Context-Aware Digital Documents Described In A High-Level Petri Net-Based Hypermedia System

Jin-Cheon Na and Richard Furuta

Department of Computer Science, Texas A&M University
College Station, TX 77843-3112, USA
{jincheon, furuta}@cs.tamu.edu

Abstract. As mobile computing becomes widespread, so will the need for digital document delivery by hypertextual means. A further trend will be the provision of the ability for devices to determine *where* they are, e.g., through the inclusion of GPS sensors, as will the need for devices to operate on heterogeneous networks. Consequently, hypertext systems supporting these devices will benefit if they also can become *context-aware*. In this research, we introduce caT (for Context-Aware Trellis), a context-aware hypertext model and associated tools, which supports flexible user (or agent) adaptation to changes in environmental information, such as time, location, bandwidth/cost, etc. Firstly a context-aware hypertext model is proposed by incorporating both high-level Petri net features and user-modeling into the previously-described Trellis hypertext model. Major features of the high-level Petri net that are being explored are structured tokens and flexible net description. Fuzzy knowledge (context) handling is supported by the integration of a fuzzy logic tool with the Petri net, and a flexible information presentation tool has been designed to support Web-based browsing of documents specified in the caT model. Additionally, authoring and analysis issues are explored to support structured authoring and verification of application models, respectively. A simple example, a university digital library, is introduced in the paper to explain the concept of the caT model.

1 Introduction

As mobile computing becomes widespread, so will the need for document collections that can customize themselves for reading in different environments. If mobile documents properly reflect their readers' context, they can provide more relevant information to meet their readers' dynamically changing contextual requirements. Mobile systems should provide context-aware services without requiring readers to provide explicit context data. For this purpose, over the last decade, some researchers have built context-aware applications [6, 17] that take advantage of environmental information to provide better interaction with readers. Context-aware applications typically focus on a mobile user who is carrying a portable system, such as a Personal Data Assistant (PDA), that has been augmented with environmental sensors, such as GPS receivers, active badges, electronic compasses, etc. These environmental sensors could detect location, orientation, time of day and time of year, temperature,

companions or objects nearby, user identification, etc. Additionally, high-level sensors (software sensors), which correlate information from lower level sensors in order to deduce some higher level state, could detect more complex situation data.

Some context-aware document applications have been developed. Peter Brown [3] provides a framework for discrete context-aware applications and services based on a simple *Post-It note* metaphor, where discrete pieces of information are attached to individual contexts, to be triggered when the user enters those contexts. Mobile tour guides [1] have been developed to familiarize a visitor with a new area. Office awareness systems [21, 22] sense users' locations, help people find each other, and keep up awareness. Context-based retrieval applications [16] collect and save context information and support subsequent information retrieval based on context information. Conference Assistant [7], which supports conference attendees and presenters, assists users in taking notes on presentations and aids in the retrieval of conference information after the conference concludes.

Hypertext systems, including the WWW (World-Wide Web), have been widely accepted as document navigation and search tools. Hypertext systems supporting mobile systems will benefit if they also can become *context-aware.* For example, we can think of a university digital library, which wishes to provide a full version of a Web document to its on-campus patrons and a limited one to its off-campus patrons. The library system also might give more detailed documents for reference material located on the current floor than for other floors. Some documents should be made available for general use outside of working hours or when they are no longer secret.

A focal point in adaptive hypermedia systems [2, 4] has been support for user modeling. Adaptive hypermedia systems use knowledge represented in a user model to adapt the information and links being presented to the given user. Adaptive hypermedia systems are mainly used now in educational hypermedia areas where the hyperspace is reasonably large and where individuals with different goals, knowledge and background use a hypermedia application. Providing context-awareness will require user modeling.

In this research, we will introduce extensions to a previously-described hypertext model in order to support context-awareness. Trellis [8], based on colored timed Petri nets [12], is chosen since it provides good facilities for dynamic adaptations and supports formal analysis techniques. This paper is organized as follows. Related work is discussed in section 2. The research objectives and detailed research work are described in section 3. Finally, future work and conclusions are discussed in section 4.

2 Related Work

2.1 Petri Nets

Petri nets [12, 15, 24], as graphical and mathematical tools, provide a uniform environment for modeling, formal analysis, and design of systems. Formally, a basic Petri net is a bipartite directed graph defined as follows:

A Petri net structure is a tuple, <P, T, F>, in which

- $P = \{P_1, P_2, \ldots, P_n\}$ is a finite set of places with $n \geq 0$;
- $T = \{T_1, T_2, \ldots, T_m\}$ is a finite set of transitions with $m \geq 0$ and $P \cap T = \varnothing$;

- $F \subseteq (P \times T) \cup (T \times P)$ is the flow relation, a mapping representing arcs between places and transitions.
  For a marking of a Petri net structure <P, T, F>
- M: $P \rightarrow I$, I = {0, 1, 2, …}, is a function that associates a marking to each place in the net.

