **Chapter 2**: An exploration of the Web's history, investigating the underlying content delivery model inherent in the system. Specifically, I discuss the highly recommended three-entity factorized model of Web content delivery and identify a fourth entity, modality, which will become important with respect to the next-generation of Web interactions.
2. **Chapter 3**: A survey of related research in Web interaction management which identifies various design perspectives related to the model discussed in Chapter 2. Each perspective is organized by levels of implementation abstraction in which I situate several showcased systems. All of these systems follow the three-entity factorized model; none instantiate the fourth.

3. **Chapter 4**: A detailed description of the WS://IM framework highlighting the design decisions and implementation details necessary to enable rich, contextual, multimodal dialogs between users and Web sites.

4. **Chapter 5**: A discussion of three case studies: Project Vote Smart, a site providing content on the US Congressional Officials, the Online Virginia Tech Timetables of Courses, and the CITIDEL digital library. For space considerations, these case studies are depicted through at most two modalities or presentation styles supported by my framework. Lastly, this chapter evaluates my framework with respect to the systems surveyed in Chapter 3.

5. **Chapter 6**: A summary of conclusions drawn from the implemented framework as well as a discussion of possible future directions for this research.

# Chapter 2

# Background

Over the past decade, the Web has evolved into an interactive medium, driven primarily by new requirements and technologies. From its modest beginnings as a simple content delivery system, the Web now extends into more fields and modalities than other information delivery systems such as newspapers, telephones, and bulletin boards.

It is instructive to examine the history and evolution of the Web in its basic role as an information delivery mechanism between providers and consumers. A key milestone was when T. Berners-Lee proposed a simple hypertext system that was accessed through a well-defined, efficient protocol (HTTP), with the *content* encoded in a Standard Generalized Markup Language (SGML) derivative, aptly named HyperText Markup Language (HTML) [70]. The next major step in the development of the Web was Mosaic, which not only contributed to the rapid popularity of browsing, but also accelerated the development of HTML. Many corporations entered the fray and vied for supremacy of the medium by adding new features and functionality with little to no forethought of standardization [15]. By this time, the scope of document markup was becoming considerably expanded, to incorporate considerations of both application *logic* and style of *presentation*. This trend continues to this day; for instance, the rise of ubiquitous computing devices foreshadows the emergence of *modality* as an additional ingredient into the Web model and promises to alter the way the Web is used. The combination of the above four distinct entities (i.e., content, logic, presentation, and modality), while reflecting the modern instantiation of Web, does not accurately capture how today's Web actually works. To see why let us study this evolution more closely.

## 2.1 Evolution of the Web

The Web was arguably originally intended as a document retrieval system, whose interaction model can be given by:

$$UserGoal = Content \tag{2.1}$$

This model is an accurate portrayal of the original intention of the Web, but due to initial technological constraints (e.g., network bandwidth, screen resolution), content could not be entirely distributed directly to the user. In order to answer these constraints, Berners-Lee envisioned a hypertext system in which information shards could be connected to one another using hyperlinks that leverage Uniform Resource Locators

(URLs) [68]. To facilitate this functionality, the content provider must design interaction pathways or decision trees through the information content. Implementing these crafted interaction pathways can be handled in two ways: statically or dynamically. The older, static implementation requires developers to enumerate every node in every interaction pathway and decision tree as an individual Web page. Hence, developers handling static implementations typically incur the cost of large updates across numerous pages every time content is altered or the interaction pathway is modified. The dynamic implementation approach employs a programmatic method constructed in CGI, Java Servlets, PHP, ASP, or similar technology to algorithmically generate the multitude of pages for all of the pathways from some data repository.

Due to the statelessness of the HTTP protocol, context about interactions between a user and the content provider was not maintained from one request to another. For this reason, a logic layer is typically implemented within the Web site (statically or dynamically) with the responsibility of maintaining state and interaction information, commonly referred to as 'session' information. Thus, the interaction model extends to include some means for handling navigation and interactivity, producing Equation 2.2:

$$UserGoal = Content + Logic \tag{2.2}$$

Besides textual content and interaction logic, such stylistic objects as images, fonts, embedded applications, and layout styles assist in the presentation and readability of a document, not to mention the character of a website as a whole. It is because of these added, specific features that the explosion of the World Wide Web actually began, and helped migrate the Web from academic circles into the corporate world, drawing contenders such as Microsoft and Netscape into the development fray. The ability to transport not only content, but also style, is what defines the modern browser; unfortunately this capability also has littered the landscape of the Web with an array of platform specific and proprietary HTML tags [70] and differing implementation models [72]. Our next model of Web interaction is thus:

$$UserGoal = Content + Logic + Style \tag{2.3}$$

This model, described in numerous books and papers [33, 48], presents the main entities within the scope of modern Web development. It has been suggested [49, 70, 73] that these entities should be kept separate in order to develop the ideal Web system; more discussion on this topic is presented later.

The rise of ubiquitous devices brings a fourth entity into this model as smaller screen sizes and alternate presentation formats become more prevalent. The impending explosion of multimodal devices, foreshadowed by a call for top level domains specific to mobile devices [65], will require content providers to publish in yet another medium. Furthermore, there has been a rise in hybrid, or more aptly, multimodal techniques to facilitate user input. The rise is generally attributed to the work of Richard Bolt in his 'Put-that-there' paper [18, 44] – one of the first explorations into the effects of multimodal interactions as a means to remedy recognition and comprehension problems associated with single modality interaction. Hence, to include some means of detecting and presenting for specific modalities, a four entity equation has been defined exclusively by the author of this thesis:

$$UserGoal = Content + Logic + Style + Modality \tag{2.4}$$

This equation summarizes the rising generation of the Web, one that is highly stylized, highly accessible,

and increasingly hard to maintain and develop for. The ability to convey and attain information through more than one modality brings Web interactions closer to *dialogs*. A web interaction can be viewed as a conversation between the user and site and, with the benefit of multimodality, users will be able to interact via sequences involving both responsive and unsolicited interactions. This will lead to more natural and compelling scenarios of usage than the traditional turn-based hyperlink systems currently available. However, note that Equation 2.4 in factorized format is an *ideal* equation representing the Web as most developers advocate and, as we will see next, continues to be an elusive goal that modern web systems fall short of.

## 2.2 The Real Model

The problem with Equation 2.4 is not so much in the equation itself, but rather in its practice within the current web landscape. The content provider, in order to develop and present content to the user swiftly, would utilize a variant of the pure model in which two or more of the entities are tightly bundled into a single unit. Such 'bundled' solutions typically fall into one of the following categories:

- Template Driven Sites (User Goal = [ Content + Style ] + Logic)
  - Weblog sites (e.g., MovableType, Blosxom)
- Dynamically Stylized Sites (User Goal = [ Content + Logic ] + Style)
  - e.g., CSSZenGarden.com
- Database Driven Sites (User Goal = [ Logic + Style ] + Content)
  - News sites (e.g., CNN.com, WashingtonPost.com)
  - Web search (e.g., Google.com, yahoo.com).
- Statically Defined Sites (User Goal = [ Content + Logic + Style ])
  - User homepages

In each of the above, since at least two of the entities are tightly bundled, they cannot be easily altered without directly affecting another entity. If the model was completely factorized, one could easily afford changes to a given entity as well as support the inclusion of new entities. The lack of factorized components prevents separation and seriously hinders the usability and applicability of web systems in different settings. To understand these flaws, the bundled solutions shall be analyzed both in historical context as well as how they affect forward progress. As one might expect, the bundling of all three entities, as occurs in the aforementioned statically defined sites, would suffer from all of the flaws mentioned.

### 2.2.1 Style and Content Bundling

This bundling was conceived as a result of the inclusion of presentation tags into HTML during the Browser War [15, 73] mentioned earlier. For example, presentation tags such as the `<font>`,`<b>`, and `<blink>` were included in browser implementations to facilitate stylization of content. In the early evolution of the

Web, embedding the style into the content was the only means for including any degree of presentation with the content. As Microsoft and Netscape vied for control of the Web browser market, they added proprietary presentation tags to their respective implementations. Done without forethought to the consequences of their actions, the two companies left the Web a veritable mess with proprietary specifications of HTML that were largely devoted to presentation and not content, contrary to the original intent of the language [53].

In response to this explosive growth, developers, in an attempt to keep up with the Web browser war and ensure that their content would be available to all visitors, would produce content in a manner that was backwards compatible. The concept of backwards compatibility, as most appropriately defined in this context by Jeffery Zeldman [73], refers to the ideal of designing content that will be displayable on all browsers. It also refers to the practical version in which developers would utilize numerous proprietary tags and non-compliant syntax to create the same look for their content across the two major browsers of the time – Netscape Navigator and Microsoft Internet Explorer.

Backwards compatibility was hence originally encouraged so that content consumers could always see the content that was being delivered, even if their browser did not recognize some specific tags. The solutions adopted however are troublesome for the following reasons (aggregated from [53, 70, 73]):

- By combining style with content, all content carries the extra weight of the style. This drastically increases the amount of bandwidth necessary to provide just the content, in addition to the processing power required to render that content.
- Style information is interface-dependent and, by binding the content to a particular style, the solution approach negates the possibility of presenting content to a different interface (such as cell phones or PDAs).
- In practice, embedded style detracts from the structure of the content in that a developer might use `<font size=''14'' face=''arial''><b>Heading1</b></font>` for a heading instead of using the `<h1>Heading1</h1>` format. This lack of structure also reduces the accessibility of a Web site as it prevents screen readers from being able to use structural cues to distinguish document substructures (e.g., identifying a paragraph versus a header).
- Maintenance of the content requires the content provider to also understand style since she will have to add in the style information for their content, ensure that changes to that content do not disrupt the style, and maintain consistency across all pieces of content. This can be an enormous task, and one that most interaction management frameworks (see next chapter) seek to resolve.

The above reasons are quite substantial as they prevent both forward and backward compatibility of the content, and limit its delivery to a very narrow display medium. Furthermore, browsers do not all render the same tag in the same manner, leading to numerous issues of consistency across the same medium. In order to resolve some of these issues, the World Wide Web Consortium (W3C) released a style sheet technology specification in 1996 called Cascading Style Sheets (CSS) [1] that facilitates the separation of style from content. As good as CSS seems as a resolution, Web browsers were slow to properly, if at all, implement it as specified by the W3C. Even to this day, Zeldman believes that 99.9% of all Web sites suffer to some degree from this tight bundling of style and content [72, 73].

### 2.2.2 Logic and Content Bundling

The bundling of logic and content is one that dates back to the beginning of the World Wide Web. Web pages are interconnected to one another utilizing URLs, as envisioned by T. Berners-Lee in his original description of the World Wide Web [16]. A URL, as defined by RFC 1738 [68], is 'the syntax and semantics for a compact string representation for a resource available via the Internet.' As mentioned earlier, URLs allow developers to craft and implement interaction pathways through their content, directing users to end goals. The implementation of these pathways can be done either statically or dynamically – the former of which produces a tight bundling of the logic and content of the Web site.

Statically implemented interaction pathways imply one of the following conditions to be true:

- URLs are defined to an exact location that includes a specific static file name (e.g., http://www.example.com/index.html).
- Pathways are determined from a single snapshot of the content at development time, as opposed to a dynamic format that continually reprocesses the pathways from a content representation at request time.
- The interconnecting pathways between the pages are embedded in each page instead of utilizing a navigation centralization scheme to consolidate pathway management and maintenance.

With these conditions set forth, logic and content bundling could be simply construed as developing a Web site at a single instance of time and placing little to no forethought about future development and growth. However, most sites are not single instance developments, but are rather iterative in nature that are continually updated, added to, and removed from; any of these actions might drastically impact the interaction pathways of that Web site. Altering a tightly bound logic and content Web site would require a developer to go through every single affected Web page, identify which URLs have changed, and determine the best manner to reference the new interaction pathway. As one might conjecture, this can be an enormous task that invites the possibility of broken links, interaction pathway dead ends, and finally, a headache for the developer.

In response to these issues, scripting languages and Common Gateway Interface (CGI) implementations were developed and have since seen an enormous growth in recent years [39, 57]. The usage of scripting languages or CGI interfaces allow developers to design a Web site while still maintaining well-practiced software engineering principles, such as encapsulating the pathway logic through a Web site's content into its own entity. This engenders a dynamic approach which encourages the two entities, logic and content, to be only loosely bundled. In this dynamic format, developers can utilize some algorithmic means to determine interaction pathways through content, which, if designed correctly, should only require one development cycle. Generally speaking, this algorithmic implementation can be encapsulated into a single or a small set of scripts so that maintenance time or redesign of the interaction pathways implies only altering a small isolated population of the Web pages in the system. Due to these advantages, most modern Web sites utilize some degree of dynamic logic for maintaining the interaction pathways through their hypermedia systems.

#### Session and State

Another facet to content and logic bundling pertains to the statelessness of the HTTP protocol [32], i.e., there is no persistent information about interactions between client and server from one request and response to the next. It was designed in this manner in order to increase the simplicity of the protocol, which is directly attributed as the reason for its incredible adoption rate. However, the lack of state between interactions is flawed since information from one interaction cannot be utilized in a later interaction without some degree of external mediation such as cookies [56], thread management techniques [12], or specialized languages specifically designed for maintaining state [14, 57].

Web sites with tightly bundled logic and content generally do not address the problem of session and state within their interaction pathways. In these sites, the choices for the user to select from were made at design time, not at interaction time. When a dynamic, loosely coupled implementation is employed, the system can evaluate where the user has been, and attempt to extract some information about where the user might be heading or present information adaptively. There has been a large breadth of research into this area of personalization, especially systems that analyze web logs [54] and adaptive hypermedia [21, 22, 23]. The implications of personalization specific to this document will be examined in detail later. For now it suffices to note that these systems and the research areas they represent verify the advantage of dynamic, loosely coupled content and logic entities as opposed to a static, tightly bundled design.

### 2.2.3 Logic and Style Bundling

The bundling of logic and style is not as salient as the previous bundlings, due mainly to the limited modality in which the hypermedia of the Web is currently presented. To better expose this bundling, imagine a database driven web site in which the content of the system is continually being modified, removed, archived, or added to. To present this continually altering content body, developers will utilize fluid templates for their style with the interaction logic embedded into the style. In this way, content can be presented in a consistent manner through the templates and users can utilize interaction patterns common across all pages within the Web site. This scenario is an instance of logic and style bundling because the interaction logic is statically contained within the stylized templates and is bundled with the adopted presentation style.

As mentioned previously, the Web is being pushed beyond the desktop Web browser and into the handheld cell phone or PDA. In order to present concise content on these devices, new languages are being employed that are specifically tailored to the task. These languages (e.g., Wireless Markup Language (WML) and xHTML Basic) have different logic and style requirements than HTML, such as form handling, hyper-linking, and content headers. Since only a few content providers have even considered providing their content in more than one modality, this bundling is rarely seen as an issue. However this narrow thinking is analogous to the 'backwards compatible' scenario presented previously and therefore could be conjectured to have about the same results – a tangled Web [53]. Since multimodality has made its stake and is gaining ground with each new browser-equipped cell phone and PDA, this increasingly becomes more of an issue – one that cannot be ignored. For this reason, it is critical to ensure that the logic being applied to content is not tightly bundled to their presentation style and furthermore it is important to enumerate the fourth entity that exposes this bundling: modality.

## 2.3 Advocacy

Numerous developers have advocated the three entity factorized model presented in Equation 2.3 [7, 70, 73]. This would provide complete separation between each of the entities, allowing dedicated developers of each entity to work independently. This model was the original vision of Berners-Lee, but was lost during the Browser War between Netscape Navigator and Microsoft Internet Explorer. Advocates today propose that each content provider handle such separation individually, and even present varying means for factorization, e.g., <bigwig> [19], Magpie [28], specifications like CSS and xHTML. While these implementations attempt to resolve the issue of separating the three entities discussed so far (logic, content, and style), none present a means for handling the fourth: modality. Furthermore, with these systems, the burden of implementation and integration is placed on the content provider; this burden requires each content provider to keep abreast with the ever-changing tide of the Web. The following chapter contains evaluations such systems and identifies their resolutions to the problems addressed in this chapter, as well as potential issues that are left unresolved.

