Employing Smart Browsers to Support Flexible Information Presentation in Petri net-based Digital Libraries

Unmil P. Karadkar, Jin-Cheon Na*, and Richard Furuta

Center for the Study of Digital Libraries and Department of Computer Science
Texas A&M University
College Station, TX 77843-3112, USA
{unmil, jincheon, furuta}@csdl.tamu.edu

Abstract. For effective real-life use, digital libraries must incorporate resource and system policies and adapt to user preferences and device characteristics. The caT (context-aware Trellis) hypertext model incorporates these policies and adaptation conditions within the Petri net specification of the digital library to support context-aware delivery of digital documents in a dynamically changing environment. This paper describes extensions to the caT architecture for supporting adaptation via smarter browsers and an external resource store to provide greater flexibility in information presentation. Browsers request resources that they can best display with their knowledge of intrinsic capabilities and constraints imposed on them by the devices that they run on. The data store returns the most appropriate version of a resource in response to browser requests, thus allowing maintainers of libraries to add, modify and remove resources without any changes to the structure, presentation or document pointers in the digital library.

1 Introduction

Libraries are dynamically changing social spaces that provide access to collections of resources, physical as well as digital. In addition to the resources, libraries provide reference and other services to patrons, some all the time and others during specific hours. Patrons may incur costs for using some of these services, while others may be available at no cost. Actions of patrons affect others who access the libraries. For example, checking out the last available copy of a resource by one patron for finishing a class assignment will render the resource inaccessible to others until someone returns their copy of this resource. To ensure smooth functioning libraries set policies that must be followed by all patrons. In the case of resources being unavailable, users must wait until a copy is available for their use. Libraries must also abide by the licensing regulations set by creators of resources that they contain. For example,

* Jin-Cheon Na's current address is: Division of Information Studies, School of Communication & Information, Nanyang Technological University, 31 Nanyang Link, Singapore 637718 (tjcna@ntu.edu.sg).

libraries may not photocopy the books they buy or make copies of digital resources like VHS tapes, CDs, or DVDs.

When porting this model to the digital world, it is essential to support the dynamism of the environment created by the presence of other users in the space as well as the policies of the service provider. At the same time, the service must be convenient to access, easy to use and must account for the needs and preferences of various users and characteristics of the devices used to access the resources. Digital library support for these features can be better managed at the architecture level than at the application level. Integrating the policy and adaptation support in the architecture simplifies the development of the library application.

This paper describes extensions to the caT architecture to support flexible information presentation by incorporating smart browsers, an externally accessible resource store to decouple information resources from the digital library that includes them, and mechanisms to retrieve the most optimal resources available at the time of user access. Na and Furuta developed caT (context-aware Trellis) [14] by augmenting the Trellis Petri net-based system [21] to support context-aware adaptation and delivery of digital documents in a dynamically changing environment. We are building on the strengths of the caT model by incorporating browsers that are aware of their capabilities (that is, the media types that they support) and the constraints imposed on them by the devices that they operate on (display resolution, size, network bandwidth). Further, they may possibly be controlled by a remote manager to present information to the user on the most suitable devices in the given situation.

The remainder of the paper is organized as follows: In section 2, we review relevant aspects of our earlier work. Section 3 presents the architecture of the system along with extensions that may go into it shortly. Section 4 describes an example of how the system will serve users who are trying to access information about the bus system on a University campus in different situations. Section 5 reviews related work and systems. Section 6 provides a discussion of the various issues involved and outlines avenues for future work in providing information to users that best suits their context of use at the right time in the best possible format. Section 7 summarizes and concludes the paper.

2 Trellis and caT

caT has a long history, draws from concepts in automata theory and has applications in a variety of areas. In this section we have only tried to present the features that are essential for understanding the current system and that help punctuate the strengths of this system for implementation and use of Digital libraries and other complex structures and systems. We provide pointers to relevant literature that explains the concepts in greater detail.

