OREGON STATE UNIVERSITY COMPUTER SCIENCE DEPARTMENT

HelpDez: Colored-Petri-Net-Based Hypermedia Help System Designer

Huan Chao Keh
T. G. Lewis
Computer Science Department
Oregon State University
Corvallis, OR 97331-3902

90-60-7

HelpDez: Colored-Petri-Net-Based Hypermedia Help System Designer

Huan Chao Keh
T. G. Lewis

Computer Science Department
Oregon State University
Corvallis, OR 97331
ph. (503) 737-3273
CSNet: keh@mist.cs.orst.edu
lewis@mist.cs.orst.edu

ABSTRACT

This paper describes HelpDez, a colored-Petri-net-based approach for the design, simulation, and construction of hypermedia help systems. The storage model, system capabilities, and user interface model of a hypermedia help system are examined. Colored Petri nets are chosen to model hypermedia help systems due to their graphical expressiveness and their applicability to concurrent, asynchronous systems. Also, they make it easy to carry out general validations using previously developed analysis techniques. Furthermore, colored Petri nets allow concurrent browsing paths of a hypermedia help system to be described by a common subnet, without losing the ability to distinguish between them. A formal definition of the HelpDez model based on the colored Petri nets is presented. This model represents the structural properties of the help information (text, graphics, and sound), as well as specifications of the execution semantics associated with hypermedia help systems. The graphical representation of the colored Petri nets provides a powerful tool for specification of the information users can see and hear simultaneously when a hypermedia help system is executed.

1. INTRODUCTION

On-line help systems have been used extensively and with mixed success in a number of environments [11, 13]. As computer systems and applications become more powerful and complex, and bitmapped high-resolution workstations gain in popularity, the demand for more advanced on-line help systems increases. Traditional text-based help information is no longer sufficient to explain complex actions [10]. More advanced on-line help systems should also support the display of context sensitive graphics, animations, and/or other kinds of information, while providing powerful structuring and access mechanisms for browsing this help information. Consequently, the development of advanced on-line help systems consumes a great part of the efforts of application developers and represents a significant portion of the resulting code [4]. Tools that can be used to aid on-line help facility design and implementation are clearly useful and needed.

To support an advanced on-line help system, many different kinds of data (text, graphics, audio) need to be stored and retrieved. The hypermedia approach has been successfully applied to information management in many different application areas [2]. It provides a powerful mechanism for structuring different kinds of data so that users can see and hear the level of detail appropriate to their current context, while simultaneously providing the freedom to browse other portions of the database. Hypermedia has the capability of providing an excellent structuring mechanism and user interface model for advanced on-line help systems.

Petri nets have gained popularity over the past few years as a formal model for concurrent and asynchronous systems, given their graphical expressiveness and the capability of carrying out general validations using previously developed analysis techniques. Stotts and Furata associated the original connection between Petri nets and hypertext [12]. They describes a model for an authoring and browsing prototype called

αTrellis in which Petri nets are used to specify both the linked form and the browsing semantics of a hypertext. However, the use of Petri nets to describe a system containing classes of objects with a similar behavior often leads to a very large net which is difficult to analyze. This is particularly true of a hypermedia on-line help system. For example, to support the backup capability of a hypermedia help system, concurrent browsing paths representing navigation through different help topics must be distinguished and represented in separate subnets. This results in a number of subnet duplications representing the same help information. One response to this problem has been to generalize ordinary Petri nets as colored Petri nets, allowing process descriptions by a common subnet without losing the ability to distinguish between the different processes [3]. Colored Petri net descriptions are often more concise and suitable for the mathematical analysis of described systems.

HelpDez, a colored-Petri-net-based hypermedia help system designer, provides a suitable environment for the design, simulation, and construction of hypermedia help systems. Although HelpDez can be used as a stand-alone hypermedia authoring and browsing system, its primary purpose has been to aid in the design and implementation of on-line help systems integrated with applications generated with the Oregon Speedcode Universe (OSU) software development system [5]. HelpDez creates a logical separation between the on-line help system and the application, allowing programmers to independently develop and maintain an on-line help system as an adjunct to application development.

In this paper the storage model, system capabilities, and user interface model for the proposed hypermedia help system (HHS) are described in Section 2. In Section 3 both ordinary Petri nets and colored Petri nets are briefly reviewed and the proposed HHS is formally defined in terms of colored Petri nets. Section 4 includes a description of HelpDez and its components, while Section 5 provides a justification of the colored-Petri-net-based approach to HHS design. Section 6 concludes this paper.

2. HHS

The HHS database is a collection of hypermedia nodes, each consisting of two files. One file contains the visible information (text and graphics) and the other file contains the audio information (recorded sound). HHS maintains links and nodes separately, thus allowing easily shared help information on multiple on-line help systems.