# Chapter 3

# Interaction Management

Interaction management, in the context of this thesis, refers to any architecture or framework that coordinates user interactions and system responses to deliver modification of the available content of a Web site. An interaction manager will commonly (although within this definition is not required to) implement some notion of session management atop the stateless HTTP protocol in order to distinguish users from one another and to support rich and personalized dialogs with the user. As shall be shown in the rest of this section, interaction manager implementations range from being fully static to fully dynamic, and from generative languages to embedded software systems. The wide range of research in this area clearly identifies the legitimacy of the underlying problem and the necessity of interaction managers at large, data-intensive Web sites. This chapter is arranged in a format that presents the most concrete and static implementations first and progresses towards the more abstract or dynamic solutions. All interaction managers surveyed here, with the exception of those covered in Sec. 3.4, assume the web development model given by Equation 2.3 whereas Sec. 3.4 adopts Equation 2.4.

## 3.1 Static Interaction Management

Static interaction management accurately portrays the lowest, most concrete level of Web development currently available. At this level, there is very little management of interaction since it is all statically bound along pathways identified at development time. Very few data intensive Web sites today are managed in this manner, due to maintenance and development costs of continually redesigning and implementing the interaction pathways every time a piece of information is added, removed, or modified.

### 3.1.1 HTML

Since HTML is an open interpretive language, any developer can create a Web site by coding it entirely by hand. This procedure can produce some highly optimized HTML code that tightly fits the content being present, in contrast to automatically generated HTML. However, the downsides of hand coded HTML—maintenance and development cost—generally prevent its use for most data intensive Web sites.

### 3.1.2 Visual Editors and Site Managers

In an attempt to reduce development and maintenance costs, direct manipulation tools such as Microsoft FrontPage and Macromedia Dreamweaver became popular for light to medium data-intensive Web sites. This allows the developer to visualize at design time the exact look and feel of the Web site as well as the interconnections of that site. These tools automatically generate the underlying static HTML code for the Web site and generally result in a solution where all three entities of Equation 2.3 are tightly coupled to each other. In recent years, however, some of these tools [11] have begun to disentangle the automatically generated code and utilize Web standards such as CSS [1] and xHTML [58] for the generated code.

## 3.2 Database Centered Interaction Management

Database centered interaction managers (DCIM) are characterized by their specific focus on a database management system (DBMS) for handling the majority of interaction management tasks [33]. These interaction managers range from being no more than a Web based front end for a database view to an entire new class of DBMSs. Within this section, the Araneus project is presented as a DCIM that exemplifies a distinct class of DBMSs with its web base management system architecture.

### 3.2.1 Araneus

The Araneus web-base management system [13, 52] is an interaction manager that views Web based interactions in the same manner as database interactions. More accurately, the Araneus project architects view it as an evolutionary step in database management systems, in that it is a management system for handling collections of heterogeneous, highly structured or semi-structured data [8], specifically targeted at the Web. To this end, Araneus handles Web requests in a similar fashion to modern DBMSs by using queries, views, and updates as application classes. However the main difference between most modern DBMSs (e.g., Oracle and Microsoft SQL) and Araneus is that Araneus utilizes a page-oriented data model (called ADM), for describing pages and navigation, whereas DBMSs do not utilize any such model. Templates are provided for applying presentational style onto HTML structures.

The Araneus approach to developing a Web system presents an interesting model by which data can be stored and requested almost directly from a DBMS. A key feature of Araneus, with respect to this discussion, is the manner in which issues of structure, navigation, and presentation are kept distinct from each other. Also of interest is the two distinct tracks for application of expertise, database and hypertext, methodology that Araneus employs.

## 3.3 Model Driven Interaction Management

Model driven interaction managers (MDIMs) are held in high regard [33] due to their attention to life-cycle issues, abstract level of automation, and ability to facilitate conceptual modeling. Each MDIM proposes some conceptual model for defining how to handle interaction management. This model is then processed to generate the code necessary to instantiate the model. This genre focuses more on the software engineering

aspect of interaction management construction than the presentation functionality, although there is almost always some provision made for presentation functionality.

### 3.3.1 WebJinn

The WebJinn system [46] was born out of an attempt to alleviate software level issues inherent in modern Web development. The designers of this system identified two key issues with common Web development practices – tangling and scattering of code fragments. Tangled code in the perspective of the system developers accurately maps to the aforementioned tight bundling and intermingling of the entities of Equation 2.3. The scattered code refers directly to the lack of encapsulation for any of the entities of Equation 2.3 within modern Web development.

WebJinn system designers state that the tangling of code 'drastically increases the dependency between different development groups resulting in high development and maintenance cost' [46]. In an attempt to resolve this issue, the model-view-controller (MVC) [36] paradigm has been employed in frameworks such as Apache Struts to exert design restrictions that would discourage the intermingling of content, logic, and style.

To resolve the issue of code scattering, the designers introduced a new model, Extension Point (XP), which localizes the structure of the entire Web site. This model was arrived at by inspecting the nature of structure-dependent code and noticing that it generally appears in clusters. The designers determined that by abstracting out these clusters (creating an extension point), one could design a centralized structure framework that is completely distinct from the abstract Web application that contains it.

The resulting hybrid of the previous two solutions is sometimes referred to as domain driven web development (DDD) [46]. WebJinn/DDD makes it possible to construct Web application components as well as applications built from those components, in a flexible and loosely coupled manner. The loose coupling in the WebJinn systems reduces the development and maintenance cost of the Web application as a whole.

Of particular interest within the WebJinn framework, with respect to the topic of interaction managers, is the identification of two key problems inherent with modern Web development systems and potential solutions to those problems. Although WebJinn does not attempt to directly solve the problem of interaction management, it does present a platform for creating a loosely coupled Web system.

### 3.3.2 AutoWeb

The AutoWeb system [34] adopts a novel approach to the development of an interaction manager. The system is more of an interaction manager generator than it is an interaction manager itself. AutoWeb begins the Web development procedure by allowing the developer to visually model their Web site using the HDM-lite modeling framework, a descendant of the entity-relationship model [27]. This modeling produces a series of schemas representing the structure and organization of the desired Web site. From these schemas, AutoWeb generates an interaction manager tailored specifically to the domain that was initially modeled. This generated interaction manager is meant to be populated with content by the provider and stylized by a designer. It should be noted at this point that the current implementation of the AutoWeb system does not leverage W3C-complaint HTML or CSS. However, AutoWeb has the unique ability to apply human

interface guideline specifications to the entire site, enforcing a standard, uniform presentation manner to the user, and therefore reducing the burden of presenting a consistent Web site design.

Several of the identified benefits of the AutoWeb system include rapid development and prototyping, a wide variety of navigation modes, and true model-driven approach. From a system perspective, the AutoWeb system appears to be a very complex architecture – one that is meant to be used, not by content providers, but by advanced Web developers who have experience in modeling Web systems.

### 3.3.3 JWeb

JWeb [17] is not formally classifiable as an interaction manager, but rather attempts to provide developers with a framework for prototyping a Web site. Systems such as JWeb provide some degree of interaction management but may not be serve as an optimized, complete, and final solution. JWeb utilizes the HDM [35] methodology to direct the user through the process of identifying and describing hyperbase schemas (content) and access schemas (logic), both maintained in an XML format.

This system is designed to create iterative design prototypes of a Web site in that it only generates a Web site's schema and not the program code necessary for a working implementation. However, schemas are relatively meaningless for prototyping unless the developer can actually visualize and interact with them. JWeb handles this issue through an HDM interpreter which captures user input, mimicking Web interactions, and converts the user input into valid parameters for JWeb's Navigation Engine.

The JWeb architecture is important to our discussion of interaction management because it utilizes XML to achieve flexibility, modularity, and exchange of information between its software modules. In fact, the developers of the JWeb identify that the use of XML facilitated "a very easy strategy of enhancement, addition, substitution, or replacement of all different tools" [17].

## 3.4 Multimodal User Interface Interaction Management

The existence of multimodal user interface interaction mangers (MUIIMs) acknowledges the relevance of modality within the Web development model (Equation 2.4). Currently, the interaction managers in this category are primarily concerned with devising an intelligent method for detecting the capabilities of the browser and rendering presentation information accordingly.

### 3.4.1 Adaptive User Interface Technology

Adaptive User Interface Technology (AUIT) [40, 41] is a set of JavaServer Pages (JSPs) custom libraries for development of user interfaces that adapt to the device that is presenting it. These extended JSP tags describe device independent user interface elements and layouts, which are transformed at runtime into device specific user interface information and delivered to the appropriate device. Since the user information is stored in XML, the transformation is handled using the eXtensible Stylesheet Language for Transformation (XSLT) [3]. AUIT includes a visual user interface designer that allows developers to quickly prototype their interfaces for each of the possible presentation devices. Once the developer has decided on an appropriate

interface, the design is then automatically converted to the necessary JSP tags. This technology is specific to the latter two entities in Equation 2.4 as no facilitation is included for handling the content and logic entities. AUIT currently supports presentation onto desktop Web browsers, HTML, and wireless Web browsers (using WML).

### 3.4.2 User Interface Markup Language (UIML)

The User Interface Markup Language (UIML) [9, 64] is defined as a 'meta-language for describing user interfaces' [69] in a operating system, platform, and interaction agnostic manner. In order to accomplish this goal, UIML carves up any user interface into a six tuple comprising:

- Structure and flow – a structure defined from a provided domain specific vocabulary;
- Style – elements in the interface that define its look; this may include color, font, or layout information;
- Content – all text, images, and sounds that will be expressed in the interface;
- Behavior – action rules that define interface reactions to user input or action;
- Logic – includes the application logic for the interface elements; and
- Presentation – includes interface controls such as buttons or text fields.

With UIML, developers can provide a consistent interface to multiple devices, modalities, and platforms without creating an interface for each of them. As its name implies, UIML is an XML derivative thus reducing the learning requirement for development. This design decision has the benefit of allowing novices to expert programmers to rapidly design, construct, and deploy user interfaces with little to no training. Once a UIML document has been created, it requires a mapping procedure to convert the platform agnostic UIML tags into actual static program code for technologies such as Java/JFC, PalmOS, WML, HTML, and VoiceXML [50]. One distinction between the aforementioned AUIT and UIML is that UIML provides some means for both logic and behavior specification, exemplifying a multimodal interaction management framework instead of a means for producing multimodal interfaces.

## 3.5 Domain Specific Language Solutions to Interaction Management

Domain specific languages attempt to reduce the degree of technical information required to create an interaction manager by exposing a language that a domain expert would understand. These solutions provide a layer of abstraction that allow the developer to create complex systems using very simple, domain dependent languages (hence the name). Most of these languages are then processed to generate the interaction manager, which is then used without modification, completely hiding the low-level program code from the developer.

### 3.5.1 The 5S Framework

The 5S framework and its language, 5SL, while not aimed specifically at Web interaction management, is pertinent here for its approach to factoring the design of large complex information systems. It is rooted in the 5S formal theory [38], a specification of a digital library (DL) as viewed in five dimensions [37]:

- Streams – describes the type of content within a DL. Examples include text, video, and software application;
- Structures – expresses the organizational format, and aspects thereof, with respect to a DL. Examples of the model include hypertext, catalogs, and directories;
- Spaces – defines interfaces, both human and machine, of DL components. Examples include user interface and retrieval models;
- Scenarios – describes the behavior of DL services. Examples include events, triggers, and actions; and
- Societies – defines the human participants and their interrelationships that are rooted around a DL. Examples include communities of interest, managers, funders, and users.

From these five different models, a declarative language was defined, aptly named 5SL, which can be used for specifying and generating DLs. Furthermore, 5SL facilitates describing a DL at an abstract, domain-specific level, thus allowing the creation of a DL by people with only domain expertise and not necessarily any computer experience.

The 5SL framework, although being specific to DLs, provides insight into the adoption requirements of any Web system in that by using domain-specific information, the 5SL framework is capable of being used by a broad range of users, so long as they maintain some degree of domain expertise. Also by describing information in a human and natural manner, the 5SL framework can be quickly understood through tacit knowledge, instead of requiring DL-specific training. These benefits are directly associated with the abstraction layer provided by the 5SL framework, a component that should, to some degree, be present in an interaction manager as well.

### 3.5.2 Strudel

Strudel [29, 30, 31] focuses on utilizing a declarative language for a Web site's structure and content, much like 5SL is applied to DLs. Strudel was designed to leverage a variety of concepts from database management systems and Web-site creation tools. From these concepts, the designers of Strudel identified attainment of separation of entities within Equation 2.3 as a key goal. Strudel's declarative language, StruQL, is a query language for specifying the content and structure of a Web site; it also can be viewed as a software artifact as it has been designed to improve modularity and reusability of Web site designs. From this language, an interaction manager is derived and automatically generated through processing done by the Strudel query inspector.

Strudel is one of the few systems that adheres to Equation 2.3 at an initial design level. As an interaction management system, Strudel provides a refined domain specific resolution to handling the issues inherent with data intensive Web sites, while still retaining extensibility and reusability for future growth.

### 3.5.3 <bigwig>

<bigwig>[19] is a high-level, domain-specific language for programming interactive Web services. The <bigwig>language was designed to be a descendent of the MAWL research language [12]. The design of <bigwig>focuses mainly on the content and logic components of Equation 2.3, leaving a clean separation between these entities, and from the presentation style to be applied. Also the design of <bigwig>affords a strong type system since it is based on compilation instead of interpretation as employed for most scripting languages.

As an interaction manager, <bigwig>focuses on users in a session centered approach, a design element rooted in the MAWL project. This focus approaches interaction management as a collection of distinct session threads akin to remote procedure calls (RPC). Each session thread relates to an ordinary sequential program which describes the session's intended flow. Further, each session thread is addressed as a unique URL which points to the session's current HTML page; this form of session caching, however, prevents back button browsing, viewed as a non-necessary functionality.

To structure the session outputs, XHMTL templates are employed, including <bigwig>specific tags that are replaced with session generated information at presentation time. The templates are handled in a manner that facilitates document composition by allowing placement of xHTML fragments into other xHTML fragments. This allows flexible construction of site structure, providing developers with a limitless degree of options for site implementation.

To facilitate various domain-specific requirements, a declarative sub-language was developed to handle specific tasks such as form field validation. The <bigwig>language also provides functionality for dynamic caching, concurrency control, security, and database connectivity.

The <bigwig>language is, without question, one of the most robust interaction managers surveyed here. This robustness, however, comes at the cost of initial development for the content provider seeking to publish their information to the Web. <bigwig>requires the content provider to have access to software developers who can understand the <bigwig>language, syntax, and semantics.

## 3.6 Discussion

It is clear from the above survey that the problem of interaction management can be approached from diverse perspectives and that every project is indicative of a distinctive set of requirements, motivations, and technological constraints. Our goal in this thesis is to develop a web interaction framework that brings us closer to the vision of achieving human-web dialogs, that is well-factored from a software engineering point of view, and is representative of the best practices of web technology.

# Chapter 4

# WS://IM Software Framework

WS://IM (web service for interaction management; pronounced as w-sim) deviates from the examples in the previous chapter as it seeks to handle interactions between content providers and users in a manner that implements the ideal model presented in Equation 2.4, with specific attention paid to supporting rich contextual human-site dialogs. WS://IM is embodied as a centralized interaction manager that is capable of providing intelligent multimodal interactions for a potentially limitless number of content systems. Going even one step further, WS://IM offers the unique ability to facilitate interactions that are *out-of-turn*, which allow users to substantially alter and transform Web site interaction sequences. This allows users to navigate the content in a more effective and natural fashion than is facilitated by purely directed, in-turn interactions. To understand what this capability means, it is helpful to consider a practical setting of multimodal web interaction.