2.1 Trellis

The Trellis project [21, 22] has investigated the structure and semantics of human-computer and human-human interaction in the context of hypermedia systems, computer-supported collaborative work (CSCW), program browsers, visual programming notations, and process models. Trellis uses a Petri net [15] based representation of hypermedia documents and applications. Information elements map to *places* (denoted by circles) in a Petri net and links to *transitions* (denoted by bars). *Tokens* in places represent the current location of users in the hyperdocument in terms of the information elements that they see. When a place is *marked* (contains a token), the corresponding content is displayed, and when a transition is *enabled* (all places leading to the transition contain a token for each arc that connects the two), the corresponding link can be selected (*firing* the transition; i.e., removing a token per arc from each place that leads to the transition and adding one to each place that is led to from the transition). Due to lack of identity for tokens, the basic Petri net is not convenient for representation and analysis of complex systems. Trellis uses colored timed Petri nets (CPN) [10] where colors represent various users, thus allowing different access specifications for various (groups of) users. The timing values on transitions can be set to enable and fire transitions automatically. Thus, Trellis treats users and processes identically as processes may browse the hypertext by waiting for automatic firing of the transitions. We can conceive of the CPN as the task description and the associated contents as the information required by the task. A detailed overall description of how Trellis works is given in [7].

The dual nature of Petri nets allows specification of document structure as well as the browsing semantics. While the graph structure describes the nodes and links in a hypertext, the associated automaton semantics, when applied, specify its browsing semantics or the sequence of information elements displayed and the links made available while browsing. Trellis browsing semantics are said to be programmable [6] since localized change to the specification can be used to change the browsing behavior. As the behavior of a document depends on its initial state, a virtual change in the document specification can modify the browsing semantics for the document.

Trellis browsing semantics provide a mechanism by which a hypertext can respond to events in its environment. As examples, Trellis hypertexts can be designed to dynamically permit or limit access to information, to select different information for presentation on initial and subsequent accesses for information elements, and to respond differently if multiple readers are viewing a section of a hypertext simultaneously. This feature can be used to implement specification of dynamically changing access policies for documents. The system implementation makes no assumptions about these but simply interprets the document specification.

Trellis documents incorporate parallelism, in that multiple information elements can be displayed simultaneously. The parallelism allows synchronous as well as asynchronous traversal of multiple content elements. The document specification can direct simultaneous display of multiple information threads, each of which can be accessed independently or synchronized display of related components of a multimedia presentation, all of which must change together.

The Petri net structure is based on formal specification and can be analyzed for behavioral properties as well as performance assessment [25]. Thus, the Petri net

structure can be formally analyzed to test the reachability of specific nodes starting with various initial states and for finding nodes that match specific properties to analyze and debug complex systems, for example, finding all nodes that do not have any outgoing links.

Over the years, the Trellis document specification has been applied to a wide variety of domains. These include applications in quick prototyping of process protocols in the areas of software engineering for specifying control algorithm for an elevator [4] and Computer Supported Collaborative Work for simulating virtual meetings that follow various protocols [5].

2.2 caT (context-aware Trellis)

Like Trellis, caT [14] is a document-centric system and directly inherits all its advantages mentioned earlier. caT was developed to support context-aware information delivery by augmenting Trellis with a high-level Petri net specification, context-awareness, user models and a fuzzy knowledge engine. The high-level Petri net tokens in caT include additional state information that expresses a user's context and preferences such as time, location, available bandwidth, acceptable costs and so on, thus enabling adaptation based on changes in the user's environmental information. caT also enhances the support for authoring, browsing and analysis of complex hypertexts by incorporating an authoring and analysis tool that supports hierarchical Petri Nets [14]. The fuzzy engine allows authors of documents to specify behavior based on abstract conditions like "if the user is on-campus", "if it is daytime". The Petri net engine forwards the inputs received from environmental sensors to the fuzzy engine, which infers new information with the specific inputs and a specified rulebase.

caT also includes a HTTP interface [3] for tapping into the potential of the WWW. The interface allows creators of caT nets to specify templates; essentially HTML files with placeholders indicating where information is to be embedded when the corresponding places are marked and transitions enabled. From caT's point of view, the template file is simply another content type. It takes control when its corresponding place is marked. The template file specifies whether and how to render the active content and transitions. The node that contains a template file can be viewed as specifying the directions by which a virtual composite node is constructed from the active content elements. As the composite node only displays elements that are currently active the content of the composite node changes as the user browses the hypertext.

A formal definition of caT and more information about the system prototype can be found in [13, 14].