Based upon the hypermedia concept, HHS is context sensitive, supporting help screen displays associated with the context currently in use by the user. For example, when a user selects an object in an application and then invokes the "help" command, the help screen(s) related to the object selected should be displayed. When no single object is selected, the user will be shown the application's top level help screen(s). When more than one object is selected, HHS must provide a mechanism to allow the user to simultaneously view all of the help screens related to the selected objects . It also allows users to traverse the help screens to view and hear information using hypermedia access mechanism. Furthermore, HHS must provide a backup capability which will allow a user to return to help screens previously viewed. There can be any number of textual and/or graphical hotspots on each help screen which the user can click with the mouse to get to other help screens in the system. These hotspots represent hyper-text/graphics links where a user can get help. Hotspots are visually distinguished so the user can in each help screen easily determine where additional help is available. When one help screen is not adequate to fully describe a concept, either a "more" hotspot should be provided for user selection to get the next help screen or multiple screens should be displayed simultaneously, dependent upon the designer's choice. The way HHS normally works is that every time the user views a new help screen by selecting a hotspot, HHS removes the previous screen(s).

Although help screens may contain any combination of text and graphics items, several conventions are useful in order to provide a common help screen format [6]. Figure 1 is an illustration of these conventions.

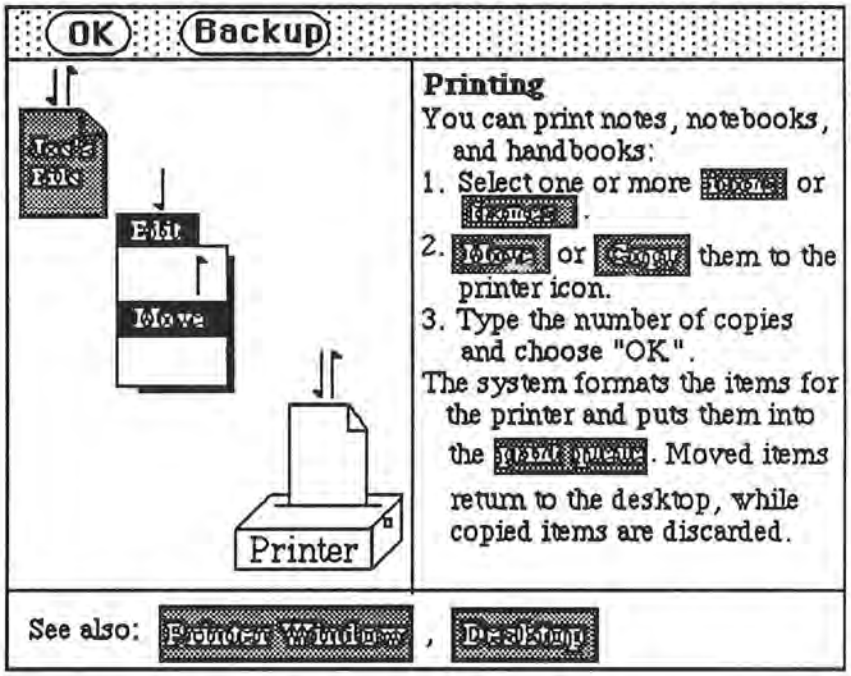

Figure 1. Example Help Screen

The help screen consists of a menu bar and three distinct areas which may or may not be shown, dependent upon help screen design. Each area in a given design is separated by vertical or horizontal lines. The menu bar contains two buttons. Each button has a predefined meaning initiating the same action each time it is selected. The OK button allows the user to remove the help screen(s), losing all backup information related to the removed help screen(s). The backup button provides the user with the ability to retrace previous steps through the help hyperspace and is disabled when the user has reached the topmost help screen(s). The remaining three sections contain, respectively, graphic illustration, a text information of help information explanations, and a section of references to related topics. In Figure 1, hotspots are shown as 50% shaded items. The recorded

sound for the textual information in a help screen is saved in a resource file and the user may also hear the textual information as the help screen is displayed.

3. DEFINITIONS

The HelpDez model is based primarily on a colored Petri net representation of the linked structure and execution semantics of an HHS. To provide a complete presentation of the HelpDez HHS model, definitions for both ordinary and colored Petri nets that are required for the understanding of the HelpDez HHS definitions are first presented, followed by definitions appropriate to the HelpDez HHS model. These definitions will be presented both informally and formally. Note that the definitions given for both ordinary and colored Petri nets are tailored specifically for HHS modeling, and are not entirely general. For a more general theory of ordinary and colored Petri nets, see, respectively, the studies by Peterson [8] and Reisig [9] and the paper presented by Jensen [3].

3.1 Ordinary Petri Nets

A Petri net is represented by a bipartite directed graph. This bipartite directed graph (PN) contains two kinds of nodes, the places (P) and the transitions (T), drawn respectively as circles and bars. These nodes are connected by directed arcs from places to transitions and from transitions to places. A marking (M) is an assignment of tokens to the places of a Petri net. On a Petri net graph, tokens are represented by small dots in the circles representing the places of a Petri net.