## 4.1 Setting

We pay careful attention to studying how interactions in a modality-rich environment differ from traditional web access paradigms. A simple example can be seen in the use of speech to augment traditional hyperlink based interaction. Notice that speech permits natural ways to perform certain types of tasks and helps compensate for deficiencies in traditional hyperlink access (which can get cumbersome on small form-factor devices). Second, speech-enabled websites help improve accessibility for the more than 40 million visually impaired people in the world today. As a result, using speech leads to the possibility of a conversational user interface [51] that combines the expressive freedom of voice backed by the information bandwidth of a traditional browser. More importantly, while the common use of speech on a website is to support navigation of existing site structure via voice, speech can actually be used to support new functionality at a website, not otherwise possible.

### 4.1.1 Motivating Example

Consider the following dialogs between an information seeker (Sallie) and an automated political information system.

*Dialog 1*

1 **System:** Welcome. Are you looking for a Senator or a Representative?
2 **Sallie:** Senator.
3 **System:** Democrat or Republican or an Independent?
4 **Sallie:** Republican.
5 **System:** What State?
6 **Sallie:** Montana.
7 **System:** That would be Conrad Burns. First elected in 1998, Burns ...
(conversation continues)

*Dialog 2*

1 **System:** Welcome. Are you looking for a Senator or a Representative?
2 **Sallie:** Senator.
3 **System:** Democrat or Republican or an Independent?
4 **Sallie:** Not sure, but represents the state of Montana.
5 **System:** Then it is either a Democrat or a Republican, there are no Independents from Montana.
6 **Sallie:** I see. Who is the Republican Senator?
7 **System:** That would be Conrad Burns. First elected in 1998, Burns ...
(conversation continues)

It is helpful to contrast these dialogs from a conversational initiative standpoint. In the first dialog, Sallie responds to the questions in the order they are posed by the system. Such a dialog is called a *fixed-initiative* dialog as the initiative resides with the system at all times. The second dialog is system-initiated till Line 4, where Sallie's input becomes unresponsive and provides some information that was not solicited. We say that Sallie has taken the initiative of conversation from the system. Nevertheless, the conversation is not stalled, the system registers that Sallie answered a different question than was asked, and refocuses the dialog in Line 5 to the issue of party (this time, narrowing down the available options from three to two). Sallie now responds to the initiative and the dialog progresses to complete the specification of a political official. Such a conversation where the two parties exchange initiative is called a *mixed-initiative interaction* [10].

At this point, it must be clear that system-initiated modes of interaction are easiest to support and are the most prevalent in web interactions today. For instance, a webpage displaying a choice of hyperlinks presents such a view, so that clicking on a hyperlink corresponds to Sallie responding to the initiative. The reader can verify that the first dialog above can be supported by a three-level tree-structured HTML site presenting options for branch of congress, party, and state. The second dialog becomes practical with the advent of multimodal interfaces. For instance, if Sallie has the ability to talk into her browser, she can provide unsolicited information using voice when she is unable to make a choice among the presented hyperlinks. In addition, if the system can process such an out-of-turn input, it can continue the progression of dialog and tailor future webpages so that they accurately reflect the information gathered over the course of the interaction.

The above example highlights the unique perspective that WS://IM brings to bear upon web interaction management — namely that of a flexible and personalized dialog, with shifts of initiative that can happen at any step. How to support such flexibility in a factored system architecture is the contribution of this thesis.

## 4.2 Design Considerations

In designing the WS://IM framework, several equally important, design considerations were asserted early in the approach. The first was to directly address the issue of maintaining separation of the entities (Content, Logic, and Style) within Equation 2.3. This makes WS://IM flexible enough to incorporate new entities, such as modality, as they become prevalent. This view is validated in that WS://IM is not only feasible for multimodal interaction, but in fact facilitates it. The second design decision was to seamlessly support the co-ordination of human to site multimodal dialogs, irrespective of the conversational shifts of initiative that might happen. The third design decision was to develop the architecture for WS://IM in a manner that could be distributed across many machines as easily as it could be encapsulated on a single machine. Finally, WS://IM was designed to facilitate a centralized approach to Web development such that the workload burden is placed not on the content provider; rather, the emphasis shifts to facilitate easy adoption of the technology. Some of these considerations are elaborated below.

### 4.2.1 Factorization

Web professionals, such as Zeldman and Veen, advocate that the separation of content, logic, and style should be the goal of all Web developers [70, 72]. WS://IM follows suit by viewing the ideal, factorized E.q. 2.3 as more than just an ideal and making it an initial design requirement. Each piece is encapsulated, separated, and loosely coupled so that any of the entities can easily be removed, modified, or made anew without affecting the other entities. In addition, by incorporating the fourth consideration of modality, WS://IM is uniquely positioned to leverage the aforementioned next generation Web.

### 4.2.2 Multimodality

The issue of modality is a new entity that in all existing web interaction managers had been statically fixed to screen-based Web browser interfaces and traditional forms of user input. The rise of ubiquitous devices, as mentioned previously, has suggested that such a tight coupling of style and logic would be a poor design decision. For this reason, WS://IM sought to easily and uniformly deliver the same content in an unlimited number of modalities and uniformly capture input from different input mechanisms. Further requirements stemming from this facet include:

- All modalities should be immediately available to all content.
- New modalities should be easy to incorporate into the system.
- The system should be capable of determining the best modality (or modalities) for presenting the content and rendering it in such fashion.
- The capability of simultaneous modalities, e.g. utilizing voice and hypertext at the same time, should be retained.
- The modality handling should not restrict a content provider's distinctive creative requirements or intentions per modality.

Within the WS://IM architecture, an entire manager component is implemented to handle these requirements. This manager is responsible for determining the most effective modality or modalities for presentation, depending on the capabilities of the browser. This determination is made through a series of evaluations ranging from browser provided HTTP request header information to JavaScript plug-in detection. Once an appropriate combination of modalities is made, WS://IM applies a series of eXtensible Stylesheet Language for Transformations (XSLT) [3] to cascade each modality onto the content. The term cascade in this context is analogous to its usage in the cascading style sheets technology, in that each modality is layered atop all previously applied modalities. This form of modality application allows WS://IM to present not just one, but many different modalities in a single package.

The WS://IM modality manger functions in a manner that accommodates a system default styling per modality as well as a site dependent styling per modality, and any combination thereof. This flexibility has been designed into WS://IM to allow each content provider the ability to provide a specific look and feel for their content while still operating in a centralized framework, as explained further below.

### 4.2.3 Distributable Architecture

In designing the architecture for WS://IM, particular attention was placed on establishing an architecture that facilitates and encourages growth and adoption. This is a vital component to any Web system since it will operate in a realm that is constantly changing and reinventing itself. In order to facilitate growth, WS://IM was designed to be distributed across numerous networked computers, allowing space and processing power requirements to be identified and adjusted on a component basis, not on a system basis. As for encouraging growth, the architecture of WS://IM leverages only open standard, XML based protocols. This design decision was made to prevent proprietary lock-in and provides a language-agnostic communication medium [26]. Through these two critical design decisions, the architecture of WS://IM was made flexible enough to facilitate rapid changes, distribute computing, and comply to open standards.

### 4.2.4 Centralization

Centralization is an interesting issue to introduce at this point, especially since it may initially seem to contradict the previous design decision. However, centralization in this sense addresses the management of interaction and not the components that make up the system. It refers to the handling of multiple content providers through a single, centrally maintained interaction manager. This differentiates WS://IM from most of the previously mentioned interaction managers, in that WS://IM is a single instance manager that is capable of handling multiple Web content systems, simultaneously. The decision to centrally host the interaction manager provides the following benefits over deliverable system distributions:

- Reduced start up costs for content providers, since they need only provide their content and do not need to install, configure, and maintain an entire Web site.
- Single point maintenance of the system which, by virtue of the system design, updates all of the Web sites created by the system.
- System maintenance and upgrades are handled by experts with the architecture and design of the system.

- Hosting costs for companies that utilize WS://IM are drastically reduced; since they only need to deliver their content, WS://IM handles the rest.

## 4.3 WS://IM Architecture Overview

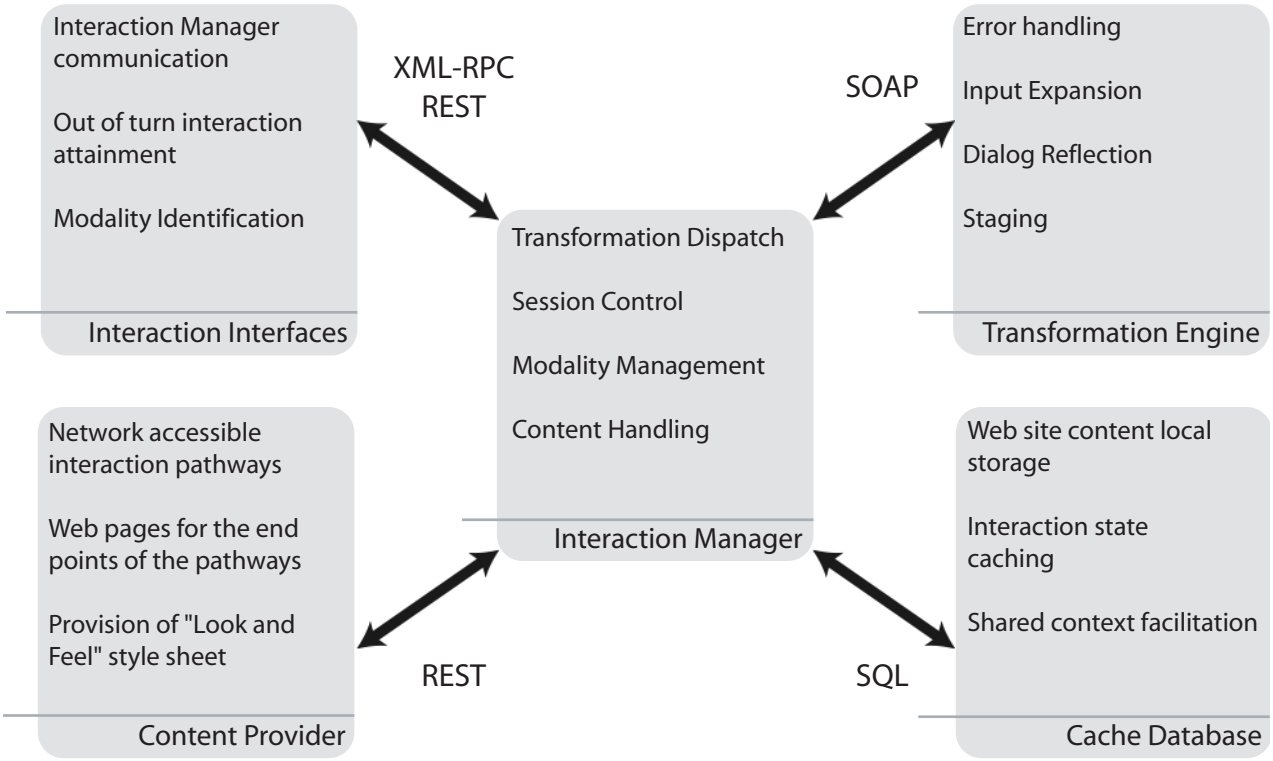


Figure 4.1: Component architecture of the WS://IM framework.

WS://IM was designed to be a distributable component architecture, allowing various components to be physically separate, while still acting as a whole. Fig. 4.1 presents the distributable components within the WS://IM architecture as well as the set of functionalities associated with each component. As one may immediately notice, the set of requirements on the content provider is at a minimum, a benefit attributed to the design decision of system centralization. Another property of this architecture is that it is not limited in scope to interaction management; in fact interaction management is merely a component of the framework, and covers all of the functionality involved in mediating between the end user and the content provider. Finally, this figure also identifies the connections between components, as well as the communication protocols employed for connecting them. The rest of this section shall describe each component and finally the communication protocols between the components.

### 4.3.1 Interaction Interfaces Component

The interaction interfaces component can be viewed as the user's gateway into the WS://IM framework (see Fig. 4.2). Interaction interfaces provide a means for interacting with WS://IM in either a conventional (in-turn) or non-conventional (out-of-turn) manner. The initial, and at this point most common, interaction interface for WS://IM is the Web browser, which is capable of only facilitating in-turn interaction sequences. The second (Extempore), leverages users' familiarity with toolbar interfaces, and provides a way to supply out-of-turn textual input. The third (SALTII, for SALT Interaction Interface, pronounced 'salty') utilizes a rapidly emerging technology for integrating speech into Web browsing. The final interface (WMLOOT, for Wireless Markup Language Out Of Turn, pronounced 'wml-oot'), presents the user with a simple WML form for providing text-based out-of-turn input on a cell phone or other handheld device. The Extempore and SALTII interfaces employ a common JavaScript toolkit which handles communication with the interaction manager (see next section). This toolkit also is designed to reduce the development costs for future interaction interfaces (e.g., PDAs and 3G phones).

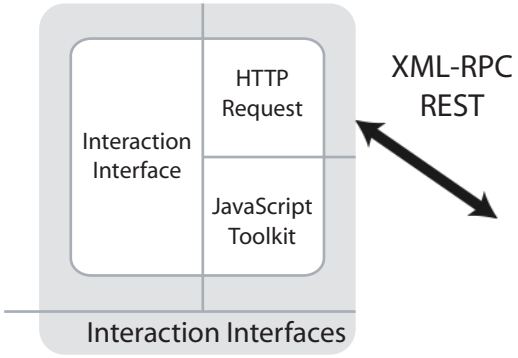


Figure 4.2: The interaction interfaces component implementation within the WS://IM framework.

#### Web Browser

The WS://IM operates as a normal Web site would in any Web browser offering conventional, in-turn interaction that directs the user to their end goal. It is important to note that the WS://IM framework will work in *any* Web browser as it complies with the W3C standards for xHTML 1.0 [58] and CSS Level 1 [1] when using the default styles. Since content providers have the ability to alter the presentation style specific to their own content, xHTML compliance cannot be assured. While this is the content provider's prerogative, WS://IM as a system does provide the basis for providing a Web site that is not, and for the foreseeable future will not become, obsolete [72].

#### Extempore

The Extempore toolbar [61], shown in Figure 4.3, was developed using the XML User interface Language (XUL) [2] and JavaScript for the cross-platform Mozilla Web browser. It was designed to be non-invasive

and to become active only when the user is visiting a Website that leverages the WS://IM framework. By displaying a lightweight, text-based interface, Extempore leverages users prior knowledge to provide a familiar and easy method of interaction. It is important to note that Extempore is embedded in the Web browser, and not the Web site. It also is not a site-specific search tool that returns a flat list of results similar to the Google toolbar.


Figure 4.3: The Extempore toolbar, an interaction interface within the WS://IM framework.

### SALTII

The SALTII interface is built using the SALT [5] XML-based markup language, which facilitates the embedding of speech and grammar tags into HTML to realize Web pages capable of speech input and output. The current SALTII implementation requires the SALT voice recognition plugin for Microsoft Windows Internet Explorer 6.0. Using this interface, users could potentially carry out an entire dialog with speech alone, using speech for not only out-of-turn interaction, but in-turn as well. Since it blends screen and voice modalities, SALTII can be considered a hybrid modality interaction interface. Such cascading modality is exposed in greater detail in the following chapter.

### WMLOOT

WMLOOT, shown in Fig. 4.5, is a unique interface that allows users to quickly provide input in an environment that suffers from constrained screen space and reduced bandwidth – the wireless environment. The WMLOOT interface brings the functionality of the Extempore toolbar into the cell phone and handheld arena. It differs from Extempore, however, in that it is actually delivered as an embedded component to the accessed WML page. This was done to allow the interface to be displayed regardless of the browser that is used, which is important given the plethora of WML browsers currently available.


Figure 4.4: SALTII, a multimodal voice and text interaction interface within the WS://IM framework.


Figure 4.5: WMLOOT, an interaction interface for wireless devices within the WS://IM framework, as seen in a Sony Ericcson T68i.

**Future Interfaces**

The aforementioned JavaScript toolkit was designed with future development in mind and factors the entire system to roughly five simple functions. Since it is developed using the vcXMLRPC JavaScript library, the functionality of the interaction manager can be remotely accessed, enabling shared context scenarios [25]. This toolkit has been made available at the project's Website (http://pipe.cs.vt.edu) for developers of other interaction interfaces.