3 Architecture

caT supports RPC-based client-server interaction in a 4-layer architecture comprising of the server, services, clients and devices. The server layer stores global data like Petri net structures, resources, user profiles, etc. Various processes in the server

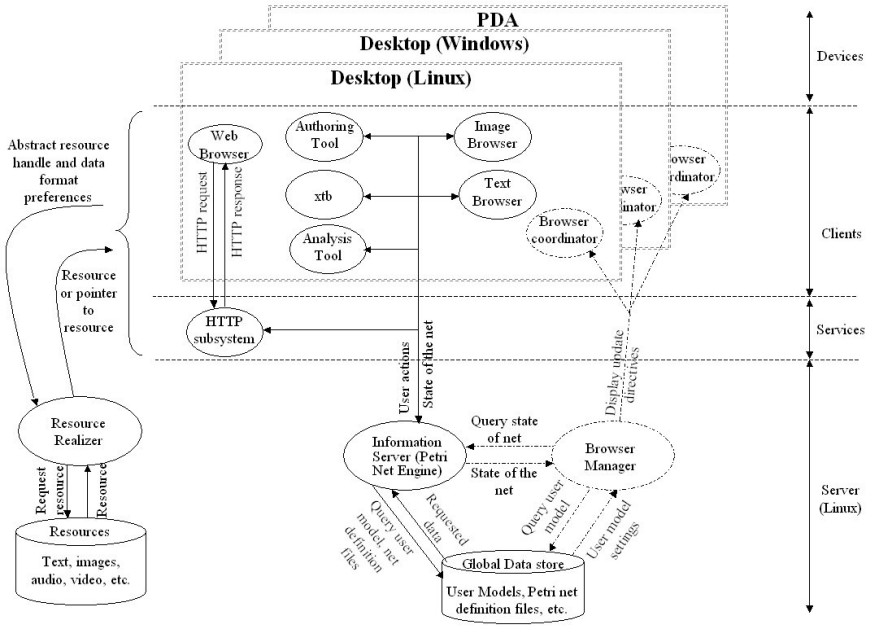


Fig. 1. Extended caT Architecture

provide access mechanisms for the clients to support user browsing of the hypertext by executing the Petri nets, help clients realize resources from abstract handles and in the near future may aid in adaptation of the hypertext by directing browsers to display resources based on various parameters. The services layer provides an HTTP interface for accessing the Petri net engine via Web browsers. Web-specific and other browsers located on a variety of devices display the hypertext structure and resources to the users. This section describes the architecture shown in Figure 1 from the perspective of its importance in structuring and accessing resources in a caT-based digital library. Technical aspects pertaining to the implementation of various components are explained in greater detail by Na and Furuta in [13, 14].

3.1 Information Server

The Information Server (also called the Petri net engine) provides function interfaces for creation, editing, annotation and execution of Petri nets to remote processes and to any client applications that may be running. It runs in the background and has no visible user interface. The Petri net engine reads in the Petri net structure that may define the structure and browsing semantics pertaining to the digital library along with user models from the Global Data Store. These are used to define, constrain, and

reflect user actions, in effect the Petri net engine executes user browsing of hypertext structures. While creating and analyzing hypertexts, the authoring tool updates the net structure stored in the server. As users browse the hypertexts the clients facilitate two-way communication between users and the Information Server. The net's document specification refers to resources available in the library by their conceptual and semantic content (in the form of abstract resource handles) that are passed on to clients that must then procure the appropriate resources.

3.2 Browser Clients

The browsers are the only part of the system that the users interact with directly. Users receive resources and information contained in caT hypertext nets via the browsers. The browsers convey actions performed by the users (for example, following active links) that may result in a change to the current state of the net. As actions of other users in the environment affect a user and change the state of the document, the browsers periodically check with the server for any updates to the state of the net that result in changes to the display. The server provides the current state of the browser and returns abstract handles for resources that are to be displayed. The browser must then contact the Resource Realizer and retrieve the resource in a format that it best supports or is most optimal for presentation to the user in the current context.

3.3 Resource Realizer

The Resource Realizer provides dual advantage by incorporating a layer of abstraction between specification of various resources in a digital library and their physical manifestation in various formats. Conceptually, it encapsulates all resources that contain similar or interchangeable information irrespective of the presentation format. Thus, a photograph of the launch of a space shuttle, its video and its text description may all be stored as a single conceptual unit and information in either or multiple formats may be presented to the user upon request by a browser. Practically, it decouples the hypertext structure that includes the resource from the location and representation of the resource. Thus, if the video file of the shuttle, in the earlier example, were to be corrupted later, it could be removed from the corpus or replaced with another video without the need to modify any other part of the system including the net structure that refers to it.

