View metadata, citation and similar papers at core.ac.uk brought to you by CORE
provided by Tamkang University Institutional Repository

A CASE Tool Supports the Software Life Cycle of Participator Dependent Multimedia Presentations

Timothy K. Shih, Yih-Jia Tsai, Jason C. Hung, and Ding-Rong Jiang
Department of Computer Science and Information Engineering
Tamkang University, Tamsui, Taiwan, R.O.C.
email: TSHIH@CS.TKU.EDU.TW
fax: Intl. (02) 620 9749

Abstract

This paper presents a CASE tool for the specification, design, implementation, testing, and maintenance of multimedia presentations. The proposed system is based on a Petri net machine to control event-based resource synchronization. Data flow diagrams and control flow diagrams are used as the specification of a multimedia presentation. The system then allows the specification to be stepwise refined toward the final multimedia presentation.

1. Introduction

Structured analysis and design have been used in software development. The methodology can be used in multimedia presentation designs. The development of a multimedia presentation involves the analysis of a presentation script, the collection of multimedia resources, and the realization of a presentation layout and navigation control flow. In line with the growing needs of multimedia presentations, many authoring systems were developed. However, in the community of multimedia computing, there is no discussion of the need of a good presentation development methodology. We have surveyed a number of presentation design tools. We found that, not many authoring system focus on the concept of stepwise refinement. Moreover, multimedia presentation development life cycle and paradigm were hardly find in the literature of multimedia computing. We believe that, the development of a multimedia presentation is similar to software development. Using Water Fall paradigm, the presentation developer will benefit from experiences and techniques of the structured analysis research of Software Engineering. However, designing a presentation is similar to but different from writing a software. We give a summary of presentation development life cycle, which is assisted by our paradigm and tool:

- **Analysis**: Presentation Requirements Analysis
- **Specification**: Presentation DFD/CFD
- **Design**: Interactive Multimedia Petri Net
- **Implementation**: Presentation Generation
- **Testing**: Presentation Testing
- **Maintenance**: Presentation Documentation

This paper is organized as the following. We present the life cycle of multimedia presentations in section 2. The analysis of presentation requirements is discussed in section 2.1. Presentation specification and the proposed multimedia data flow/control flow diagram is discussed in section 2.2. An interactive Petri net program serves as the presentation design assistant tool is proposed in section 2.3. Section 2.4, as an extension to the interactive Petri net, proposes a new direction of multimedia presentation designs. Multimedia presentation implementation, testing, and maintenance are discussed in section 2.5 and section 2.6. A short conclusion summarizes our contributions is also presented in section 3.

2. Presentation Life Cycle

As mentioned in the introduction section, the development life cycle of multimedia presentations consists of several phases. In this section, we discuss these phases in detail.

2.1. Analysis of Presentation Requirements

In a typical software development cycle, the first step is to conduct an analysis of the problem and the requirement. If the problem is suitable and worth it to be solved by software programs, the system analyzer starts to prepare for a system specification of the software. In the development of a multimedia presentation, the main theme of the presentation has to be decided first. The analysis of presentation requirements is the most important factor achieving the success of a multimedia presentation. How to attract the audience and demonstrate the main theme of a presentation through text, sound, and graphic illustrations are the purposes of presentation analysis.

Authorized licensed use limited to: Tamkang University. Downloaded on March 24,2010 at 02:44:48 EDT from IEEE Xplore. Restrictions apply.

2.2. Presentation Specification

We propose a revised DFD/CFD mechanism for producing multimedia presentation specifications. The design of a multimedia presentation, utilizing message passing for navigation, has a similar concept to writing a software specification. The proposed multimedia data flow/control flow diagram is based on DFD and CFD, with the extension of some new objects:

- **Multimedia Resource**: similar to the data store in a DFD, a multimedia resource is denoted by a pair of thick parallel lines.
- **State Variable**: besides resources, an interactive presentation may contain state variables store presentation data, such as the audience's name. A state variable is represented by a pair of thin parallel lines. State variables not only keep information, but also control *dynamic presentations* (to be discussed in section 2.4).
- **Resource Data**: similar to the data link in a DFD, resources are passed to a presentation program by a link with a regular arrow.
- **Navigation Message (NM)**: similar to the control link in a CFD, a navigation message is passed by a link with a light-weighted arrow.
- **Dynamic Mutation**: to support dynamic multimedia presentations, the dynamic mutation link is introduced, which is represented by a curved arrow. There are four types of mutations in a multimedia DFD/CFD: state variable change, layout change, resource change, and navigation change.
- **External Entity (EE)**: similar to one in DFD/CFD, an external entity could represent a user, a hardware device, or another system which pass data/controls to the multimedia presentation. An EE is denoted by a box surrounded by thick lines.
- **Presentation Window (PWin)**: similar to a process in a DFD, a presentation window is denoted by a circle.