Definition: Petri net PN = <P, T ,IO, M> , where
P is a finite and non-empty set of places $P = \{ p_i \}$;
T is a finite and non-empty set of transitions $T = \{ t_j \}$, and $P \cap T = \emptyset$;
IO : $(P \times T) \cup (T \times P) \rightarrow \{0 , 1\}$ is a mapping representing arcs between P and T;
M: $P \rightarrow N$, where N is the set of non-negative integers.

A place p_i is an input (or output) place for a transition t_j iff $IO(p_i, t_j) = 1$ (or $IO(t_j,p_i) = 1$). The marking can also be defined as an n-vector, $M = (m_1, m_2, \ldots, m_n)$, where $n = |P|$ and $m_i = M(p_i)$, representing the number of tokens in the place p_i. The execution of a marked Petri net is controlled by the number and distribution of tokens in the Petri net. A Petri net executes by defining its initial marking (M_S) and firing transitions until either its final marking or a state in which no transitions are enabled is reached. Therefore, the execution of a Petri net results in two sequences: the sequence of markings (M_S, M_1, M_2 ...) and the sequence of transitions which were fired (t_1, t_2, t_3 ...). A transition may fire if it is enabled and is enabled if each of its input places has at least one token.

Definition: A transition $t_j \in T$ in a Petri net PN = <P, T ,IO, M> is enabled iff $(\forall p_i \in P)$ such that $IO(p_i , t_j) = 1$, $M(p_i) \geq 1$.

A transition fires by removing one token from each of its input places, creating a new token which is then distributed to each of its output places.

Definition: A transition t_j in a Petri net PN = <P, T ,IO, M> may fire whenever it is enabled. Firing an enabled transition p_i results in a new marking M' defined by $(\forall p_i \in P)$ $M'(p_i) = M(p_i) - IO(p_i , t_j) + IO(t_j , p_i)$.

For example, consider the Petri net in Figure 2a. With the initial marking M_S = (2,1,1,1,0,1,2,0,0), both transitions t_1 and t_3 are enabled; transition t_2 is not enabled since one of its input places (p5) does not have a token. Firing transition t_1 produces the new marking (1,0,0,1,1,2,2,0,0) shown in Figure 2b. If, instead, transition t_3 fires, the resulting marking (2,1,1,1,0,0,1,0,1) is shown in Figure 2c.

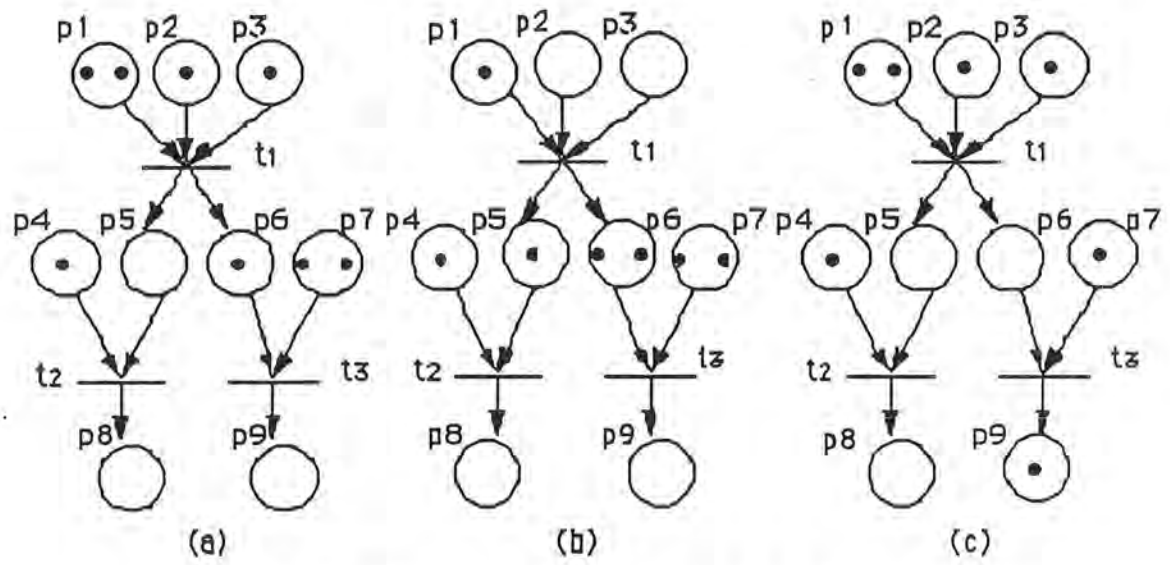

Figure 2. Firing a Transition in a Petri Net

3.2 Colored Petri Nets

A colored Petri net is a generalized Petri net. In colored Petri nets each token has attached a color, indicating the identity of the token. Moreover, each place and each transition has attached a set of colors. A transition can fire with respect to each of its colors. By firing a transition, tokens are removed and added at the input and output places in the normal way, except that a functional dependency is specified between the color of the fired transition and the colors of the involved tokens.