### 4.3.2 Transformation Engine Component

The transformation engine component of WS://IM, shown in Fig. 4.6, currently leverages *staging transformations* [24, 55], a formal theory for reasoning about hierarchical staging notation and for simplifying dialogs using program transformation techniques such as partial evaluation [43]. A key facet of this transformation engine is that the underlying simplification of the dialog is abstracted to a point that it doesn't require distinguishing between in-turn and out-of-turn input. This property is crucial to the non-anticipatory nature of staging transformations. The WS://IM interaction manager (see next section) preserves this property to the fullest. To apply the staging transformations theory, the transformation engine represents interaction pathways within a Web site as XML documents and implements the transformations using XSLT technology. The XML documents summarize the hyperlink structure, the hierarchical staging notation, and indirectly the vocabulary comprising of legal partial input.

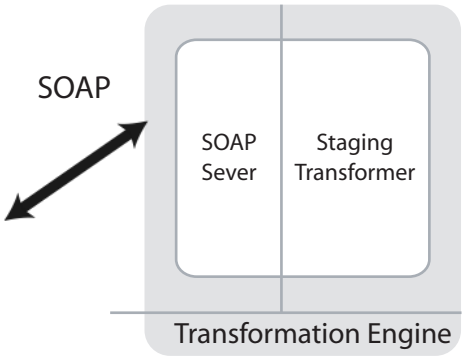


Figure 4.6: The transformation engine component implementation within the WS://IM framework.

The original version of the staging transformer, designed by Saverio Perugini, accepted the input necessary for a transformation through a series of command line parameters. To accomplish the WS://IM design decision of a distributable architecture, we resolved to bind a Simple Object Access Protocol (SOAP) [6] server onto the Staging Transformer using the gSOAP toolkit [4]. This process was relatively trivial since the toolkit was capable of automatically generating the entire SOAP server when given a function header stub file. This generated SOAP server was then compiled directly into the staging transformer producing the transformation engine component.

### 4.3.3 Interaction Manager Component

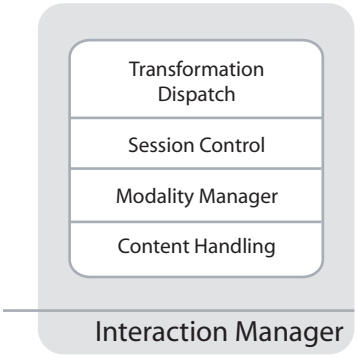


Figure 4.7: The interaction manager component implementation within the WS://IM framework.

The interaction manager, shown in Fig. 4.7, is the focal point for the WS://IM framework. It is essentially responsible for the coordination of communication between one or many content providers, a single, non-concurrent transformation engine (explained later), and one or many interaction interfaces. It is at this point that the WS://IM framework embodies the ideal Equation 2.4 as each facet is enumerated into its own subcomponent and the interaction manager merely serves as the summation of those parts. The following sections provide a brief overview of each of the interaction managers subcomponents.

#### Content Handler

The primary responsibility of the content handler is to dynamically determine if a Web site is manageable by the WS://IM framework. To determine this, the content handler uses a simple representation state transfer (REST) request for a document named 'content.xml' located at the content provider's root Web directory (i.e., the Web site http://www.example.com would be manageable if there exists a file at http://www.example.com/content.xml) . This document is meant to supply the representation of the Web site's entire content structured in a manner that conveys the desired in-turn hierarchy (out-of-turn interactions are handled by explicit manipulation in the transformation engine). If the content file is not present, it indicates that the Web site cannot be managed by the WS://IM framework. The content handler is also responsible for retrieving, caching, and updating content from compliant Web sites. The content file is cached in an external database (MySQL is used in the current implementation) for fast transformation computations. It also will signal a trigger event that will initiate the activation of the aforementioned interaction interfaces, as appropriate. From this point, the content handler is responsible for ensuring the currency of the representation and updating the file as appropriate.

Interaction state caching, also managed by the content handler, was originally implemented by trivially associating intermediate dialog states with content files generated over the course of an interaction. A more sophisticated solution was later employed that utilizes a caching policy to exploit the structure of program transformations. Under this caching policy, if a user is requesting a partial evaluation with respect to 'digital library multilingual,' but the cache only contains a document that has been evaluated with respect to 'digital library,' we can partially evaluate this document internally with respect to the remaining input, namely

'multilingual', thus removing the need to partially evaluate from the root document. This revised approach to interaction state caching has drastically reduced the storage and processing complexity of the overall WS://IM system; a highly beneficial improvement especially when dealing with large content systems.

### Transformation Dispatch

The transformation dispatch is responsible for handling communication with the transformation engine component. It handles connecting to the transformation engine as well as notifying the interaction interfaces if such a connection cannot be made. This notification is done to prevent potential invalidation of users' expectations due to a system error, abstracting the user from dealing with internal errors until they can be fixed. If a connection can be made, transformation dispatch is responsible for attaining user input, agnostic of in-turn or out-of-turn provision, from any of the interaction interfaces. With this received partial input, transformation dispatch formulates a message including the partial input and the content file that the input applies to, for delivery to the transformation dispatch. Recall the transformation engine component will not know, or need to know, whether the partial input is a result of browsing or of supplying some information out-of-turn; this is why the transformation dispatch formulates the message the same regardless of input manner. Finally, transformation dispatch supports the marshalling and un-marshalling of transformation requests into Simple Object Access Protocol (SOAP) messages, as well as the transmission and reception of those messages across HTTP in accordance with the SOAP specification.

### Session Control

Session control has a responsibility that differs from most other Web systems' concept of session management. The notion of 'state' within a dialog is just its representation, since it succinctly summarizes all remaining dialog options. Furthermore, the transformation engine does not explicitly manipulate state and is therefore a purely functional entity. Thus the goal of this system's session control is merely to distinguish one user's interaction session from another. Due to our uniform handling of in-turn and out-of-turn inputs and the variety of interaction interfaces available, session tokens (in this case a ten decimal digit session identifier, size and format was arbitrarily chosen) are stored in multiple different places, one in each interaction interface and a final in the browser itself. This multi-headed session format negates the application of most modern session management packages, which are primarily concerned with tracking browsing interactions. Session control was specifically designed to handle this issue as well as to handle the normal session management issues (e.g., back button browsing and threaded browsing).

### Modality Manager

The modality manager is responsible for transforming the information returned from the transformation engine into the visitor's native presentation format. Towards this goal, the modality manager must first identify the user's browser capabilities and determine how best to stylize the content. Notice that the modality manager evaluates and potentially utilizes all of the presentation styles, instead of just picking a 'safe' presentation style, like HTML. To handle the multiple possible combinations of modalities, the modality manager enumerates the following two types of modalities:

1. Base – a modality type whose syntax, structure, or language requires a complete and total restructuring of the existing document. Examples of base modalities include HTML, WML, and XML – as they cannot be embedded into any other language as per their respective specifications.

2. Extension – a modality type that can be embedded into another language's syntax. Examples of extension modalities include SVG, SALT [5], and VoiceXML [50].

With these two enumerations, the modality manager will first determine and apply the most appropriate base modality to the content. This is generally determined through analysis of the HTTP request headers sent by the Web browser in a field labeled 'USER_AGENT.' With the base modality applied to the content, the modality manager will analyze which extension modalities are also appropriate for application to the modified content. Notice that potentially more than one extension modality may be applied, whereas only a single base modality is ever applied. In order to fulfill the requirements of some modalities, such as SALT based voice modality, the modality manager might have to define and introduce suitable content specific tags into the HTML page, which is accomplished by analyzing the remaining dialog options. Currently, the modality manager supports xHTML, WML, and SALT, but is capable of easily supporting any structured presentation format including X3D, VRML97, PDF, and raw text by simply defining a XSLT style sheet to convert an XML file to that format.

The modality manager is also responsible for evaluating if a content provider has developed a look and feel that is specific to the content. Development of a content-specific look and feel can be done by downloading the publicly available modality style sheets (http://pipe.cs.vt.edu/pipe/style/) and modifying them to accomplish the desired look and feel. If a content-specific look and feel has been developed, it can easily be used within the WS://IM framework by placing it in the same directory as the previously mentioned content file. The modality manager will utilize a REST request to obtain the file and use it in place of the system version of the same file. This content-specific look and feel application allows WS://IM to accommodate multiple Web sites, each having a unique look and feel, to coexist through the same framework.

### 4.3.4 Cache Database Component

The cache database component, as previously mentioned, is used to store both pure XML content documents as well as interaction state caches. This component can be any SQL compliant database server and is implemented in the WS://IM instantiation at the time of this writing using MySQL version 4.1. The database cache component for WS://IM is trivially implemented in two tables – one for the pure XML content and one for the cached interaction states. These two tables are separated in order to ensure a distinction of the original content and the reduced content that represents interaction state. Since the database component is an open standard format, it facilitates the potential for various shared context initiatives [25]. A shared context system could easily search and retrieve a user 's interaction state from this database component, based on information such as time and user identification. In the current WS://IM framework, the user identifier is automatically generated, as described in session control, but it conceivably might be replaced, or used in tandem, with a shared context login and password identifier.

### 4.3.5 Content Provider Component

Within the WS://IM framework, the content provider component's responsibility was deliberately kept to a minimum. This is done in order to reduce the degree of labor required for *any* content provider, regardless of Web experience, to present a quality multimodal Web site. The content provider component has a series of requirements that require adherence to, in order for the content to be considered 'WS://IM compliant' and, therefore, capable of utilizing this framework. The following sections present the relatively small list of requirements in the suggested order of completion. They assume that the content provider maintains the Web site located at the URL http://www.example.com in order to illustrate naming issues.

#### Providing the Content

The main catalyst, and hence requirement, for the WS://IM framework is the content of the Web site. The content of the Web site should be structured as a well formed XML document that adheres to the following specifications:

1. Each element within the document is representative of a single interaction step.
2. All element names should have meaning within the application domain *except* for the root element, which must be a <db> tag.
3. All element names must be lowercase.
4. In order to facilitate multiple word elements, use an underscore (_) in place of a space.
5. The children and descendants of any element represent the possible interaction pathways from that point.
6. Since these specifications are somewhat limiting of the presentation of the information the following two attributes are supported:
   - category – used to present the genre that information at this level represents. An example would be would be a category value of 'Scholastic Level' for an element named 'graduate.' This information is also useful for specializing levelwise dialogs and supporting sophisticated dialog reflection capabilities.
   - display – used during presentation of this pathway, facilitating the usage of any allowable attribute value, including spaces and upper case letters. An example of this would be a display value of 'Graduate' for the previously mentioned graduate element.
7. The leaf node of the XML document will always be an absolute URL that links to the Web page presenting that information.

Fig. 4.8 provides an example content file within the application domain of Commonwealth of Virginia's Congressional Senators. Notice that the element tags use an underscore instead of a space for two word element names like 'senior seat' and 'junior seat.'

```
<?xml version="1.0"?>
<db>
  <virginia display ="Virginia" category="State">
    <senate display ="Senate" category="Branch of Congress">
      <republican display ="Republican" category="Party">
        <junior_seat display ="Junior Seat" category="Seat">
          http://www.vote-smart.org/bio.php?can_id=CNIP9093
        </junior_seat>
        <senior_seat display ="Senior Seat" category="Seat">
          http://www.vote-smart.org/bio.php?can_id=S0920103
        </senior_seat>
      </republican>
    </senate>
  </virginia>
</db>
```

Figure 4.8: An example XML content file showing the Commonwealth of Virginia's Congressional Senators.

Once this content file has been created, it should be made accessible to the WS://IM framework by placing it in the root Web accessible directory of the Web site with the name 'content.xml.' In the current example, this would mean that an XML document meeting the aforementioned specification would be made publicly available at the URL http://www.example.com/content.xml. At this point, the WS://IM framework will automatically detect and cache the content file (see content handler) when the first user with a WS://IM interaction interface accesses the Web site. Notice that the content provider does not have to register with the WS://IM framework; the framework will automatically detect if the Web site is compliant or not.

### URL Redirection

The astute reader might observe that, since it is the content provider's prerogative to allow the 'content.xml' file to reside in the defined location, how does one access the WS://IM version of the site? This is a valid question, and one that brings up the second requirement to be WS://IM compliant – a simple URL redirection. WS://IM requires a URL within the Web site's domain in which the 'machine name' identifier 'personalize' is mapped to the canonical name of pipe.cs.vt.edu. In technical terms, this implies adding a CNAME reference for personalize.example.com to pipe.cs.vt.edu in the administrative DNS server for the domain, a relatively trivial network administration operation. This is required of the content provider due to the dynamic and flexible nature of the WS://IM framework. Since it will automatically detect and register sites as they become 'compliant,' WS://IM must have some means for always calling back to itself, while still retaining the Web site's domain name information.

This method was chosen from a variety of mechanisms, ranging from those that involve the full participation of the Website, to proxy-based bypass schemes. The chosen solution has the attractive property that mixed-initiative interaction can be enabled at as fine or coarse a level of granularity as desired by the content provider. This gradient of granularity allows the content provider to determine at which point they would like the WS://IM framework to handle all, some, or none of the interaction management. The proxy-based approaches were far less configurable solutions and required a high degree of participation from the content

provider to avoid loss of functionality. Also in contrast to the other solutions, addition of a DNS name mapping to a defined location required the least amount of work from the content provider, an original design goal for both this component and the WS://IM framework as a whole.

### 4.3.6 Component Communication Protocols

Throughout the architecture description, component communication protocols have been hinted at; we now expose these communication protocols and the justification for their usage, in greater detail. The following section will be broken into subsections according to the connection points enumerated in Fig. 4.1.

#### Interaction Interface to Interaction Manager

Since the interaction interfaces reside on end user systems, the connection between them and the interaction manager will potentially pass through one or many high latency communication channels (e.g. dial-up). For this reason, size of the communication packets is critical to the success of the framework. Two lightweight solutions—REST and XML-RPC—were employed as the communication protocols of choice for this connection. Since REST is merely an architecture applied to the World Wide Web, its implementation within this communication was trivial. XML-RPC however was a different story, especially since the interaction interfaces were to communicate using browser interpreted JavaScript. To accomplish the XML-RPC connectivity, a third party library known as vcXML-RPC was employed to create the JavaScript client module as well as the interaction manager server module.

#### Interaction Manager to Cache Database

All communication between the interaction manager and the cache database is done through the standard SQL protocol. Messages are passed between the two using the SQL interface specification for connecting and SQL message structure for interfacing with the database. This connection was trivial to develop as the programming language that was used for the interaction manager had native support for connection and communication with most SQL databases, including MySQL.

#### Interaction Manager to Transformation Engine

In contrary to the interaction interface to interaction manager connection, the connection between the interaction manager and the transformation engine components will most likely only be separated by negligible latency communication channels (e.g. LAN connection, gigabit ethernet, localhost). Because of this property, the packet size for the information does not need to be at a minimum. Also since the information being transfered is naturally XML format which requires to stay intact, it was determined that the simple object access protocol (SOAP) would be the most desirable, and functional, solution for this connection. As was already mentioned, the transformation engine was revised to include a SOAP server automatically generated by the gSOAP toolkit. The interaction manager portion of the connection required some integration and extension of the third party NuSOAP library in order to get it to marshall and un-marshall the content files. This process however created a very interesting problem since the SOAP server of the transformation

attempted to bind to the network port 80, which the interaction manager was already bound to listening for HTTP requests. The SOAP server was hence modified to listen on network port 18083, allowing the WS://IM interaction manager to listen on port 80.

#### Interaction Manager to Content Provider

As previously discussed, the connection between the interaction manager and the content provider is a simple REST request to attain the new or updated content file. The utilization of REST was decided mainly because it requires no extra work from the content provider and must be implemented for the content provider to provide any manner of information delivery on the Web. Also bandwidth may be an issue for content providers and moving the entire content file is already a large task in and of itself; thus the extra overhead commonly associated with XML-RPC or SOAP would arguably do more harm than good.