2.3. Presentation Design

A presentation window must be refined. Stepwise refinement of a presentation allows the presentation to be specified in different levels of details. A multimedia DFD/CFD of multiple levels is to help a presentation designer to organize the script structure of a presentation. However, it is not powerful enough to define the precise schedule or layout of that presentation. Incorporated with an interactive multimedia Petri net diagramming mechanism, the last level of a presentation window is refined to a Petri net, which describes the temporal behavior of a presentation window.

The spatial organization of the presentation is specified by using the graphical user interface of our system. Our interactive multimedia Petri net is a variation of timed Petri net, with the addition of *User Transitions* and *Sync Arc*. Since a multimedia presentation is interactive, it is necessary for us to introduce the above two objects for participant dependent synchronization.

A timed Petri net is a bipartite graph with two types of nodes: the transition nodes and the place nodes. A transition controls synchronization and a place holds a token and a time duration. A transition is fired only after each place adjacent to the transition releases the token. A place holds a multimedia resource to be played for the time duration. Transitions and places are connected by *sync arcs* in our revised Petri net. We add *user transitions* and *user arcs* to the timed Petri net. A user transition receives a navigation message from the user before it is fired. A user transition is directly connected to some transitions. The activation of the user transition interrupts the demonstration of the presentation window and causes the activations of the connected transitions simultaneously.

The followings are components of the Petri nets:

- **Transition**: is for synchronization control. Transitions are shown as vertical dark bars.
- **User Transition**: accepts a message and causes the activation of connected transitions. User transitions are shown as vertical light bars.
- **Place**: is for playing a multimedia resource. Places are shown as ellipses.
- **User Arc**: connects from a user transition to a transition. User arcs are shown as curved arrowheaded arcs.
- **Sync Arc**: connects from a place to a transition, or from a transition to a place. Sync arcs are shown as straight arrowheaded lines.

In the specification of a presentation, the designer does not need to have all multimedia resources ready at the beginning. The presentation system allows dummy resources. For example, dummy text or picture resources are represented as labeled boxes. Video or animation resources not only have labeled boxes but also have animation ICONs showing the duration of those resources. Sound or MIDI resources also have animation ICONs. The presentation designer can test the presentation prototype before resources are ready for the final version of presentation.

We have discussed the global view of our diagramming mechanism. However, a multimedia presentation in our system is dynamic and mutable. In the following subsection, we propose other diagramming techniques to achieve our goal – allowing dynamic multimedia presentation.

2.4. Dynamic Presentations

Usually, when a multimedia presentation is designed, the usage of resources, the layout, and the navigation sequences are all fixed until the presentation is re-designed

Authorized licensed use limited to: Tamkang University. Downloaded on March 24,2010 at 02:44:48 EDT from IEEE Xplore. Restrictions apply.

again. We want to improve this approach by allowing dynamic replacement of resources, layouts, and controls. This makes the presentation generator have the ability of computing the presentation representation at the run-time. Possible dynamic events are: asserting/retracting of information, changing resources, changing layouts, and changing navigation controls.

On the top of the revised timed Petri net, we further add the following components:

- **Selection**: selects one of the outgoing navigation messages. A selection could be a push button, a menu item, or a key stroke of the presentation window. A selection holds an internal representation of which outgoing navigation messages will be sent upon the activation of the selection. A selection is represented as a circle labeled with an "S" (or a name) in the Petri net diagram.
- **Assignment**: assigns a value to a state variable. An assignment uses a state variable. When an assignment received a navigation message (with a parameter holding a value), the assignment set the state variable to the value. An assignment is denoted by a box labeled with the "Var <= Val" sign.
- **Condition**: decides whether to proceed with a change. A condition has pairs of state variables and values. The value is checked against the state variable. If they match each other, the associated outgoing change is fired. A condition is represented by a box with the "Var ?= Val" sign.

The Petri nets of a presentation may contain a number of selections. Usually, a selection is associated with only a navigation message. However, if the selection is tight to a condition, the selection may have more than one outgoing messages. Which message to send out is decided by the condition's value. We allow an assignment statement to change the value of a state variable. State variables are used in a presentation to hold internal states. These variables will be saved on the disk when the presentation terminates (upon the user's decision), and will be loaded when the presentation starts again. All state variables are global. That is, they are accessible from all Petri nets. Therefore, information sharing is feasible. A condition decides whether to proceed with a change to the propagation of a navigation message. A condition may change the execution flow of a Petri net.

2.5. Presentation Implementation and Testing

After the user designs the schedule and layout of the presentation via our Petri net tool and graphical user interface, our presentation system generates the presentation automatically. Unlike a typical software development, no program is written. This generation strategy is also used by most multimedia presentation systems available on the market. The advantage is that if the specification and design of a presentation is correct, our system guarantees the correct implementation.

