The Just-In-Time Hypermedia Engine

Zong Chen[1], Li Zhang[2]
[1](School of Computer Sciences and Engineering, Fairleigh Dickinson University, USA)
[2](Computer Science Department, New Jersey Institute of Technology, USA)

ABSTRACT: *Many analytical or computational applications create documents and display screens in response to user queries "dynamically" or in "real time". Hypermedia features must be generated "just in time" – automatically and dynamically. This paper presents a just-in-time hypermedia engine to provide hypermedia support of virtual documents.*

Keywords *- Just-in-time, Hypermedia, Virtual document*

1. INTRODUCTION

Many analytical applications, especially legacy systems, create documents and display screens in response to user queries "dynamically" or in "real time". These documents and displays do not exist in advance, and thus hypermedia must be generated "just in time"—automatically and dynamically.

A Just-In-Time Framework has been proposed [1]. To implement and evaluate the just-in-time hypermedia framework, the JIT hypermedia engine (JHE) needs to be designed. The Just-in-time Hypermedia Engine (JHE) executes as a middleware between an application and its user interface, providing additional hypermedia navigational, structural and annotation functionality, with minimal modification to the application.

JHE should have the following functions to supplement hypermedia functionalities for the "just-in-time" environment:

(1) JHE assigns unique, persistent identifiers for documents, elements and anchors in the documents;

(2) JHE stores links, comments, etc., in an external link base;

(3) JHE does the regeneration by intercepting the messages between the user interface and the applications.

(4) JHE relocates and re-identifies the anchors in the documents efficiently.

This paper proposed a Just-in-time Hypermedia Engine design based on the Dynamic Hypermedia Engine (DHE) [2]. DHE is a hypermedia system that can supply hypermedia services for analytical applications by intercepting documents between application computation and the user interface. DHE can automatically generate links based on the application's underlying relationships. But DHE currently does not support re-identification, relocation or regeneration. The "Just-in-time Hypermedia Engine" (JHE) is an extension to DHE. It can supplement analytical applications with hypermedia functionality for virtual documents.

2. DYNAMIC HYPERMEDIA ENGINE

2.1 Analytical Applications

Analytical or computational applications logically can be split into two parts: the computational part that performs calculations and generates screens and documents for display; and the user interface — the users' viewer or browser that displays the screen or document [3]. Microcosm's Universal Viewer and Freckles seamlessly integrate with applications but provide only manual linking. OO-Navigator provides a seamless hypermedia support for computational systems that execute within a single Smalltalk environment. The Dynamic Hypermedia Engine (DHE) [2] supplies dynamic linking that applies to any Web-based service (including, but not exclusively, "Web services"). The Just-in-time Hypermedia Engine (JHE) extends the concepts in DHE for just-in-time hypermedia and fully supports hypermedia for virtual documents.

2.2 DHE Architecture

To supplement analytical applications with hypermedia functionality, the Dynamic Hypermedia Engine (DHE) [2] intercepts documents and screens as they are about to be displayed on the browser, adding link anchors dynamically over elements it can recognize. When the user selects one of these supplemental anchors, DHE generates a set of relevant links. Choosing one of these links prompts DHE to send a command (e.g., a query) to the target analytical application that will cause it to generate a virtual document containing the calculation results. The target application can be the same one that generated the original display or a different one. DHE can often provide this supplemental hypermedia functionality with minimal or no changes to the analytical applications through the use of application wrappers [4]. A wrapper is an interface between the hypermedia engine and the application, which mediates between them. Wrappers are described in more detail later.

DHE provides the following features in a dynamic environment:

- A Web interface for legacy systems.
- Automatically generated meta-information and hypermedia functionality.
- Support for virtual documents. Virtual documents in DHE are source documents from analytical applications. The application wrapper has the ability to parse elements in the source documents and dynamically generates links based on underlying relationships. However, DHE does not provide rich hypermedia functionality for elements and does not support dynamic regeneration, re-location and re-identification.
- Mapping rules to represent an application's internal structure. Mapping rules provide a mechanism to add commands to a link. When a user selects a link from the list of links generated, its mapping rule defines the commands for that link and the application wrapper will send those commands to the application to execute.
- Wrappers to integrate applications with minimal changes.