## 4.4 Implementation Notes

The WS://IM framework was developed in a wide variety of languages including PHP, JavaScript, XSLT, and C. The framework currently works in the Apache Web server, but can operate in *any* Web server environment on almost any platform. As mentioned, MySQL server was utilized to implement the cache database component of the framework, but any database could be used with little to no modification of the current implementation. In the current instantiation, all of the components of the system coexist on the same computer system, but the system has been tested in a distributed manner across a local area network with only a negligible performance degradation to the end user.

## 4.5 The WS://IM Workflow

This section shall present an end to end workflow for the WS://IM framework. Within this scenario, the user, Joe, will be interacting with the Web site www.example.com, a simple hierarchical Web site that contains the information sought by Joe. The workflow for this interaction is illustrated in figure 4.9, but only shows a single interaction; multiple interactions repeat the same process with minor alteration of parameters specific to the interaction.

Joe begins by entering the address “www.example.com” into his Mozilla Web browser. At this point, www.example.com transparently redirects Joe’s Web browser to “personalize.example.com,” turning over the interaction management to the WS://IM framework. At this point (1), the request from Joe’s Web browser (interaction interface) is sent to the interaction manager component. The request is parsed in order to extract a session ID; if none is found, a unique session ID is generated, starting a session for Joe’s current interaction interface. At this point, a simple NOOP request is made to the transformation engine, ensuring connectivity and operation of the system. If the request can be completed, the interaction manager will redirect the user back to the root Web site, www.example.com with a ‘refer’ HTTP parameter indicating that the redirection came from the interaction manager.

The interaction manager delegates the request to its content handling sub component, which will first check

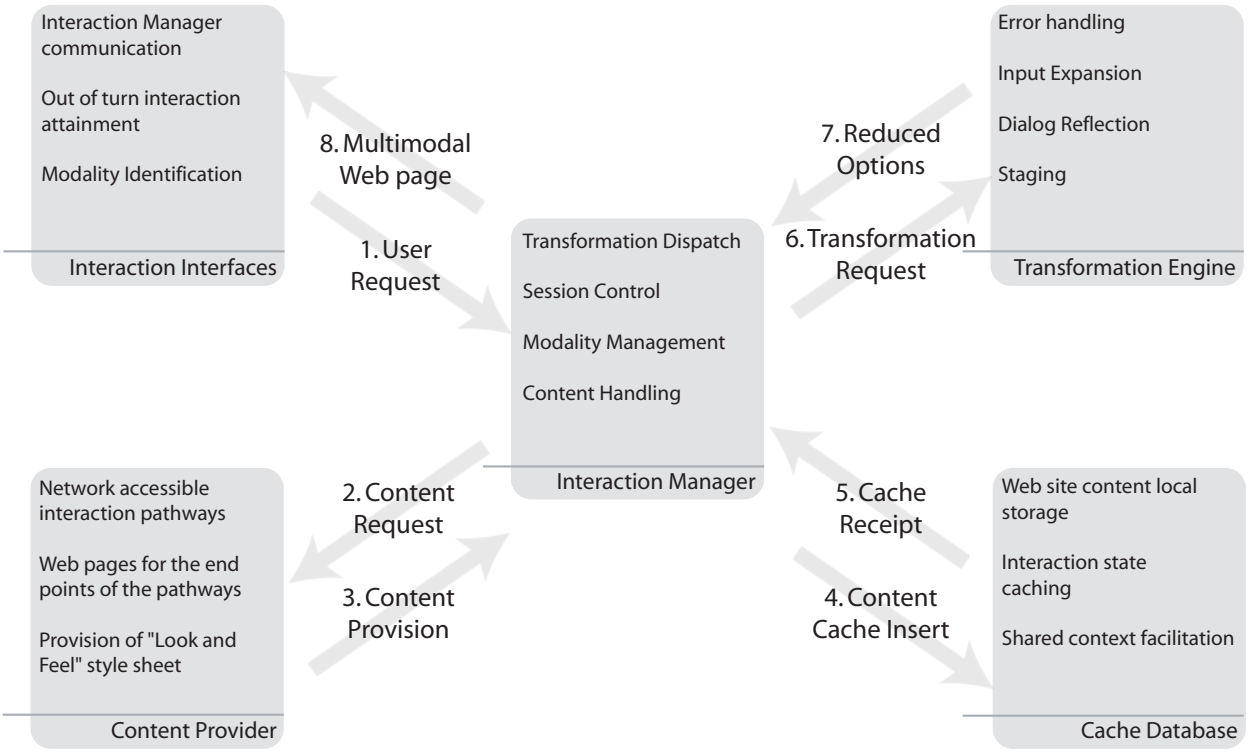


Figure 4.9: Workflow process of the WS://IM framework for a single interaction.

the cache database to see if there exists a stored content file for that Web site. For the sake of this scenario, it will be assumed that no such content file is stored in the cache database, therefore requiring the content handler to make a HTTP GET request for the content file from the content provider (2). If the file does not exist or is not a valid XML document, the interaction manager will immediately send a redirect to Joe's interaction interface in the same manner described above. In this scenario, though, the url www.example.com/content.xml returns a valid XML document to the content handler (3). The content handler then stores this file in the cache database for later use (4), reducing the delivery time of the content for future interactions (5), since communication with the cache database will be through either a local or low latency connection.

Once the content file has been attained, the content handler will provide it to the transformation dispatch subcomponent, triggering it to initiate a simple object access protocol connection with the transformation engine. Transformation dispatch is responsible for formulating a proper message to the transformation engine which will include the content file, or reduced content file if there has already been previous interactions between the user and WS://IM, explained later, and the user's request, which in this case would be an empty string since Joe has not identified anything as a request (6). Upon receipt of this request, the transformation engine will process the two pieces through a series of input expansion, dialog reflection, and staging. This transforms the provided content, reducing the available options for interaction, and returns this reduced content in the form of a SOAP message to the interaction manager. If the transformation engine identifies an error with the input with respect to the content, it will return a SOAP exception message to the interaction manager, which will propagate the message to the user in a manner appropriate to the interaction interface, but since Joe's request is empty and the content file is valid this situation does not happen (7).

The session control component of the interaction manager will cache both the returned content file and the request that was made, including the entire history of requests made by Joe to get to this point in the dialog. This is done in order to facilitate back button browsing as well as advanced context aware caching in which if another user makes a similar request, the aforementioned content handler only needs to return the reduced content. This advanced context aware caching has the added benefit of reducing overall system operation time, since it eliminates the request and processing steps associated with the transformation engine component.

At this point, the reduced interaction options, which are embodied in the transformed content file, are ready to be delivered to the user; however they still remain in a raw XML format. At this point, the modality management subcomponent will determine the best modalities and styles to apply to the content. This is done through analysis of the HTTP request headers set by Joe's interaction interface. Once the appropriate modalities and styles are identified, the modality management subcomponent will attempt to pull those stylesheets from the content provider's Web site; if a stylesheet for any modality or style cannot be found, the WS://IM default stylesheet for that missing modality or style will be employed. Each modality or style is applied in cascading fashion to the reduced interaction options producing a Web page that is specifically tailored to Joe's browser and its capabilities. This page is returned to Joe's browser (8), which is then responsible for appropriately interpreting and rendering the information. This concludes the workflow process description of the WS://IM framework for a single interaction. Joe's iterative interactions are handled in the exact same manner, with the provision of the request being something other then the empty string used in this example.

# Chapter 5

# Case Studies

In order to exercise the WS://IM framework, three Web sites, each from very different application domains, were chosen as case studies. These sites were selected on the basis of their hierarchical content structures, supporting rich dialog completion capabilities:

- Project Vote Smart – http://www.vote-smart.org/
- Virginia Tech Timetables – https://banweb.banner.vt.edu/
- CITIDEL Digital Library – http://www.citidel.org

To attain the content necessary from these Web sites, each one was crawled using a simple PHP script, except in the case of the CITIDEL Web site, where access to its back-end database was available. Since this means of information retrieval is prone to error, a human verifiable subset of the total information was used for these case studies (explained later).

For each of these sites, a pseudo Web site was made in the Internet domain smallbox.org. This was done for a variety of reasons, the greatest of which was to be able to accomplish the second requirement for WS://IM, a URL redirection, without affecting the actual Web site owners. Once these two requirements were fulfilled, the WS://IM framework showed its powerful nature, presenting the three Web sites in a multimodal, out-of-turn capable manner in a variety of interfaces, with no further work required.

## 5.1 Project Vote Smart

### 5.1.1 Overview

The Project Vote Smart web site, henceforth referred to as PVS, is dedicated to providing unbiased, accurate information about the elected members of and current candidates for the United States government. This Web site presents a range of information designed to assist voters in making educated electoral decisions. It purports itself as 'a national library of factual information, Project Vote Smart covers political candidates and elected officials in five basic categories: biographical information, issue positions, voting records, campaign finances and interest group ratings.'

A section of the PVS Web site is dedicated to providing information pertaining to the almost 450 currently elected officials of the United States Congress. This subset was the chosen for the case study since it was narrowly defined, well understood by users, and lent well to being structured hierarchically within an XML document. To this content, a hierarchical structure was extracted using the following as interaction keys: (i) state, (ii) branch of Congress, (iii) political party, and (iv) either district or seat depending on branch information. This well-formed XML document was then stored, as specified by the WS://IM requirements, at the URL http://www.votesmart.smallbox.org/content.xml, which made the PVS website accessible through WS://IM.

In order to make the WS://IM version of PVS appear like the real version, a style sheet for the desktop screen modality was developed which nearly replicated the PVS look and feel. This also exposes the capabilities of the WS://IM's modality manager component to apply a specific look and feel to a content provider's system.

We showwcase the PVS case study in the Extempore and WMLOOT interaction interfaces, utilizing desktop screen and cell phone screen modalities respectively. It should be noted however that all of the interaction interfaces and modalities are available to the Project Vote Smart Site by nature of the WS://IM framework.

### 5.1.2 Extempore Interaction Interface and Web Browser Modality

The Extempore interaction interface was the first interface designed to facilitate the novel technique of out-of-turn interaction [63]. Through this technique, users are capable of providing partial information in a manner *unsolicited* by the website. To present the capabilities of the WS://IM framework, the user goal of finding the Republican Senator of Montana (ref. previous chapter) is assumed.

Figure 5.1 presents the conventional method for determining such information. Notice that this task can be completed without the assistance of the Extempore toolbar; it is only done in this figure to show that the Extempore interface can accomplish in-turn interactions in the same manner as the browser. The interaction model for the pictured sequence could be expressed as Montana $\Rightarrow$ Senate $\Rightarrow$ Republican $\Rightarrow$ Senator Conrad Burns, where a $\Rightarrow$ indicates an interaction. This shows that the WS://IM can be used in the exact same manner as every other interaction management suite, providing conventional in-turn based dialogs through a desktop screen based modality.

However, WS://IM is not limited to in-turn interaction, as exemplified in Fig. 5.2. Notice that with the Extempore interface, located at the bottom of the screen, a user can provide their partial information even when such information is unsolicited. Also, the utilization of Extempore does not preclude future in-turn interaction as is shown in the second screen of Fig. 5.2. Through the utilization of Extempore, the interaction sequence can be reduced by one interaction: Republican Senator $\Rightarrow$ Montana $\Rightarrow$ Senator Conrad Burns.

### 5.1.3 WMLOOT and Cell Phone/PDA Modality

The WMLOOT interface provides the user with a similar means for providing partial input to WS://IM, but for a very different reason. The cell phone/PDA modality is characterized by extremely limited bandwidth, small available screen size, and a monochrome to limited color palette, a drastic difference from the aforementioned screen modality. To further complicate the setting, there is no easily definable dominant browser in this domain, thus creation of a toolbar interface like Extempore would bind the interface to a browser

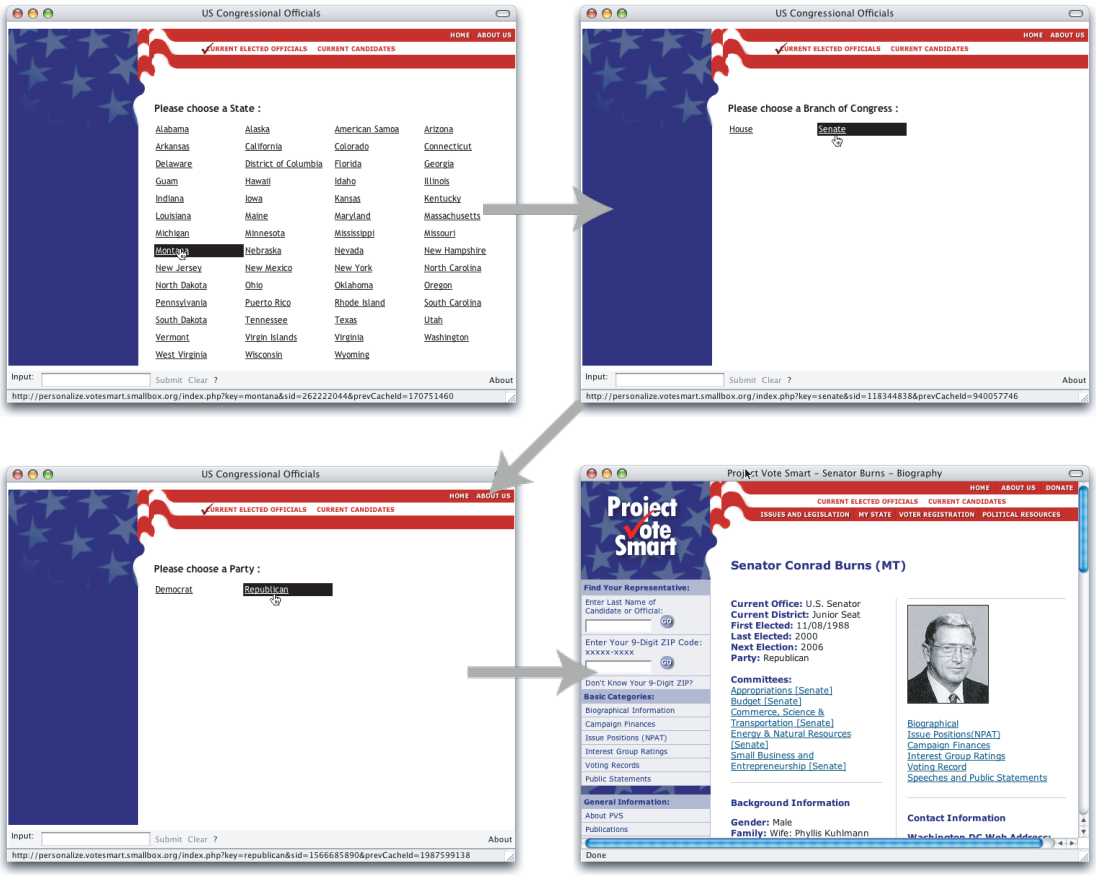


Figure 5.1: A conventional in-turn dialog with the PVS site using the desktop screen modality to reach the webpage of the Republican Senator from Montana.