It is possible to obtain the software metrics of a presentation. Our system is able to calculate the metrics of a presentation based on the following criteria:

- number of presentation windows
- number and size of multimedia resources
- number of navigation messages
- number of state variables
- number of dynamic mutations
- number of selections
- number of assignments
- number of conditions

Software metrics of a presentation indicates the complexity and the amount of efforts to test the presentation. Complexity of the presentation also indicate the effort to collect presentation resources and the size of potential disk storage used. Presentation testing is essentially important to ensure a smooth demonstration without missing resources and non-existing navigation sequences. It is hard to perform a complete testing by the user due to the amount of navigation sequences. Fortunately, our system facilitates automatic testing by means of an tool. The tool traverses the presentation based on the following testing criteria:

- every presentation window should be tested
- every multimedia resource should be tested
- every navigation message should be tested
- every selection should be tested
- every assignment should be tested
- every condition should be tested

The testing includes all Petri nets, which may accept navigation messages sent to transitions or user transitions. Synchronization of multimedia resources used by places are tested as well. Moreover, selections, assignments, and conditions are tested. Since there is no particular order that the user navigates the presentation, the path topology of testing is arbitrary. Our testing tool traverses a graph consists of nodes (i.e., selections, assignments, conditions, and starting transitions/user transitions) and links (i.e., navigation messages) based on a modified breadth first search (BFS) algorithm. Upon the activation of a presentation window, each selection, assignments, or conditions are kept in a queue for further traverse. When the internal state of the presentation is changed due to the change of state variables (which may cause navigation change, layout change, or resource change), some modified nodes are added to the queue used in the BFS algorithm for re-testing. The testing tool keeps a testing log file which indicates the testing order.

Authorized licensed use limited to: Tamkang University. Downloaded on March 24,2010 at 02:44:48 EDT from IEEE Xplore. Restrictions apply.

Another feature of our system is an evaluation tool of the presentation design status. It is very likely that a user design an incomplete or inconsistent presentation. The following is a list of potential mistakes that a user may create in a presentation:

- non-used state variables
- non-used multimedia resources
- non-used assignments
- non-used conditions
- incomplete navigation messages
- incomplete selections
- incomplete assignments
- incomplete conditions
- incomplete presentation windows (i.e., if any component of the window is incomplete)
- inconsistent presentation windows (i.e., unbalanced links)
- undocumented presentation windows

Non-used items, such as state variables not used in any condition, multimedia resources not linked to places, or assignments and conditions not triggered by navigation messages, are marked by the tool. Incomplete items, with one of their properties missing, are also marked. A presentation window is inconsistent if the window and its refinement have different links (i.e., the unbalanced link problem). An inconsistent or undocumented presentation window is also marked. The user is able to check the list of problematic items before releasing the presentation.

2.6. Presentation Maintenance

It is possible that the user will release different versions of a presentation. Therefore, presentation maintenance is an important issue. A presentation specification in our system contains a number of presentation windows. Each presentation window has a documentation box. It is important to document the presentation script in the box for further references. Moreover, how to keep track of different versions of a presentation window is important. In a multimedia presentation database we developed [3, 2], presentation windows as well as presentation resources are reusable objects. Each of these objects has several versions. A presentation is a composed object from these reusable objects. The database system serves as an underlying supporting tool for several multimedia presentation systems that we have developed [4, 2], including the one we propose in this paper.

3. Conclusions

The preliminary experiences of using the system show that, the proposed multimedia DFD/CFD is easy to learn since data flow diagrams have been used by many engineers and managers for decades. The concept is adapted easily. Comments from many undergraduate students taking a multimedia course show that it is quite easy to use our system. Contributions in this paper are: firstly, we apply the Water Fall paradigm to presentation development and revise data flow/control flow diagrams for the use of multimedia presentations. Next, we proposed a multimedia Petri net for dynamic multimedia presentations. Finally, a system was developed on MS Windows 95/NT to justify our approach.

References

[1] T. K. Shih. “Multimedia Abstract Machine”. In *Proceedings of the Third Joint Conference on Information Science*, 1997.

[2] T. K. Shih and R. E. Davis. “IMMPS: A Multimedia Presentation Design System”. *IEEE Multimedia*, Summer, 1997.

[3] T. K. Shih, C.-H. Kuo, and K.-S. An. “An Object-Oriented Database for Intelligent Multimedia Presentations”. In *Proceedings of the IEEE International Conference on System, Man, and Cybernetics Information, Intelligence and Systems Conference*, 1996.

[4] T. K. Shih, C.-H. Kuo, and H.-J. Lin. “Visual Multimedia Presentation Designs”. In *Proceedings of the 1996 Pacific Workshop on Distributed Multimedia Systems*, pages 306 – 315, 1996.

Authorized licensed use limited to: Tamkang University. Downloaded on March 24,2010 at 02:44:48 EDT from IEEE Xplore. Restrictions apply.