Definition: A colored Petri net CPN = <P, T , C, IO, M> , where
P is a finite and non-empty set of places P = { p_i };
T is a finite and non-empty set of transitions T= { t_j} and $P \cap T = \emptyset$;
C is a color function such that $C : P \cup T \rightarrow 2^{CS} - \emptyset$, where CS is a finite set of colors;
IO is a function with domain $(P \times T) \cup (T \times P)$ such that
$IO(p_i, t_j), IO(t_j,p_i): C(t_j) \rightarrow [C(p_i) \rightarrow \{0 , 1\}]$;
M ,the marking of CPN, is a function with domain P such that $M(p_i) : C(p_i) \rightarrow N$.

Elements of $C(p_i)$ and $C(t_j)$ are called colors. A place p_i is an input (or output) place for a transition t_j iff $\exists c' \in C(t_j), \exists c'' \in C(p_i)$ such that $IO(p_i , t_j)(c')(c'') = 1$ (or $IO(t_j,p_i)$

(c')(c") = 1). All tokens on a place p must be labelled by a color $c \in C(p)$. The marking M of CPN can also be defined as an n-vector $M = (m_1, m_2, \ldots, m_n)$, where $n = |P|$ and $m_i = \{M(p_i)(c) \mid c \in C(p_i)\}$, representing the set of numbers of tokens for each color in the place p_i.

Definition: A transition $t_j \in T$ in a colored Petri net CPN = <P, T, C, IO, M> is enabled iff $\exists c' \in C(t_j), \exists c'' \in C(p_i), (\forall p_i \in P)$ such that $IO(p_i, t_j)(c')(c'') = 1$, $M(p_i)(c'') \geq 1$; and firing the enabled transition t_j results in a new marking M' such that $M'(p_i)(c'') = M(p_i)(c'') - IO(p_i, t_j)(c')(c'') + IO(t_j, p_i)(c')(c'')$.

A transition t in a colored Petri net may fire in several different ways corresponding to the different colors of t. To describe an HHS, the firing rule of a colored Petri net can be stated as follows: A transition t is enabled if each of its input places has at least one token labelled by a color, c, which is one of the colors associated with t. The firing of a transition t, with color $c \in C(t)$, removes one token with color c from each of its input places while adding a new token with color c to each p of its output places, where $c \in C(p)$. For example, consider the Petri net of Figure 3a, where $C(p_1)=C(t_1)=\{c_1\}$, $C(p_2)=C(t_2)=\{c_2\}$, and $C(p_3)=C(p_4)=C(p_5)=C(t_3)=C(t_4)=\{c_1, c_2\}$. With this initial marking $M_S = (\{1\}, \{1\}, \{0,0\}, \{0,0\}, \{0,0\})$, both transitions t_1 and t_2 are enabled. If the firing sequence of transitions is (t_1 with color c_1, t_2 with color c_2, t_3 with color c_1, t_3 with color c_2), the corresponding markings are

({0}, {1}, {1,0}, {0,0}, {0,0})
({0}, {0}, {1,1}, {0,0}, {0,0})
({0}, {0}, {0,1}, {1,0}, {0,0})
({0}, {0}, {0,0}, {1,1}, {0,0})

as shown in Figures 3b, 3c, 3d, and 3e respectively. Two concurrent execution paths sharing a common subnet can be identified from the execution of this colored Petri net. The movement of the token with color c1 represents the execution path corresponding to the firing sequence of transitions (t_1 with color c_1, t_3 with color c_1), and the evolution of

the token with color c_2 describes the remaining execution path with firing sequence (t_2 with color c_2, t_3 with color c_2).