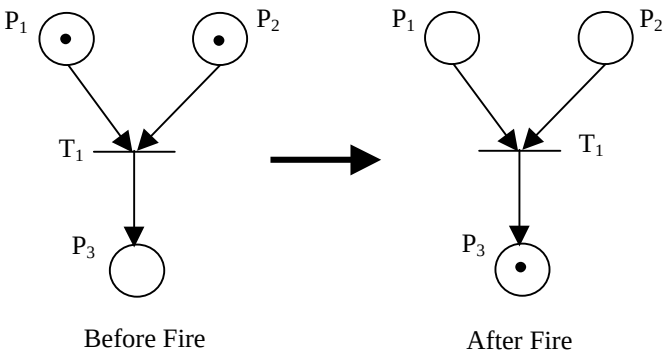


Fig. 1. A Simple Petri Net

Graphically, a Petri net is represented by indicating its places by circles, transitions by bars, arcs by arrows, and tokens by small black dots. A place containing one or more tokens is said to be marked. When each place incident on a transition is marked that transition is enabled. An enabled transition may fire by removing one token from each of its input places and putting one token into each of its output places. For example, consider the marked Petri net shown in Figure 1. The initial marking is M_0 = [110] (i.e., $P_1 = 1$, $P_2 = 1$, $P_3 = 0$) and T_1 is an enabled transition under M_0. If T_1 were fired, the resulting next state would be M_1=[001], as shown in the right side of the Figure.

The basic Petri net is not always convenient for representing and analyzing complex systems because tokens in the basic Petri nets have no identity. In order to overcome this problem, Petri nets that allow tokens to have distinct identity, called high-level Petri nets, were proposed. These nets include predicate-transition nets [9], colored Petri nets [12], object-oriented Petri nets [10], and others. In high-level Petri nets, a token can be a composite object carrying data, which may be of arbitrary complexity including integers, reals, text strings, records, lists and tuples. Nonetheless, it should be noted that ordinary and high-level Petri nets have the same descriptive power, even though high-level Petri nets supply much better structuring facilities than basic nets [24]. A more thorough exposition of net theory can be found in the texts by Peterson [15] and Jensen [12].

2.2 Trellis

The Trellis project [8, 18] has investigated the structure and semantics of human computer interaction, in the context of hypertext (hypermedia) systems, program browsers, visual programming notations, and process models. The Trellis model is

formally defined using colored timed Petri nets as the structure of a hyperprogram, and this gives the model an elegant structure that can be both programmed and analyzed [24].

In the form of colored timed Petri nets (CPN) used by the Trellis model, tokens have color types and a token of one color is discernible from a token of another color. However, within a color class individual tokens cannot be distinguished from one another. Each transition in a Trellis net has two time values, representing respectively a delay and a timeout. Time values in Trellis are thought of as defining ranges for the availability of an event. Formally, a colored timed Petri net in the Trellis model is defined as follows [18]:

A CPN is a tuple, $<\Sigma, P, T, F, \tau>$, in which

- Σ is a finite set of token colors, such as {black, maroon, …};
- P, T, and F are as defined earlier ;
- $\tau : T \rightarrow \{0,1,2, \ldots\} \times \{0, 1, 2, \ldots, \infty\}$ is a function mapping each transition to a pair of values termed release time and maximum latency respectively. For any transition $t \in T$, we write $\tau(t) = (\tau^r, \tau^m)$ and we require that $\tau^r \leq \tau^m$.

A Trellis net is completed by annotating for the components of the CPN. We can conceive of the CPN as the task description and the different annotations as the information required by the task. One important category of annotation is content. Fragments of information (text, graphics, video, audio, executable code, and other hyperprograms) are associated with the places in a CPN. Another category of annotation is events, which are mapped to the transitions of the CPN. A third category of annotation is the attribute/value (A/V) pair; a list of A/V pairs is kept with each place, each transition, each arc, and for the CPN as a whole. Annotations can be used by the Trellis implementation to build client tools, such as hypertext documents browser and authoring tools. Formally, a hypertext in the Trellis model is defined as follows [18]:

A hypertext in the Trellis structure is a tuple, $<CPN, M_0, D, W, B, P_l, P_d>$, in which

- CPN is a colored timed Petri net $<\Sigma, P, T, F, \tau>$;
- $M_0 : P \rightarrow$ token instances of Σ is an initial marking (or initial state) for CPN;
- D is a set of document contents;
- W is a set of windows;
- B is a set of buttons (or links);
- P_l is a logical projection for the document, $P_l = <D_l, W_l, B_l>$, mappings from components of a Petri net to the human-consumable portions of a hypertext;
- P_d is a display projection for the document, a collection of mappings that associate the logical buttons and windows of a hypertext with physical screen representations and locations.

3 caT

With the increased availability of mobile personal computers, in modern hypertext systems, adaptation support for users in dynamically-changing environments will be essential to meet the needs of mobile users. The main goal of this research is to continue the development of caT [14], a context-aware hypertext model and associated tools. Beginning with the χTrellis implementation by Stotts, et al., the following main research issues are being explored.

