$TSPN_{UI}$

A Petri net model for Specifying User Interactions in Multimedia Presentations

By

Kendra M. L. Cooper

B.A.Sc. Electrical Engineering, The University of British Columbia, Vancouver, Canada, 1993.

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENT FOR THE DEGREE OF

MASTER OF APPLIED SCIENCE

in

THE FACULTY OF GRADUATE STUDIES

ELECTRICAL ENGINEERING

We accept this thesis as conforming

to the required standard

THE UNIVERISTY OF BRITISH COLUMBIA

September 1995

In presenting this thesis in partial fulfilment of the requirements for an advanced degree at the University of British Columbia, I agree that the Library shall make it freely available for reference and study. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by the head of my department or by his or her representatives. It is understood that copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Department of ELECTRICAL ENGINEERING

The University of British Columbia
Vancouver, Canada

Date Oct 11/95

Abstract

The first objective of this thesis is to analyze and group the synchronization requirements for a generic, distributed, multimedia presentation application [Gon95a]. The purpose of this analysis work is to determine what synchronization requirements a general purpose specification approach for distributed multimedia applications must be able to describe. The results reveal a set of related requirements, which are classified into four categories: data stream; communication; computer-human interaction; and expressiveness requirements. The analysis and classification work provides a template which may be tailored for specifying applications.

The second objective of this thesis is to review Petri net models. Established Petri net models are reviewed in addition to recent multimedia extensions [Gon95b, Vuo95a]. The multimedia extensions are the Object Composition Petri net (OCPN), Extended Object Composition Petri Net (XOCPN), Dynamic Timed Petri net (DTPN), and the Time Stream Petri net (TSPN). The purpose of this analysis is to determine which requirements each of the Petri net approaches may model. The analysis work is carried out using an illustrative example, which includes interstream and intrastream synchronization, time dependent and time independent data streams, fixed playout duration, coarse and fine grained synchronization requirements, user interactions, and jitter constraints. The example is specified in the natural language and the multimedia Petri net extensions. The analysis of these specifications focuses on whether or not a model may be used to describe the requirements and how complex the resulting model is in terms of the number of symbols used to describe the example. The results of this work indicate that the TSPN approach provides a good basis for describing multimedia applications,

but needs to be enhanced to describe pre-emptive user interactions, quality of service, and multiple destination site requirements.

The third objective of this thesis is to extend the TSPN model to permit the formal modelling of pre-emptive user interactions. The newly enhanced version is called the $TSPN_{UI}$ model. The TSPN model is used as the basis of the extension because it models a large number of the synchronization requirements for a generic multimedia application. The pre-emptive user interaction approach in the DTPN model is simplified, generalized into a toolkit of interactions, and integrated with the TSPN approach. To insure the temporal correctness of the $TSPN_{UI}$ extension, the example is simulated using a visualization tool developed with MATLAB [MAT94]. MATLAB is a commercial application which has been successfully used to simulate Petri nets [Rez95]. The simulation tests different scenarios of user interactions and timing of events to verify the model maintains synchronization under diverse conditions. Errors in the specification are found and corrected using the Visualization Tool.The practical significance of the tool is that error detection and correction are less expensive early in the development cycle.

Acknowledgement

I would like to thank my supervisor, Dr. Mabo Ito, and my co-supervisor, Dr. Son Vuong, for originally suggesting this thesis topic and for their technical guidance throughout the work. A special thank you to Dr. Ito for the financial support he provided during the work under NSERC grant #58-9054. Without the graduate research assistance this degree would not have been possible. Acknowledgements to Mohammad Rezai for his assistance with Petri nets and the tutorials on the simulation tool he developed in MATLAB. Warm thanks to Raymond Cheng, Jeffrey Chow, Chong Ong, John Jay Tanlimco, and Darren Tsang (aka The Boys) for brightening my days with laughter and general weirdness. Noteworthy episodes included the stapler-napping incident, the Teamwork windows that were captured and set to the background image, the mouse "feature", flashing neon blue and orange background images, voices from the computer telling me to go home, and the messages from God across my screen. (:

1 Introduction

1.1 Background

Multimedia (mm) refers to the integration of two or more media streams such as text, bitmap (rastor) graphics, vector graphics, metafile graphics, animation, audio, or video in an application. The mm applications can be divided into two broad categories: distributed and non-distributed applications. Non-distributed mm applications include those where the source site is the same as the destination site. These presentations are composed of locally supplied data, for example, from CDROM, hard disk, or camera. Distributed mm applications, the more complex category, support the integration of various media supplied by data sources over a network. For both categories the data may be from a live, real-time or stored, pre-orchestrated media source. The type of application focused on in this work is a distributed application with pre-orchestrated data sources.

Since the multiple streams of data may be supplied by different sources, the various streams need to be coordinated both spatially and temporally for presentation. Temporal synchronization is the process of coordinating the real-time presentation of mm information and maintaining the time ordered relations among component media [Iin94]. The synchronization task involves coordinating among streams of distinct media (interstream synchronization) as well as coordinating within each stream of media (intrastream synchronization). Synchronization is applied to the normal playout of streams of data and in response to events from a human user or the supporting network.

The problem of synchronizing time dependent and time dependent (continuous) and time independent (discrete) data is fundamental to mm applications. The problem includes

synchronizing data that are:

- heavily time dependent and require stringent interstream synchronization control. For example, lip synchronize the audio and visual data in a motion picture for 10 seconds;
- time independent and do not have stringent synchronization requirements. For example, present the image for 10 seconds;

The specification of an application with temporal constraints requires a description of what the application needs to do and when the application needs to do it. Multimedia synchronization requirements may be specified using informal, semi-formal, or formal techniques. The informal approach is a natural language specification. Using natural language for requirements specification tends to result in ambiguity and requirements which may be mutually exclusive [Hin93]. Semi-formal techniques include real-time structured analysis and object oriented analysis approaches [War85, Boo91]. They provide the notation and heuristics for developing a specification, but the semantics are not described by a mathematical model. Formal methods provide a mathematically defined approach which is unambiguous and precise, promising correct specifications and automated development [Hin93, Pla92]. Formal approaches include Petri nets, Harel state charts, formal description techniques, high level languages, and path operators [Hoe92, Gib91, Pet81, Bol88, Log92, Har87]

Extended Petri net models are the most commonly used formal method in the literature for specifying mm synchronization requirements. They offer a variety of useful features for graphical representation of concurrent systems including hierarchy and the occurrence of external events. Another advantage of Petri net models is that they are easy to learn.

1.2 Objectives

The first objective of this thesis is to analyze and group the synchronization requirements for a generic, distributed, mm presentation application [Gon95a]. The purpose of this analysis work is to determine what synchronization requirements a general purpose specification approach for distributed mm applications must be able to describe. The results reveal a set of related requirements, which are classified into four categories: data stream; communication; computer-human interaction; and expressiveness requirements. The analysis and classification work provides a template which may be tailored for specifying applications.

The second objective of this thesis is to review Petri net models. Established Petri net models are reviewed in addition to recent mm extensions [Gon95b, Vuo95a]. The mm extensions are the Object Composition Petri net (OCPN), Extended Object Composition Petri Net (XOCPN), Dynamic Timed Petri net (DTPN), and the Time Stream Petri net (TSPN). The purpose of this analysis is to determine which requirements may be modelled by each Petri net approach. These results may be used to select a specification approach for an application. The analysis work is carried out using an illustrative example, which includes interstream and intrastream synchronization, time dependent and time independent data streams, fixed playout duration, coarse and fine grained synchronization requirements, user interactions, and jitter constraints. The example is specified in the natural language and the mm Petri net extensions. The analysis of these specifications focuses on whether or not a model may be used to describe the requirements and how complex the resulting model is in terms of the number of symbols used to describe the example. The results of this work indicate that the TSPN approach provides a good basis for describing mm applications, but needs to be enhanced to describe pre-emptive user interactions, quality of service, and multiple destination site

requirements.

The third objective of this thesis is to extend the TSPN model to permit the formal modelling of pre-emptive user interactions. The newly enhanced version is called the Time Stream Petri net with User Interaction ($TSPN_{UI}$) model. The TSPN model is used as the basis of the extension because it models a large number of the synchronization requirements for a generic mm application. The pre-emptive user interaction approach in the DTPN model is simplified, generalized into a toolkit of interactions, and integrated with the TSPN approach. To insure the temporal correctness of the $TSPN_{UI}$ extension, the example is simulated using a visualization tool developed with MATLAB [MAT94]. MATLAB is a commercial application used successfully to simulate Petri nets [Rez95]. The simulation tests different scenarios of user interactions and timing of events to verify the model maintains synchronization under diverse conditions.

1.3 Outline

The thesis work begins with an analysis of the distributed mm synchronization requirements. This analysis reveals what a specification approach needs to be able to model. The requirements analysis is followed by a review of existing Petri net models and four proposed mm extensions. The strengths and weaknesses of the proposed extensions lead to the development of a new Petri net model, the $TSPN_{UI}$ model. A specification in the $TSPN_{UI}$ model is simulated with a Visualization Tool, developed using MATLAB.

The organization of this thesis is as follows:

- Chapter 2 presents a requirements analysis for a generic, distributed mm presentation application. The synchronization requirements are identified and classified into four groups. The relationships among the requirements are also presented in a high level entity relationship diagram.
- Chapter 3 reviews a selection of established and recently proposed mm Petri net models. The mm extensions include the OCPN, XOCPN, DTPN, and TSPN models. A common example is worked out in each of the proposed models and the mm extensions are analyzed.
- Chapter 4 presents the proposed $TSPN_{UI}$ model. The example specified in Chapter 3 is also worked out in the proposed model. The $TSPN_{UI}$ model is compared with the existing mm extensions.
- Chapter 5 presents the Visualization Tool for simulating the $TSPN_{UI}$ specification, the experimental scenarios, and the results of the simulation. The strengths and weaknesses of the tool are discussed.

- Chapter 6 presents the conclusions and future work.

- The natural language specification of the example used throughout the thesis is available in Appendix A. The logic necessary to provide the user interactions for the DTPN model and the $TSPN_{UI}$ model are provided in Appendix B.

2 Multimedia Synchronization Requirements

2.1 Introduction

A requirements analysis is an essential step in understanding and documenting what the system has to do and when it has to do it [Hof93, ANS84]. Although multimedia is a very popular research area, a requirements analysis has not been available in the literature for a generic distributed mm application. In this section, the synchronization requirements for a generic, distributed mm presentation application are identified and classified [Gon85]. Issues in the development of a classification scheme include finding a logical breakdown that is straightforward and easy to allocate requirements into. Following the classification of the synchronization requirements, the relationships among the requirements are presented in an entity relationship diagram.

2.2 A Classification of the Synchronization Requirements

A structured analysis software engineering technique provides both a framework for determining what the synchronization requirements are in addition to the heuristics for grouping the requirements into categories. Grouping places functionally related requirements together makes the specification easier to create, review, and correct. A set of requirements are identified and organized into four groups: data stream; communication requirements; computer-human interface (CHI); and expressiveness requirements. The requirements groups are discussed in the following sections by group and summarized in Figure 1.

2.2.1 Data Stream Requirements

A data stream is defined in this thesis as the data from a unique source site used in contiguous presentation intervals. The data stream group consists of requirements which answer questions about the data streams. The questions and the requirement names are summarized in Table 1.

2.2.1.1 Source Sites. Each data stream in a mm presentation originates from a unique source site. In this thesis, a source site is not equivalent to a physical site. Multiple data streams may originate from a single physical source site and multiple physical sites may be required. Therefore, the data stream source site: physical location relationship for each physical location is N:1. The rationale for defining the source site requirement this way is that all data streams delivered over the network must be synchronized at the destination site whether or not they originate from one or many physical locations. Additional physical sites do not add complexity to the specification because the data transmission does not need to be coordinated among the source sites.

