A Journal of Software Engineering and Applications, **2013, 6, 32-36**
doi:10.4236/jsea.2013.65B007 Published Online May 2013 (http://www.scirp.org/journal/jsea)

Just-In-Time Hypermedia

Zong Chen[1], Li Zhang[2]

[1]School of Computer Sciences and Engineering, Fairleigh Dickinson University, Teaneck, NJ, USA; [2]New Jersey Institute of Technology, Newark, NJ, USA.
Email: zchen@fdu.edu, zhangli73@yahoo.com

Received 2013

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. Conversely, the hypermedia features may have to cause target documents to be generated or re-generated. This paper presents a just-in-time hypermedia framework 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.

Suppose an analyst wants to determine projected profits for different sales levels in her company. She performs the analysis within a sales support application, and makes comments on each resulting profit calculation. A few days later when preparing her final report, she knows that she will need to include the calculation results and comments. She wishes to return to them without having to remember the input parameter values for each and then manually re-performing each calculation. Therefore, she creates a bookmark to the display screen containing each calculation result before discarding it. Invoking each bookmark later causes the sales support application to re-execute its calculations automatically, and the hypermedia interface to relocate her comments in the application's newly re-generated display.

A just-in-time (JIT) hypermedia system is different from a traditional hypermedia system in that it adds hypermedia functionality to dynamically generated virtual documents, while traditional hypermedia systems usually deal with static documents. The major challenges encountered in JIT hypermedia research are dynamic regeneration, re-location and re-identification.

In analytical or computational applications, documents do not exist in advance, they are generated when users do queries or execute some commands. When the windows close, these documents no longer exist. If a user adds some comments to a document, next time when he tries to visit the comments, the virtual document that was annotated should be re-generated. Some Web systems regenerate analysis screens by storing all relevant parameters in the virtual document's URL (Uniform Resource Locator), but many Web-based and non-Web based legacy systems have no such mechanism. Other Web systems do not allow URLs with detailed parameters for security or other reasons. When the user bookmarks and returns to its URL, the system returns the user to the home page or the initial input page. The JIT hypermedia system must provide automatic regeneration without asking the user to reenter parameters. Because a virtual document does not have a file name and a specific location to store the file content, dynamic regeneration requires a unique and persistent identifier to reference each virtual document. Dynamic regeneration also requires a criterion and procedure to decide whether the re-generated document is same as the previously marked document.

When a virtual document is regenerated, the virtual document's structure and content could be changed. It could also be possible that virtual elements in the document have changed. If a user has put some anchors on the virtual document, the position and content of the anchors could change. It is necessary to find the location of the predefined anchors in the virtual document, and decide whether the content is same as the previously marked one. This is called re-location and re-identification. Moreover, when analytical applications generate new query results, the JIT hypermedia system should relocate and re-identify the anchors for known elements in the old query results. Re-location requires flexible and efficient mechanisms to record the location and content of anchors. Re-identification requires a criterion to revalidate the virtual elements and some information about the virtual elements are needed.

2. Static vs Dynamic Links

Hypermedia is a concept that encourages authors to structure information as an associative network of nodes and interrelating links [1]. Hypermedia researchers view the terms hypertext and hypermedia as synonymous and use them interchangeably. Hypermedia nominally applies hypertext concepts to multiple media. A hypertext system is made of nodes (concepts) and links (relationships). A node usually represents a single concept or idea. It can contain text, graphics, animation, audio, video, images or programs. In most hypermedia systems, a node usually is a document. It can be semantically typed (such as detail, proposition, collection, summary, observation, issue) thereby carrying semantic information [2]. Nodes are connected to other nodes by links. Links represent relationships between two nodes. Static links are direct, persistent connections from one node or anchor to another. Dynamic links on the other hand are computed each time an attempt is made to traverse them.

The static links suffer from a variety of problems [3]. These include:

- Dangling links: when a link points to a node that is subsequently moved or deleted, that link becomes stale; attempts to traverse it will fail.
- Inexpressiveness: some conventional systems lack link semantics information. Even in systems that do provide link types, if the number of link types is set in advance, then the information content that can be attached to a link is restricted to the defined set.
- Expensive construction: manually constructing links is time-consuming and expensive.
- Inflexibility: manual links are created once and are thereafter fixed; they are unable to rearrange themselves to suit the needs of the moment.
- Duplication: The semantic knowledge implicit in a particular link or link type cannot easily be reused or generalized.
- Irrelevance: When a node's content changes or other conditions change, the link may no longer valid.