1. A context-aware hypertext model: Even though previously-described hypertext models provide a powerful modeling framework, they have a weakness in supporting context-aware adaptation, mainly due to a lack of incorporation of dynamically-changing information from the external environment. We need a new context-aware hypertext model that supports context-awareness in dynamically-changing environments.
2. Fuzzy knowledge handling: By introducing the fuzzy logic [23] concept into a hypertext model, the model can have the ability to handle uncertain knowledge (i.e., fuzzy context) in real world applications. Fuzzy logic primarily is motivated by observing that human reasoning can use concepts and knowledge that do not have well-defined, sharp boundaries (i.e., vague concepts). For example, the new model may support the following link display condition: *"shows a link when access right of the current user is rather high"*. In this case, the model needs to use a fuzzy rulebase for inferring the *access right* value from the current user's information (e.g., access time and location).
3. Flexible information presentation: The Trellis model supports good separation between document specification and presentation. This allows multiple presentations of a particular document specification. To provide a WWW-based presentation, which has been widely accepted recently, we need to specify how active content elements and links of the Trellis model are to be displayed and embedded in WWW pages, respectively. The currently-existing χTrellis prototype uses a separate window (or frame) to display the net's active content and links, but in the WWW context it is necessary to have a mechanism that combines these elements into one larger document. A composition mechanism, based on the composite node concept introduced in the Dexter model [11], is explored for the presentation of multiple active content elements into one flexible (or dynamic) document.

To address these requirements, we are developing caT, which supports flexible user (or agent) adaptation to changing environments by incorporating context-awareness, user modeling, and fuzzy knowledge handling features to the current Trellis system. caT also supports Web-based browsing to enable the high usability of the model. The detailed methods of the suggested research work are described in the following subsections.

3.1 Context-Aware Hypertext Model

The foremost objective of this research work is to develop a context-aware hypertext model. Beginning with the Trellis model, the following features are added:

1. Structured tokens: Each colored token caries its own local variable/value pairs. Thus, it can represent dynamically-changing characteristics, such as access time and access location. Providing individual identity to tokens requires a mechanism to determine what values should be used if tokens are combined or replicated during transition firing. In caT, these transformations are specified through predicates associated with the outgoing arcs from the transition. For hypertext applications, each colored token may represent each person who is in a dynamically-changing environment. With this token information, caT can provide different behavior to the user under different contexts (e.g., different access times, such as morning and evening, spring and winter, etc.). Users can share the same Petri net, but have context-aware, customized views from the net.
2. User modeling: Each colored token also can have a link to a user-modeling profile that contains the user's information (e.g., preference, background, etc.). caT can use this user modeling information in addition to local token values to customize its behavior to different users. The distinction between a token's local values and the user model profile is that the local values are expected to reflect dynamic and environmentally changing data, while values in the user modeling profile are less dynamic, reflecting user profile data. In addition, the user model profile is globally visible, while the local variables are associated with an individual token. For context-aware applications, user modeling will help supplement the limits of current sensor devices. In the real world, some sensor devices are not available to common users, and it is impossible to get some kinds of environmental data from sensor devices, such as a user's current working organization type. User modeling can address these shortcomings by enabling derivation of missing values.
3. Link adaptation: Conditional statements attached to transitions are evaluated with values from the current user model and token; conditional statements determine the threshold values for transition firing, and thus provide link adaptation. Additionally, assignment statements attached to output arcs are used for changing current token values, and function calls for getting environmental data or invoking a fuzzy inference engine are supported in conditional and assignment statements.

   The formal definition of Petri net used in the caT model is as follows:

   A caT Petri net structure is a tuple, $<\Sigma, P, T, F, \tau, C, G, E>$, in which
   - Σ is a finite set of token types, called color sets;
   - P, T, F, and τ are as defined earlier;
   - $C : P \rightarrow \Sigma$ is a color function;
   - $G : T \rightarrow$ *Boolean Expression* is a guard function;
   - $E : (T \times P) \rightarrow$ *Arc Expression* is an arc expression function.

The caT model has additional (or enhanced) properties (or functions) in addition to the common ones in the Trellis model. In Σ, in caT, each token can have a color value and optional local token variables. Therefore, each place can have different type of

tokens (i.e., tokens with different local variables), declared by "$C : P \rightarrow \Sigma$". The color function, C(p), maps each place to a color set (type), and each token on the place must have a token that belongs to the color type. The guard function, G(t), is used for mapping a Boolean expression to each transition, which specifies an additional constraint (threshold) which must be fulfilled before the transition is enabled. The arc expression function, E(t×p), is used for mapping an assignment expression to each output arc, which changes current token values when the transition is fired.

3.2 Fuzzy Logic Tool Integration

In context-aware applications, we need to handle uncertain user contexts. Consequently, caT incorporates a fuzzy logic engine for evaluating conditional statements used for link adaptation as well as for updating values in local variables. Matlab's Fuzzy Logic Toolbox [13] is used as caT's fuzzy logic engine and is invoked on transition firing. Some fuzzy Petri nets [5] use fuzzy tokens, but keeping the current Petri net model and invoking an external fuzzy logic engine when necessary seems like a simple and reasonable solution, rather than introducing fuzzy logic into the current model and making a complex fuzzy Petri-net-based model.