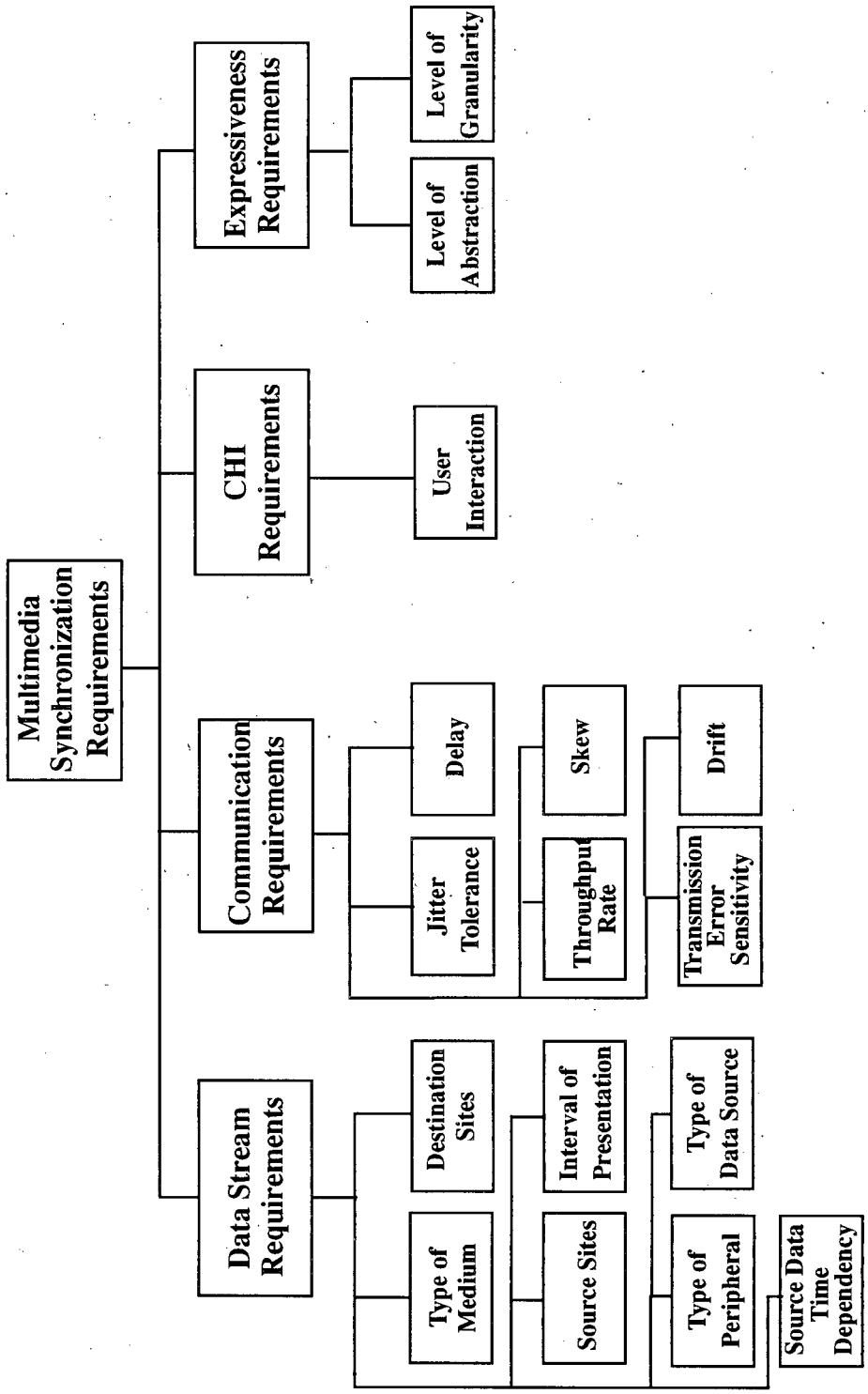


Figure 1. A Classification of Synchronization Requirements for Distributed Multimedia Applications

Question	Requirement Name
Where are the data streams coming from?	Source Sites
Where are the data streams going to?	Destination Sites
What kind of data are in the streams?	Type of Medium
How time sensitive is the data stream?	Source Data Time Dependency
What interval is the data stream presented in?	Presentation Interval
Is the data stream from a real-time or a pre-orchestrated data source?	Type of Data Source
What kind of peripherals are required for the data stream to be presented?	Type of Peripherals

Table 1. Data Stream Requirements

2.2.1.2 Destination Sites. The destination sites for the mm presentation describe the unique physical sites which accept data streams for presentation. The physical location:data stream relationship at each destination site is 1:M. Multiple destination sites may be required in a mm application such as multipoint videoconferencing. The presentation of synchronized data at two or more destination sites requires the various media streams to observe the same temporal relation at different sites on a network. The additional destination sites do add complexity to the specification.

2.2.1.3 Type of Medium. The type of medium identifies the data in a stream. For example, the data may be voice quality audio, CD quality audio, text, video, audio and video, bit-mapped image, vector image, metafile image, or a digital animation. A review of different types of media and common interchange formats is available in [Aud94, Mur94].

2.2.1.4 Source Data Time Dependency. The data supplied for each data stream may be time dependent or time independent. Time dependent data depend upon the continuous pre-

sentation of the medium in order for the semantics of the presentation to be meaningful. In addition, the time dependent data is not considered complete until the recorded message or the video clip is complete. Speech, video, and animation are examples of time dependent media. The quality of these media depends upon the original sampling rate, compression/decompression techniques, and playback rates. Time independent media, such as graphics, do not depend on the sampling or playback rate. The time independent data is considered complete as soon as the image or graph is displayed. The quality of the presentation depends on the size of the original sampling, compression/decompression techniques, and the conformance of the playback system to the presentation requirements [Sai93].

2.2.1.5 Interval of Presentation. The interval of presentation requirement for a data stream describes when the presentation of data begins and ends. An important and often used representation of time is the temporal interval [All83]. Temporal intervals consist of time durations characterized by two endpoints, or instants. A time instant is a zero-length moment in time, such as "4:00 PM." By contrast, a time interval is defined by two time instants, such as "4:00 PM to 4:30 PM".

An interval is formally defined as follows [Lit93]:

Let $[S]$ be a partially ordered set

Let $a \in S$, $b \in S / a \leq b$

$X = \{x / a \leq x \leq b\}$ is called an interval of S denoted by [a,b].

Furthermore, any interval [a,b] has the following three properties:

- $[a, b] = [c, d] \Leftrightarrow a = c \wedge b = d$
- If $c \in [a, b], d \in [a, b], e \in S, c \leq e \leq d$ Then $e \in [a, b]$

- #([a,b]) > 1

Time intervals are described by their endpoints (e.g., a and b in the definition above). In the case of fixed intervals, such as the requirement to "play the video stream for 10 seconds", the presentation interval is simply b - a. For variable length intervals, such as the requirement to "play the video stream until the user indicates to stop to a maximum time of 30 minutes", the endpoint b is initially defined as the maximum end point in time or infinity if a maximum time is not specified. The duration may be described as a subset of the non-negative rational numbers union infinity [Sen94, Wah94]. The duration may be a single number, indicating the value is fixed, or a range of numbers. The interval based approach is suitable for interactive media environments which have non-deterministic characteristics [Wah94].

Specifying intervals with respect to each other rather than by using the endpoints decouples the intervals from an absolute or instantaneous time reference. Given any two intervals, there are thirteen distinct ways in which they may be related [All83]. Using Allen's representation, these relations are shown graphically in Figure 2. The thirteen relations may be represented by seven cases because six of them are inverses (the equality relation has no inverse). For inverse relations, given any two intervals, it is possible to represent their relation by using the non-inverse relation and exchanging the interval labels. For example, "A after B" is the inverse relation of "B before A".

Temporal intervals may be used to model a mm presentation by letting each interval represent the presentation of some mm data element, such as a still image or an audio segment. This modelling concept is referred to as temporal-interval-based modelling (TIB). Two types of intervals are defined in [Lit93]: atomic and non-atomic. An atomic interval is one which may not be decomposed into subintervals, as in the case of the presentation of a single

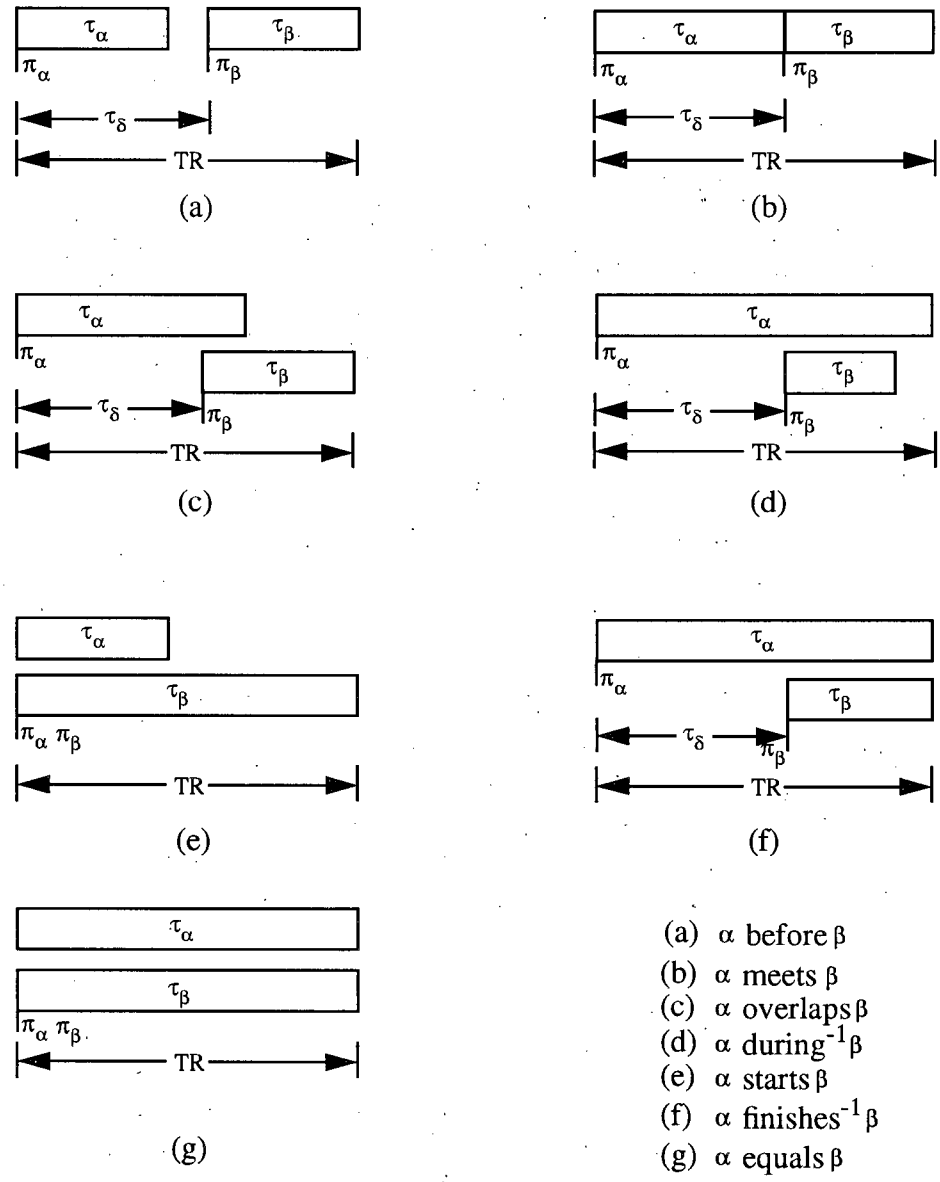


Figure 2. Binary Temporal Relations [Lit93]

frame of a motion picture. Non-atomic intervals are composed of atomic intervals, such as presenting five seconds of motion-picture frames. Intervals indicate the start times (π_α, π_β), durations $(\tau_\alpha, \tau_\beta)$, and end times for data elements α and β. The relative positioning between them is captured by a delay from the beginning of the first interval (τ_δ), as is their overall

duration, TR.

2.2.1.6 Type of Data Source. MM applications may have real-time and/or pre-orchestrated data sources. For example, a video conference application may include real time sources such as cameras and microphones, in addition to a database on a hard disk for supplying pre-orchestrated data. A video on demand application has only pre-orchestrated, stored sources of media. The specification approach should provide a device independent way to specify the type of data source and the operations supported by each one.

2.2.1.7 Type of Peripheral. The playout device for a data stream may be a computer monitor, speaker, or something somewhat more unusual such as a visor display or a chair. A data stream which has audio and video interlaced, such as a VHS data stream, requires both a monitor and at least one speaker for the presentation. Non-interlaced streams such as pure audio, for example, would only require one or more speakers. The characteristics of the display and the speaker required to support the mm presentation need to be specified, in order to support the quality required by the application. For example, a speaker must support a dB and frequency range. Displays are specified in terms of resolution, pixel size (dots per inch), and number of colours supported.

2.2.2 Communication Requirements

The communication group contains all of the requirements which answer questions about the communication requirements for the application. The media involved in a distributed mm application have widely varying communication requirements. The type of medium imposes different requirements for traffic characteristics, error handling, and delay on the

underlying communication system. For example, discrete media (time-independent) include bursty traffic, high sensitivity to transmission errors, and low sensitivity to delays. Conversely, continuous media (time-dependent) require a guaranteed, continuous presentation, and are relatively insensitive to transmission errors. Table 2 shows this variability for media in delay, bandwidth requirements, jitter, and error rates [Heh90]. The result is a need to negotiate with the underlying communication system for the appropriate support to obtain the desired quality of service. QoS parameters are used to negotiate this service during the connection establishment phase. QoS parameters express the mm application communication requirements such as throughput, tolerable delay and jitter, and acceptable error rates.

QoS	Max Delay (s)	Max Jitter (ms)	Avg Throughput (Mb/s)	BER	PER
Voice	0.25	10	0.064	$<10^{-1}$	$<10^{-1}$
Video (TV quality)	0.25	10	100	10^{-2}	10^{-3}
Compressed Video	0.25	1	2-10	10^{-6}	10^{-9}
Data (file transfer)	1	-	-	0	0
Image	1	-	-	10^{-4}	10^{-9}

Table 2. QoS Requirements

2.2.2.1 Delay. Design constraints such as the maximum network delay for the transmission between the source and destination sites may be included in a specification as a non-functional requirement. The delay includes the processing, serialization and propagation delays. Additional processing delays on the source and target sites may include compression/decompression or encryption/de-encryption processing. Some of these design constraints may be specified indirectly, for instance, in a requirement such as "the image shall be obtained from

source site A in the JFIF format". The JFIF format only supports the JPEG compression standard, so compression design details are already included in the specification. Compression techniques are reviewed in [Com05].

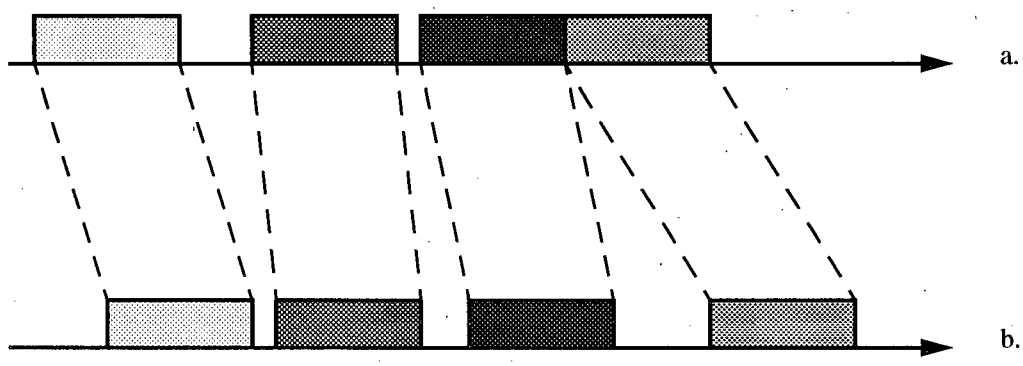
2.2.2.2 Throughput Rate. The throughput requirements for a data stream is derived from the type of medium. The rates assume no additional processing is done on the data, such as compression or encryption. For audio samples, there is an emerging tendency to standardize on only a few sampling rates and encoding styles, even if the file formats may differ. The suggested rates and styles are summarized in Table 3.