Operationally, the Resource Realizer receives an abstract resource handle from various browsers along with the browsers' preferred media type(s). The browser requests the resources based on the knowledge of its intrinsic capabilities (for example, a text browser may request text file while an image browser may request a jpeg file). The Resource Realizer returns the resource to be displayed along with the access method to be used by the browser. The Realizer could either return the contents of the resource (which can be directly processed by the browser), or with a

location pointer within the browser's file system (disk path) or via a globally accessible location pointer (a URL).

3.4 Browser Manager

The Browser Manager and Coordinators (sections 3.4 and 3.5) are not yet operational and hence considered as a future extension. Figure 1 shows these components differently than the system that is currently in place, with dotted lines. However, these sections are included here for the sake of completeness and coherence.

The Browser Manager is expected to function as the centralized display controller. It will communicate with the Browser coordinators on various devices to manage the routing of information based on use and user information from user profiles, current tasks and status of the user, system policies, and knowledge about other users in the system and their actions, among other factors. For example, it may send critical information to a user's cell phone instead of send it to the PDA or a notebook computer if the user is driving. The Browser Coordinator is responsible for invoking browsers on the local device upon the Browser Manager's instructions.

In order for the Browser Manager to be effective, the Information Server treats it like it treats all other clients. The Manager must ensure that it receives updates to the state of the net and decide if it needs to act upon these, based upon the user's current state (available from the net), user profile (available from the Global Data Store) and other policies that may be specified in the net.

3.5 Browser Coordinators

The Browser Coordinators are device-specific clients that contain information about capabilities of the device and various browsers available on the device. They invoke these browsers when the Browser Manager directs them to present the contents of a marked place or a set of places. The browser that is opened must then communicate with the Information Server and the Resource Realizer to present information to the user.

The following section describes a scenario to illustrate an example of user interaction with caT. The scenario is presented from the user's perspective with tie-ins to the architecture at relevant points.

4 Example of Browsing a caT-based Digital Library

Large Universities often provide information resources for students, employees and visitors via their Web servers. These University-wide information systems include resources on academic information, organizational and administrative information, catalogs of books in the libraries, networking and computing information, shuttle and commuter information and overview of reasearch among other things. In spite of the

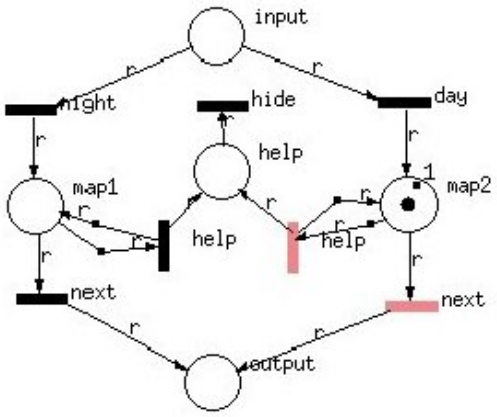
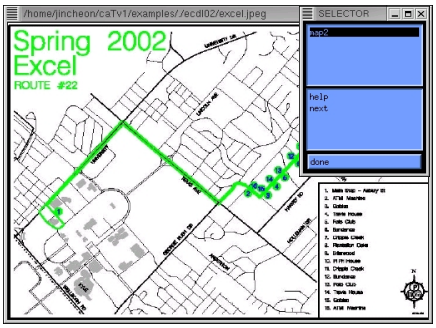