A function for invoking an external fuzzy logic engine is supported in caT. The first argument is a rulebase name and the following arguments are either local token values or user profile values. For example, the user's access right is inferred by invoking the external fuzzy logic engine with a target rulebase name, user's access time, and current location. A sample rulebase for this purpose is as follows:

1. *If (time is day) and (distance is close) then (accessRight is high) (1)*
2. *If (time is day) and (distance is middle) then (accessRight is middle) (1)*
3. *If (time is day) and (distance is far) then (accessRight is low) (1)*
4. *If (time is not day) then (accessRight is high) (1)*

Generally, the rulebase infers an access right value from the current user's access time and distance. When the current time is *day*, users who are in close distance get high access rights. But when the current time is not *day*, all users get high access rights, regardless of their distance. Fuzzy terms, such as *day*, *close*, *middle*, and *far*, are defined by membership functions; a membership function is a curve that defines how each point in the input space is mapped to a membership value (or degree of membership) between 0 and 1.

3.3 Flexible Information Presentation Tool

In caT, a template file specifies how active content elements are to be displayed and how links are to be embedded. The template file is simply another Trellis content type associated with a place. It takes control when the place is marked; consequently multiple template files can be active at different times during a browsing session. The node that contains the template file can be viewed as a virtual composite node constructed from several atomic nodes. Only active atomic nodes in the composite node are used for generating the current presentation of the composite node. Thus, the

content of the composite node changes dynamically based on the current net state.

As an example, consider the case of a university digital library, which provides a full version of a Web page to its on-campus patrons and a limited one to its off-campus patrons. However, faculty members off-campus retain access to the full version. Because the page includes access to a real-time help desk, only the limited version is available outside of normal operating hours. For exemplifying the above scenario in the caT model, some of the current existing library services at Texas A&M University are used. Figure 2 shows a small net fragment implementing these restrictions. The top portion of the net, invoked automatically by caT's time specifications, invokes the fuzzy inference engine to classify the user as being "on-campus" or "off-campus" based on the time of day, distance from campus, access rights, and incoming IP address. The bottom portion of the net specifies the information display in the Web browser, shown in Figure 3. The detailed behavior of the Petri net in Figure 2 is described in the following subsections.

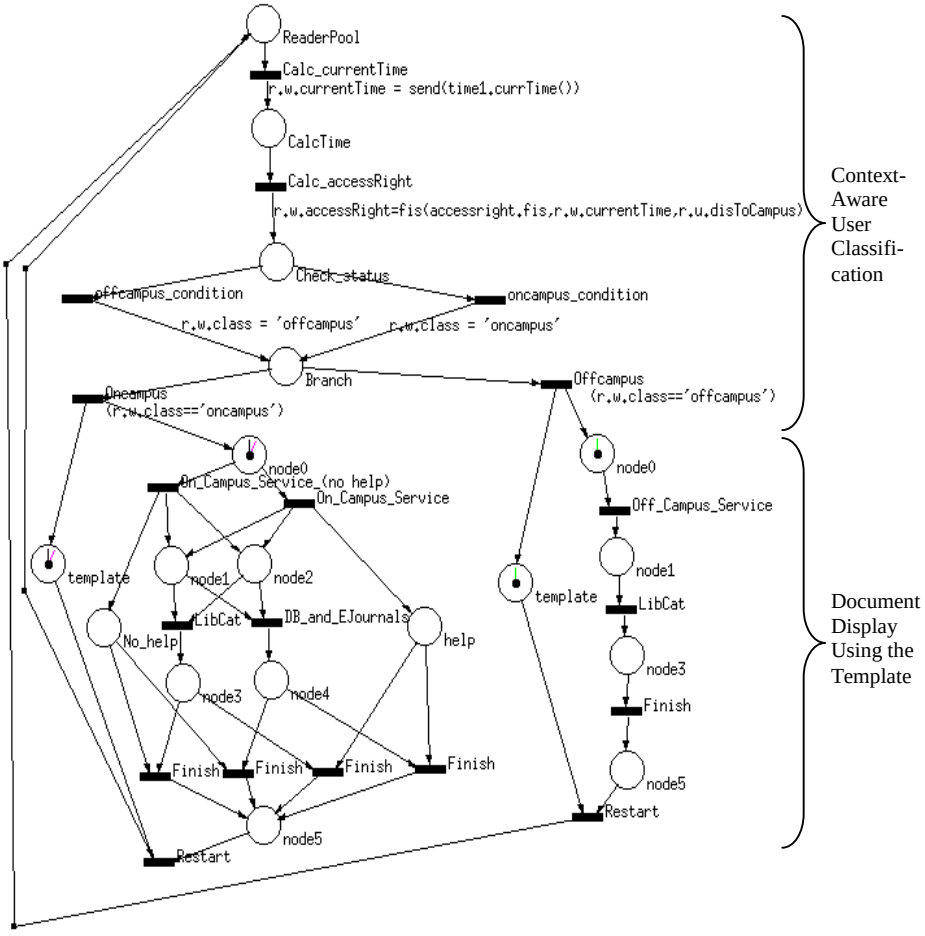


Fig. 2. Petri net of a simple digital library tour

Context-Aware User Classification. Initially, user tokens will be placed in *ReaderPool* place. Each colored token has the following local token variable/value pairs: *user name, accessRight, user class, network,* and *currentTime*. The *user name* and *network* values will be marked initially, and the other values are inferred or calculated by the system. A user profile for each user has the following values: *disToCampus* (distance from a current location to campus) and *userType* (user occupation).