The current DHE architecture consists of several well-defined and separate processes, each possibly running on different platforms. It is shown in Figure 1.

2.3 Component Functionality

Component functionality is described as follows:

- User Interface (UI): This usually runs on the user's computer (Web browser).
- Gateway (GW): This enables the communication among the DHE modules and works as the router for all the DHE internal messages. All DHE messages pass through the GW, which then redirects them to the appropriate module.

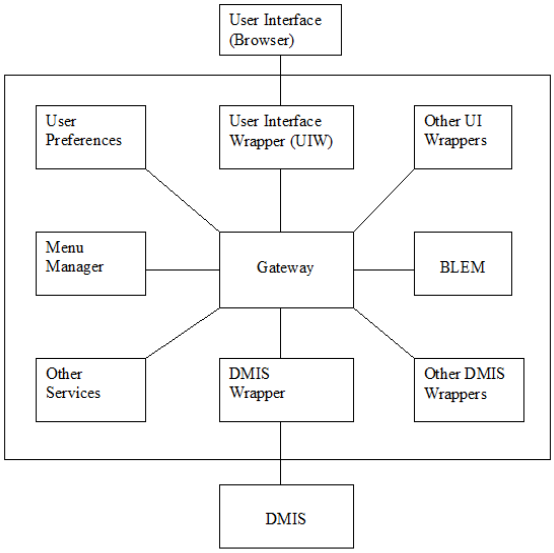

Figure 1. DHE Architecture

- User Interface Wrapper (UIW): This serves three important functions: First, it translates the displayable portion of the internal messages, from DHE's standard format to a format the UI can process, and vice versa; second, it handles the communication between the UIW and the UI; and third, it implements any functionality the DHE requires of the UI, which the UI cannot provide itself.
- Bridge Law Element Wrapper (BLEM): This maps the application data and relationships to hypertext objects at run-time. Bridge laws are another name for mapping rules [3]. BLEM maps the element instances in the virtual document to the global element types, and finds the links for them.
- Dynamically Mapped Information System (DMIS): This is an application system that dynamically generates the data requested by the user.
- Dynamically Mapped Information System Wrapper (DMISW): This manages the communication between the DMIS and the application system, translates the user requests from the DHE internal format to the application API, receives the output from the DMIS and converts it to the DHE format. The DMISW also identifies and marks the elements within the DMIS output to which hypermedia components are mapped.

3. THE JUST-IN-TIME HYPERMEDIA ENGINE

While DHE dynamically generates link anchors and links for virtual documents, it currently does not support re-identification, relocation or regeneration. The "Just-in-time Hypermedia Engine" (JHE) is an extension to DHE. It can supplement analytical applications with hypermedia functionality in this "just-in-time" environment. JHE's architecture is shown in Figure 2. The following sections describe identifiers, the architectural components and the information flow within JHE.