Fig. 2. Bus Information Service: subnet specification

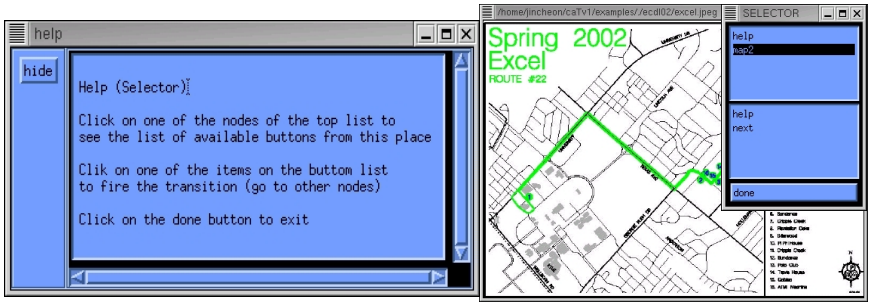
(a) Text browser output: map2 place



(b) Image browser output: map2 place

Fig. 3. Text and image displays for the Bus Information Service

availability of these resources students, employees and visitors on large university campuses often have trouble getting from one place to another. Typically, the users must ensure that all needed information is available to them before they set out and stick to their planned route as much as possible. However, little support is provided for shuttle users to access the information while they using the system. The users must rely on information from others around them, who may also not know answers to the users' specific questions. The recent surge in browsing the Web via PDAs or cell phones is yet to make its mark on campus information systems. Here, we show how a user may access information on the go using any of the various available network-enabled devices and caT infrastructure that is currently available.



(a) Text browser output: help place (b) Image browser output: map2 place

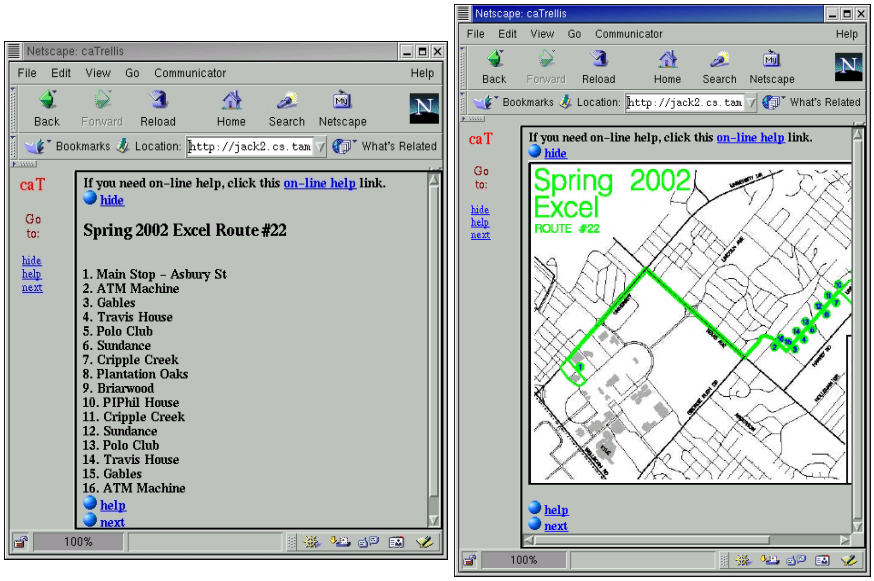
Fig. 4. Help (text) and map displays for the Bus Information Service

The user starts a session when she accesses the information pages for the caT-based shuttle and commuter information pages. At the start of the session, caT identifies the user and the state of the user, for example whether the user is a student or employee, the current time is day or night and if the student is on or off campus. Not all information that caT gathers may be used during an interaction.

Figure 2 shows a part of net specification that was used to display the bus route information. Here we only show the simple net used to control the display of two information resources, the day and night route maps for #22, Excel, for the Spring 2002 semester along with the "help" resource that explains how to use the information resource and interact with the system. This net is a part of a larger hierarchy of nets that form the infrastructure and specification for browsing all resources in the shuttle system. In the net, *input* and *output* places are linked to the places in its parent subnet. The *nigh*t and *day* transitions control flow of tokens based on the current state (day or night) as determined by the system. In figure 2, the system has determined that it is daytime, causing the token to move through the *day* transition to the *map2* place that has associated map information of daytime bus route. The places, *map1* and *map2*, contain abstract resource handles to various representation formats of information for the bus route in question. Thus, when the user accesses the net via a text browser, presumably located on a small hand-held device with limited network bandwidth and display resolution, the Resource Realizer returns the information in a textual format as shown in figure 3 (a). When the same net is viewed via an image browser, that could be located on a networked notebook computer possibly with higher bandwidth, the resource handle resolves to an image file of the bus route as shown in figure 3(b). Thus, caT-based hypertexts provide good separation between the net structure and its related contents with the help of Resource Realizer.

The selector shown on top of the route image in figure 3(b) is the main browser that allows firing transitions (the caT equivalent of following links) to browse the hypertext.