Type of Medium	Sample Description	Average Throughput (b/s)
Voice Quality Audio U Law	8000 samples/s, 8 bits/sample, mono	64,000
Half the CD sampling rate	22,050 samples/s, 8 bits/sample, mono	176,400
Half the CD sampling rate	22,050 samples/s, 8 bits/sample, stereo	352,800
CD sampling rate	44,100 samples/s, 8 bits/sample, mono	352,800
CD sampling rate	44,100 samples/s, 8 bits/sample, stereo	705,600
Video (TV quality)	8 bit pixel depth, 352x240 frame size, 30 samples/s	20,275,200
Video (HDTV quality)	8 bit pixel depth, 1280x720 frame size, 24 samples/s	176,947,200

Table 3. Throughput Rates

2.2.2.3 Jitter. MM applications may be either sensitive to jitter, the instantaneous variation in delays, or relatively insensitive. The effect of jitter is the introduction or removal of

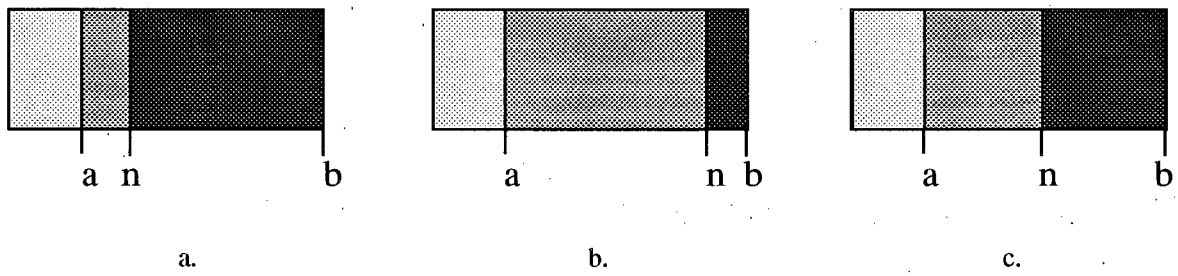
gaps in the delivery of data at the destination site (refer to Figure 3). The anomalous delivery



a. Transmission of Data
b. Reception of Data

Figure 3. Effects of Jitter

may change the semantics of the data, and result in an unintelligible presentation. Jitter may be introduced by the communication subsystem and other components (device controller, operating system, etc.). The specification of the mm presentation needs to include how sensitive the presentation of each unit is in terms of being early or late with respect to the nominal presentation time of the data. Jitter is typically defined as the difference between the latest and the earliest the data may be presented. This definition assumes the range is symmetric around the ideal presentation time. However, the range is not necessarily symmetrical (refer to Figure 4). Three scenarios may be specified: the ideal presentation time is shifted toward the smallest acceptable presentation time; shifted toward the longest acceptable presentation time; or is symmetrical. For example, if a point in a presentation requires that the presentation of the data may be very little on the early side, but is more lenient on the late side, the interval may be specified as [a,n,b] where a is the earliest firing time, n is the nominal firing time, b is the latest



a.Nominal Shifted Toward Smallest Presentation Time
b.Nominal Shifted Toward Longest Presentation Time
c. Symmetrical About Nominal Presentation Time

Figure 4. Types of Jitter

firing time and |b-n| >> |a-b|.

Two additional measurements are derived from the jitter over the entire presentation: skew and drift. Skew is the average of the jitter over the duration of the presentation. Although used frequently in the earlier literature, the overall average measure of the jitter is not very useful as a specification characteristic. A mix of early and late units may average out to a low value for the skew, yet specify a mediocre quality presentation. Drift is the sum of the jitter over the duration of the presentation. This measurement indicates how far the data streams are from being synchronized by the end of the presentation.

2.2.2.4 Sensitivity to Transmission Errors. For media which are sensitive to transmission errors, the application specification needs to describe what to do if a packet is lost, delayed too long, duplicated, or damaged. For example, an electronic transfer of funds is extremely sensitive to both bit and packet transmission errors and requires retransmission of

damaged or lost packets as well as a means of handling duplicate packets. If the quality of service degrades, the application may renegotiate the QoS parameters, continue the presentation with the available quality of service from the network, or abort the presentation. The behaviour of the application needs to be specified for the possible variation in network quality.

2.2.3 Computer-Human Interaction Requirements

The user interaction requirement describes the set of user inputs the application must support. For example, a video on demand application is expected to support a number of interactions including start, freeze, resume, change presentation speed, restart, toggle playout direction, terminate, etc. Applications using real-time sources of data may only have a small set of interactions such as start and stop.

2.2.4 Expressiveness Requirements

Expressiveness requirements fall into two categories: application driven and developer driven. An application driven requirement is the level of granularity (coarse and/or fine) for the synchronization control. Developer driven requirements include the level of abstraction and the simplicity of using the specification approach.

2.2.4.1 Level of Granularity. The granularity determines when the synchronization control is applied in the application. If the grain is coarse, for example, synchronization is enforced on a larger presentation unit such as a scene. A fine grain synchronization control enforces synchronization on a smaller presentation unit, for example, on the frames that compose a scene. As the granularity increases, the more the data streams may drift out of synchronization. In addition, the granularity of synchronization control constrains the specification of

the user response time, assuming that no interrupts of the processing is allowed. With this assumption, a user interaction is not processed until the synchronization control is exerted. The coarser the granularity of control, the longer the user must wait for a response to their interaction.

2.2.4.2 Level of Abstraction. The developer driven requirements includes the level of abstraction of the specification approach. The level of abstraction means whether the approach supports a high level and/or low level specification from the end user's point of view. Ideally, a specification approach is flexible enough to support more than one level of abstraction.

2.2.4.3 Complexity of the Specification. The developer driven requirements also include the complexity of the specification approach. The number of symbols required to express the requirements indicates how powerful the modelling approach is. More powerful approaches may produce simpler models, but usually take longer to learn. The symbols may be graphic symbols (places, transitions, arcs, or labels) or textual symbols, such as lines of code in a natural language specification.

2.3 The Relationships Among the Synchronization Requirements

The requirements discussed in the previous sections are not independent of one other. An entity relationship diagram (ERD) for the generic mm application is illustrated in Figure 5. The ERD is one means of showing the relationships among the requirements [War85]. Documenting the relationships among the requirements is important in the development of a specification because it provides an aid to the developer in maintaining a consistent specification. For example, if a live, mono voice data stream is replaced by a requirement for music from a stereo CDROM source, the following requirements must be updated: type of medium, throughput, type of data source, type of peripheral (additional speaker), and possibly the source site as well. The documented relationships assist the developer in determining which requirements must be updated so that the model is consistent.

The ERD also provides an aid in prioritizing the requirements and understanding the trade-offs with the requirements in the design phase. For example, the impact of requiring an image of an object as opposed to a voice description of an object is significant on the throughput requirements of the system. Using the image, a compression and decompression mechanism may be necessary. This design decision affects the choice of hardware for compression/decompression support, memory, processor, and input/output rates for the communication system.

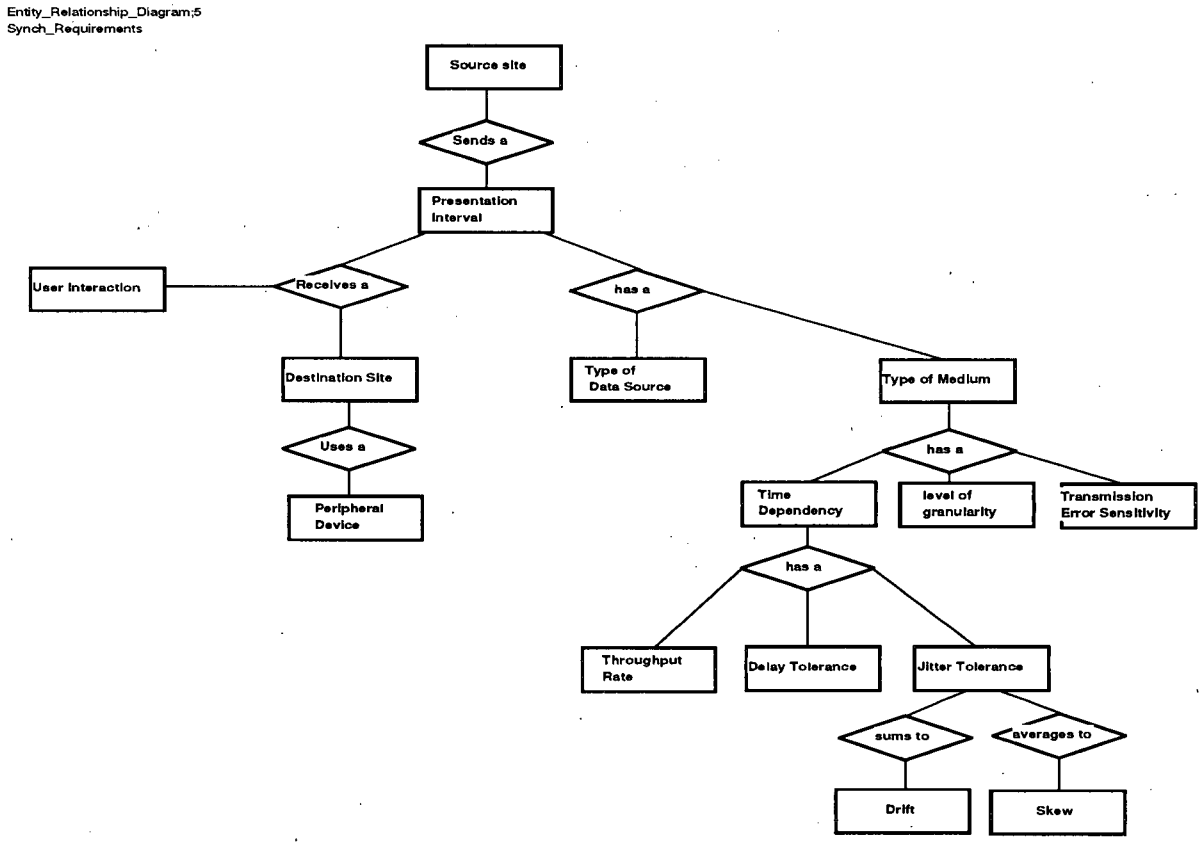


Figure 5. Entity-Relationship Diagram for Synchronization Requirements

2.4 Conclusions

New work presented in this chapter includes the requirements analysis and a high level ERD which describes the relationships among the requirements. The requirements analysis identifies a set of sixteen requirements which are classified into four categories. The analysis work is useful for two reasons. Firstly, the set of requirements for the generic distributed mm application may be used as a template to begin the analysis work on a specific application. The template may be tailored to fit the application being developed. Secondly, the classification of the requirements into categories makes the analysis and review work easier. If all sixteen requirements are described in one group, the size of the group exceeds the "7 +- 2" rule (i.e. humans analyze groups of objects best when the size of the group is between 5 and 9 objects), and the task becomes difficult [War85]. The classification is straightforward, and it is easy to classify the requirements into this organization. The ERD documents the relationships at a high level and provides an aid in maintaining the consistency of the specification during its development and in evaluating the trade-offs among requirement choices early in the development cycle.

3 Petri Net Modelling of Synchronization Requirements

3.1 Introduction

This chapter reviews three established Petri net models [Ram74, Mer76, Wal83] and presents an analysis of four recent Petri net models used for describing multimedia applications [Sen94, Lit90, Woo94, Pra93]. These recent extensions to Petri nets build on the established models reviewed. Petri nets are defined in varying but equivalent notations in the literature. The models are presented in a single notation in this thesis to make the relationships among the definitions of the models clear [Rez95]. As a result, the notation in this thesis varies from the notation used in the original papers.

Following the review and analysis work, the discussion of the multimedia Petri nets focuses on how well these methods support the specification of the requirements in the example and the complexity of the model developed [Vuo95b]. The same example is specified in each of the approaches for comparison purposes. The example includes four different types of media, varying levels of synchronization control requirements, user interactions, and jitter constraints. The natural language specification for the example is provided in Appendix A.

3.2 Established Petri Net Models

Petri net models are formal, graphic modelling techniques suitable for describing concurrent, asynchronous, distributed, non-deterministic and/or stochastic systems [Mur89]. Application areas for Petri net models are reviewed in [Rez95]. The underlying graph of a Petri net is a directed graph consisting of two kinds of nodes: places and transitions. Pictorially, a Petri net is represented by places with circles, transitions with bars, arcs with arrows, and the tokens by small black dots (refer to Table 4) [Pet81]. The places and the transitions

Name	Graphic Symbol	Informal Semantics	
Place	○	Input Place	Output Place
		Preconditions Input data Input signal Resources needed Conditions Buffers	Postconditions Output data Output signals Resources released Conclusion(s) Buffers
Transition	|	Event Computation step Signal processor Task or job Clause in logic Processor	
Arc	→ Wpt or Wtp	Indicates a relationship between a place and a transition. A weighted arc may be interpreted as parallel arcs. Arc weights equal to one are omitted by convention. Wpt is the weight for the arc relating a place to a transition. Wtp is the weight relating a transition to a place.	
Token	●	The interpretation of the place is True.	