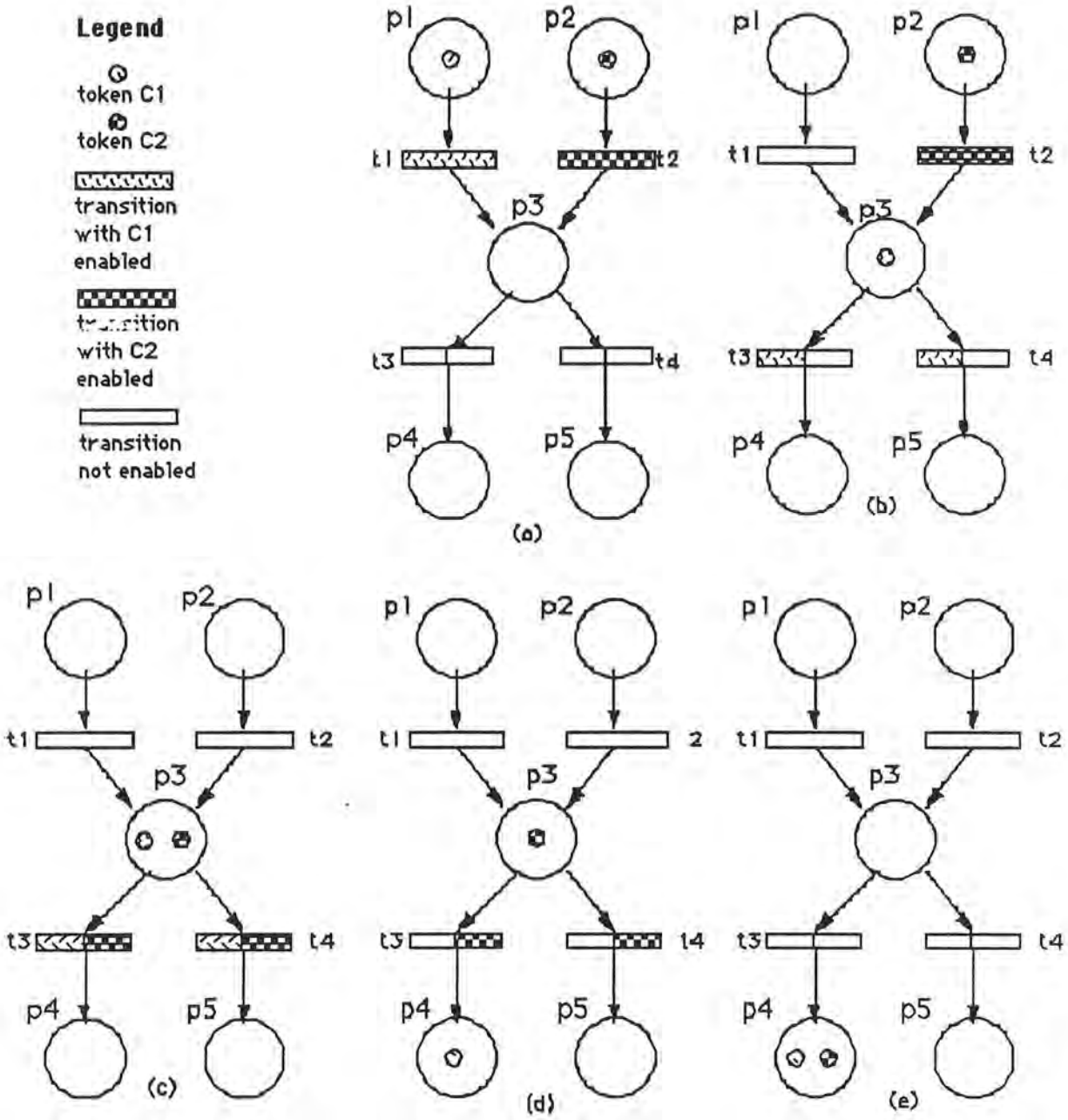

Figure 3. Firing a Transition in a Colored Petri Net

3.3 HelpDez Model of Hypermedia Help Systems

The HelpDez model employs a colored Petri net to specify both the linked structure and the execution semantics of an HHS. The logical structure of the colored Petri net can then be mapped through two collections of mappings onto a displayed form for user consumption. The colored-Petri-net-based HelpDez model permits the separation of the linked structure from the stored help information and allows the same chunk of information to be simultaneously displayed at different locations.

Definition: A hypermedia help system HHS is a 7-tuple
HHS = <CPN, TG, SR, LW, LH, LM, PM>, where
CPN = <P,T, C, IO, M> is a colored Petri net structure;
TG is a set of visible resources (text and graphics);
SR is a set of sound resources;
LW is a set of logical windows (window locations);
LH is a set of logical hotspots;
LM is a collection of logical mappings;
PM is a collection of presentation mappings.

A hypermedia help system consists of a colored Petri net representing the linked structure and execution semantics of the system, four sets of user-perceptible components (visible resources, sound resources, logical windows, and logical hotspots), a collection of logical mappings from the components of a colored Petri net to the user-perceptibe components, a collection of presentation mappings between the user-perceptible components and the presentation mechanisms. Each visible resource from set TG is a binary file containing stored text and graphics. Each sound resource from set SR is a sound resource file containing recorded audio information. A logical window from set LW describes the type and location of a window in which the help information (text and graphics) is to be displayed. A logical hotspot from set LH represents the area of a help screen on which the user can click to get to next help screen.

Definition: Given the colored Petri net CPN = <P, T, C, IO, M> in a hypermedia help system HHS = <CPN, TG, SR, LW, LH, LM, PM>, the collection of logical mappings of HHS is a 4-tuple LM = <MPTG, MPSR, MPC, MTC>, where

MPTG : P → TG;
MPSR : P → SR;
MPC is a function with domain P such that ($\forall$ p $\in$ P) MPC(p) : C(p) → LW;
MTC is a function with domain T such that ($\forall$ t $\in$ T) MTC(t) : C(t) → LH.

LM consists of four mappings between the colored Petri net and the user-perceptible components. The function MPTG associates an element in TG (visible resource) with each place in the CPN; the function MPSR associates an element in SR (sound resource) with each place in the CPN; and the function MPC associates a logical window with each color for each place in the CPN. Note that each place in P is associated with an element in TG and a set of logical windows in LW. This means that the same chunk of help information may be simultaneously displayed at different locations. The function MTC associates a logical hotspot with each color for each transition in CPN. Each transition in T is associated with a set of logical hotspots. This implies that a transition t in a colored Petri net may fire in several different ways with respect to the different colors associated with t. A transition can be fired by selecting one of its associated logical hotspots. Hotspots can thus be thought of as links that are used to interconnect individual information chunks into networks or structures of related information chunks. Each hotspot (link) is a typed (colored), directional connection between a source information fragment and a destination information fragment.