When the user selects (or fires) the *help* transition for acquiring help information the system moves the token into the *help* place, causing the browser to display its contents. In this example, the help information is only available as a text file and the



(a) Display on mobile devices

(b) Display on desktop computers

Fig. 5. Web browser displays for the Bus Information Service

image browser is unable to display this information. Figures 4(a) and 4(b) show the display of the text and image browsers, respectively, when the user views the help information. In the future the Browser Coordinator could help resolve such situations by negotiating with the various browsers and the Resource Realizer to either obtain the best resource or invoke the most capable browser for presenting the information that is available.

Browsers that are capable of processing information in multiple formats may use their full capability to provide optimal information to the user. If the user prefers to view the information via a Web browser, the HTTP interface assists with presentation of the information in figure 3 in a suitable format. Figure 5 shows composite views of the information when viewed via a Web browser. The system may display the textual information as shown in figure 5(a) when a browser running off of a PDA requests information considering the lesser bandwidth, while it may display the image of the bus route as shown in figure 5(b) when the user views the information from a desktop system connected to a fast network.

5 Related Work

Various project groups have researched aspects of digital library infrastructure, policies, and access mechanisms among other aspects. Schilit, et al. [18], envision widespread use of mobile appliances for accessing digital repositories in the near future. We can already see the surge in wireless connectivity via the notebook computers, cell phones and PDAs. We believe that in the near future there is an urgent need to support information work via a plethora of appliances and have aimed the development of caT towards supporting users in various work settings armed with a variety of devices.

Sistla, et al. [19], propose various cost structures for accessing digital libraries and predict that future digital library access tools will have cost optimizers built into them. They mention that various cost models like consumer-paid, advertiser paid, free resources (paid for by the provider) will co-exist and the users may be able to optimize their costs for accessing resources.

Furuta, et al. [2], have emphasized the importance of physical objects in scholarship and research. Marshall [12] provides examples where patrons of digital libraries search for physical holdings using a digital access mechanism. Brown's stick-e documents [1] are another example of gaining more information about physical objects via digital annotations. The Topos 3-D spatial hypermedia system [8] associates hypermedia information with representations of real world artifacts to compose mixed metaphorical-literal workspaces. In the following section, we describe how caT can adaptively guide users to physical resources.

Several Web-based systems provide resource indirection by intermediate resolution. Some of the most commonly used systems include OCLC's Persistent Universal Resource Locators (PURLs) [17] and the Handle system [9] that gained acceptance via its incorporation in Dienst (the Cornell Digital Library) and then NCSTRL [11]. These systems provide assured access to given resources in the face of frequent location changes. However, they tie in the pointers with a specific resource in a specific format. We believe that the Resource Realizer provides more features than simple location-independent access. More recently, systems like KeyLinking [16] provide facilities for dynamic resolution of soft and implied links. This research focuses on "providing access to the most appropriate available resource at the time of usage". Though the phrase sounds similar to what the Resource Realizer does, it differs from KeyLinking in that the Resource Realizer makes no judgment about what is appropriate but leaves the decision to the browser.

Among the contemporary multimedia presentation systems, SMIL [20] provides mechanisms that permit flexible specification of multimedia presentations that can be defined to respond to a user's characteristics. However, this support is restricted to an instantiation of the presentation. Once instantiated, the changes in browsing state cannot be reflected to other browsers that the user may open later, or to other devices. Similarly, XML [23] and XSLT [24] implementations allow flexible translations of given static structures, but these transformations are static and irreversible. Generation of a new transformation requires repetition of the process with a different XSLT template.

6 Discussion and Future Work

The Resource Realizer decouples the conceptual resource from its instantiation. It enables creators of digital libraries to package all resources that are semantically equivalent. The abstract resource may then be appropriately resolved to retrieve its most suitable presentation for each user taking into account the user's preferences, task environment and other factors like the presence and location of other users in system. This encapsulation can be further extended to support versioning in digital libraries. All versions of a resource may also be linked together, to be resolved to present the right version(s) at access time. For the example in section 4, traffic pattern analyzers could compare the bus routes for various semesters to plot trends and assess the future needs of the campus.

The Realizer also may act as a resource catalog that points to various resources owned by different entities, some or all of which may charge the users for providing access. The Resource Realizer could then act as the cost optimizer [19] for caT and negotiate the costs for resources to provide the best possible deal for the users depending on their preferences.

The resources that the system includes also may include physical resources, for example, books on shelves in a physical library. The resource realizer may also provide the location of the physical resources. The Resource Realizer is simply the gatekeeper for accessing resources. It does decide if or how the resource will be used by the entity that requests it. Thus, the information regarding location of physical resources, if provided by the Realizer, may or may not be presented to the users depending on their context.