Table 4. Petri Net Modelling Notation [Mur89]

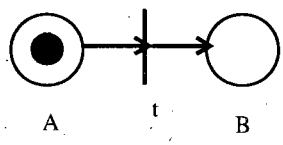
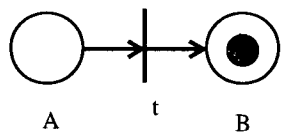
Name	Graphic Symbol	Informal Semantics
Enabled Transition t Before Firing		ex/ Precondition A is true, Event t may occur.
Enabled Transition t After Firing		ex/ Event t has occurred, Postcondition B is true.

Table 4. Petri Net Modelling Notation [Mur89]

have a variety of interpretations, depending on the application area.

The firing rules for the Petri net model are straightforward. The holding of a condition is represented by placing a token in the place. A transition may fire if all of the input places to that transition have at least the arc weight, W_{pt}, number of tokens. If more than one transition in the model are enabled, only one of the transitions may fire. The selection of the enabled transition to fire is non-deterministic. The transition fires instantaneously. When a transition fires, it removes W_{pt} tokens from each input place and puts W_{tp} tokens in each output place.

Formally, a marked Petri net is a bipartite, directed graph defined by the tuple (P,T, W_{pt}, W_{tp},M(0)) where:

$P = \{p_1, p_2, p_3, \ldots, p_m\}$ is a set of places, $m \geq 0$

$T = \{t_1, t_2, t_3, \ldots, t_n\}$ is a set of transitions, $n \geq 0, P \cap T = (T \cap P) = \varnothing$

$A \subseteq \{T \times P\} \cup \{P \times T\}$ is a set of arcs,

$W_{pt} : \mathrm{A} \rightarrow I$, $I = \{0, 1, 2, 3 \ldots\}$ the weighting function maps an integer value to an arc,

$W_{tp} : \mathrm{A} \rightarrow I$, $I = \{0, 1, 2, 3 \ldots\}$ the weighting function maps an integer value to an arc,

$M(0) = [M_1(0) M_2(0) M_3(0) \ldots M_n(0)]^T$, $M(0) = P \rightarrow I$, $I = \{0, 1, 2, \ldots\}$ M(0) is the initial

marking,

At any arbitrary time k, the marking is represented as

$$M(k) = [M_1(k) M_2(k) M_3(k) \ldots M_n(k)]^T .$$

The dynamics of the Petri net model are tracked by the movement of tokens via firing transitions. A transition may fire when it is enabled. Firing removes W_{pt} tokens from each input place and adds W_{tp} tokens to each output place for the firing transition. Formally, this is described below:

$^\circ t = \{p \mid (p, t) \in A\}$ = the set of input places of t

$t^\circ = \{p \mid (p, t) \in A\}$ = the set of output places of t

$^\circ p = \{t \mid (p, t) \in A\}$ = the set of input transitions of p

$p^\circ = \{t \mid (p, t) \in A\}$ = the set of output transitions of p

$\neg\exists t^\circ \mid t = t^\circ \neq \varnothing, \neg\exists p^\circ \mid p = p^\circ \neq \varnothing$ are assumed, meaning there are no isolated places and transitions.

A transition $t \in T$ is enabled under a marking M(k) of a Petri net iff

$$\forall p \in {}^\circ t: \mathrm{M}_p(k) \geq W_{pt}(p, t) .$$

When an enabled transition fires, the new marking is defined by:

$$\mathrm{M}_p(k+1) = \mathrm{M}_p(k) - W_{pt}(p, t) ,$$

$$\mathrm{M}_p(k+1) = \mathrm{M}_p(k) + W_{tp}(t, p) ,$$

$$\mathrm{M}_p(k+1) = \mathrm{M}_p(k) \text{ otherwise} .$$

3.2.1 Timed Petri Nets

The first introduction of time in Petri nets is in the Timed Petri net model [Ram74]. In

this model, a time duration is associated with each transition. The firing rules in this model are that the transition must fire as soon as it is enabled, and firing a transition takes a fixed, finite amount of time. The notion of instantaneous firing of transitions is not preserved in the Timed Petri net model. When a transition becomes enabled, the tokens are immediately removed from its input places. After the time delay, tokens are deposited in the output places. The result is that the state of the system is not always clearly represented during the process. A second weakness is that the Timed Petri net model does not describe variability in timing. Playout duration may be described with this approach, but not jitter constraints.

3.2.2 Time Petri Nets

The Time Petri net model may describe timing uncertainty with a fixed, finite interval [x,y] that is associated with each transition [Mer76]. The firing rules of Time Petri nets assume that if a transition t with a firing interval [x,y] is enabled at the absolute time τ then, while being continuously enabled:

- t may not fire before time $\tau + x$;
- t must fire on or before time $\tau + y$.

The notion of instantaneous firing of transitions is preserved in the Time Petri net model. The marking of the net is not updated until the transition fires. If several places are inputs of one timed transition, the timer associated with that transition only starts when the transition is enabled. Therefore, the latest place always determines when the transition becomes enabled. This approach is suitable for describing intrastream synchronization, but does not allow the correct specification of interstream synchronization. Synchronization intervals are permitted in the model, making interstream synchronization impossible. If this condi-

tion occurs, the presentation stops.

3.2.3 Arc Time Petri Nets

In the Arc Time Petri net model, the temporal intervals are associated with the arc [Wal83]. When a place receives a token a local timer is started for the place. If the place P_i is marked at absolute time τ_i and $[x_{ij},y_{ij}]$ is the interval associated with the arc (p_i,t_j) then, while being continuously enabled:

- t_j may not fire before time $\tau_i + x_{ij}$;
- t_j must fire on or before $\tau_i + y_{ij}$;
- $[\tau_i + x_{ij}, \tau_i + y_{ij}]$ is called the dynamic firing interval of the arc (p_i,t_j).

The notion of instantaneous firing of transitions is preserved in the Arc Time Petri net model because the marking of the net is not updated until the transition fires. Interstream synchronization is addressed in the Arc Time model. For a transition that synchronizes multiple places, the firing conditions must hold for all of the arcs. This means that the transition t_j may fire if the intersection set of all dynamic intervals related to all of its input arcs is not empty. At least one value of time allows the firing of all arcs entering the transition t_j. Time mismatches or empty intersections of the dynamic intervals are prohibited, but nothing in the model ensures the continuity of the presentation if these events occur. Like in the Time Petri net model, the presentation may stop.

3.3 Multimedia Synchronization Petri Net Models

A review of four recent Petri net model extensions for describing multimedia synchronization requirements is presented in this section. The extensions are: object composition, extended object composition, time stream, and dynamic time Petri nets. Their definitions are provided and their strengths and weaknesses in specifying the synchronization requirements identified in Chapter 1 are discussed. A common example is worked out in the four approaches as a basis for comparison. The informal, natural language specification for the example is available in Appendix A. The example contains a variety of requirements including time independent and time dependent data, jitter constraints, four different types of media, and a set of user interactions.

3.3.1 Object Composition Petri Nets

The Object Composition Petri net (OCPN) model is an enhanced version of the established Timed Petri net model [Lit90]. The OCPN has resources, input data, and durations associated with the places. For example, a single place may represent the presentation of 10 seconds of video data. The approach is different from the original Timed Petri net model, in which the time duration is associated with the transition. Assigning the time to the transition, as in the Timed Petri net approach, or to the place, as defined in the OCPN method, are equivalent [Sif80].

The transitions of the net are the explicit inter-object synchronization points. The semantics of the model are controlled by a set of firing rules which preserve the instantaneous firing of a transition. This is also in contrast with the original Timed Petri net model, in which firing a transition takes a fixed amount of time. Assuming a weighting function of one, the fir-

ing rules for the OCPN model are:

- A transition t_i fires immediately when each of its input places contain an unlocked token.
- Upon firing, the transition t_i removes a token from each of its input places and adds a token to each of its output places.
- After receiving a token, a place p_i remains in the active state for the interval specified by the duration τ_j. During this interval, the token is locked. When the place becomes inactive, or upon expiration of the duration τ_j, the token becomes unlocked.

3.3.1.1 Formal Definition of the OCPN. Formally, the OCPN is defined as a tuple (T, P, W_{pt}, W_{tp}, D, Rs, M(0)), where:

$(P, T, W_{pt}, W_{tp}, M(0))$ defines a marked Petri net;

$D : \mathrm{P} \rightarrow R$ is the nominal playout duration, $R \geq 0$;

$Rs : \mathrm{P} \rightarrow (r_1, r_2, r_3, \ldots r_k)$ represents the resources required at each place.

3.3.1.2 Graphic Notation and Example Specification. The graphic notation for a single place and transition is provided in Figure 6. Note that the label for the place identifier is not included in the formal definition. This convention is followed in Petri net modelling. The example is specified using the OCPN approach in Figure 7. Note that the specification is very concise and easy to review. However, a number of the example's requirements are not described in the model including: user interactions, jitter, and the fine grained synchronization

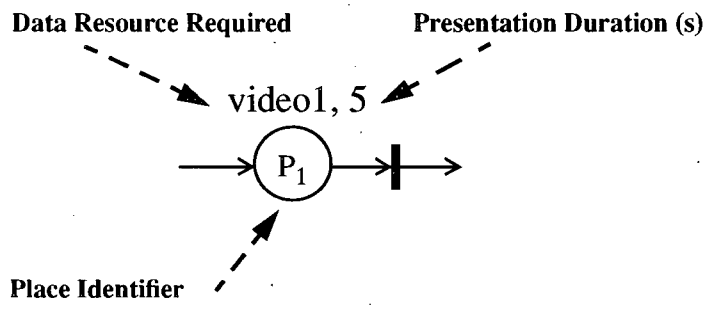


Figure 6. The OCPN Graphic Notation

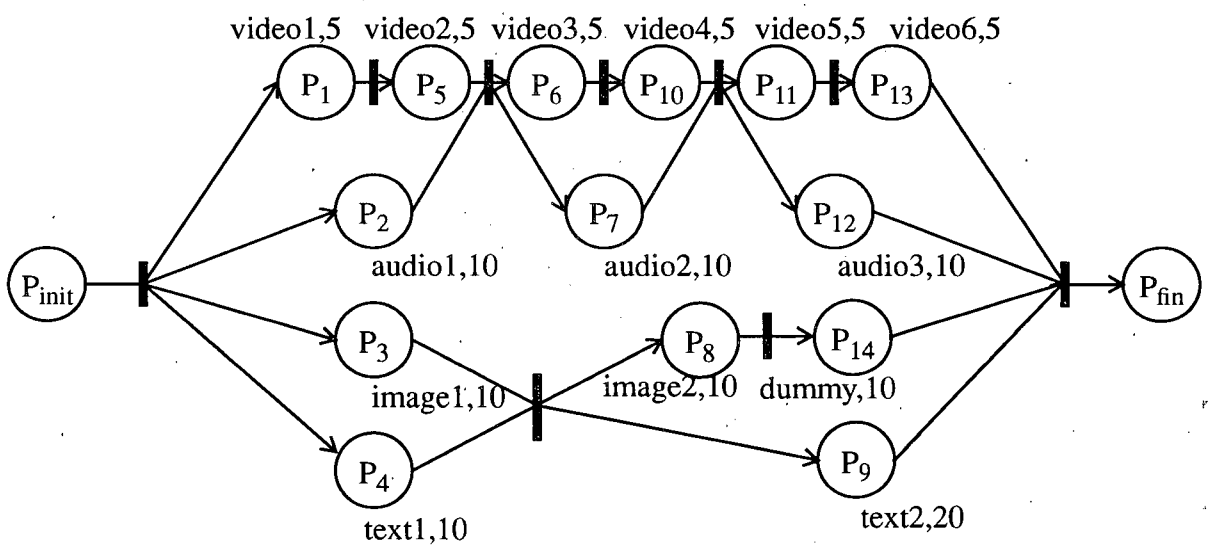


Figure 7. The OCPN Specification

requirements. Also note that the specification only supports the playout of the presentation in the forward direction.

3.3.2 Time Stream Petri Nets

The time stream Petri net (TSPN) model integrates the Arc Time Petri net and the OCPN approaches and further extends the hybrid model with typed transitions[Sen94]. The TSPN model has several advantages: it addresses the finer grained synchronization requirements for isochronous data; ensures the continuity of presentation with the use of typed transitions; and may be used to specify jitter requirements.

In the TSPN model, arcs leaving the places are labelled with a tuple $[\alpha, \eta, \beta]$ representing the earliest firing time (EFT), the nominal duration (D), and the latest firing time (LFT), respectively. The tuple, derived from the Arc Time Petri net approach, is extended with the nominal duration time, D, from the OCPN model. The intervals are used to describe the jitter requirements for the presentation.The difference between the EFT and the D specifies the tolerable jitter on the early side while the difference between the LFT and D specifies the tolerable jitter on the late side. The TSPN model defines six synchronization time instants which are used to state time correctness. These instants may be distinguished between two points of view: a modelling and a run-time point of view. The synchronization times are summarized in

Table 5.