Definition: Given the colored Petri net CPN = <P,T, C, IO, M> and the collection of logical mappings LM=<MPTG, MPSR, MPC, MTC> in a hypermedia help system HHS = <CPN, TG, SR, LW, LH, LM, PM>, the collection of presentation mappings of HHS is a triple PM = <DF, SF, DH>, where DF is a mapping with domain P such that ($\forall$p $\in$ P)

DF(p) : {{tg, lw} | tg = MPTG(p) , lw $\in$ {MPC(p)(c) | c $\in$ C(p)}} → a set of on-screen representations and locations of visible resources ;

SF : {MPSR(p) | p $\in$ P} → a set of audio effects;

DH is a mapping with domain T such that ($\forall$t $\in$ T);

DH(t) : {MTC(t)(c) | c $\in$ C(t)} → a set of visual effects.

PM also consists of three mappings between the user-perceptible components associated with places and transitions in a colored Petri net and the presentation mechanisms. The function DF maps each of the visible resource and logical window pairs associated with a place into an on-screen representation and location; the function SF maps each of the sound resources associated with a place into an audio effect; and the function DH maps each of the logical hotspots associated with a transition into a visual effect. The collection of presentation mappings PM allows different presentation mechanisms to be specified to provide different visual and audio effects without affecting the logical structure of the HHS. For example, given a visible resource displayed in a physical window, one way to present logical hotspots on the screen is to map them onto mouse-selectable text/graphics items actually in the physical window; this approach is currently in use in the HelpDez HHS model. The visual effect of the physical hotspots can be achieved either by the use of reverse-video to highlight mouse-selectable text/graphics items or by their display in a different color background. Another approach is to map these logical hotspots onto a menu of mouse-selectable items located in a special section next to or below the physical window. Unfortunately, this approach requires extra screen space to display the menu of mouse-selectable text/graphics items.

The marking M of CPN represents the state of an HHS during execution. When an HHS is first activated, the information fragments presented to the user correspond to the places that contain tokens in the initial marking M_S of the CPN. For example, considering an HHS for a UNIX desktop, if both the printer and the mailbox icons are selected and the on-line help system is activated, the user should be able to simultaneously view two help screens; one displaying help information for the printer icon and the other displaying help information for the mailbox icon. Note that the user can also concurrently hear the audio information at the same time the help screens are displayed. A token in a place p indicates that the help information associated with p, denoted by MPTG(p) and MPSR(p), is

presented for both viewing and hearing. Two tokens with different colors in a place p indicate that the same help information MPTG(p) can be simultaneously displayed at two different screen locations; display locations are determined by information provided from the logical windows associated with the colors of each place.

The movement of tokens in a colored Petri net represents the functioning of an HHS. When a token with color c moves into an empty place p, the associated visible information is mapped to a specific location on the screen and the associated audio information is also presented to the user through a sound generator. When a token with color c is removed from a place p, the corresponding help screen, denoted by DF(p)({MPTG(p), MPC(p)(c)}) , is removed from the display. The movement of colored tokens in the net is accomplished by selecting logical hotspots associated with the colors of each transition. Selection of a hotspot by the user fires one of the enabled transitions with respect to an individual color. When a transition t is enabled with respect to a color c, the logical hotspot MTC(t)(c) is mapped to a mouse-selectable area on the screen, denoted by DH(t)(MTC(t)(c)).

In summary, the HHS as defined above employs both the linked structure and the execution semantics of a colored Petri net to describe all possible execution paths that a user may follow through the help system. A colored Petri net CPN can describe an HHS representation model in which the Petri net structure represents the linked structure of the system; each color represents a help topic of the system; the marking describes the state of the system; the marking with respect to each color describes the state during the browsing of each help topic; the evolution of the marking describes the functioning of the system; and the movement of tokens of each color describes the browsing of a help topic. During execution, the current marking M of CPN determines which information fragments are presented to the user and at which locations they should be displayed. The transitions enabled with respect to some colors under the current marking M determine which hotspots are displayed in which windows. When the user click on one of the displayed hotspots,

one of the enabled transitions with respect to some color is fired. Firing an enabled transition with respect to some color results in a new marking M', thus causing the presented information to change as well. The execution of an HHS terminates when a marking M is reached in which no transitions are enabled.

4. HelpDez

HelpDez provides a colored Petri net based environment for the design, simulation, and construction of an HHS. It allows an HHS to be separately developed, tested, and maintained even in the absence of the application for which it is implemented. HelpDez consists of three tools for graphically constructing an HHS : HelpDraw, Help Screen Sequencer, and Control File Generator. HelpDraw, which is used to create help screen objects and save them as binary encoded files, is a tool similar to MacDraw in scope. In this section the Help Screen Sequencer and the Control File Generator are discussed in detail, followed by consideration of the integration of an OSU application with an HHS designed with HelpDez. Note that in this paper discussion is confined to the visual help screens presented to the user.