While browsing, caT maintains the current state of the net in the server. This state is reflected in all browsers that connect to the server and request updates. Users only see their state (the places that they are viewing) even though others may be accessing the system simultaneously. However, this permits interactions between users. For example, caT net specifications may allow only a certain number of users to access a resource simultaneously, thus providing support for access control policies. Once all available copies of a resource are in use, the system may temporarily prohibit access, and indeed, cause the resource to vanish (if the designer of the system so desires) until a license of the resource is available for use.

As the browsing state is maintained on the server, users accessing it via multiple browsers on various devices see the same information on all devices. Also, when the user browses the digital library, the changes effected via an action in one browser are reflected (almost) instantaneously in all other browsers on all devices. Thus, our user who is trying to find out more about the bus system, may start viewing the shuttle pages on her desktop, locate the nearest bus stop for the bus she wants and realize that she needs to hurry to get onto the next bus and leave without closing her browsing session on the desktop computer. On her way to the bus stop, she can open another browser, presumably from her PDA to check on the fare, the bus route and decide if she can get other things done during this trip. While coming back home, she can start reading an article that interests her and after coming home, continue reading it on her desktop, where she may also be able to watch relevant videos or see images that her PDA could not adequately display.

The Browser Management system described in sections 3.4 and 3.5 (Manager and the Coordinators) is still in the design stage. Upon integration, the Browser Manager is expected to communicate with the Browser Coordinators on the devices in order to present various active information elements on different devices, in response to user profiles, device capabilities and the current state of the user. The user may configure the behavior of the Manager and the Coordinators through configuration files. For example, the user may decide to allow the manager to open all possible browsers that are capable of displaying an information element, or to allow it to open only the browser that can best present it. Thus, in the example in section 4, the first case would allow the Manager to open both, the text as well as the image browsers, and if it exists, the Web browser. This implies that the user viewing the bus route via the image browser would still be able to get the textual help information. The latter case would however require that the Manager opens only the Web browser as it can handle all media types, if the device supports the high bandwidth and display resolution.

If two or more browsers are capable of displaying a media type, the Browser Coordinator may act as the local conflict resolver by closing one of the applications. Media types do not have to match the conventional types. Creators of resources may create multiple media types for a conventional type, for example, to support Web-based browsing from different devices, creators of the digital library may support different versions of an html document (i.e., “full html” and “short html”). Web browsers on mobile devices may request and display “short html” files while desktop computers may present “full html” files.

The interaction between the browser and the Realizer can be further enhanced to enable them to reach a mutually acceptable resolution for a given resource handle. Currently, the Realizer either provides a resource, or does not. For the earlier example, the *help* resource could not be realized for the image browser. The proposed enhancement would cause the Realizer to respond with the best available resource, which can then be displayed by another browser that may be capable of presenting this resource. This negotiated resource resolution may probably be better handled by the Browser Coordinators than the browsers themselves.

7 Conclusion

In this paper we have described extensions to caT (context-aware Trellis) for supporting flexible information presentation in digital information repositories via browsers on a variety of devices. caT provides a consistent display of the current browsing state on multiple browsers and reflects the changes caused by user actions in one browser to all open browsing instances. Browsers are aware of their capabilities and the constraints of the devices that they run on. The browsers receive abstract handles for resources they must present to the users from the server. An abstract handle may point to possibly many equivalent instantiations all of which are semantically identical and hence interchangeable. The Resource Realizer resolves abstract handles to return resources in a media formats described by the browser. This architecture can be further extended by the incorporation of a central Browser Manager and device-specific Coordinators to deliver information to browsers on

various devices and presumably at specified time in order to help users view the relevant information in its most suitable form based on their preferences and the immediate task at hand.

8 Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No. DUE-0085798.