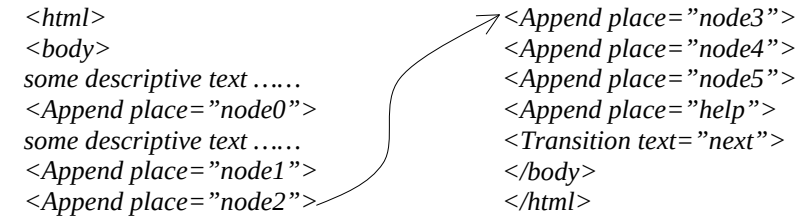
The *Calc_currentTime* transition has a timing value of (0,0). (0,0) means no delay and no timeout, therefore the *Calc_currentTime* transition will be executed immediately by the system when it is enabled. When the *Calc_currentTime* transition is fired, the *currentTime* value is calculated by using an expression that is attached to the output arc of the *Calc_currentTime* transition. "*r.w.currentTime"* is used for accessing a local token variable *currentTime* of the current colored token: "*r*" is a color variable that is instantiated to the current color value, and "*w*" denotes that the following variable *currentTime* is a local token variable. "*send()*" function is used for calling system-support functions. When the *Calc_currentTime* transition is fired, a colored token in *ReaderPool* caries its own local variable/value pairs to the output place, i.e., *CalcTime*, with a new *currentTime* value.

In the next step, the *Calc_accessRight* transition is fired, and the user's *accessRight* value is inferred by invoking an external fuzzy logic engine. *"fis()"* is a function for invoking an external fuzzy logic engine. The *accessright.fis* rulebase is the one introduced in section 3.2. "*r.u.disToCampus"* is used for accessing the user profile variable *disToCampus* of the current token user. *"u"* denotes that the following variable *disToCampus* is a user profile variable. The user profile name is stored in the current token's *user name* variable.

When a token arrives at the *Check_status* place, either the *offcampus_condition* or the *oncampus_condition* transition is enabled after each condition statement attached to its transition is evaluated with the current token and user profile values. The condition in *offcampus_condition* is *"(r.w.accessRight < 0.8 && r.w.network != '128.194.147' && r.u.userType != 'faculty')";* The condition in *oncampus_condition* "*(r.w.accessRight >= 0.8 || r.w.network == '128.194.147' || r.u.userType == 'faculty')"*. When the current token's inferred *accessRight* is larger than or equal to 0.8 (i.e., rather high), or *network* is '128.198.147', or *userType* is *'faculty'*, the *oncampus_condition* transition is enabled; otherwise, the *offcampus_condition* transition is enabled. When the transition is fired, the assignment statement (*r.w.class = 'oncampus'* or *r.w.class = 'offcampus'*) attached to output arc of the transition is executed.

After the user is classified, a token is in the *Branch* place. A token with the user class value '*oncampus' (*or '*offcampus')* has the *Oncampus* (or *Offcampus*) transition enabled and fired. Now the user (on-campus or off-campus) will have tokens in both *template* and *node0* places, and the user is ready for a guided digital library tour. Up to this point, all transitions are executed by the system without interaction with the user.

Document Display Using the Template. Each classification includes a separate template file to display a full version of the page to the on-campus user and a limited one to the off-campus user. The content of "template" node for on-campus users is as follows:



This template file is used for combining contents of active nodes (nodes that have tokens) specified in an <*Append*> construct. When only *node0* and *template* places are active, the content of *node0* is used for generating the current Web page. Then, when *node1, node2* and *help* including *template* are active (when the next transition *On_Campus_Service* of the on-campus user, who accesses the net during regular office hours, is fired), the contents of these nodes are used for generating the current Web page. Also when the on-campus user accesses the net at a non-regular office hour, the content of *help* is not used for generating the output Web page. Thus, only the limited version (no on-line help) is available outside of normal operating hours.

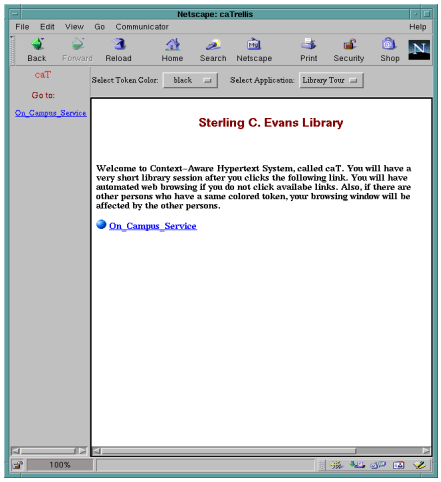
The *template* node for off-campus users may have the same template file as on-campus users. But because there are no *node2, node4,* and *help* in the subnet of off-campus users, only *node0*, *node1*, *node3*, and *node5* are used for generating the output Web page. Therefore, off-campus users will have a limited version compared to on-campus users. When tokens move around the net, the contents of the Web page change dynamically.