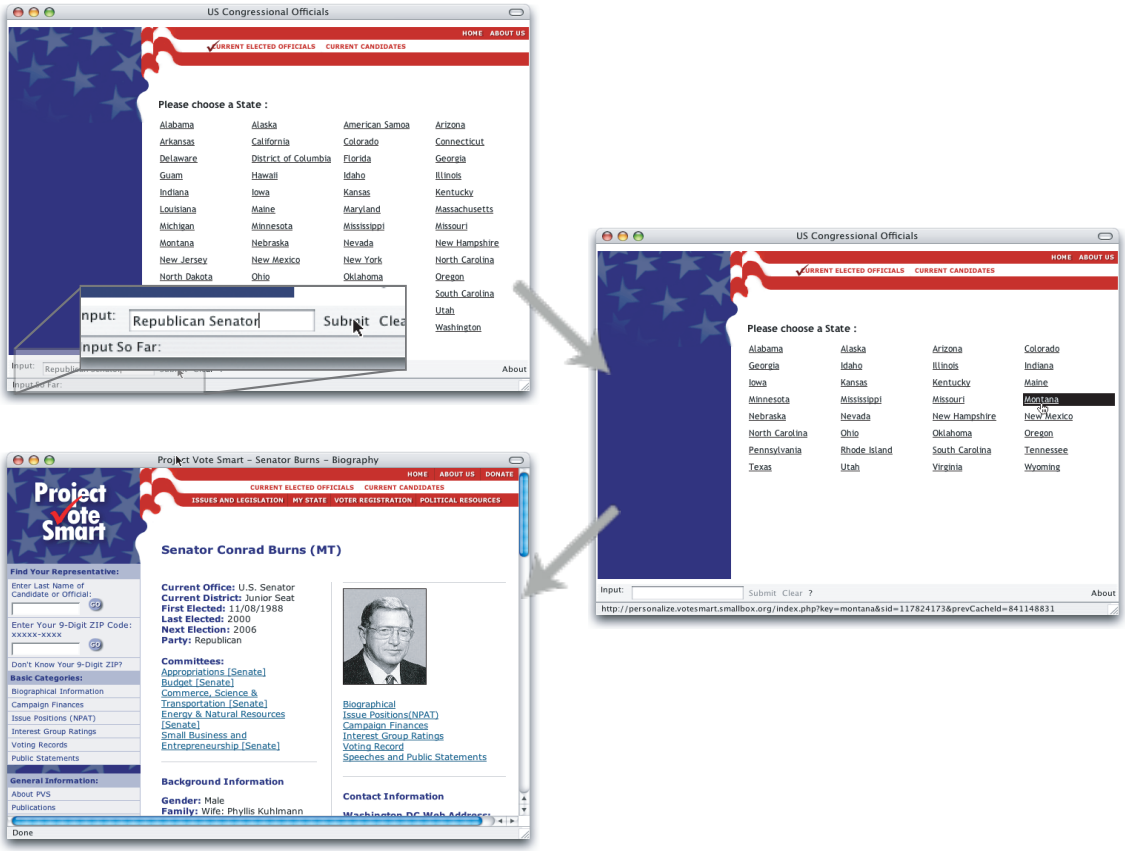


Figure 5.2: A mixed-initiative dialog with the PVS site as facilitated by the Extempore interaction interface.

that may not exist in the coming year. In order to create this interface, it was decided to actually send it embedded within the returned Web page, and yet kept distinct by being on a separate 'card[1].' This manner of interaction interfacing—embedding it into the resulting Web page—has the disadvantage of binding it directly to the modality but, in this case, the binding is present regardless since it is only presented to users operating in a cell phone/PDA modality.


Figure 5.3: The PVS site as presented in the cell phone/PDA modality on a Sony Ericsson T68i.

The presentation of the WS://IM managed PVS web site in this modality is provided in Fig. 5.3. One of the most notable issues with this modality is the strictness of the language that is employed for delivering content to it – Wireless Markup Language (WML). Implementations of the WML specification are extremely strict in comparison to modern HTML Web browser implementations, a trait that may seem too aggressive for most HTML developers. Two more key issues particular to this modality, as apparent in Fig. 5.3, are the reduced screen size and lack of an input device to assist the user with scrolling, such as a mouse or trackball. Due to these two traits, navigation through content becomes a hard task for the user. Due to the design of WML, however, since WMLOOT is always accessible from the top, a user can swiftly navigate the content by providing their partial information out-of-turn. The provision of the partial information, much like Extempore, will reduce the number of potential links applicable at a given point, making browsing a relatively simple task even in a constrained environment.

## 5.2 Virginia Tech Timetables

### 5.2.1 Overview

Virginia Tech provides its course timetables online so that students can make educated decisions about course selections for the coming semester. The information regarding courses is currently provided in a table-based manner that is arguably hard to navigate through. In order to improve usability the developers of this site have provide a search functionality at the top of every page, but the search functionality does not

[1]The term card is used here in the manner most closely associated with a hypercard deck. This is the native deployment format for the WML language.

cover all of the possible views of the information that a student may desire. The Virginia Tech Timetables, henceforth referred to as VTTT, was therefore selected as the next case study to utilize the WS://IM framework. Much like the previous case study, in fact using an extended version of the same script, the VTTT Web site was crawled for all of the available information pertaining to courses in eight departments:

- Accounting and Information Systems
- Business Information Technology
- Computer Science
- Electrical and Computer Engineering
- Industrial and Systems Engineering
- Mathematical Sciences
- Mathematics
- Statistics

From these eight departments, information about more than 630 courses was extracted and placed into a hierarchical XML file according to the following keys:

- Department
- Academic Level
- Course Type
- Instructor
- Days
- Time
- Credits
- Location

Similar to the PVS Web site, a pseudo Web site was established at the URL http://www.vttimetables.smallbox.org/ and the created content file was made available at the specified address. To prevent duplication, this case study is illustrated through two alternative presentation options that are available in the WS://IM framework instead of different modalities.

### 5.2.2 Table Interface

Table based interfaces represent a drastically different format and structuring mechanism for information. Issues such as navigation and browsing are drastically different modes of operation for a user in comparison to the normal hyperlink, turn based presentation format that was exemplified in the Extempore discussion for PVS and in Fig. 5.4 for VTTT. Finally, since the original presentation format for this information was in fact a table interface, it was important to verify that even while using the WS://IM framework, this look and feel could be retained.

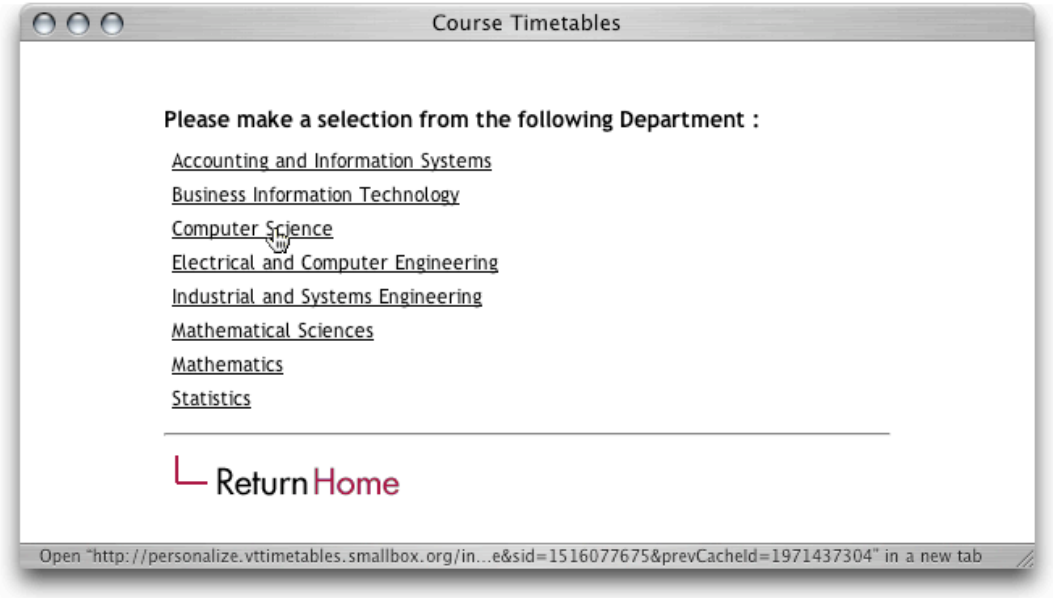


Figure 5.4: The Virginia Tech Timetables Web site as presented using the default, in-turn dialog format.

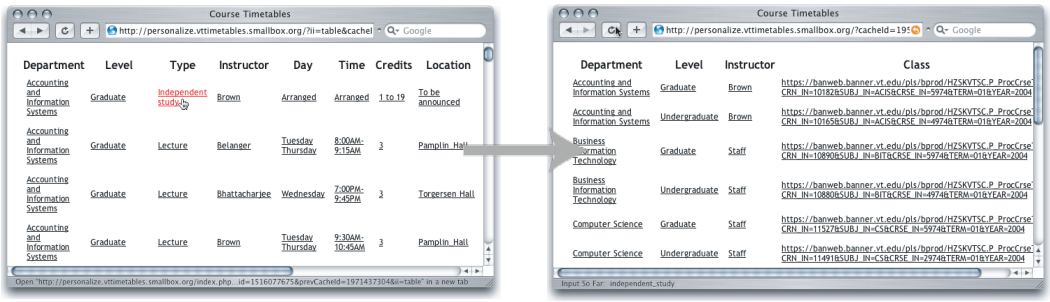


Figure 5.5: The Virginia Tech Timetables Web site as presented in a table format showing all of the information at once.

Since the modality manager utilizes an XSLT [3]-based style sheet, the development of a table based interface was a one time task, owing to the centralized methodology of WS://IM. The table based interface is presented in Fig. 5.5; notice how once a selection has been made, 'Independent Study' in this case, the entire table is reformed, pruning out all of the entries that are not 'Independent Study' course format.

### 5.2.3 Faceted Classification Interface

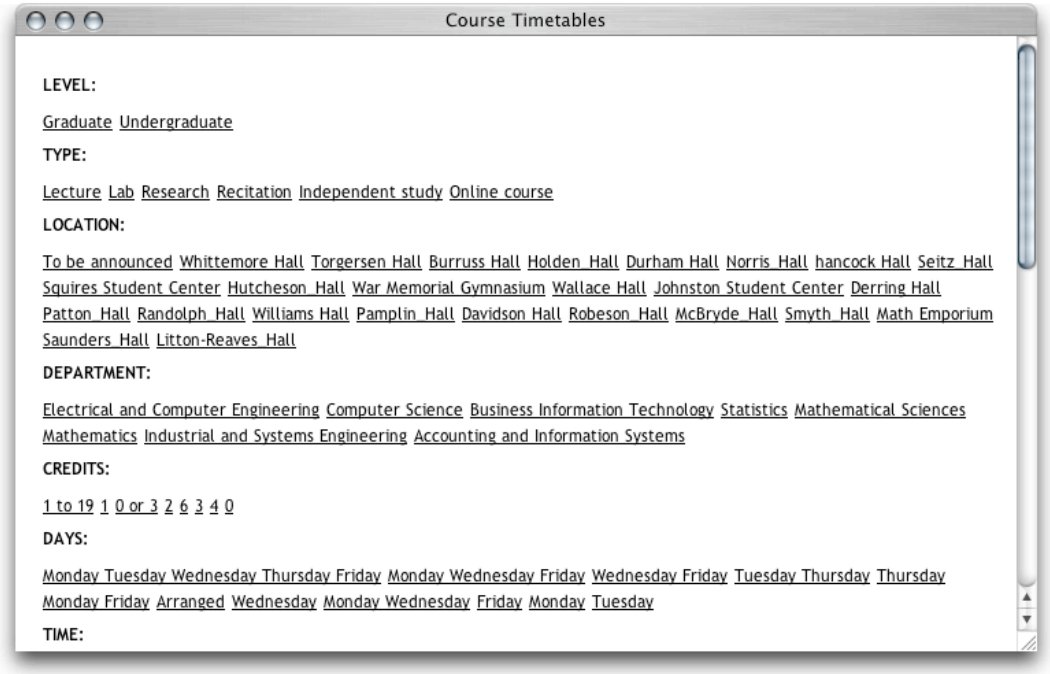


Figure 5.6: The Virginia Tech Timetables Web site rendered in a faceted classification interface.

In the same manner as the Table interface, faceted classifications also represent a very different structure and interaction manner for a Web site. Faceted classification systems present all of the possibilities for interaction in a clustered manner and mirror the levelwise nature of information hierarchies. Web sites employing faceted classification interfaces assist users by presenting them with all possible means for providing their partial information while still supporting traditional in-turn mechanisms. This interface is depicted in Fig. 5.6.

## 5.3 CITIDEL

### 5.3.1 Overview

At the time of this writing, the Computing and Information Technology Interactive Digital Educational Library (CITIDEL) is an information repository that houses articles, papers, graphics, etc. pertaining to the

broad topic of computing and information technology, which in itself has exploded in recent years. This information was classified using the following four different, but overlapping hierarchical structures, the first of which (CCS1998) was selected for this case study [60]:

- Association for Computing Machinery's (ACM) Computing Classification System 1998 (CCS1998)
- ACM/IEEE Computing Curricula 2001 (CC2001) System
- Computing Research Repository (CoRR) System
- AMS Mathematics Subject Classification 200 (MSC200)

CITIDEL is organized alongside a series of SQL tables which required a processing step to convert to an XML representation in order to be compliant with the WS://IM framework. The developed script to handle this conversion step could be executed upon a request for the URL http://personalize.citidel.smallbox.org/content.xml that would allow for the XML representation to always be a true reflection of the information. Support for this feature was not actually implemented in the CITIDEL pseudo site, but is presented here as a means to expose the potential for dynamic data retrieval by the content provider. It should be noted at this point that other options, such as Open Archive Initiative (OAI) [47, 67], for retrieving this information were investigated. Upon investigation, these options, in their current instantiation were decidedly not capable of producing an XML file with the hierarchical structure of the entire system embedded in its metadata. However since the OAI utilizes an XML based represnetation, it has been discussed that adaptations could be made to the developed script which would convert the OAI XML to a WS://IM compliant XML format.

### 5.3.2 SALTII – Speech and Hyperlink Hybrid Modality

The SALTII interface brings the modality of speech to bear on all of the Web sites managed by the WS://IM framework. As has already been mentioned, the hybrid or mixed modality presentation of the SALTII interface, providing both hyperlink and voice modalities, nearly embodies the 'concerted, natural user modality' [18]. SALTII is constructed upon the Microsoft SALT plug-in, which currently only work in Microsoft Internet Explorer on the Microsoft Windows platform. That being stated, the component which handles the SALTII interface was designed not just to look for those specific qualities, as future growth may extend it into the Macintosh or Smart Phone platforms, but to determine the presence of the SALT plug-in and embed the proper tags into the return format, making SALTII agnostic of the format itself.

Figure 5.7 presents CITIDEL as rendered with the SALTII interface. The only difference between the SALTII enhanced user interface and the site's non-SALTII interface is the addition of a 'Speak' button in the upper left corner of every page. This button allows the user to initiate a speech input session as indicated by the presence of a 'listen' box which graphs the speech input as it arrives. This input is then processed according to a WS://IM generated grammar file, defined from the remaining interaction options. If a valid input can be identified, a request for a new page is issued to WS://IM. Since SALTII defines its grammar for allowable inputs, the user can provide both in-turn *and* out-of-turn input through the speech modality. Also it should be noted that the in-turn interaction capability inherent with the non-SALTII portion of the interface is still retained, thus facilitating input through both speech and point-and-click modalities. Currently SALTII has been tested on many platforms and browsers (to use the speech modality, a microphone is required):

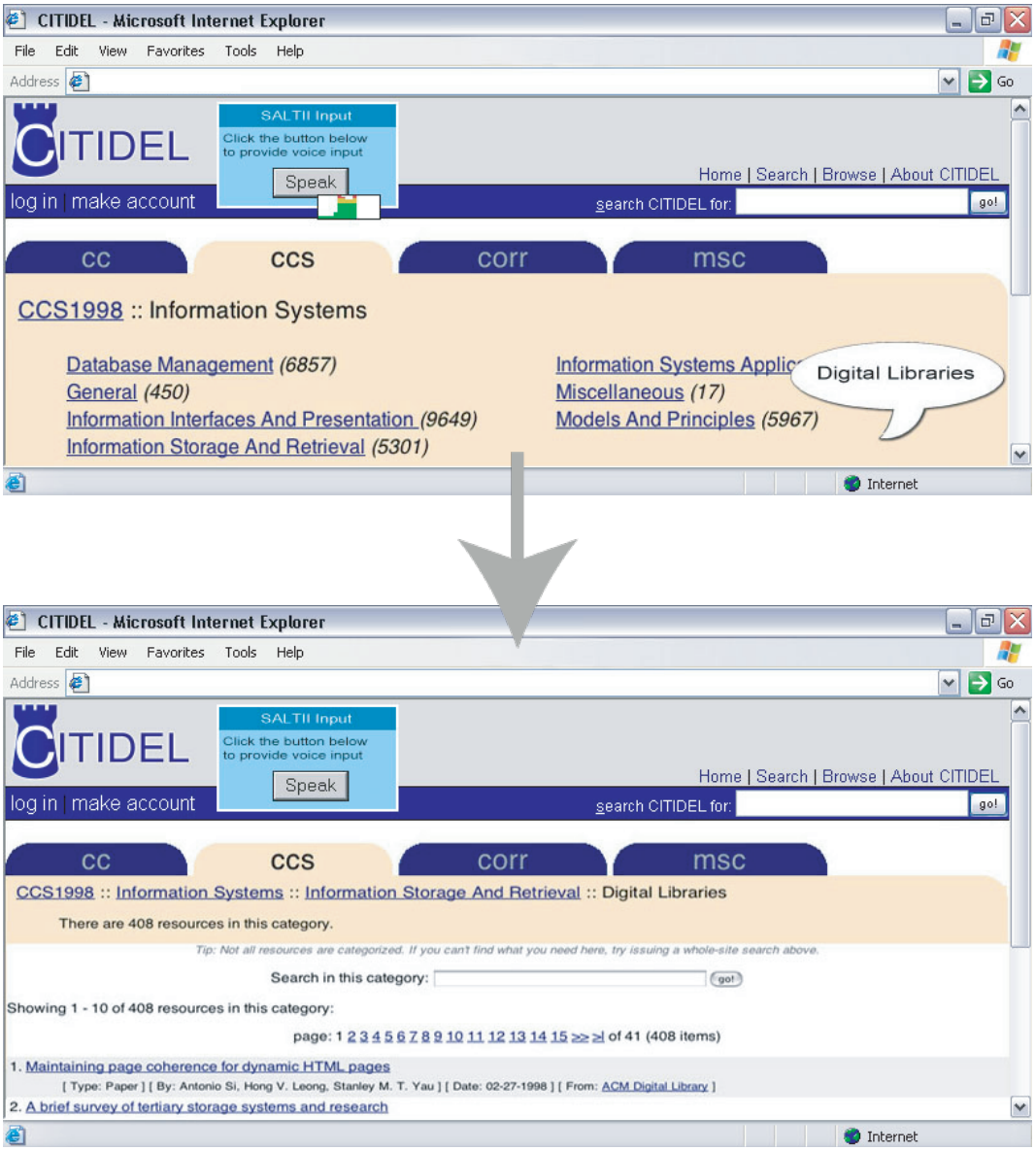


Figure 5.7: A example using the CITIDEL digital library with multimodal interaction as facilitated by the SALTII interface.