Synchronization Instants	Modeling Point of View	Runtime Point of View
τ_i^{tok}	When the token enters place p_i and enables the arc leaving it.	When the multimedia object starts to be processed.
τ_i^{min}	When the age of the token in place p_i reaches the minimum temporal bound x_i (i.e. $\tau_i^{tok} + x_i$)	When the processing time reaches x_i time units (its minimal time semantics). The runtime should commit for a minimal playout duration of x_i time units for the information unit related to the arc a_i.
τ_i^{end}	This instant is not covered by the modelling point of view. So the firing rules for TSPN defined later on induce a firing interval, not only one time instant, for admissable synchronization instants.	When the processing of the multimedia object actually ends. Ideally, this time equals $\tau_i^{tok} + n_i$, where ni is the nominal duration of the multimedia object; however, the temporal semantics of the multimedia object is preserved if it occurs during the temporal interval: $[\tau_i^{tok} + x_i, \tau_i^{tok} + y_i]$.
τ_i^{max}	When the age of the token in the place p_i reaches the maximum temporal bound y_i (i.e. $\tau_i^{tok} + y_i$) The duration yi is the maximum playout duration acceptable by the user for the information unit associated with arc a_i.	When the processing time reaches y_i time units (its maximal time semantics). The runtime should commit for a maximal playout duration of y_i time units for the information unit related to the arc a_i.
τ^{tra}	When the transition becomes enabled, i.e. the first instant when all arcs are enabled, $max_i\left(\tau_i^{tok}\right)$	When the processing of all the multimedia objects associated with a synchronization point has begun.
τ^{fir}	When the transition is fired. Defined by the firing rules as an interval of admissible firing times.	When the synchronization point is met. Defined by the runtime scheduler as a unique time instant function of runtime events.

Table 5: Synchronization Time Instants [Sen94]

The firing rules for intrastream synchronization use these concepts to guarantee that,

for each medium, $\tau^{fir} \in [\tau^{min}, \tau^{max}]$ even if $\tau^{end} \notin [\tau^{min}, \tau^{max}]$. In the case of interstream synchronization this condition must hold for all arcs entering the transition, i.e., $\forall i, \tau_i^{end} \cap i[\tau_i^{min}, \tau_i^{max}]$. In the later case, if the condition doesn't hold, interstream strategies labelled at the transitions are applied to preserve the temporal semantics of the presentation.

The dynamic firing interval represents the range in time during which one of the enabled transitions must fire. It is updated after a transition is fired. The minimum value for the dynamic firing interval is always the minimum of all of the EFTs for the calculated firing intervals for each transition. The maximum value for the firing interval is set to the minimum values of all of the LFT for the calculated firing intervals for each transition. The firing time calculation for each transition is determined by the type of transition. The rules for the three basic types of transitions are discussed in the sections on strong-or, weak-and, and master below. After a transition fires at some time during this composite firing interval, the earliest, nominal, and latest firing times for each of the previously enabled arcs are updated to show time has passed. The new dynamic firing interval is calculated from the updated intervals of the previously enabled arcs and from the newly enabled arcs. The newly enabled arcs are initialized with their static interval values.

Nine models for interstream synchronization are described in [Sen94]: or, strong or, and, weak and, master, or-master, and-master, weak-master, and strong-master. The three basic models of interstream synchronization are provided by the strong-or, weak-and, and master transition types.

3.3.2.1 Strong-or. In the strong-or strategy the presentation of one medium to its EFT is sufficient to satisfy the global presentation semantics. The firing interval for the strong-or type

transition is calculated as follows. The EFT for the transition is set to the smallest EFT of all of the arc intervals entering the transition. The LFT for the transition is set to the smallest LFT of all of the arc intervals entering the transition. The result is that the earliest time during which the synchronization event may occur is determined by the presentation object with the smallest EFT and the latest time is determined by the presentation object with the smallest LFT. This transition type does not guarantee the temporal correctness of the streams with larger EFTs because firing the transition may cut off the presentation of the other streams before their earliest firing times haves arrived. Formally, the composite firing interval for a strong-or transition type is a transition t is firable at time $\tau^{tra} + \theta$,where θ is the relative firing time to the absolute first instant of firable time τ^{tra}:

$$MIN_m = min_i\{\alpha_i\} \leq \theta \leq min_i\{\beta_i\} = MAX_m.$$

3.3.2.2 Weak-and. In the weak-and strategy the presentation of all media to their EFTs satisfies the global presentation semantics. The firing interval for the strong-or type transition is calculated as follows. The EFT for the transition is set to the largest EFT of all of the arc intervals entering the transition. The LFT for the transition is set to the largest LFT of all of the arc intervals entering the transition. The result is that the earliest time during which the synchronization event may occur is determined by the presentation object with the largest EFT and the latest time is determined by the presentation object with the largest LFT. This transition type does not guarantee the temporal correctness of the streams with smaller LFTs because firing the transition may occur after their LFTs have occurred. Formally, the composite firing interval for a weak-and transition

type is a transition t is firable at time $\tau^{tra} + \theta$ where θ is the relative firing time to the absolute first instant of firable time τ^{tra}:

$$MIN_m = max_i\{\alpha_i\} \leq \theta \leq max_i\{\beta_i\} = MAX_m.$$

3.3.2.3 Master. The master strategy ensures the temporal correctness for the master stream. The entire presentation of at least the master medium is necessary in order to satisfy the global presentation semantics. The firing interval for the master type transition is calculated as follows. The EFT for the transition is set to the EFT of the master stream. The LFT for the transition is set to the LFT of the master stream. This transition type does not guarantee the temporal correctness of the streams with smaller LFTs because firing the transition may occur after their LFTs have occurred. Formally, the composite firing interval for a master transition type is a transition t is firable at time $\tau^{tra} + \theta$ where θ is the relative firing time to the absolute first instant of firable time τ^{tra}:

$$MIN_m = \alpha_m \leq \theta \leq \beta_m = MAX_m$$

3.3.2.4 Formal Definition of the TSPN. A TSPN is defined as a tuple (T, P,W_{pt},W_{tp},M(0),IM, SYN) where:

$(P, T, W_{pt}, W_{tp}, M(0))$ defines a marked Petri net;

$A \subseteq \{T \times P\} \cup \{P \times T\}$ is a set of arcs;

$B \subseteq A, B = \{b = (p, t) \in P \times T / W_{pt} \neq 0\}$ is a set of arcs in the flow direction from the place to the transition;

$IM : B \rightarrow (Q^+ \cup \infty) \times (Q^+ \cup \infty) \times (Q^+ \cup \infty)$ IM is a function that maps arcs entering transitions into rational numbers representing the EFT, D, and the LFT, respectively; Q^+ is the set of rational numbers greater than or equal to zero.

$SYN : T \rightarrow$ {or, strong_or, and, weak_and, master, or-master, and-master, weak-master, strong-master} SYN is the typing function that distinguishes between the different firing rules (i.e. the different semantics for interstream synchronization). Note that the intrastream synchronization transitions are not typed. They use the Time Petri net firing rules.

3.3.2.5 Graphic Notation and Example Specification. The graphic notation for a single place and transition is provided in Figure 8. The TSPN approach does not define the resources for a place formally. In this model, the resources are identified in the notation as an informal label. The example is specified using the TSPN approach in Figure 9. The TSPN specification

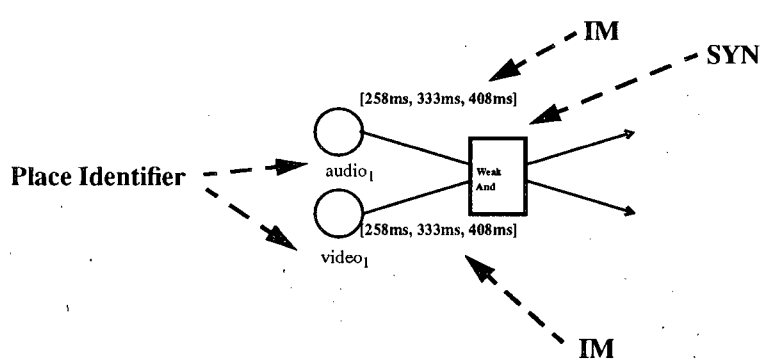


Figure 8. The TSPN Graphic Notation

is more complex than the OCPN specification. There is considerable repetition in the specification and the lack of hierarchies contribute to the complexity. The benefits, however, include

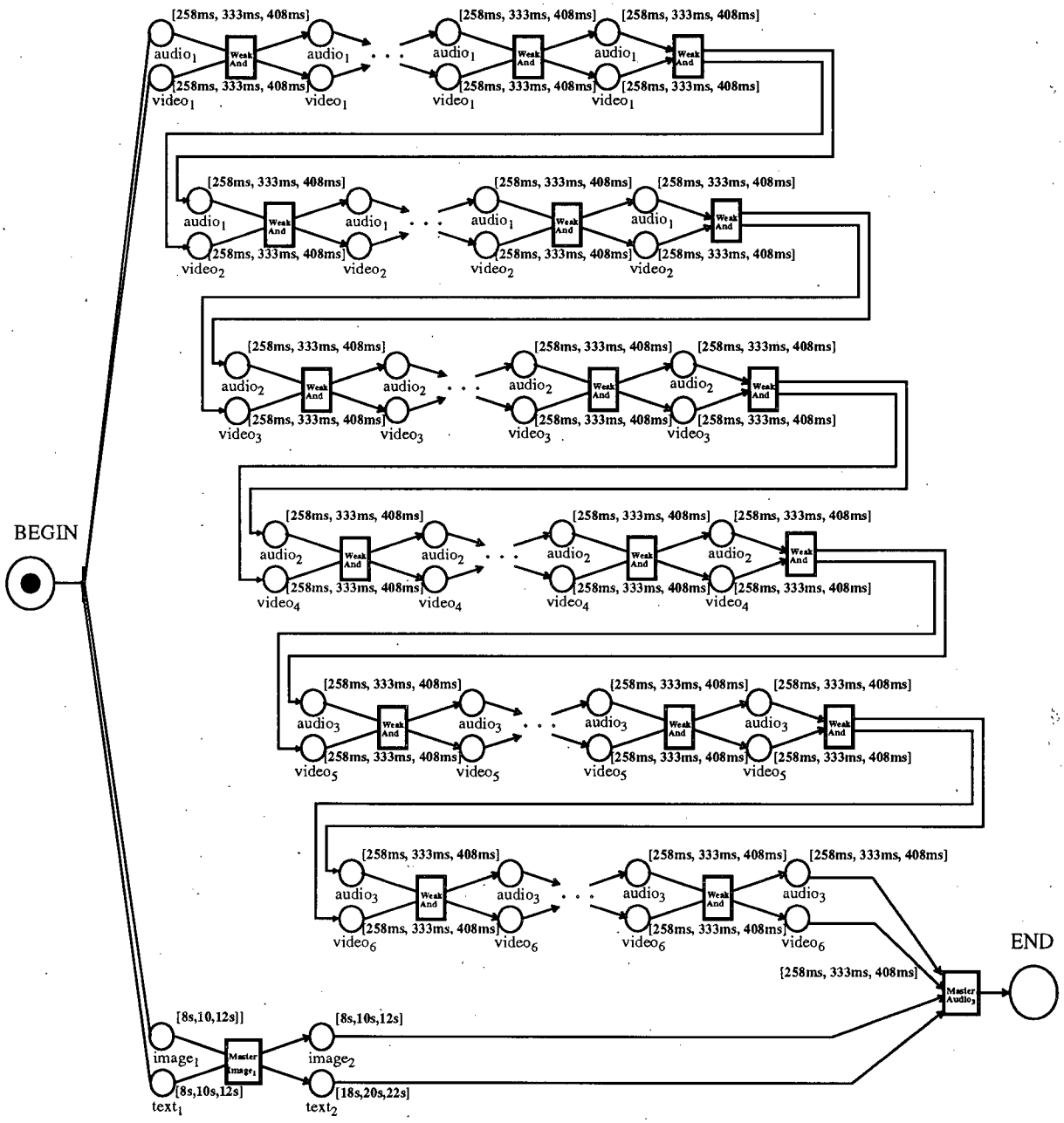


Figure 9. The TSPN Specification

that it does model the jitter and the fine grained synchronization requirements. The maximum jitter tolerance for the processing place is derived from the IM definition as the difference between the LFT and the EFT.

3.3.3 Extended Object Composition Petri Nets

The XOCPN model is an enhanced version of the OCPN method [Woo94]. It allows both coarse and fine grained synchronization for continuous media objects. Therefore, the XOCPN model overcomes the limitation of the coarse grained, object level synchronization in the OCPN approach. In addition, the XOCPN model addresses modelling the quality of service (QoS) specification requirements. Note that the QoS requirements are not described in the OCPN approach.

To achieve finer synchronization control the temporal interval associated with an object is divided into a sequence of smaller units, called synchronization interval units (SIUs). For example, a video object may be divided into multiple SIUs, each one holding information for a video frame. The size (duration) of a SIU depends on the type of the object. In addition, a number of intermediate synchronization points, Interstream Pacing Points (IPP), are inserted at appropriate, precalculated points in the string of SIUs. These IPP places are used to represent fine grained interstream synchronization control processing. The type of media and the level of human perception involved dictates the interval between two IPPs. For the object level synchronization, the XOCPN model defines the Inter-object Synchronization Point (ISP), which corresponds to the transition points in OCPN. For isochronous data this level of synchronization may be too coarse.

Processing in the places is also described in more detail in comparison to the OCPN

model. The places in the XOCPN may be of two types: object places and control places. Object places specify actions for the playout or the transmission of the object. Control places denote controlling actions according to the XOCPN semantics. These actions are used to setup a virtual channel, negotiate QoS parameters, release resources upon the playout, and define interstream synchronization policies.

Unlike the OCPN model, the number of specifications models described in the XOCPN is two: distinct models are defined for the source and receiver sites. The transmitter model schedules the transmission of multimedia objects. The receiver model plays out the multimedia objects and introduces actions for maintaining the synchronization during the presentation. The use of the two models is advantageous when there are requirements that are unique to a specific site. A disadvantage is the duplication of common requirements at both the transmitter and receiver sites.

The intrastream synchronization makes use of the concept of blocking and restricted blocking policies. In the blocking policy the playout action may be suspended until that time by which the late SIU arrives. In the restricted blocking policy the playout process blocks for a pre-specified period of time only while it waits for the current SIU to arrive. If a time-out occurs the most recent stored SIU is played back, skipping the late SIU. This synchronization policy is not general enough to describe the requirements for some applications. For example, if restricted blocking is used to ensure the playout of a presentation continues, replaying the previous audio SIU may not be appropriate. Blocking and restricted blocking may also be applied to interstream synchronization.