4.1 Help Screen Sequencer

The Help Screen Sequencer allows the designer to specify how the help screen objects should be sequenced in the final HHS, accomplished by simulating each final sequence as it is built. The Help Screen Sequencer collects information from the designer by a combination of dialogs and direct manipulation of the help screen objects displayed on the screen. For example, considering a designer in the process of sequencing the help screens for an HHS, if a text/graphics item in the help screen A is clicked, the item is highlighted to indicate that it is a hotspot; the designer is then asked which help screen(s) should be invoked. The designer can then designate that help screen B will appear in the final on-line help system by selecting it from a scrolling box displaying the file names of all of the help

screen objects created with the use of HelpDraw. Once a file name is selected, help screen A is removed from the screen and help screen B, corresponding to the selected file name, is displayed. With a mouse, designer may drag help screen B to any place on the screen to specify its actual location. The designer can continue this process in a depth first manner to specify all of the execution paths for an HHS.

A colored Petri net is automatically constructed as the designer sequences the help screens. At any point in building the sequence, the designer can switch from building to running in order to simulate what has been specified to that point in the building phase During the building phase, OK and Backup buttons are disabled to prevent the designer from selecting them; their functions are automatically enabled when the designer switches to the running mode in order to test the sequencing. When this mode is selected, the colored Petri net constructed to that point, along with a small palette containing different colors (patterns) is graphically displayed. The palette provides the colored Petri net with the "OK" and "Backup" capabilities. The designer is then allowed to place tokens in the colored Petri net to specify the initial marking. Simulation can be controlled either by the firing of enabled transitions from the colored Petri net or by directly selecting the hotspots displayed on the help screens. The "OK" and "Backup" actions are controlled either by selecting the enabled colors (patterns) from the palette or by directly selecting the enabled buttons on the help screens. At any point in this mode, the designer can refer to a graphical representation of the colored Petri net to obtain a global view of the HHS and the status of current location(s).

An alternative approach to the construction of the help screens' sequence is to provide a colored Petri net editor so the designer can explicitly build a colored Petri net and specify the mappings of places to help information fragments and transitions to hotspots. However, this approach requires the designer to have more extensive knowledge of the

colored Petri net. Moreover, the use of a net editor will not gracefully support the direct manipulation of the help screen objects.

4.2 Control File Generator

When the designer is finished specifying the sequence, the internal representation of the colored Petri net, together with the information associated with places and transitions in the net, is used as input to the Control File Generator to produce the control file. Consequently, the control file includes information on the relationships (links) among the help screens, the locations where the help screens will be displayed on the screen, and the locations and sizes of the hotspots which will be displayed on the help screens.

4.3 Integration with OSU

To integrate an HHS with an OSU-produced application, three things are needed: help screen objects designed and stored in binary encoded files using HelpDraw, sequence information stored in a control file, and a drawing library which may be written in advance. The drawing library is a collection of routines that are called to display help information stored in binary encoded files on the screen and to achieve visual effects representing hotspots [4]. Integration can be accomplished with the use of the OSU graphical sequencer. The graphical sequencer is used to display and directly manipulate the user interface objects to produce a sequence command script, which is used to generate source code by the program generator [1]. In the process of sequencing the user interface objects, the designer must specify the relationships between application objects and the HHS top level help screens. For example, considering the integration of an HHS with a Unix desktop, the designer can relate the desktop object "printer" and the top level help screen A for a printer by clicking on the icon representing the "printer" and then selecting the file name associated with help screen A. The relationships between application objects and the HHS top level help screens, along with sequence information stored in the control file, can

then be used to generate source code containing mainly procedure calls to the drawing library routines. The source code generated for the help system, the drawing library, and other OSU-generated source code implementing the user interface and functionality, are compiled and linked to produce an executable application supporting a context sensitive hypermedia on-line help system.

5. JUSTIFICATION

The use of Petri-net-based models offers the advantages of interactive simulation, graphical representation, and a wide development of analysis tools. Since colored Petri nets are abbreviations of ordinary Petri nets, they present the same advantages. Although ordinary Petri nets and colored Petri nets are equivalent with respect to descriptive power, the modeling of complex systems using ordinary Petri nets often leads to very large nets which are difficult to analyze or to display on graphic screens. Colored Petri nets allow for the description of concurrent execution paths in a common subnet, thereby reducing net size, but without loss of the ability to distinguish between paths. Much of the Petri-net-based analysis is accomplished using the reachability graph [8] and invariants [9]. This is true of colored Petri nets also [3, 7]. We discuss below how the colored-Petri-net-based HelpDez can benefit from these analysis techniques and interactive simulation.