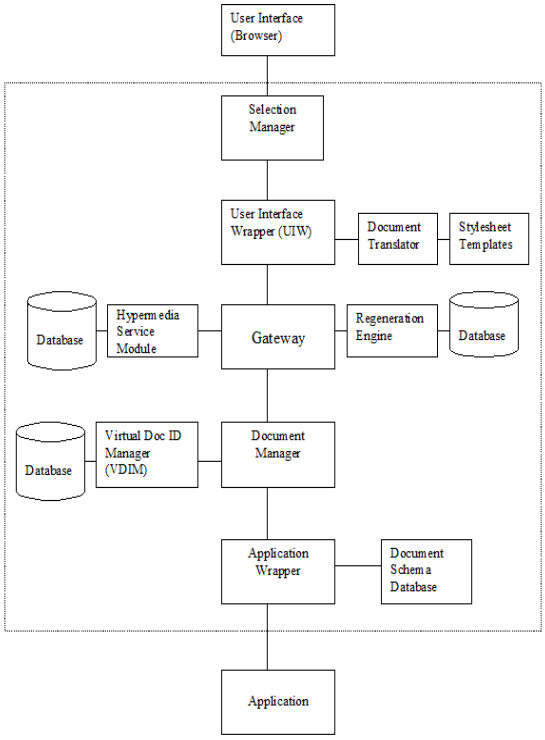

Figure 2. JHE Architecture

3.1 Component Functionality

JHE's architecture from Figure 2 uses many of the same component modules as DHE. Underlined components are entirely new to JHE and support re-identification, relocation and dynamic regeneration.

User Interface (UI): This usually runs on the user's computer (e.g., a Web browser providing a Web interface for the underlying analytical system). JHE displays virtual documents, anchor markers, and lists of comments and links on the UI.

Document Translator (DT): This translates a page in JHE's internal XML format to an HTML page for display if required according to the stylesheet template file. How virtual documents display varies on different Web browsers. Some Web browsers need stylesheet templates only; others may only be able to display HTML pages. In the latter case, a Document Translator is needed.

StyleSheet Templates: A stylesheet template is supplied for a virtual document or a set of virtual documents for display on the Web browser.

Selection Manager (SM): When the user selects a span of content on the UI in order to create an anchor, the SM gets the selection, and generates the anchor location information.

Hypermedia Service Module (HSM): This receives the hypermedia construct information from the user, then stores hypermedia construct information into the database. When user visits the hypermedia construct, HSM retrieves hypermedia construct information into the database and sends back to the user.

User Interface Wrapper (UIW): This serves three important functions: First, it translates the displayable portion of the internal messages from the JHE's standard format to a format the UI can process, and vice versa; second, it handles the communication between the UIW and the UI; and third, it implements the hypermedia functionality to allow users add hypermedia constructs (comments, links, bookmarks).

Application: A computer application external to the hypermedia system. This research focuses on analytical applications that dynamically generate virtual documents as the result of user queries. These may be any kind of user requests, not only database queries.

Application Wrapper (AW): This manages the communication between JHE and the application system, translates the user requests from the JHE internal format into the protocols the application requires and receives documents and screens for display from the application and converts each to an XML document and embeds it in an internal JHE message.

Document Schema Database (DSS): Application specific metadata for virtual documents and virtual elements are generated and maintained by the Application Wrapper and stored into the Document Schema Database.

Regeneration Engine (RE): This serves three important functions: First, the RE gets the necessary commands and parameters from the JHE database according to the virtual document ID. Second, to regenerate documents it sends commands to the appropriate AW for execution and gets back resulting virtual documents from the AW in XML page format. Third, it compares the newly-generated virtual document with the history information stored in RE database to revalidate it.

Document Manager (DM): This looks for the hypermedia components (links, comments, etc.) associated with a re-generated virtual document and virtual elements within it, marks the pre-existing anchors as elements, and generates a table of the hypermedia components for the virtual document. Re-locates and re-identifies the virtual elements to decide if they are the same as previously marked ones.

Virtual Document Identifier Manager (VDIM): This generates and maintains a table of identifiers for virtual documents, virtual elements and hypermedia components (such as anchors, links, and comments).

Gateway (GW): This enables the communication between the JHE modules and works as the router for JHE internal messages.

3.2 DHE Architecture

Information flows through the architecture are

shown as flowcharts in Figure 3 and Figure 4. Figure 3 shows how to visit a virtual document. Figure 4 shows how to add or display a hypermedia construct. Details are described as follows.