3.3.3.1 Formal Definition of the XOCPN. Formally, the XOCPN model is defined as a

tuple (T, P, Wpt, Wtp, D, Rs, M(0), Y, Z) where:

$(P, T, Wpt, Wtp, M(0))$ defines a marked Petri Net;

$D : \mathrm{P} \rightarrow (d_1, d_2)$ represents the delay before an action and the action duration, respectively;

$Rs : \mathrm{P} \rightarrow (r_1, r_2, r_3, \ldots r_n)$ is a mapping from places to resources;

$Y : \mathrm{P} \rightarrow$ (Resource_setup, Resource_release, SIU_playout, SIU_transmit, Interstream_synchronization) is a mapping from places to actions to be performed during communication;

$Z : \mathrm{P} \rightarrow$ (address_for_QoS, address_for_SIU, address_for_synchronization_requirements) .

3.3.3.2 Graphic Notation and Example Specification. The graphic notation for a single place and transition is provided in Figure 10. The XOCPN approach defines the resources for a place formally, like in the OCPN approach. The example shows a SIU place. In this model, the resources are identified in the notation as an informal label. The Z component of the formal definition is not part of the graphic notation. The example is specified using the XOCPN

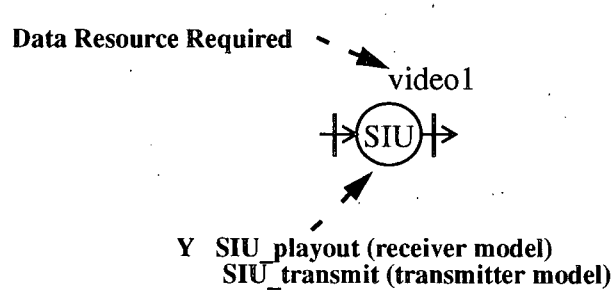
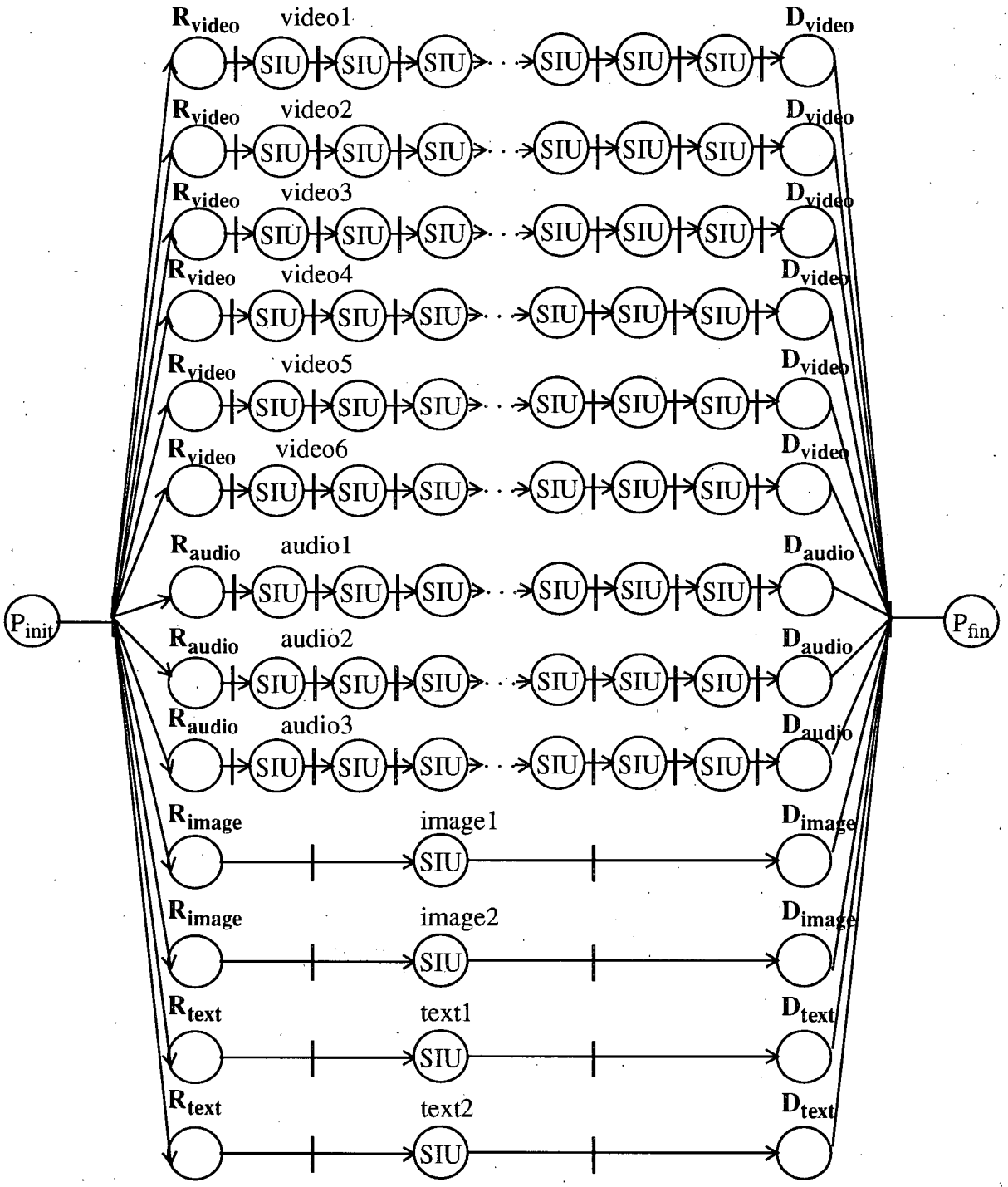


Figure 10. The XOCPN Graphic Notation

approach in Figure 11 and Figure 12. The specification is very large, and yet does not model all of the requirements in the example. The XOCPN may not describe the user interactions or



R Resource allocation
D Resource release

Figure 11. The Transmitter XOCPN Specification

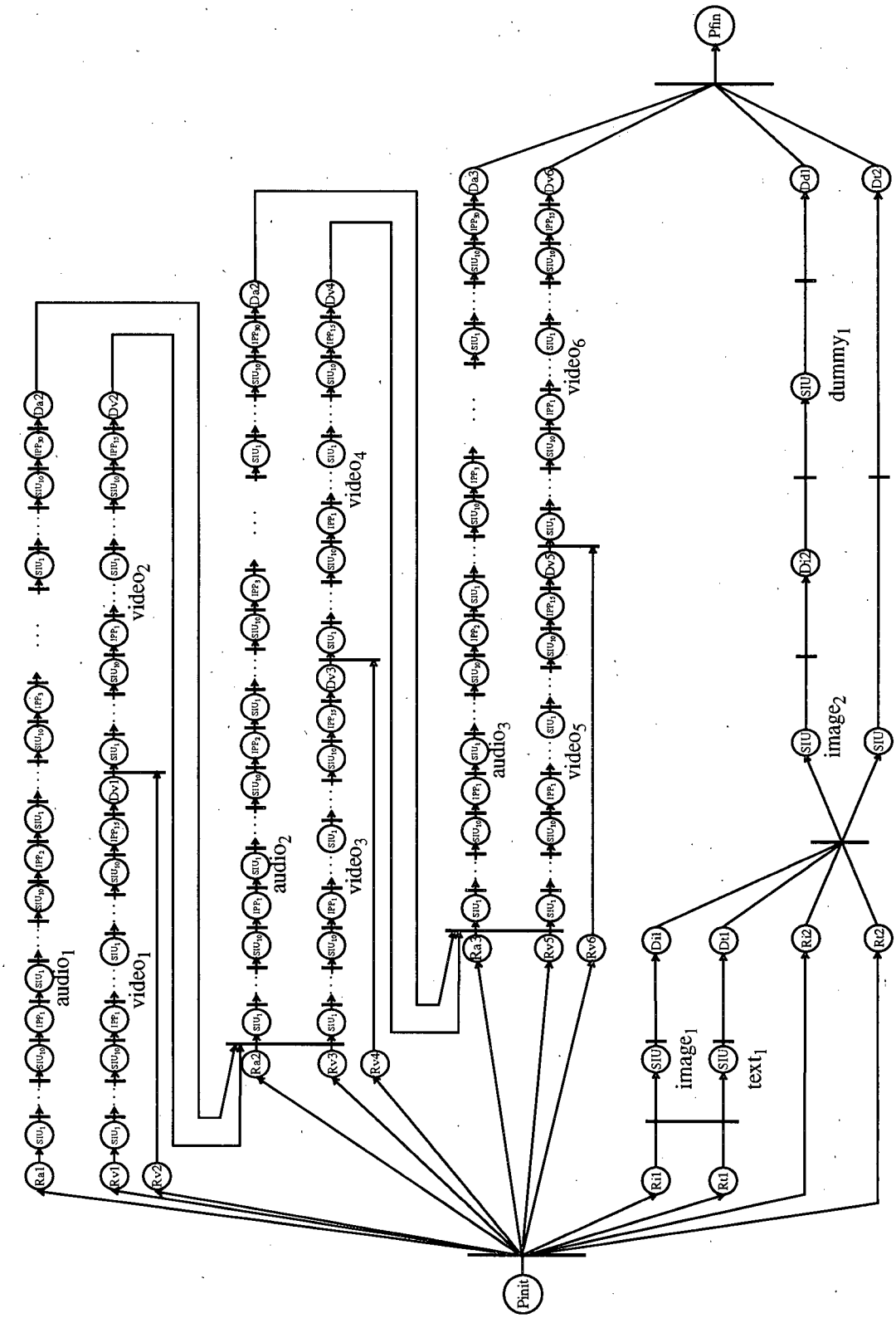


Figure 12. The Receiver XOCPN Specification

the jitter requirements. From the graphic specification, it is also not clear how the IPPs relate the different streams. The approach of using places to control synchronization instead of transitions creates a very streamlined graphic specification at the expense of showing the synchronization relationships explicitly.

3.3.4 Dynamic Timed Petri Nets

The Dynamic Timed Petri Net (DTPN) model is an extension of the OCPN approach which supports the specification of a number of user interactions [Pra93]. Like the OCPN model, the places represent the processing and the transitions represent the synchronization events and fire instantaneously. The DTPN model extends the OCPN model to include typical user inputs such as requests to skip, reverse, freeze, restart, and scale the speed of a presentation to make it faster or slower. User interactions are supported by allowing the playout duration to be pre-empted and the execution or duration time to be updated. The specification of user interactions is not tied to the coarsest granularity of the presentation units. Recall that in the OCPN model, playout durations in the places are not pre-emptable.

The user interactions may be added as required in the DTPN model, and are encapsulated using hierarchy. The number of different combinations of user interactions required by the application determines the number of hierarchies in the model. In the DTPN model, a place is pre-emptable and its execution duration value may be modified. When a place is pre-empted, its execution duration may be modified temporarily (P & TM), modified permanently (P & PM), terminated (P & T), or deferred (P & D). Temporary modification of the execution time refers to setting the execution duration for the current playout to a positive real number. Permanent modification refers to setting the nominal playout duration for every activation of

the Petri net place to a positive real number and updating the currently active or frozen execution times. Termination of the execution is accomplished by setting the execution duration for the place to zero for the current execution. The deference of execution involves storing the execution time, pre-empting the active place, and then terminating the presentation playout by setting its execution time to zero.

Three types of data streams are defined in the DTPN model: primary; dependent secondary; and independent secondary. The stream to which the interaction is applied is defined as the primary stream; others are secondary streams. The secondary streams are classified into dependent or independent secondary streams, depending on when the next synchronization event occurs with respect to the primary stream. A dependent secondary stream is one in which the next synchronization event in the secondary stream occurs before or at the same time as the next synchronization event in the primary stream. An independent secondary stream is one in which the next synchronization event in the secondary stream occurs after the next synchronization event in the primary stream. The terms before and after are defined relative to the current presentation playout direction. Note that the playout durations of the "independent" streams are affected by the user interactions and require playout duration adjustments.