Dynamic links are created at run-time rather than being generated earlier (pre-computed links). One approach to dynamism in hypertext focuses on computing links based on relationships or similarities between texts or passages of text. In this approach, the link is not defined as a pointer from one hypertext node to another, but rather as a query that leads to a different node [4]. Pre-computed links can be constructed at any time, whereas dynamic links are computed at the moment they are required [5]. A dynamic link can take into account the specifics of the current user interaction. The query might be based on a combination of the browsing history, user profile, content of the current document, etc.

Compared to static links, dynamic links are more flexible than static links. Dynamic links are dynamically computed according to the underlying relationships between nodes or elements. Since dynamic links are computed at the time when users request to do so, there is no need to duplicate the links manually and it is not time consuming. On the other hand, dynamic links also can suffer from the dangling link problems. Dynamic link production, however, can detect a dangling link problem when it occurs and can guarantee that no dangling links are displayed. Dynamic links can allow links to be created from and to material that the link author does not control by overlaying the link within the document sent to a browser. When a node's content changes or other conditions change, dynamic link production has the potential to detect that the link is no longer valid.

Advantages of dynamic links include: a simplified interface to search functionality, reduced authoring effort, and greater opportunity for customization based on the current user's interaction history or specific task context. An additional benefit is decreased cost of maintenance. However, disadvantages include computation of each link at run-time (instead of using stored, pre-computed links), over-completeness and problems with false links. Computation links at run-time can be expensive in a large system with many users [5]. Over-completeness occurs when the reader is presented with more links than he/she can comprehend [6]. False links may occur because of polysemy, *i.e.*, multiple words and sentences that lead the search algorithm to incorrectly judge the similarity of text fragments. This may lead to unsound links [6] in the hypertext. JIT systems support both static and dynamic links.

An anchor in the proposed JIT system is a selection (which could be part of the document or the whole document) in a dynamically-generated virtual document. A link in the JIT system is the connection between two selections or multiple selections in one or more documents. JIT supports static and dynamic links. Static links in JIT are manual links created by users, which is similar to traditional systems. What JIT offers new are re-location and re-identification processing when previously-displayed documents are redisplayed. Dynamic links in JIT system are computed, based on the inherent relationships between elements, metadata structure, document structure, relationships among application specific commands and parameters, user group relationships, etc. These relationships could be analyzed before the hypermedia interface is designed or at the time an anchor is selected, and JIT then dynamically determines the links. JIT can support node types by analyzing the dynamically-generated virtual document types, through classification, and through categorizing the commands and parameters.

3. Just-In-Time Hypermedia Framework

The differences between dynamically-generated virtual documents and static documents are shown in **Table 1**. It gives the following conclusion:

(1) Dynamically-generated documents can be created in many flexible ways.

(2) It is more difficult to reference and version virtual documents. To reference a virtual document, a unique document identifier is required and a resolution scheme is needed. The "just-in-time" (JIT) hypermedia system should be able to generate unique identifiers and maintain them. Every time a virtual document is referenced, the JIT hypermedia system should resolve this ID to the specifications of the virtual document to regenerate.

(3) Maintaining hypermedia within dynamically-generated virtual documents leads to requirements that are different from static documents, including dynamic regeneration, relocation and re-identification.

Table 1. Dynamically-generated vs. Static Documents.

	Dynamically-Generated Virtual Document	Static Document
Status	Dynamic and virtual: does not exist in advance, only exists when the user visits the virtual document.	Static and real, stored persistently in some physical location with a specific file name.
Storage	Only specifications or links and other information about the document are stored; requires much less room.	The entire content of the document is stored.
Reference	Could be referenced by a unique document identifier or some specifications about the document.	Referenced by file name, location or a unique document identifier.
Generation	(1) Dynamically generated by query, search, system commands or user actions, depending on parameters specified by user. The document contains results from the database or computation module; or dynamic information, such as date or time. (2) The creation date and time, file size and content could change with each generation.	(1) Often created manually. (2) Once created, the date, time, file size and content are consistent.
Regeneration	Regeneration is required every time the user revisits the document.	No regeneration is required.
Template	Usually some kind of document templates, composition algorithms, scripts or skeletons facilitate the document's composition, organization and components during dynamic regeneration.	Can follow a pre-defined format or be free-form.
Metadata	Usually it is not difficult to analyze the metadata. Because the document is the result of query, search or calculation, useful metadata should be available. For example, the query results from database have some definite metadata, the user actions (commands + parameters) are known, the dynamic information (date, time) has clear meanings, and the calculation has definite metadata.	Metadata may be provided. If not, then it often will be difficult to infer.
Editable	Unless the virtual document can be saved as a static document, the ability to edit it is irrelevant. For the purposes of dynamic hypermedia functionality, virtual documents are not editable and only can be generated by the underlying application.	Usually editable. The date, time and file size information could be used to indicate whether the file has been edited.
Versioning	In order to version the document: (1) History information about the old version should be kept and sometimes a minimal copy of the old version is required. (2) Date and time information is not very useful to indicate that this is another version. (3) The file size could be used to indicate that it is a new version to the extent that two versions are treated as the same (further discussion later). (4) Byte-by-byte comparison is not possible, because only a minimal copy of the document is kept (this also depends on the two versions are treated as the same, as discussed later).	Usually performed by a document versioning system. Usually the original version of the document is kept. Then for each version, the system does a file comparison and keeps the file differences ("deltas"). When displaying the new version, the file difference is added to the old version to recreate the new version. Date, time and file size information is very useful for versioning purposes.
Anchoring elements in a document	To record the anchor information, internal document addressing is required, possibly using the same methods as with static documents. Every time a user revisits the document, element relocation and re-identification are required (details will be discussed later).	As long as the document has not been edited, there is no need to re-locate and re-identify the anchors. However, if anchors are stored in an external base, they technically need to be relocated, but this is straightforward.