Reachability graph represents information about the reachable markings and occurrence sequences in a net. Although the reachability graph for a net representing a reasonable HHS is finite, it can be infinite for a general Petri net. To overcome this problem, methods have been developed to obtain a finite graph known as coverability graph [9] or reachability tree [8]. From analyzing the reachability graph of a colored Petri net, we can gain information about the properties of an HHS. The maximum number of tokens in all reachable markings determines the maximum number of help screens required to be simultaneously displayed on the screen. This information can be used to plan screen

layouts in order to avoid overcrowding the screen. By scanning all the reachable markings in the reachability graph for an HHS, we can determine which help screen can not be viewed, or which help screens can or cannot be simultaneously viewed, when the HHS is in execution. This information may be used to locate design errors.

Invariants are invariant assertions over the marking, which can be calculated directly from the solutions of an equation upon the incidence functions (matrix) of the net [3]. It is often unnecessary to calculate the invariants since from the very beginning the designer has a clear idea about the set of invariants that he expects the system to fulfil. Normally, this is the situation whenever Petri nets are used during the design of new systems. The set of proposed invariants can then be checked using a previously developed tool. For example, assume that the designer has decided to use only one help screen at a time and, when necessary, a "more" hotspot to describe a concept. For the HHS to be modeled correctly, it must have the property "the total number of tokens for each color in any one of the reachable markings is less than or equal to 1". That is, it has to be shown that

$$\forall_{j=1,2...,q} \left[\sum_{i=1}^{r_j} M(P_i^j)\ (C_j) \leq 1 \right]$$

where q is the number of all possible independent execution paths, C_j is the color representing an execution path j , and r_j is the number of places involved in the execution path j. This means that when the user clicks a hotspot or a button on the help screen A, either the help screen A is simply removed from the screen or the help screen A is removed from the screen and replaced by the help screen B.

The interactive simulation of an HHS can be controlled by manipulation of a colored Petri net graphically displayed on the screen. Interactive simulation provides an easy and very effective way to reveal bugs during design. The graphical representation of the colored Petri net allows HelpDez to highlight those transitions that are enabled. This gives

a very nice form of dialog where the user can see different transitions that are enabled, choose some of them to fire, and immediately see the result, that is computed by HelpDez. Colored Petri nets also provide the interactive simulation of the HHS with a concise way to support backup capability. This allows the designer to investigate how a given marking was reached or to try out different execution paths between a set of concurrently enabled transitions.

6. CONCLUSION

This paper describes a colored-Petri-net-based approach to the design, simulation, and construction of an HHS. It also presents a formal definition of HelpDez model in terms of the colored Petri nets. Colored Petri nets can be used to describe both the structural and behavioral properties of an HHS in a more useful and succinct manner than ordinary Petri nets. Analysis techniques which have been developed for both ordinary Petri nets and colored Petri nets can be directly used for proving the right properties of an HHS. In addition, integration of an HHS with OSU-produced applications has been described. Currently, HelpDez is under development for the Apple Macintoshes, with the expectation of producing a running prototype within the next six months.

REFERENCES

1. Armstrong, J.R. Code Generation in the Oregon Speedcode Universe. Tech. Report 88-60-15, Dept.of Computer Science, Oregon State Univ.,Corvallis, Ore.

2. Conklin, E.J. Hypertext: An Introduction and Survey. *IEEE Computer* 2, 9 (Sept. 1987), 17-41.

3. Jensen, K. Coloured Petri Nets and the Invariant-Method. *Theoretical Computer Science* 14 (1981), 317-336.

4. Keh, H.C. Drawing Library Design Document. Internal Correspondence, Sequent Computer Systems Inc., Beaverton, Ore., Sep. 1989.

5. Lewis, T.G., Handloser, F.T., Bose, S. and Yang, S. Prototypes from Standard User Interface Management Systems. *IEEE Computer* 22, 5 (May 1989), 51-60.

6. Macky, L. Help System Requirements Specification. Internal Correspondence, Sequent Computer Systems Inc., Beaverton, Ore., Aug. 1989.

7. Narahari, Y. and Viswanadham, N. On the Invariants of Coloured Petri Nets. *Advances in Petri Nets 1985*. Rozenberg, G. ed., L.N.C.S.,Springer-Verlag (1986), 330-341.

8. Peterson, J.L. *Petri Net Theory and the Modeling of Systems*. Prentice-Hall, Englewood Cliffs, N.J. 1981.

9. Reisig, W. *Petri Nets: An Introduction*. Springer-Verlag, New York, 1985.

10. Shneiderman, B. *Designing the User Interface*. Addison-Wesley, Reading, MA, 1987.

11. Sommerville, I.,Welland, R. Potter, S.and Smart, J. The ECLIPSE User Interface. *Software - Practice and Experience* 19,4 (Apr. 1989), 371-391.

12. Stotts, P. D. and Furuta, R. Petri-Net-Based Hypertext: Document Structure with Browsing Semantics. *ACM Trans. Off. Inf. Syst.* 7,1 (Jan.1989), 3-29.

13. Walker, J.H. The Document Examiner. *SIGGRAPH Video Review*, Edited Compilation from CHI'85: Human Factors in Computing Systems 1985.