Microsoft Windows 2000, Windows XP, Windows Mobile (PocketPC 2002 and 2003), Internet Explorer 6.0, Internet Explorer 5.5, and Microsoft Pocket IE.

#### Continuous Speech Modality

The most natural facilitation of speech modality for the user would be simply begin speaking to the Web page unhindered with the requirement of pressing a 'speak' button to initiate the input session. The SALT plug-in does facilitate a means for continuous listening; however in the current implementation of SALTTI this feature is disabled. When this feature is enabled, SALTII would listen for and analyze all of the user's speech input to determine which elements can be used as valid input as determined by its grammar specification. Future work for the SALTII interface would include the enabling of this functionality, but is not suggested to be done until after the first full release of the SALT plug-in from Microsoft, which was at Beta 3 level at the time of this writing.

## 5.4 Evaluation

In order to compare the WS://IM framework with respect to the previously mentioned interaction managers, a series of metrics were defined. These metrics are focused on the two different classes of stakeholders: the content providers and the users. Due to the nature of most interaction managers, the content providers group must be extended to include designers, developers, and maintainers as well. The rest of this chapter shall evaluate the WS://IM framework in comparison to database driven, model driven, and domain specific language solutions to interaction management.

### 5.4.1 Content Provider Perspective Evaluation Metrics

The first set of metrics characterize the average requirements placed on a content provider if any interaction manager was utilized. The evaluation metrics defined in this section are not specific to a content provider as they are meant to include all human participants that are necessary to produce, develop, and maintain the Web site.

#### System Setup

The initial impression of an interaction manager is generally in the system setup phase. This is where a content provider gets a glimpse into the complexity of the interaction management solution adopted. This metric is broken down into the various stages of system setup that start from installation of the interaction manager and range to establishment of the actual interaction manager in working order.

- Installation Requirements

  The installation requirements metric takes into account the amount of time and external knowledge required to establish the correct environment for the interaction manager. This metric is designed to

evaluate all of the requirements placed on the content provider in order to operate or use the interaction manager.

The identified database driven, model driven, and domain specific language solutions require application download and installation, a step common for interaction managers. This step requires a varying degree of work, ranging from installation of language specific runtimes to specific SQL databases. Most of the systems require installation of a specific type of Web server or the presence of a specific library. As one might conjecture at this point, installation of these systems requires parsing through numerous instructions and specifications in order to establish the correct local environment for the system to operate. Such server configuration is necessary to be completed before any progress can even begin with the interaction manager.

The WS://IM framework, however, makes system setup and installation as simple as providing an XML file, which can be developed with applications ranging from simple text editors to specialized XML creation tools like SyncRO's <oXygen/> or Altova's XMLSPY. Therefore, no special software, libraries, or servers are required to use the WS://IM framework. This provides a drastic improvement in terms of usability and the overall system cost for the content provider to the other interaction managers. In terms of time and external knowledge required to establish the environment for the interaction manager, the WS://IM drastically outperforms the other interaction managers.

- Network Requirements

  To make the XML file available to the WS://IM architecture, a content provider will require a domain name and web accessible network space, both of which are required by any interaction manager; otherwise it cannot be easily accessed from the Web[2]. An interesting feature of WS://IM is that since the content provider no longer directly handles interactions with users, the bandwidth and network space requirements for the content provider are drastically reduced. This reduction is in contrast to the previously mentioned interaction managers which require space for not only the content and style, but also for the interaction manager and the libraries, languages, servers that it might require.

  The WS://IM framework, by centrally handling interactions, reduces the bandwidth and processor requirements for each content provider. Furthermore, since the WS://IM framework can handle a high degree of simultaneous Web sites (bound only by processor and bandwidth requirements), the cost for such a system can be apportioned across all participating Web sites, instead of each Web site paying more for the same amount of processing power and network space. This distributable cost system is a benefit of the architecture that was chosen for WS://IM, a design detail that is absent in any of the other interaction managers. Therefore, one could attribute the centralized design strategy for the WS://IM framework as the main contributor to its success in this metric.

- System Construction

  The system construction metric refers to the amount of time and external knowledge required to establish a working implementation of the interaction, withstanding the time required to install the application and all of its dependencies. Most of the aforementioned interaction managers have developed a direct manipulation or interactive graphical user interface to specifically address this issue. Such an interface is designed to assist the user in understanding the concepts that are being utilized in

[2]The use of static IP addresses instead of domain names has been suspended from this discussion as it decreases usability for the end user.

the interaction manager, while still constructing the interaction manager, structuring the content, and making presentation design issues.

In this metric, WS://IM may initially appear to fall short of the others, providing no graphical user interface with which the content provider can construct the system. However, WS://IM allows the content provider to use any proprietary text editor to produce their content in the specified format or use a script to automatically extract the information from a data repository, as was done in the CITIDEL case study. By using a standard data format in XML, WS://IM leverages applications that the content provider is currently familiar with, like the previously identified XML creation tools instead of requiring the content provider to learn a new suite of tools and technologies.

However, in the current implementation of the WS://IM framework, end points within the framework point to actual web pages maintained by the content provider. This is an identified issue in the current implementation and has been projected as future improvement for the WS://IM framework (see the Future Work section for more details). This however must be taken into account as part of the evaluation for the framework, which decreases the WS://IM framework's value in the system construction metric.

- Styling

  Since the WS://IM framework operates 'out-of-box' with default styles for *all* supported modalities, a content provider need not design, develop, and provide such styles until they are ready to do so. This is in direct contrast to other interaction managers that require definition, or at least modification, of HTML templates before the content can be made visible on the Web and only in one modality.

  Furthermore, due to the adoption of Eq. 2.4 by WS://IM, complete separation of content and style developments is facilitated. This in turn means that the content provider need not require anything from the style provider and vice versa. For more stylistic Web sites, it is common practice, and one retained but not enforced by the WS://IM framework, to define a standard naming and programming interface between the two parties.

**System Maintenance**

The system maintenance metric includes two different forms of maintenance that affects the interaction manager. First, content providers will frequently modify content to reflect changes that have occurred over time. The second form of maintenance is general system maintenance and improvements for the interaction manager system itself. This metric constitutes one of the most important desiderata of any interaction manager as it represents the bulk of work done at any Web site.

- Content Modifications

  Content modifications can be divided into two categories: minor and major. Minor modification include spelling, grammar, on nomenclature alterations. In this category, almost all of the systems fare equally well, requiring the content provider to simply modify their content storage (XML file, relational database row, etc.). Major modifications, such as structure alteration, section removals, section additions, provide a starkly different picture of the interaction managers. In this category, domain specific languages suffer drastically as they require re-generation of the interaction manager

to reflect the modification. Database driven and model driven sites also require a significant amount of re-design and re-implementation in order to handle such drastic alterations.

WS://IM, however, only requires the content file to be modified and it will dynamically adjust itself to handle the modification. Due to this, major modifications to the content are achieved using the exact same workflow process as minor modifications, making major system modifications drastically easy in comparison to the other systems. Because WS://IM handles style rendering, logic application, and modality management for the content provider, system maintenance becomes a rather trivial task.

- Interaction Manager Maintenance

  In the metric of interaction manager maintenance, WS://IM is unique. Since it features a centralized architecture, maintenance is handled in one place and applies immediately to all of the Web sites that utilize the framework. Furthermore, by being centralized, WS://IM maintenance can be handled by a single set of trained professionals, instead of each Web site requiring a set of professionals who are familiar with the employed interaction manager.

  Due to the distributed nature[3] of the aforementioned interaction managers, updates may rarely be applied or, worse still, by erroneous updates may cause the Web site to become unusable or unavailable. These issues are veritable nightmares for information technology professionals who are commonly in charge of system maintenance, especially since they may not be fully, or even partially, trained in the interaction management aspects.

  Throughout these metrics, WS://IM either performs at par or exceeds the database driven, model driven, and domain specific language solutions for interaction management. This indicates that, from a content provider perspective, utilization of the WS://IM framework would reduce the overall cost, in terms of time, monetary, and required knowledge, for providing content on the Web.

### 5.4.2 User Perspective Evaluation Metrics

The other human side, opposite of the content provider, for any interaction manager would be the user. The following is a series of metrics to evaluate an interaction manager from the users' perspective. They were defined by analyzing the average web interactive techniques as well as the goals that are implied by those techniques.

#### Interactive Navigation

Interactive navigation is one of the key aspects that defines the Web. Users and web sites conduct dialogs in order to direct a user to some end goal. Facilitating this in some usable manner is the ideal goal of any Web site, and in turn the goal of any interaction manager. All of the identified interaction managers provide an in-turn navigation scheme where the partial information is to be supplied by the user in a rather restricted format.

WS://IM is the only interaction manager to take the next step and allow the user to interject and control the direction of the Web site, even if it is not designed in this manner. This allows users to provide their

[3]The word 'distributed' here refers to each Web site being in possession of its own, individual copy of the interaction manager application.

partial information at their own convenience and in their own vernacular. Preliminary usability analysis of this out-of-turn interaction functionality has proven very positive [61].

### Modality

As mentioned, the rise of ubiquitous computing indicates the increasing need to present information in a multimodal manner. A few of the identified interaction managers, <bigwig>, AUIT, and Autoweb, offer the ability, or planned capability, to present content in both HTML and WML formats, but none of them extend beyond the screen based interface. The WS://IM framework, in its current implementation, takes this logical next step and can present and interact with users through speech modality. To further extend this multimodal functionality, future work for the WS://IM framework includes the ability to present content in a 2D interactive graphics format (SVG), a 3D virtual environment manner (X3D), and highly interactive speech modalities (Semantics Synchronous Understanding (SSU) [71]).

### Dialog Management

Provision of information in multiple modalities facilitates an interaction manager to conduct dialogs with users instead of turn based interaction sequences. This dynamic communication provides a natural interface for information retrieval by the user, but brings an added complexity to deciphering user input for the interaction manager [51, 66]. Following from the previous evaluation criterion (modality), the only interaction manager which is capable of a spoken language interface would be the WS://IM framework with its speech modality. Furthermore, WS://IM, through its out-of-turn interaction capabilities, implements a means for conducting dialogs between the user and the website.

### Accessibility

In 1998, the United States Congress enacted the Rehabilitation Act that include a accessibility clause that pertains directly to information technology. This clause, Section 508, was included to ensure that people with disabilities have access to public information equal to everyone else. Attributed mainly to the date created, none of the identified interaction managers directly address the Section 508 accessibility issue.

Towards Section 508 compliance, WS://IM provides all content in well-structured xHTML in order to provide meaningful metadata for screen readers directly in the content instead of through visual cues. Furthermore, as previously mentioned, WS://IM provides multimodal interaction for all of its users, meaning that people with disabilities can directly interact with a WS://IM managed Web site using the modality that best suits them, be it visual, voice, or tactile.

# Chapter 6

# Conclusions and Future Work

## 6.1 Conclusion

The main contribution of this thesis is the validation of my original hypothesis – rich and contextual multimodal Web interactions can be supported through a factorized software framework using distributed components and centralized management of interaction. Further, the enumerated software framework, embodied in WS://IM, handles multimodal web interactions in a manner that reduces the requirements of the content provider to a minimum. The provision of an entire abstraction layer to the interaction management paradigm removes the content provider from having to perform any web development. In this framework, a means for providing unique and different presentation styles per modality per content system has also been presented. Similar to the previously mentioned abstraction layer, handling of styling is done in a manner that does not require the style provider to interface with the rest of the system. With its provision of default stylings for all modalities, WS://IM allows style providers to only develop presentation styles that are required to be distinct, reducing the number of styles and modalities required for development to minimum.

This software framework further validates the staging transformation approach to interaction that is employed at the core of the WS://IM architecture. It provides a toolkit for future development of interaction interfaces in any modality that can leverage the mixed-initiative interactions presented in this document. In this sense, the research that has been presented in this document could be viewed as a web accessible interface to the staging transformations modeling methodology.

The three case studies expose the ability of the WS://IM framework to simultaneously manage different content systems in numerous different presentation and modality formats. The benefits of strict adherence to the four entity, factorized equation of the Web (Equation 2.4) have been shown through the flexibility of the architecture employed, the capabilities of the WS://IM system, and the extensibility of the framework to facilitate future growth. Finally, an interaction management framework has been presented that encourages the utilization of personalization techniques such as out-of-turn interaction for any content system.

## 6.2 Future Work

This work has provided a flexible framework capable of extension to incorporate rising technologies. Future directions for this work include:

1. **Enhancing user-site dialog functionality:**
   The dialogs currently supported by WS://IM are primarily of the drill-down nature that utilizes a fixed-length, hierarchical structure. We say that these dialogs are *destructive* in nature, since dialog options are progressively narrowed. An interesting direction of future work is to relax this restriction to support *constructive* dialogs, where the user can decompose a complex information-finding task into a series of smaller subtasks, each of which involves simple mixing of initiative. Such dialogs will become important with the rise of 'social networking frameworks' such as the Friend-of-a-Friend (FOAF) specification [20].

2. **Context creation and reuse**
   An important element of realistic dialogs is the ability to address and reuse context from earlier interactions. Shopping carts at electronic commerce sites are a simple example of context creation and use, but more sophisticated forms of context [25] could potentially be employed in user-site dialogs. A second direction of future work would be to store WS://IM interactions as an addressable artifact, that can be retrieved at a later date and possibly utilized in a different interaction.

3. **Domain specific transformation engine component**
   The architecture of WS://IM was designed in a manner that the components could be replaced with different components so long as they implement the interface in the specified language. Future work could lead to development of domain specific transformation engines that implement that interface and replace the current Web domain transformation engine. Examples of possible domain specific transformation engines include the following two examples.

   - Map coordinate processing engine that could dynamically determine localized information when provided with geographical coordinate references.
   - Semantic Web processing engine which could parse rich site summary (RSS) news feeds to identify current information relating to some user provided topic.