Besides this, other requirements include virtual document support; document generation; unique identifiers for virtual documents and elements; internal document addressing mechanisms for elements; application specific metadata support; hypermedia functionality; integration with viewers and integration with information systems. These requirements are discussed in detail as follows:

• Virtual document support: Traditional hypermedia systems deal with static documents, which have persistent file names and locations. A JIT system should be able to reference a virtual document, record its information and enable hypermedia services for these virtual documents. Also it should enable hypermedia services for static documents.

• Document generation: A JIT system should be able to generate a document when a user asks for it. Document generation may be handled by an independent underlying information system. However, the JIT system should have the ability to control the generation process, add semantic information and display the documents on viewers.

• Document regeneration: When a user traverses to a link anchor or bookmark that was put on a virtual document, the JIT system should regenerate the document and then revalidate the regenerated document.

• Hypermedia functionality support: Hypermedia functionality should work for the dynamically generated virtual documents with a similar effect as for static documents.

• Re-location and re-identification: When a user traverses a link or comment that was put on a virtual element in a virtual document, the JIT system should be able to find the location of the anchor for this element in the document. This is called re-location. The JIT system also should be able to re-identify the anchored virtual elements in the virtual document.

• Unique, persistent identifiers for virtual documents and virtual elements: Reference to virtual documents and virtual elements requires unique and persistent identifiers across the hypermedia system.

• Internal document addressing mechanisms: Re-location and re-identification of virtual elements require flexible and efficient internal document addressing mechanisms.

• Application specific metadata: The metadata of the applications should be analyzed and supported to the extent that it is used when the JIT system regenerates and revalidates the virtual documents and elements.

• Integration with viewers: A hypermedia system could support multiple viewers (e.g., Web browsers and interfaces of non-Web applications). Because application-generated documents could be displayed by different viewers, this requires integration with viewers. Integration with viewer means the JIT system has an interface (middleware) between each viewer, which could transform the JIT data formats to data formats that different viewers could understand or display.

• Integration with information systems: To supplement hypermedia functionality for the underlying independent information systems, integration with information systems is required. Different applications give out different source documents in different formats. Each requires an interface or middleware to transform its source documents into the universal documents that the JIT system can understand and process.

A conceptual JIT architecture is proposed in **Figure 1**. A brief description of the conceptual architecture is as follows:

Viewer Wrapper: integrated with viewers. It is composed of the Selection Manager; the Document Presentation Manager and the Communication Manager. The Selection Manager gets the selections (content, location) from the screen. The Document Presentation Manager handles the layout of the internal virtual documents and translates the virtual documents to the data format that the viewer can recognize. The Communication Manger manages the communication between the viewer and the hypermedia engine.

Application Wrapper: manages the communication between the application and the hypermedia engine. It passes the application outputs to identify any potential elements which JHE may wish to identify as link anchors. It then translates the application specific data formats to any internal document data format that the hypermedia engine requires.

Hypermedia Engine: Hypermedia Engine is composed of the Hypermedia Gateway, the Regeneration Engine, the Hypermedia Service Module and the Virtual Document Manager. The Hypermedia Gateway deals with the communication between viewer wrapper and the hypermedia engine and the communication between the application wrapper and the hypermedia engine. The Regeneration Engine deals with virtual document generation and revalidation. The Hypermedia Service Module supplements virtual documents with hypermedia functionality. The Virtual Document Manager manages unique, persistent identifiers and other related information for virtual documents and virtual elements. It also does re-location and re-identification for the regenerated virtual documents and virtual elements.