A <*Transition*> construct is used for defining an embedded link that allows the user to move to the next state. If the output transition of the *template* node is enabled, that transition link is included in the combined Web page. This enables the user to fire the transition to move to the next state. Possible internal links, such as links between *node0* and *node5*, are generated automatically in the combined Web page if these are enabled transitions.

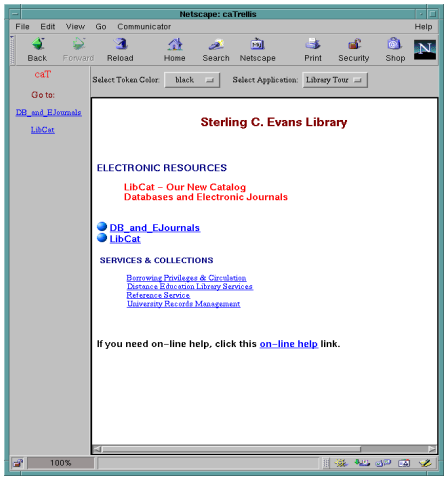
For the detailed behavior of the bottom portion of the net, consider the case of on-campus users who access the net during regular hours. The *On_Campus_Service* transition has a condition statement *"r.w.currentTime >= 9.0 && r.w.currentTime <= 17.00"*; *Off_Campus_Service(no help)* transition has *"r.w.currentTime < 9.0 || r.w.currentTime > 17.00"*. Therefore, when the user accesses the net during regular office hours, *On_Campus_Service* will be enabled, otherwise *On_Campus_Service(no help)* is enabled. The resulting page is shown in Figure 3 (a). Introductory text (the content of *node0*) and the *On_Campus_Service* link are displayed in a main frame, and a duplicate *On_Campus_Service* link in a left control frame. Since *On_Campus_Service* has a time value $(0,\infty)$, the system will wait for the user's input (i.e., a mouse click). When a time value is not declared, $(0,\infty)$ is the default value.

When the *On_Campus_Service* transition is fired, the contents of *node1, node2,* and *help* are used for generating the current Web page (see Figure 3 (b)). *LibCat* and *DB_and_Ejournals* links are displayed with additional text and WWW links, and the

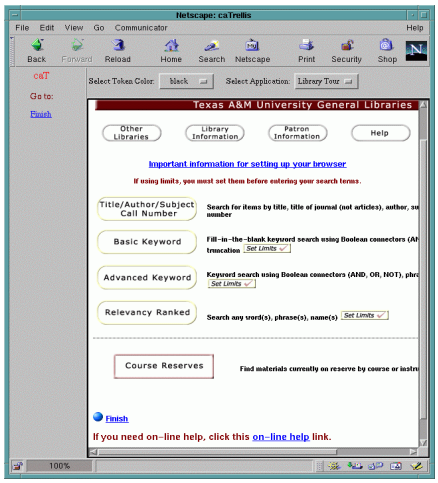
user can select one of these links for the next navigation. The caT links that are mapped to active transitions in the net (called “caT links” in this section for distinguishing them from WWW links) is distinguished with a preceding circle bullet, and normal WWW links are shown without preceding bullets. When the user clicks caT links, the corresponding transition is fired, the net state is changed, and the new content is regenerated using *template*. When the user clicks WWW links, the resulting effect is the same as normal WWW browsing semantics, but the user still can move to the next state using caT links placed in the left control frame. Since *LibCat* has a time value (0,20) and *DB_and_Ejournals* (0,∞), *LibCat* will be selected automatically by the system unless the user responds within 20 time units (i.e., seconds).



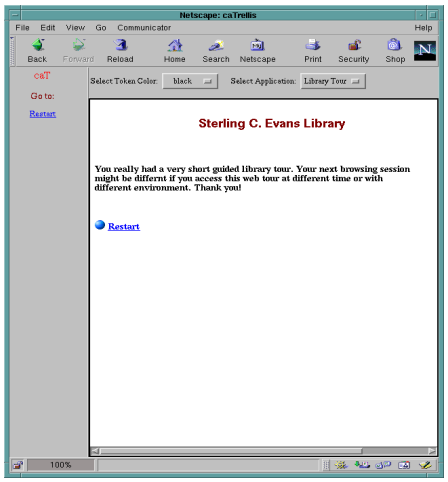
(a) Initial page



(b) After *On_Campus_Service* clicked



(c) After *LibCat* clicked



(d) After *Finish* clicked

Fig. 3. Context-aware display

When *LibCat* is clicked, LibCat (the catalog of Texas A&M University General Library) will be shown (see Figure 3 (c)). Now the user can search for library items using the LibCat search engine. Since the *finish* link has a time value (0,60), the user has to finish his (or her) search work within 60 time units, otherwise the user's window will be forced to move to the next state window (see Figure 3 (d)). The time value can easily be adjusted to a reasonable value by an author of this net, or, indeed, adjusted dynamically based on the reader's performance [19]. When the *restart* link is clicked, the user will be reclassified as being an "on campus" or "off campus" user based on new environmental values: time of day, distance from campus, and incoming IP address (in current example, only time of day is dynamically changing).