The definition of a secondary stream as either dependent or independent may depend on when the user interaction occurs. This is illustrated using the on-going example. If the video stream is defined as the primary stream and the skip ahead user interaction occurs at $0 < t_{ui} \leq 5$, where t_{ui} is the execution time from the start of the presentation until the user interaction occurs, then the first audio, text, and image streams are defined as independent data

streams (refer to Table 6). However, if the user interaction occurs such that $5 < t_{ui} \leq 10$,then the first audio, text, and image streams are defined as dependent data streams. The reason the secondary data streams flip from being independent to dependent is because the primary stream is based on a smaller presentation duration. The presentation units which change from independent to dependent are bolded in Table 6. The dynamic nature of these definitions is modelled

Data Stream	$0 \leq t_{ui} < 5$	$5 \leq t_{ui} < 10$	$10 \leq t_{ui} < 15$	$15 \leq t_{ui} < 20$	$20 \leq t_{ui} < 25$	$25 \leq t_{ui} < 30$
Audio 1	Independent	Dependent	NA	NA	NA	NA
Audio 2	NA	NA	Independent	Dependent	NA	NA
Audio 3	NA	NA	NA	NA	Independent	Dependent
Video 1	Primary	NA	NA	NA	NA	NA
Video 2	NA	Primary	NA	NA	NA	NA
Video 3	NA	NA	Primary	NA	NA	NA
Video 4	NA	NA	NA	Primary	NA	NA
Video 5	NA	NA	NA	NA	Primary	NA
Video 6	NA	NA	NA	NA	NA	Primary
Image 1	Independent	Dependent	NA	NA	NA	NA
Image 2	NA	NA	Independent	Dependent	NA	NA
Text 1	Independent	Dependent	NA	NA	NA	NA
Text 2	NA	NA	Independent	Independent	Independent	Dependent
Dummy 1	NA	NA	NA	NA	Independent	Dependent

Table 6: Data Stream Definitions Video Primary

using an additional hierarchy. It is composed of the union of the hierarchies for the independent and the dependent secondary streams, making the model more complicated. In the cases where a different data stream is defined as the primary stream (e.g. audio, image, or text), the secondary data streams do not change definitions. The results of using the audio stream as the primary data stream are summarized in Table 7. The results of using the image or text streams as the primary stream are very similar to Table 7, and are not included. In the DTPN model,

Data Stream	$0 \le t_{ui} < 5$	$5 \le t_{ui} < 10$	$10 \le t_{ui} < 15$	$15 \le t_{ui} < 20$	$20 \le t_{ui} < 25$	$25 \le t_{ui} < 30$
Audio 1	Primary	Primary	NA	NA	NA	NA
Audio 2	NA	NA	Primary	Primary	NA	NA
Audio 3	NA	NA	NA	NA	Primary	Primary
Video 1	Dependent	NA	NA	NA	NA	NA
Video 2	NA	Dependent	NA	NA	NA	NA
Video 3	NA	NA	Dependent	NA	NA	NA
Video 4	NA	NA	NA	Dependent	NA	NA
Video 5	NA	NA	NA	NA	Dependent	NA
Video 6	NA	NA	NA	NA	NA	Dependent
Image 1	Dependent	Dependent	NA	NA	NA	NA
Image 2	NA	NA	Dependent	Dependent	NA	NA
Text 1	Dependent	Dependent	NA	NA	NA	NA
Text 2	NA	NA	Independent	Independent	Dependent	Dependent
Dummy 1	NA	NA	NA	NA	Dependent	Dependent

Table 7: Data Stream Definitions Audio Primary

the number and complexity of hierarchies in the model is determined by which data stream is defined as the primary data stream.

The number of hierarchies used in a specification may be reduced by removing the P & TM user interactions which are not necessary for the model. In the example worked out in Figure 6, the dependent secondary streams never need a skip ahead (P & TM) user interaction because their intervals always end in synchronization with the primary stream (refer to Table 7). The hierarchy which provides this skip ahead (P & TM) user interaction is deleted.

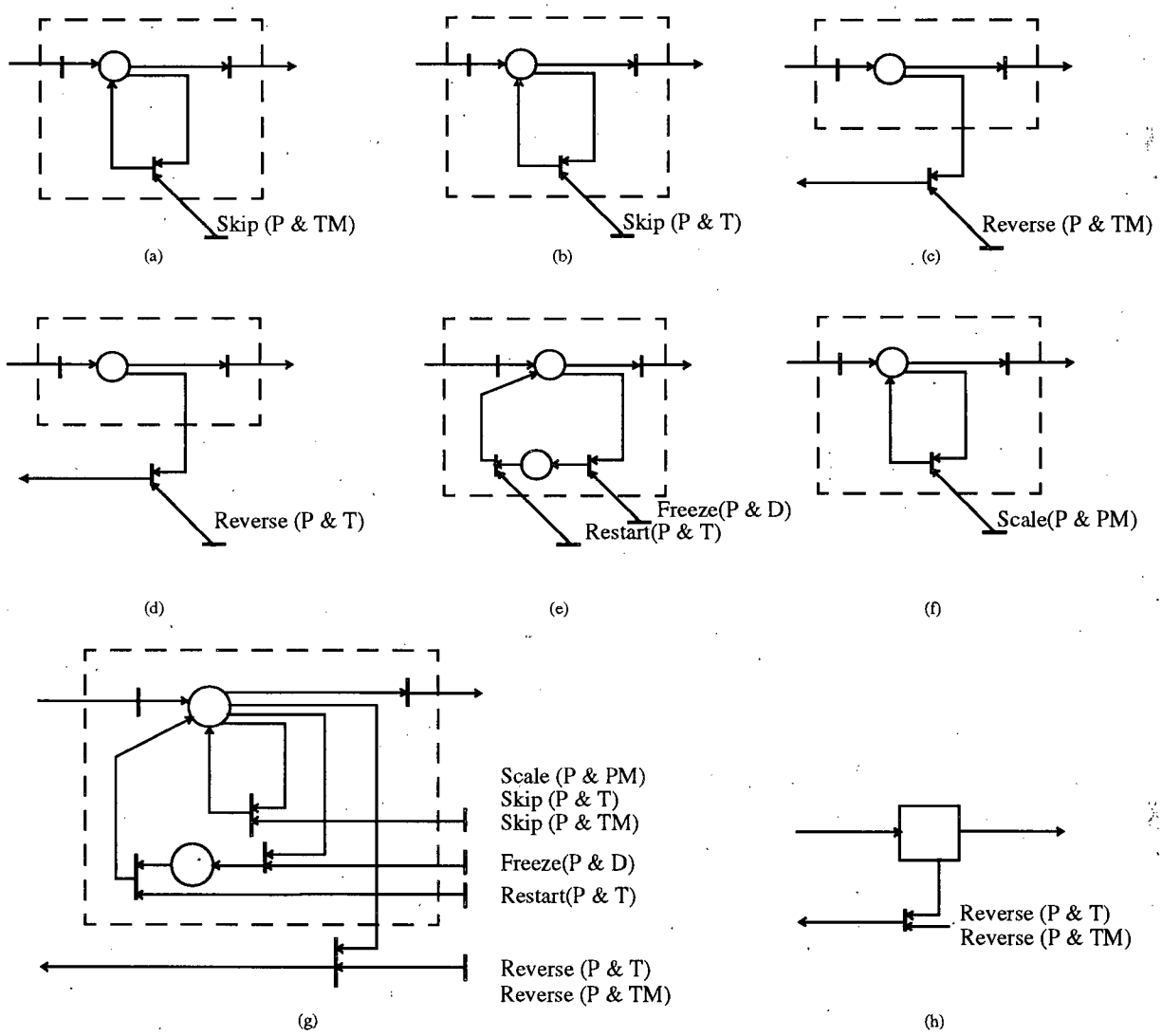
Finding the simplest model, defined by the lowest number and complexity of hierarchies, involves working through each possible definition of primary stream and simplifying the resulting model. For larger applications, this activity is time consuming and prone to errors.

The user interactions supported by the DTPN model are reviewed in the following sec-

tions. The graphic notation for the components is illustrated in Figure 13.

3.3.4.1 Skip Ahead . When a skip interaction occurs, the active presentation unit in the primary data stream is pre-empted and terminated, allowing the presentation to skip to the next presentation unit in the primary data stream. The granularity of the primary stream determines the number of time units to skip ahead. Each of the dependent secondary streams are pre-empted and terminated until the events of the dependent secondary stream synchronize with the primary stream. This specification approach assumes that a dependent secondary source's playout duration ends at the same time as the pre-empted and terminated place in the primary stream. A small correction allowing the presentation units which do not end exactly at the same time as the primary unit to pre-empt and temporarily modify the execution time is necessary to make the approach general. The independent secondary data streams are pre-empted and the playout execution times are decreased to synchronize with the primary stream. Figures 13 (a) and 13 (b) illustrate how to model the skip user interaction for the different data streams. The logic necessary to control updating the primary stream and synchronizing the secondary streams is provided in Appendix B. Note that only the primary data stream is defined, leaving the determination of the independent or dependent secondary stream to be included in the logic. In addition, each of the data streams is processed in series, and the occurrence of blocked interstream transitions is considered for both the primary and the dependent secondary streams. Although not addressed in the DTPN model, handling the interstream transition blocking is essential, and adds complexity.

3.3.4.2 Reverse. When a reverse presentation interaction occurs, the primary data stream is pre-empted and terminated, and the playout direction is reversed. The first presentation unit



P & TM: Pre-emption and Temporary Modification

P & PM : Pre-emption and Permanent Modification

P & T: Pre-emption and Termination

P & D: Pre-emption and Deference

(a) Skip on Independent Secondary Source Stream
(b) Skip on Primary or Dependent Secondary Source Stream
(c) Reverse on Independent Secondary Source Stream
(d) Reverse on Primary or Dependent Secondary Source Streams
(e) Freeze and Restart on Any Source Stream
(f) Scale Presentation Speed on Any Source Stream
(g) Combined User Interaction Operations
(h) Replacement Hierarchy for User Interaction Operations

Figure 13. The DTPN User Interaction Hierarchy

played in reverse in the primary stream is the last presentation unit that was played out completely in the forward direction. To achieve this, the currently active presentation unit in the primary stream is pre-empted and terminated, and the current playout direction is reversed. The active presentation units of dependent secondary streams are pre-empted and either terminated or modified, and the current playout direction is reversed. Subsequent presentation units of the secondary data streams are pre-empted and either terminated or modified until they synchronize with the primary stream. The execution times of the currently active presentation unit in the independent secondary data streams are pre-empted and modified, and the current playout directions are reversed. The independent data streams' execution time are modified to equal their current execution time plus the difference between the normal duration and the execution time for the presentation unit in the primary stream that is active when the user interaction occurs. Figure 13 (c) illustrates how to model the reverse presentation user interaction for all the data streams. Once playing in reverse, however, there is no user interaction defined to return to playing out the presentation in the forward direction. Including the forward and other additional user interactions is addressed in Chapter 4. The logic necessary to control updating the primary stream and synchronizing the secondary streams is available in Appendix B. As for the skip ahead user interaction, checks to determine the type of secondary stream are necessary in the logic as is handling blocked streams during the re-synchronization.

3.3.4.3 Freeze and Resume. When a freeze interaction occurs, the active presentation unit in all data streams is pre-empted and the execution is deferred until a restart user interaction occurs. The value of the execution time is stored before the place is pre-empted. A new

place is activated and its execution value is set to its normal playout duration, which is a very large positive real number. When a restart user interaction occurs, the place is pre-empted and terminated, and the frozen place is re-activated. Figure 13 (d) illustrates how to model the freeze and restart user interactions for all the data streams. The logic necessary to control updating the primary stream and synchronizing the secondary streams for the freeze and restart requests is provided in Appendix B. The logic for these interaction is straightforward, although the check to determine an independent stream from a dependent secondary stream is still made, even though the processing is exactly the same for all of the streams.

3.3.4.4 Scale Presentation Speed. The speed of the presentation may be scaled by a factor to make the presentation faster or slower as a whole. When a scale presentation speed interaction occurs, all of the presentation units in all of the data streams are pre-empted and permanently modified by updating the normal playout duration value. Figure 13 (e) illustrates how to model the scale presentation speed user interactions for all of the data streams. The logic necessary to control updating the primary stream and synchronizing the secondary streams is provided in Appendix B. As with the freeze and resume logic, the check to determine an independent stream from a dependent secondary stream is made, even though the processing is exactly the same for all of the streams.

3.3.4.5 Formal definition of the DTPN. The formal definition of the DTPN model is a tuple (T, P,W_{pt},W_{tp},D,Rs,M(0),C,E) where:

$(P, T, W_{pt}, W_{tp}, D, Rs, M(0))$ defines the OCPN;

$A \subseteq \{T \times P\} \cup \{P \times T\}$ is a set of arcs;

$C \subseteq P \times T$; $A \cap C = 0$ C represents a set of escape arcs;

$E : P \rightarrow R, R \geq 0$ is a function that maps the remaining execution time from the set of places to the set of real numbers greater than or equal to zero [Mer76]. E maintains the remaining duration for which execution is to be carried out when the place is made active by the firing of an input transition. D maintains the nominal allowed duration. One assumption of this model includes the initialization of E to the value of D in a housekeeping function.

3.3.4.6 Graphic Notation and Example Specification. The graphic notation for the user interaction components and an example of a replacement hierarchy is illustrated in Figure 14.

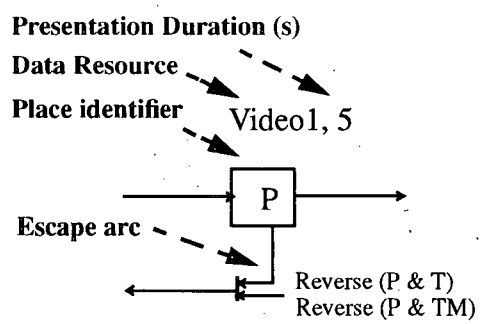


Figure 14. The DTPN Graphic Notation

The example is specified in the DTPN approach in Figure 15. The transitions labelled with the letters indicate that a single transition is illustrated in two separate places on the diagram. For example, the two transitions labelled A are actually the same transition. This approach is used to remove the crossing arcs which makes the diagram easier to read. Like the OCPN approach, the DTPN specification is concise. The most significant advantages of this model