4. Future Work

The differences between a just-in-time hypermedia system and a traditional hypermedia system are analyzed. A JIT hypermedia framework was identified based on the

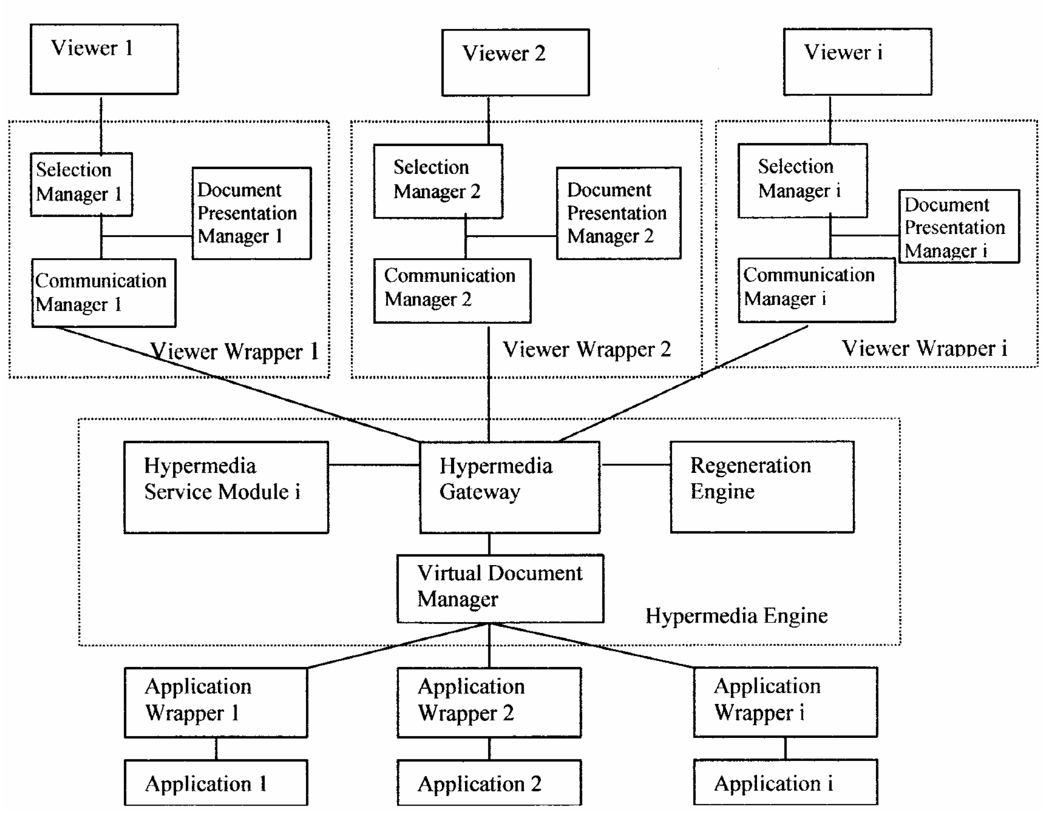

Figure 1. A conceptual JIT hypermedia architecture.

analysis. In the future, a JIT hypermedia engine is planned to be built based on the proposed architecture. Furthermore, a prototype of JIT system is planned to be built to provide dynamic virtual document support.

REFERENCES

[1] M. Bieber, F. Vitali, H. Ashman, V. Balasubramanian, and H. Oinas-Kukkonen, “Fourth Generation Hypermedia: Some Missing Links for the World Wide Web,” *International Journal of Human-Computer Studies, World Wide Web Usability Special Issue,* Vol. 47, No. 1, 1997, pp. 31-65.

[2] U. Rao and M. Turoff ,“Hypertext Functionality: A Theoretical Framework,” *International Journal of Human-Computer Interaction,* 1990. doi：10.1080/10447319009525989

[3] J. Mayfield, “Two-Level Models of Hypertext,” *Intelligent Hypertext: Advanced Techniques for the World Wide Web,* Vol. 1326, 1997, pp. 90-108.

[4] R. Bodner and M. Chignell, “Dynamic Hypertext: Querying and Linking,” *ACM Computing Survey,* Vol. 31, No. 4, 1999. doi：10.1145/345966.346002

[5] H. Ashman and J. Verbyla, “Dynamic and Externalized Linking: Requirements of Large-Scale Hypermedia Management Systems,” *Hypertext '93 Workshop on Hyperbase Systems,* 1993.

[6] P. Thistlewaite, “Automatic Construction and Management of Large Open Webs,” *Information Processing and Management,* Vol. 33, No. 2, 1997, pp. 161-173. doi：10.1016/S0306-4573(96)00060-X