Implementation. In the caT implementation, beginning with χTrellis for the X-windows environment, a distributed client/server network achieves cooperative separation between net and interpretation. Every caT model is an instance of an information server – an engine that receives remote procedure call (RPC) requests for its services. Clients are separate processes that have visual user interfaces and communicate with one or more engines via RPC. The Trellis browser has been implemented in C, C++, and Motif, and it runs in a Unix environment. A new Web version of the browser has been designed and implemented using Java language: the Java applet first communicates with an intermediate message-handling server, and the server passes RPC requests using Java JNI (Java Native Interface) to the information servers. The general architecture of the new browsing tool is shown in Figure 4.

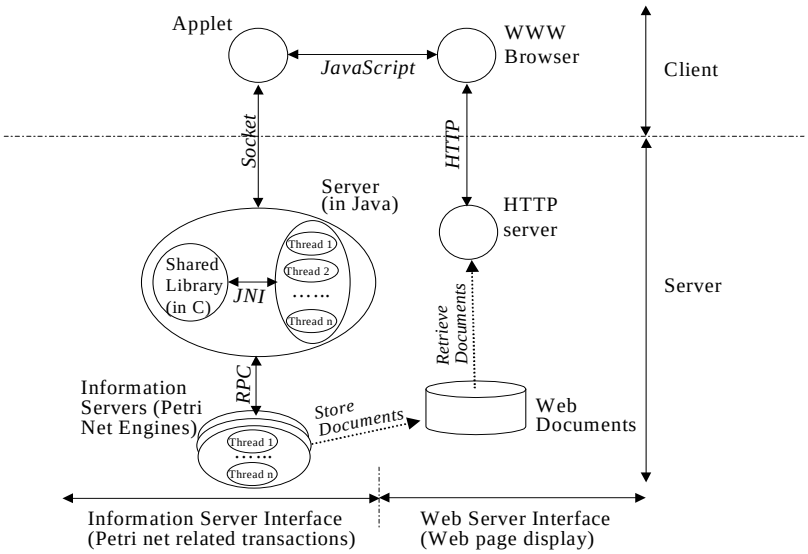


Fig. 4. General architecture of a flexible information tool

4 Discussion

One of the primary advantages of using Petri net models is that the same model is used for the analysis of behavioral properties and performance assessment, as well as for systematic building of systems [24]. Therefore, authors can use Petri net analysis techniques to verify and validate the behavioral characteristics of developed hypertext systems before they are handed over to users [20]. Firstly it is possible to verify that all nodes in a hypertext can be reached via some path; more important, it also is possible to verify that certain nodes cannot be reached from particular initial markings, giving the basis for access control. Additionally, the following characteristics may be verified (or simulated): terminal state existence (i.e., if a state *m* exists in which no transitions are enabled), maximum or minimum time for a specific document navigation (using timing statements in transitions: a release and a maximum latency times), certain collections of information that can be viewed simultaneously, etc. To support these features, the extended model has been designed to be consistent with Petri net theory. However we expect that dynamic characteristics of environment data including user-modeling data may make the verification of the system difficult. For example, the combinations of many dynamic data may generate very large net states. We plan to analyze the system with only a small amount of dynamic data considered at one time. We are developing analysis tools to be added to the current authoring tool, so authors can verify behavioral characteristics of developed hypertext documents.

Additionally, top-down or bottom-up structured authoring support will be essential for developing large-sized Petri net applications since it reduces graphical complexity problems that are common to graph-based modeling tools. Current Trellis supports hierarchical net browsing using a hyperprogamming feature, but it does not support built-in hierarchical nets. In other words, when a token arrives into a place where another net (i.e., a subnet) is declared to be invoked, a browsing tool can only handle the subnet invocation using a content attribute of a place, and show the output interface of the subnet in a new process. But an information server itself does not invoke the subnet since it ignores the content attribute of the place. Moreover, there are no closely coupled interactions between hierarchical nets, such as data value (or token value) passing. Shifting hierarchical net handling from content-level (i.e., a browsing tool) to net structure-level (i.e., an information server) will be more appropriate for flexible net interactions. When data passing between nets is supported, subnets can be used as templates, like functions or procedures in high-level programming languages. This will increase reusability of subnets. The structure-level hierarchical net feature will be very useful for structured authoring, and also will support easy system tracing for simulation and debugging purposes.

In summary, we have introduced caT, a context-aware hypertext model and associated tools, which supports flexible adaptation in a dynamically-changing environment. caT is an extension of the earlier Trellis model in the following ways:

- It supports both high-level Petri net and user modeling features to provide flexible context-aware adaptation,
- It integrates a fuzzy logic tool with the current model to support fuzzy (or uncertain) knowledge handling, and

- It supports a Web-based browsing tool.

The successful result of the current work shows the potential usability of context-aware hypertext systems. Potential target applications of this model are context-aware hypermedia applications running on mobile systems with attached sensors to capture their environment. The research is continuing in order to enhance the usability of the caT model.