There are two ways to visit a virtual document, either by creating a new document or revisiting a bookmark. In the first case, the user should enter parameters on the application's home page (which is already integrated into JHE), and then submits it to UIW. In the latter case, he chooses the bookmark from the bookmark list displayed on the UI menu, and submits it to UIW. UIW gets the user commands from the UI, translates the necessary information into an internal JHE message and passes this message to the Gateway. The Gateway forwards it to the AW, which passes the commands to its application for execution. The application performs the request and sends the resulting screen or document to the AW for display.

When the AW receives the resulting document, it translates the source document into an XML document. If it is a regenerated document, the Regeneration Engine revalidates the document by applying the criteria information recorded in database. If it is a new document (i.e., there is no bookmark information about this document), the Document Manager generates a unique identifier for the virtual document and records the virtual document information (document identifier, application command, parameters, etc.) into database. Then the Document Manager looks for the hypermedia objects associated with this virtual document from the database. The Document Manager also relocates and re-identifies virtual elements in this virtual document. Then this virtual document is sent back to the browser for display.

After the virtual document is generated and displayed on the screen, users can add new hypermedia constructs for this document. Users also can browse the hypermedia construct information. Following explains how to add or display a hypermedia construct. Figure 4 shows the information flow.

When a user adds a hypermedia construct, he chooses a function ("add bookmark", "add comment", "add link", etc.) from the UI menu list. To add a comment or a link, he needs to select some texts from the screen. The Selection Manager generates anchor location information for the selected texts. Then the user enters other anchor information, such as name, granularity and re-identification criteria on the screen. Then UIW packs the commands into a message and sends them to Gateway. To add a bookmark, the user does not need to select some texts from the screen. He just enters the bookmark information, such as name and revalidation criteria. Then UIW packs the commands into messages and sends them to Gateway. Gateway receives the message, and calls the Hypermedia Service Module (HSM) to process the commands. For a bookmark, HSM stores bookmark information into database. To add a comment or a link, HSM stores the anchor information into database, and then it stores comment or link information into database. After all the information has been stored into database successfully, Gateway sends a message to UIW. Finally, UIW refreshes the document and inserts signs at the side of the newly created anchors for UI to display.

When a document is generated, it is displayed with anchors. Each anchor has a small icon inserted at its side. When the user clicks on the icon, JHE pops a new window to display a list of comment and link titles on the screen. This is called "display hypermedia constructs" in Figure 4. In this case, HSM retrieves all comment and link information for the anchor from database. When the user clicks on the comment or a link title on the screen, HSM retrieves the comment and link information from database. For a comment, UI displays comment title and content on the screen. For a link, if the link destination is an anchor or a bookmark, Gateway calls AW and RE to do regeneration and UI displays the destination virtual document. If the link destination is a URL, UI displays the page content on the URL.

For a bookmark, it is shown in the UI menu. When JHE starts up, Gateway retrieves all bookmark information and all bookmark titles are listed in the UI menu.

4. CONCLUSION

A Just-In-Time Hypermedia Engine has been designed in this paper. The JHE provides a solution to supplementing virtual documents for real time hypermedia work. The future works include implementing the JHE and evaluate its performance.

REFERENCES

[1] Z. Chen, L. Zhang, "Just-In-Time Hypermedia", Journal of Software Engineering and Application, Vol. 6, No. 5B: 32-36, 2013.

[2] R. Galnares, "Augmenting Applications with Hypermedia Functionality and Metainformation." Ph.D. Dissertation, New Jersey Institute of Technology, 2001

[3] M. Bieber, "Automating Hypermedia for Decision Support." Hypermedia. Vol. 4(2): 83-110, 1992.

[4] A. Bhaumik, D. Dixit, R. Galnares, M. Tzagarakis, M. Vaitis, M. Bieber, V. Oria, A. Krishna, Q. Lu, F. Aljallad, and L. Zhang, "Towards Hypermedia Support for Database Systems", Proceedings of the 34th Hawaii International Conference on System Sciences, 2001.