4. **Bi-directional mixed initiative interactions**
   This work has presented a means for facilitating mixed initiative interaction, allowing a user to supply their partial information to the system. The current instantiation of the interaction, however, requires the computer to wait until the user is finished providing all of their input, meaning that users do not receive any feedback from the system until after they have finished. A bi-directional mixed initiative interaction scheme would allow the system to begin processing the input *while* the user is still providing it, making the dialog highly interactive. Modern technologies such as Semantics Synchronous Understanding (SSU) [71] already facilitate this level of interactivity; SSU would merely need to be added as an interaction interface to the WS://IM framework using the publicly available JavaScript toolkit.

Web based multimodal interfaces and the growth of ubiquitous computing devices represent the rise of the next generation World Wide Web. The WS://IM framework provides a solid base for content providers

to present their information using all modalities without having to continually redevelop their presentation system to facilitate new modalities and presentation technologies.

# Bibliography


[1] Cascading Style Sheets, level 1, December 1996. URL: http://www.w3.org/TR/CSS1.

[2] XUL Planet, 1998. URL: http://www.xulplanet.com.

[3] XSL Transformations (XSLT) Version 1.0, November 1999. URL: http://www.w3.org/TR/xslt.

[4] gSOAP: SOAP C++ Web Services, 2000. URL: http://www.cs.fsu.edu/∼engelen/soap.html.

[5] Speech Application Language Tags (SALT) Specification. Technical report, SALT Forum, July 2002. Version 1.0.

[6] SOAP Version 1.2 Part 1: Messaging Framework, June 2003. URL: http://www.w3.org/TR/2003/REC-soap12-part1-20030624/.

[7] K. Aberer, A. Datta, Z. Despotovic, and A. Wombacher. Separating Business Process from User Interaction in Web-Based Information Commerce. *Electronic Commerce Research*, 3(1–2):83–111, 2003.

[8] S. Abiteboul, P. Buneman, and D. Suciu. *Data on the Web: From Relations to Semistructured Data and XML*. Morgan Kaufmann, 2000.

[9] M. Abrams, C. Phanouriou, A. Batongbacal, S. Williams, and J. Shuster. UIML: An Appliance-Independent XML User Interface Language. *Computer Networks*, 31(11–16):1695–1708, May 1999.

[10] J.F. Allen, D.K. Byron, M. Dzikovska, G. Ferguson, L. Galescu, and A. Stent. Towards Conversational Human-Computer Interaction. *AI Magazine*, 22(4):27–37, Winter 2001.

[11] R. Andrew and D. McLellan. Dreamweaver task force. http://webstandards.org/act/campaign/dwtf/.

[12] D. L. Atkins, T. Ball, G. Bruns, and K. C. Cox. Mawl: A Domain-Specific Language for Form-Based Services. *IEEE Transactions on Software Engineering*, 25(3):334–346, May–June 1999.

[13] P. Atzeni, G. Mecca, and P. Merialdo. Design and Maintenance of Data-Intensive Web Sites. In *Proceedings of the 6th International Conference on Extending Database Technology*, pages 436–450. Springer-Verlag, 1998.

[14] B. W. Benson, Jr. Web-based applications you can live with. *SIGPLAN Notices*, 35(3):21–24, 2000.

[15] H. Berghel. Who Won the Mosaic War? *Communications of the ACM*, 41(10):13–16, 1998.

[16] T. Berners-Lee, R. Cailliau, A. Luotonen, H. F. Nielsen, and A. Secret. The world-wide web. *Communications of the ACM*, 37(8):76–82, 1994.

[17] M. Bochicchio and R. Paiano. Prototyping Web Applications. In *Proceedings of the 2000 ACM Symposium on Applied Computing*, pages 978–983. ACM Press, 2000.

[18] R. A. Bolt. Put-That-There: Voice and Gesture at the Graphics Interface. In *Proceedings of the 7th Annual Conference on Computer Graphics and Interactive Techniques*, pages 262–270. ACM Press, 1980.

[19] C. Brabrand, A. Møller, and M. I. Schwartzbach. The `<bigwig>` Project. *ACM Transactions on Internet Technology*, 2(2):79–114, 2002.

[20] D. Brickley and L. Miller. FOAF Vocabulary Specification, May 2004. URL: http://xmlns.com/foaf/0.1/.

[21] P. Brusilovsky. Methods and Techniques of Adaptive Hypermedia. *User Modeling and User-Adapted Interaction*, 6(2–3):87–129, 1996.

[22] P. Brusilovsky. Adaptive Hypermedia. *User Modeling and User-Adapted Interaction*, 11(1–2):87–110, 2001.

[23] P. Brusilovsky and M. T. Maybury. From Adaptive Hypermedia to the Adaptive Web. *Communications of the ACM*, 45(5):30–33, May 2002.

[24] R. Capra, M. Narayan, S. Perugini, N. Ramakrishnan, and M. A. Pérez-Quiñones. The Staging Transformation Approach to Mixing Initiative. In G. Tecuci, editor, *Working Notes of the IJCAI 2003 Workshop on Mixed-Initiative Intelligent Systems*, pages 23–29. AAAI/MIT Press, 2003.

[25] R. Capra, M. A. Pérez-Quiñones, and N. Ramakrishnan. WebContext: Remote Access to Shared Context. In *Proceedings of the 2001 Workshop on Perceptive User Interfaces*, 2001.

[26] D. A. Chappell and T. Jewell. *Java Web Services*. O'Reilly & Associates, Inc., 2002.

[27] P. P. Chen. The Entity-Relationship Model – Toward a Unified View of Data. *ACM Transactions on Database Systems*, 1(1):9–36, 1976.

[28] J. Domingue and M. Dzbor. Magpie: Supporting Browsing and Navigation on the Semantic Web. In *Proceedings of the 9th International Conference on Intelligent User Interfaces*, pages 191–197. ACM Press, 2004.

[29] M. Fernandez, D. Florescu, J. Kang, A. Levy, and D. Suciu. STRUDEL: a Web Site Management System. In *Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data*, pages 549–552. ACM Press, 1997.

[30] M Fernandez, D. Florescu, J. Kang, A. Levy, and D. Suciu. Catching the Boat with Strudel: Experiences with a Web-Site Management System. In *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, pages 414–425. ACM Press, 1998.

[31] M. Fernandez, D. Suciu, and I. Tatarinov. Declarative Specification of Data-Intensive Web Sites. In *Proceedings of the 2nd Conference on Domain-Specific Languages*, pages 135–148. ACM Press, 1999.

[32] R. Fielding, U.C. Irving, J. Gettys, J. Mogul, H. Frystyk, and T. Berners-Lee. RFC 2068: Hypertext Transfer Protocol, January 1997. ftp://ftp.isi.edu/in-notes/rfc2068.txt.

[33] P. Fraternali and P. Paolini. Tools and Approaches for Developing Data-Intensive Web Applications: a Survey. *ACM Computing Surveys*, 31(3):227–263, 1999.

[34] P. Fraternali and P. Paolini. Model-Driven Development of Web Applications: the AutoWeb System. *ACM Transactions on Information Systems*, 18(4):323–382, 2000.

[35] F. Garzotto, P. Paolini, and D. Schwabe. HDM – a Model-Based Approach to Hypertext Application Design. *ACM Transactions on Information Systems*, 11(1):1–26, 1993.

[36] A. Goldberg and D. Robson. *Smalltalk-80: The Language and Its Implementation*. Addison-Wesley, Reading, Massachusetts, USA, 1983.

[37] M. A. Gonçalves and E. A. Fox. 5SL: a Language for Declarative Specification and Generation of Digital Libraries. In *Proceedings of the 2nd ACM/IEEE-CS Joint Conference on Digital Libraries*, pages 263–272. ACM Press, 2002.

[38] M. A. Gonçalves, E. A. Fox, L. T. Watson, and N. A. Kipp. Streams, Structures, Spaces, Scenarios, Societies (5S): A Formal Model for Digital Libraries. *ACM Transactions on Information Systems*, 22(2):270–312, April 2004.

[39] The PHP Group. PHP: PHP Usage Stats, 2004. http://www.php.net/usage.php.

[40] J. Grundy and B. Yang. An Environment for Developing Adaptive, Multi-device User Interfaces. In *Proceedings of the Fourth Australian User Interface Conference on User Interfaces*, pages 47–56. Australian Computer Society, Inc., 2003.

[41] J. Grundy and W. Zou. An Architecture for Building Multi-device Thin-Client Web User Interfaces. In *Proceedings of the 14th International Conference on Advanced Information Systems Engineering*, pages 728–732. Springer-Verlag, 2002.

[42] J. Hightower and G. Borriello. Location Systems for Ubiquitous Computing. *IEEE Computer*, Vol. 34(8):pages 57–66, August 2001.

[43] N. D. Jones. An Introduction to Partial Evaluation. *ACM Computing Surveys*, 28(3):480–503, 1996.

[44] L. E. Julia and A. J Cheyer. Speech: A Privileged Modality. In *Proceedings of Eurospeech '97*, pages 1843–1846, Rhodes, Greece, 1997.

[45] H. Kautz, B. Selman, and M. Shah. Referral Web: Combining Social Networks and Collaborative Filtering. *Communications of the ACM*, Vol. 40(3):pages 63–65, March 1997.

[46] S. Kojarski and D. H. Lorenz. Domain Driven Web Development with WebJinn. In *Companion of the 18th Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications*, pages 53–65. ACM Press, 2003.

[47] C. Lagoze and H. Van de Sompel. The Open Archives Initiative: Building a Low-Barrier Interoperability Framework. In *Proceedings of the 1st ACM/IEEE-CS Joint Conference on Digital Libraries*, pages 54–62. ACM Press, 2001.

[48] H. W. Lie and B. Bos. *Cascading Style Sheets: Designing for the Web*. Addison-Wesley Longman Publishing Co., Inc., 1997.

[49] H. W. Lie and J. Saarela. Multipurpose Web publishing using HTML, XML, and CSS. *Communications of the ACM*, 42(10):95–101, 1999.

[50] S. McGlashan, D. Burnett, P. Danielsen, J. Ferrans, A. Hunt, G. Karam, D. Ladd, B. Lucas, B. Porter, K. Rehor, and S. Tryphonas. Voice eXtensible Markup Language: VoiceXML. Technical report, VoiceXML Forum, October 2001. Version 2.0.

[51] M. McTear. Spoken Dialogue Technology: Enabling the Conversational User Interface. *ACM Computing Surveys*, 34(1):90–169, March 2002.

[52] G. Mecca, P. Atzeni, A. Masci, G. Sindoni, and P. Merialdo. The Araneus Web-Base Management System. In *Proceedings of the 1998 ACM SIGMOD international conference on Management of data*, pages 544–546. ACM Press, 1998.

[53] E. A. Meyer. *Cascading Style Sheets: The Definitive Guide*. O'Reilly & Associates, Inc., 101 Morris Street, Seebastopol, CA 95472, 1st edition, 2000.

[54] B. Mobasher, R. Cooley, and J. Srivastava. Automatic Personalization Based on Web Usage Mining. *Communications of the ACM*, 43(8):142–151, August 2000.

[55] M. Narayan, C. Williams, S. Perugini, and N. Ramakrishnan. Staging Transformations for Multimodal Web Interaction Management. In *Proceedings of the ACM International World Wide Web Conference (WWW'04)*, New York, NY, May 2004. ACM Press.

[56] M. Nelte and E. Saul. Cookies: Weaving the Web into a State. *Crossroads*, 7(1):10–13, 2000.

[57] J. K. Ousterhout. Scripting: Higher-Level Programming for the 21st Century. *IEEE Computer*, 31(3):23–30, 1998.

[58] S. Pemberton, D. Austin, J. Axelsson, T. Celik, D. Dominiak, H. Elenbaas, B. Epperson, M. Ishikawa, S. Matsui, S. McCarron, A. Navarro, S. Peruvemba, R. Relyea, S. Schnitzenbaumer, and P. Stark. XHTML 1.0 The Extensible HyperText Markup Language (Second Edition), January 2000. URL: http://www.w3.org/TR/xhtml1/.

[59] S. Perugini. *Program Transformations for Information Personalization*. Ph.D. dissertation, Department of Computer Science, Virginia Tech, 2004.

[60] S. Perugini, K. McDevitt, R. Richardson, M. A. Pérez-Quiñones, R. Shen, N. Ramakrishnan, C. Williams, and E. A. Fox. Enhancing Usability in CITIDEL: Multimodal, Multilingual, and Interactive Visualization Interfaces. In *Proceedings of the Fourth ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL'04)*, Tuscon, AZ, July 2004. ACM Press.

[61] S. Perugini, M. E. Pinney, N. Ramakrishnan, M. A. Pérez-Quiñones, and M. B. Rosson. Taking the Initiative with Extempore: Exploring Out-of-Turn Interactions with Websites. Technical Report cs.HC/0312016, Computing Research Repository (CoRR), 2003.

[62] S. Perugini and N. Ramakrishnan. Personalizing Interactions with Information Systems. In M. V. Zelkowitz, editor, *Advances in Computers*, volume 57: Information Repositories, pages 323–382. Academic Press, September 2003.

[63] S. Perugini and N. Ramakrishnan. Personalizing Web Sites with Mixed-Initiative Interaction. *IEEE IT Professional*, Vol. 5(2):pages 9–15, March–April 2003.

[64] C. Phanouriou. *UIML: A Device-Independent User Interface Markup Language*. Ph.D. dissertation, Department of Computer Science, Virginia Tech, 2002.

[65] Microsoft PressPass. Mobile Industry Leaders Announce an Initiative To Create a Mobile Top-Level Domain. Technical report, Microsoft Corporation, 2004. http://www.microsoft.com/presspass/press/2004/mar04/03-10LeadersAnnounceTLDPR.asp.

[66] A. Shenoy. A Software Framework for Out-of-Turn Interaction in a Multimodal Web Interface. Master's thesis, Virginia Polytechnic Institute and State University, Blacksburg, Virginia, June 2003. http://scholar.lib.vt.edu/theses/available/etd-06172003-210140/unrestricted/thesis.pdf.

[67] H. Suleman. Introduction to the Open Archives Initiative Protocol for Metadata Harvesting. In *Proceedings of the 2nd ACM/IEEE-CS Joint Conference on Digital Libraries*, page 414, 2002.

[68] M. McCahill T. Berners-Lee, L. Masinter. RFC 1738: Uniform Resource Locators (URL). http://www.w3.org/Addressing/rfc1738.txt.

[69] S. Trewin, G. Zimmermann, and G. Vanderheiden. Abstract User Interface Representations: How Well Do They Support Universal Access? In *Proceedings of the 2003 Conference on Universal Usability*, pages 77–84. ACM Press, 2003.

[70] J. Veen. *The Arts & Science of Web Design*. New Riders Publishing, Indianapolis, Indiana, 1st edition, 2001.

[71] K. Wang. Semantics Synchronous Understanding for Robust Spoken Language Applications. In *Proceedings of the UIST Conference*, Redmond, WA 98052, USA, November 2003. Speech Technology Group, Microsoft Research.

[72] J. Zeldman. 99.9% of Websites Are Obsolete. *Digital Web Magazine*, September 2002. online publication.

[73] J. Zeldman. *Designing with Web Standards*. New Riders Publishing, Indianapolis, 1st edition, May 2003.

# Vita

Christopher Stephen Williams was born on March 6, 1981 to Timothy and Rebecca Williams. He has been a Hokie from birth as he spent most of his crawling and toddler years in McBryde Hall, waiting for his parents to finish their undergraduate degrees. At the age of 13, Christopher attained the coveted Boy Scouts of America's Eagle Scout award. In 1997, he began working for Cryptek Secure Communications, LLC as a Web Master. He continued to work for Cryptek throughout his high school, undergraduate, and graduate career, accepting permanent full-time employment status with Cryptek in 2001. Following in the footsteps of his parents, Christopher attended Virginia Tech for his undergraduate degree in Computer Science. With the encouragement and support of his family, friends, and employer, Christopher continued his academic career at Virginia Tech for his Masters of Science degree in Computer Science. He looks forward to continuing his full-time position with Cryptek in both the Marketing and Software Engineering disciplines.