References

1. ABOWD, G.D., ATKESON, C.G., HONG, J., LONG, S., KOOPER, R. and PINKERTON, M. "Cyberguide: A Mobile Context-Aware Tour Guide", *ACM Wireless Networks 3,* (1997), pp. 421-433
2. DE BRA, P., HOUBEN, G. and WU, H. "AHAM: A Dexter-based Reference Model for Adaptive Hypermedia", *Proceedings of the 10th ACM Conference on Hypertext and Hypermedia,* Darmstadt, Germany, ACM Press, (1999), pp. 147-156
3. BROWN, P.J. "Triggering information by context", *Personal Technologies,* vol. 2, no. 1, (September 1998), pp. 1-9
4. BRUSILOVSKY, P. "Methods and techniques of adaptive hypermedia", *User Modeling and User Adapted Interaction,* vol. 6, no. 2-3, (1996), pp. 87-129
5. CARDOSO, J., VALETTE, R., and DUBOIS, D. "Fuzzy Petri Nets: An Overview", *13th IFAC World Congress,* San Francisco USA, (30 June – 5 July 1996), pp. 443-448
6. DEY, A.K. and ABOWD, G.D. "Toward a Better Understanding of Context and Context-Awareness", *Georgia Institute of Technology, GVU Technical Report GIT-GVU-99-22,* (June 1999)
7. DEY, A.K., SALBER, D., ABOWD, G.D., and FUTAKAWA, M. "The Conference Assistant: Combining Context-Awareness with Wearable Computing", *The Third International Symposium on Wearable Computers,* (1999), pp. 21-28
8. FURUTA, R. and STOTTS, D. "Trellis: a Formally-defined Hypertextual Basis for Integrating Task and Information", In Olson, G.M., Smith, J.B., and Malone, T.W., editors, *Coordination Theory and Collaboration Technology,* to appear in 2001
9. GENRICH, H.J. and LAUTENBACH, K. "System modeling with high-level Petri nets", *Theoret. Comp. Sci.,* vol. 13, (1991), pp. 109-136
10. GUDWIN, R. and GOMIDE, F. "Object networks – a modeling tool", *Fuzzy Systems Proceedings, 1998 IEEE World Congress on Computational Intelligence, The 1998 IEEE International Conference on Volume: 1,* (1998), pp. 77-82
11. HALASZ, F. and SCHWARTZ, M. "The Dexter Reference Model", *Proceedings of NIST Hypertext Standardization Workshop,* (1990), pp. 95-133
12. JENSEN, K. "Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use Volume 1", *EATCS Monographs on Theoretical Computer Science,* Springer-Verlag, (1992)
13. The MathWorks, Inc. "Fuzzy Logic Toolbox User's Guide", (1999)
14. NA, J. and FURUTA, R. "Context-Aware Hypermedia in a Dynamically-Changing Environment, Supported by A High-Level Petri Net", short paper, *Proceedings of Hypertext '2000,* San Antonio TX, USA, (2000), pp. 222-223
15. PETERSON, J.L. "Petri Net Theory and the Modeling of Systems", *Prentice-Hall,* Englewood Cliffs, N.J., (1981)
16. RHODES, B.J. "The Wearable Remembrance Agent", *Proceedings of 1st International Symposium on Wearable Computers, ISWC '97,* Cambridge, MA, (October 1997), IEEE Press, pp. 123-128

17. SCHILIT, W.N., ADAMS, N., and WANT, R. “Context-aware computing applications”, *Proceedings of the Workshop on Mobile Computing Systems and Applications*, Santa Cruz, Ca., IEEE Computer Society Press, Los Alamitos, Ca., (1994), pp. 85-90
18. STOTTS, P.D. and FURUTA, R. “Petri-net-based hypertext: Document structure with browsing semantics”, *ACM Transactions on Information Systems*, vol. 7, no. 1, (Jan. 1989), pp. 3-29
19. STOTTS, P.D. and FURUTA, R. “Dynamic Adaptation of Hypertext Structure”, *Proceedings of Hypertext ’91*, (1991), pp. 219-231
20. STOTTS, P.D., FURUTA, R., and CABARRUS, C.R. “Hyperdocuments as Automata: Verification of Trace-Based Browsing Properties by Model Checking”, *ACM Trans. On Information Systems*, vol. 16, no. 1, (1998), pp. 1-30
21. WANT, R., HOPPER, A., FALCAO, V., and GIBBONS, J. “The Active Badge Location System”, *ACM Transactions on Informations*, vol. 10, no. 1, (1992), pp. 91-102
22. WANT, R., SCHILIT, B., ADAMS, N., GOLD, R., PETERSON, K., ELLIS, J., GOLDBERG, D., and WEISER, M. “The PARCTAB Ubiquitous Computing Experiment”, *Technical Report CSL-95-1, Xerox Palo Alto Research Center*, (1995)
23. ZADEH, L.A. “Knowledge representation in fuzzy logic”, *IEEE Trans. Knowledge and Data Engineering*, vol. 1, no. 1, (1989), pp. 89-100
24. ZURAWSKI, R. and ZHOU, M. “Petri Nets and Industrial Applications: A Tutorial”, *IEEE: Transactions on Industrial electronics*, vol. 41, no. 6, (December 1994)