References

1. Brown, P.J.: The stick-e Document: A Framework for Creating Context-Aware Applications. Proceedings of EP '96 (Palo Alto, CA, 1996) 259-272
2. Furuta, R., Marshall, C., Shipman, F.M., Leggett, J.: Physical objects in the digital library. Proceedings of the first ACM conference on Digital Libraries (Bethesda, Maryland, United States, 1996) 109-115
3. Furuta, R., Na, J-C.: Applying Programmable Browsing Semantics Within the Context of the World-Wide Web. Proceedings of the thirteenth conference on Hypertext and hypermedia, Hypertext '02 (College Park, Maryland, USA, June 11-15, 2002) 23-24
4. Furuta, R., Stotts, P.D.: A hypermedia basis for the specification, documentation, verification, and prototyping of concurrent protocols. Technical Report TAMU-HRL 94-003, Texas A&M University, Hypertext Research Lab, (June 1994)
5. Furuta, R., Stotts, P.D.: Interpreted Collaboration Protocols and their Use in Groupware Prototyping. Proceedings of ACM 1994 Conference on Computer Supported Cooperative Work, ACM (Oct. 1994) 121-132
6. Furuta, R., Stotts, P.D.: Programmable Browsing Semantics in Trellis. Proceedings of Hypertext '89, ACM, New York (1989) 27-42
7. Furuta, R., Stotts, P.D.: Trellis: a Formally-defined Hypertextual Basis for Integrating Task and Information. Lawrence Erlbaum Associates (2001) 341-367
8. Grønbæk, K., Vestergaard, P. P., Ørbæk, P.: Towards Geo-Spatial Hypermedia: Concepts and Prototype Implementation. Proceedings of the thirteenth conference on Hypertext and hypermedia, Hypertext '02 (College Park, Maryland, USA, June 11-15, 2002) 117-126
9. Handle: The Handle System. http://www.handle.net/ (2002) accessed April 2002
10. Jensen, K.: Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use Volume 1. EATCS Monographs on Theoretical Computer Science, Springer-Verlag (1992)
11. Lagoze, C., Fielding, D. Payette, S.: Making global digital Libraries work: collection services, connectivity regions, and collection views. Proceedings of the third ACM conference on Digital Libraries (Pittsburgh, Pennsylvania, United States, 1998) 134-143
12. Marshall, C., Golovchinsky, G., Price, M.: Digital Libraries and mobility. Communications of the ACM 44, 5 (May 2001) 55-56
13. Na, J-C.: Context-aware Hypermedia in a Dynamically Changing Environment, Supported by a High-level Petri Net. Ph.D. Dissertation, Texas A&M University (December 2001)
14. Na, J-C., Furuta, R.: Dynamic documents: Authoring, browsing, and analysis using a high-level Petri net-based hypermedia system. Proceedings of the ACM Symposium on Document Engineering (DocEng '01), ACM (2001) 38-47

15. Peterson, J.L.: Petri Net Theory and the Modeling of Systems. Prentice-Hall, Englewood Cliffs, N.J. (1981)
16. Pritchett, B.: KeyLinking: dynamic hypertext in a digital library. Proceedings of the fifth ACM conference on Digital Libraries (San Antonio, Texas, United States, 2000) 242-243
17. PURL: OCLC's Persistent Universal Resource Locators. http://www.purl.org/ (2002) accessed April 2002
18. Schilit, B.N., Price, M.N., Golovchinsky, G.: Digital library information appliances. Proceedings of the third ACM conference on Digital Libraries, ACM (Pittsburgh, Pennsylvania, United States, 1998) 217-226
19. Sistla, A. P., Wolfson, O., Yesha, Y., Sloan, R.: Towards a theory of cost management for digital Libraries and electronic commerce. ACM Transactions on Database Systems (TODS) 23, 4 (December 1998) 411-452
20. SMIL: Synchronized Multimedia Integration Language (SMIL 2.0) specification. http://www.w3.org/TR/smil20/, W3C Proposed recommendation (2001)
21. Stotts, P.D., Furuta, R.: Petri-net-based hypertext: Document structure with browsing semantics. ACM Transactions on Information Systems, 7, 1 (January 1989) 3-29
22. Stotts, P.D., Furuta, R.: Dynamic Adaptation of Hypertext Structure. Proceedings of Hypertext '91, (1991) 219-231
23. XML: Extensible Markup Language (XML) 1.0 (Second Edition). http://www.w3.org/TR/2000/REC-xml-20001006, W3C Recommendation (2000)
24. XSLT: XSL Transformations (XSLT) Version 1.0. http://www.w3.org/TR/xslt, W3C Recommendation (1999)
25. Zurawski, R., Zhou, M.: Petri Nets and Industrial Applications: A Tutorial. IEEE: Transactions on Industrial Electronics, vol. 41, no. 6, (December 1994)