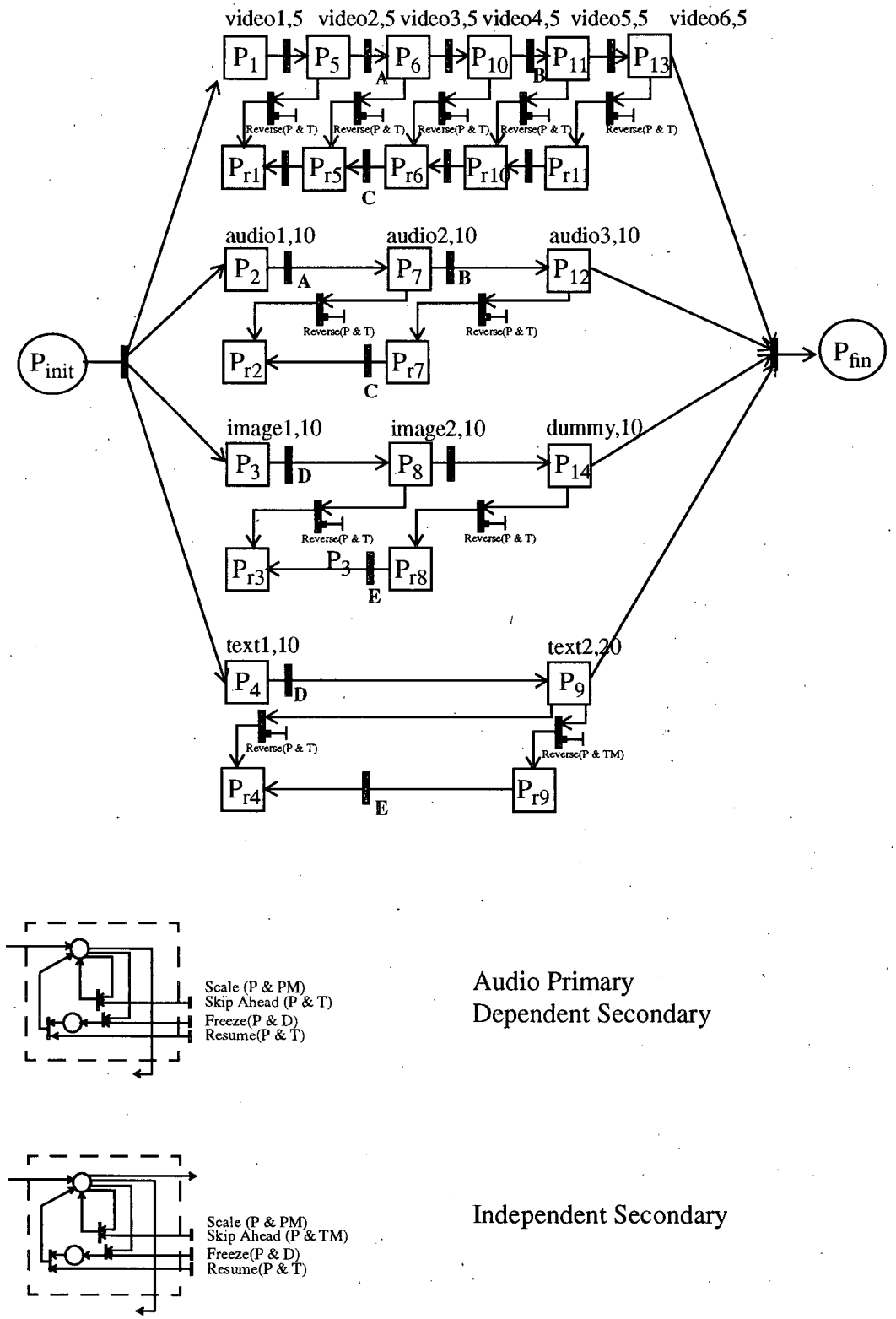


Figure 15. The DTPN Specification

are that some user interactions are addressed and the partial playout of the presentation in reverse are addressed.

3.4 Discussion of the Specifications

The OCPN, TSPN, XOCPN, and DTPN approaches are discussed in this section. The criteria used for evaluating the specification models include the complexity of the model and the ability to express the requirements of the example application in the model. The metric for measuring the complexity of the model is the number of symbols used in the specification [Vuo95b]. For the Petri Net models, the number of symbols is the sum of the number of transitions, arcs, places, and labels. The components of any hierarchy diagram which is reused are only counted once. The symbol used in the specification to indicate where to use the hierarchy diagram is counted each time. For the natural language specification, the number of symbols is the line count for the specification. Although rough, these measurements provide an order of magnitude measurement for comparison purposes. The results of the evaluation are summarized in Table 8.

The complexity of the Petri net models varies significantly between order of magnitudes 10^1 and 10^4 for the example specified: the OCPN model uses 10^1; the DTPN model uses 10^2; TSPN uses 10^3; and XOCPN uses 10^4. The natural language specification requires approximately 10^2 lines to describe the example. The OCPN approach may not describe the jitter, user interactions, or the fine grained synchronization requirements. In addition, the OCPN model only supports playout of the presentation in the forward direction. The DTPN model has the same jitter and level of granularity deficiencies as the OCPN model, but does allow a partial description of the user interaction requirements. The user interactions the DTPN approach may model include the requests to skip ahead, scale the presentation speed, freeze and resume the presentation, and reverse the presentation playout. The DTPN model

may not model user interactions such as start, skip back, play forward (once the presentation has been reversed), restart, terminate, fast forward or fast reverse. In addition, the entire presentation is not accessible for playout in reverse. The presentation units at the beginning and end of the presentation may not be partially played out in reverse. The TSPN approach may model all of the requirements in the example except the user interaction requirements. The XOCPN model, the most complex, may not model the jitter or user interaction requirements and does not support the playout of the presentation in the reverse direction.

Requirement	Synchronization Specification Models				
	OCPN	DTPN	TSPN	XOCPN	Natural Language
Pre-orchestrated Source Data	yes	yes	yes	yes	yes
Source data time dependency	implicit	implicit	implicit	implicit	implicit
Source data type of medium	yes	yes	yes	yes	yes
Multiple source sites	yes	yes	yes	yes	yes
Duration of Presentation	yes	yes	yes	yes	yes
Single destination site	yes	yes	yes	yes	yes
Jitter	no	no	yes	no	yes
User Interaction	no	partial	no	no	yes
Level of granularity	too high	too high	adequate	adequate	adequate
Total number of symbols	86	299	1117	19332	95

Table 8: Summary of Specification Approaches

The natural language is the only approach powerful enough to describe all the requirements for the example and is concise. The drawbacks of this approach are that the informal specification is not amenable to verification by simulation and the specification is open to interpretation during the development cycle.

If a formal specification is required, the TSPN approach is the best choice among the formal Petri net approaches because it models the largest number of the requirements. A sig-

nificant drawback of the TSPN model is that it does not model the user interaction requirements. The only Petri net approach in the previous review which supports the specification of user interactions is the DTPN approach. Another strength of the DTPN model is that is uses a small number of unique hierarchies to model the user interactions. The limited number of hierarchy diagrams is easy to maintain in a specification. This approach is limited to a few user interactions, though, and the logic to implement this model in a tool is quite complex. The complexity results from two characteristics. First, the dynamic nature of determining if a stream is independent secondary or dependent secondary stream adds additional checking. Second, for a skip ahead request, the DTPN approach to re-synchronizing the secondary streams is serial i.e. one data stream is re-synchronized at a time. After the primary data stream is updated, each secondary data stream is processed one at a time to synchronize with the primary stream. If only intrastream synchronization is required, the depth first approach is straightforward. The complexity with this approach occurs when a dependent secondary data stream becomes blocked by an interstream synchronization point. The blocked transition involves one or more data streams that are not re-synchronized yet. To handle this case, the blocked transition must be noted and the processing continued on the remaining data streams. When the other data streams have been re-synchronized up to the blocked transition, the re-synchronization on the blocked streams may continue. Multiple blocked transitions may occur in a specification.

3.5 Conclusions

This chapter reviews established Petri nets to provide background information for the reader and presents a new analysis and comparison of four Petri net model extensions for describing multimedia synchronization requirements: object composition, extended object composition, time stream, and dynamic time Petri nets. The analysis covers the behaviour, formal definition, and graphic notation in addition to noting errors and deficiencies in the multimedia synchronization extensions. The comparison of the models is based on a common example that is specified in each of the approaches. For each specification, the complexity and the completeness of the specification are determined.

The analysis work in this section points out the need for a new Petri net model because a single Petri net model may not be used to describe the example completely. From the analysis work, the TSPN and the DTPN models are observed to complement one another, in that the deficiencies of one modelling approach are partially or fully corrected by the strengths of the other. The TSPN model does not describe user interactions and the playout of the presentation in reverse whereas the DTPN model may do so in part. The DTPN model does not describe the jitter requirements whereas the TSPN model does. The new model needs to include a simplified, generalized version of the DTPN model integrated with the TSPN model. This model is developed in Chapter 4.

4 User Interaction Extension to TSPN

4.1 Introduction

The analysis work discussed in Chapter 3 points out the lack of a single Petri net model that may be used to describe the requirements in the example. To solve this problem, a new model is proposed in this chapter. The issues involved in developing the new model are that the new model must provide a flexible toolkit approach to describing requirements. The developer may build and re-use hierarchy diagrams and re-use primitive user interactions. The model must also have simple, straightforward logic for the control of the Petri net, be straightforward to implement in a simulation tool, and be formally defined.

The TSPN model is used as the basis of the new model because it allows the formal specification of the largest number of the synchronization requirements discussed in Chapter 1. Since the DTPN model complements the deficiencies of the TSPN model, it is used in the development of the new model, too. The new extension is developed in three steps. The first step simplifies the DTPN model's definition of data streams. The concepts of primary, independent secondary, and dependent secondary streams are replaced with a unified approach which treats all of the data streams the same. This makes the logic necessary to control the Petri net simpler than in the original DTPN model. The second step defines a set of primitive user interactions and a sample set of composite of user interactions for typical requests in pre-orchestrated presentation applications. The user interactions are described using the simplified data stream approach presented in step one. The primitive user interactions are the building blocks for more complex, composite user interactions and provide a toolkit approach to describing user interaction requirements. The third step takes the simplified, general approach

developed in steps one and two and integrates it with the TSPN model. The resulting model is called the Time Stream Petri net with User Interaction ($TSPN_{UI}$) model. Figure 16 illustrates the evolution of the $TSPN_{UI}$ model from the Petri nets discussed in Chapter 3. The shaded

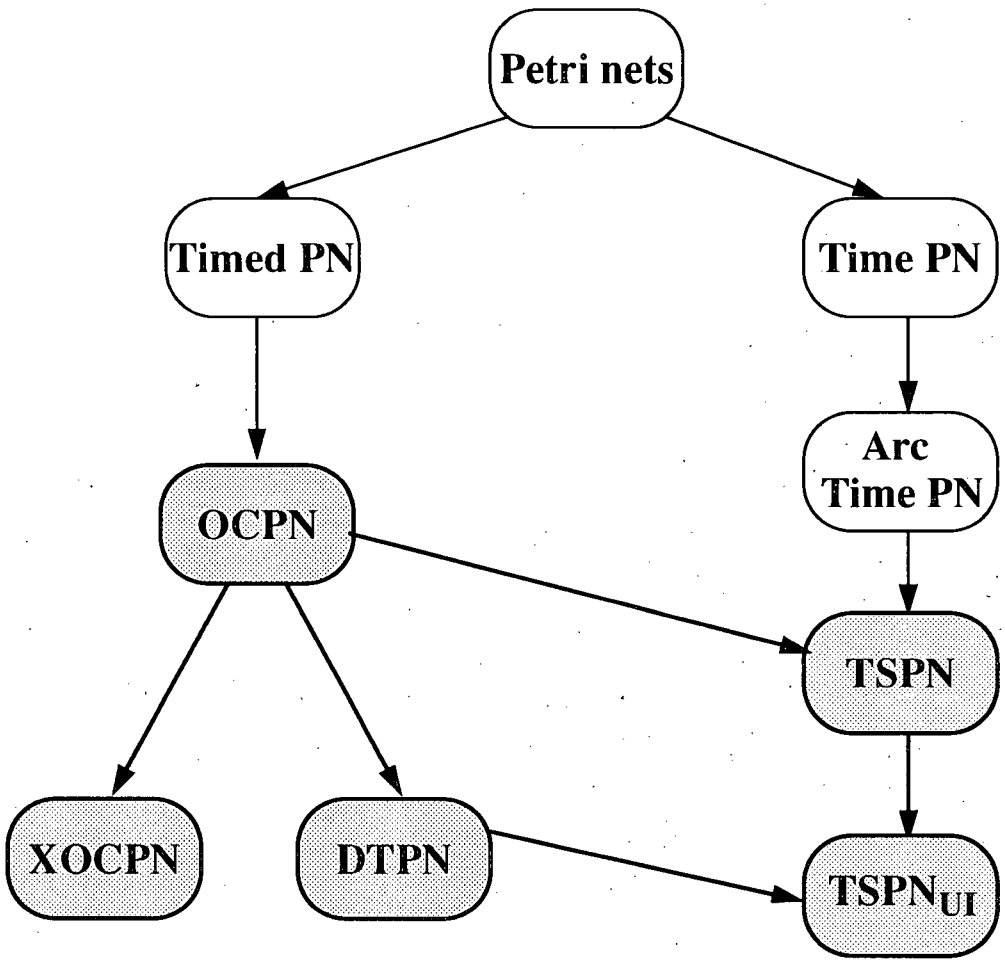


Figure 16. Evolution of the $TSPN_{UI}$ Model

bubbles indicate the model is a multimedia extension to Petri nets. The three development

steps are discussed in the following sections.

4.2 Simplification of DTPN Data Stream Definitions

The DTPN model defines each data stream as a primary, independent secondary, or dependent secondary stream. These definitions serve to support the skip and reverse user interactions defined in the DTPN model. In these interactions, the primary data stream determines the re-synchronization point for the presentation by using the granularity of its presentation units (refer to section 3.3.4). For example, if the primary data stream is playing out a presentation unit that is 10 seconds long and a skip request occurs 2 seconds into the playout, the re-synchronization point is set 8 seconds ahead by the primary stream. Subsequently, the secondary data streams are re-synchronized 8 seconds ahead. The DTPN model is not flexible because it does not allow the user to skip ahead an arbitrary number of time units. To correct this, the skip and reverse user interactions are modified in this step. The number of time units to skip ahead may be specified by the user, and the reversal of the presentation occurs at the current point in the presentation. This approach is more flexible and simpler than the DTPN approach. It is more flexible because it is independent of the granularity of the presentation unit duration and allows the playout of any part of the presentation in reverse. The modified approach is simpler because the definitions of primary stream and secondary streams are no longer needed. All of the data streams are treated the same way, so the logic necessary for synchronizing the data streams is simpler. This simplification is an advantage in the development of tool support for a model.

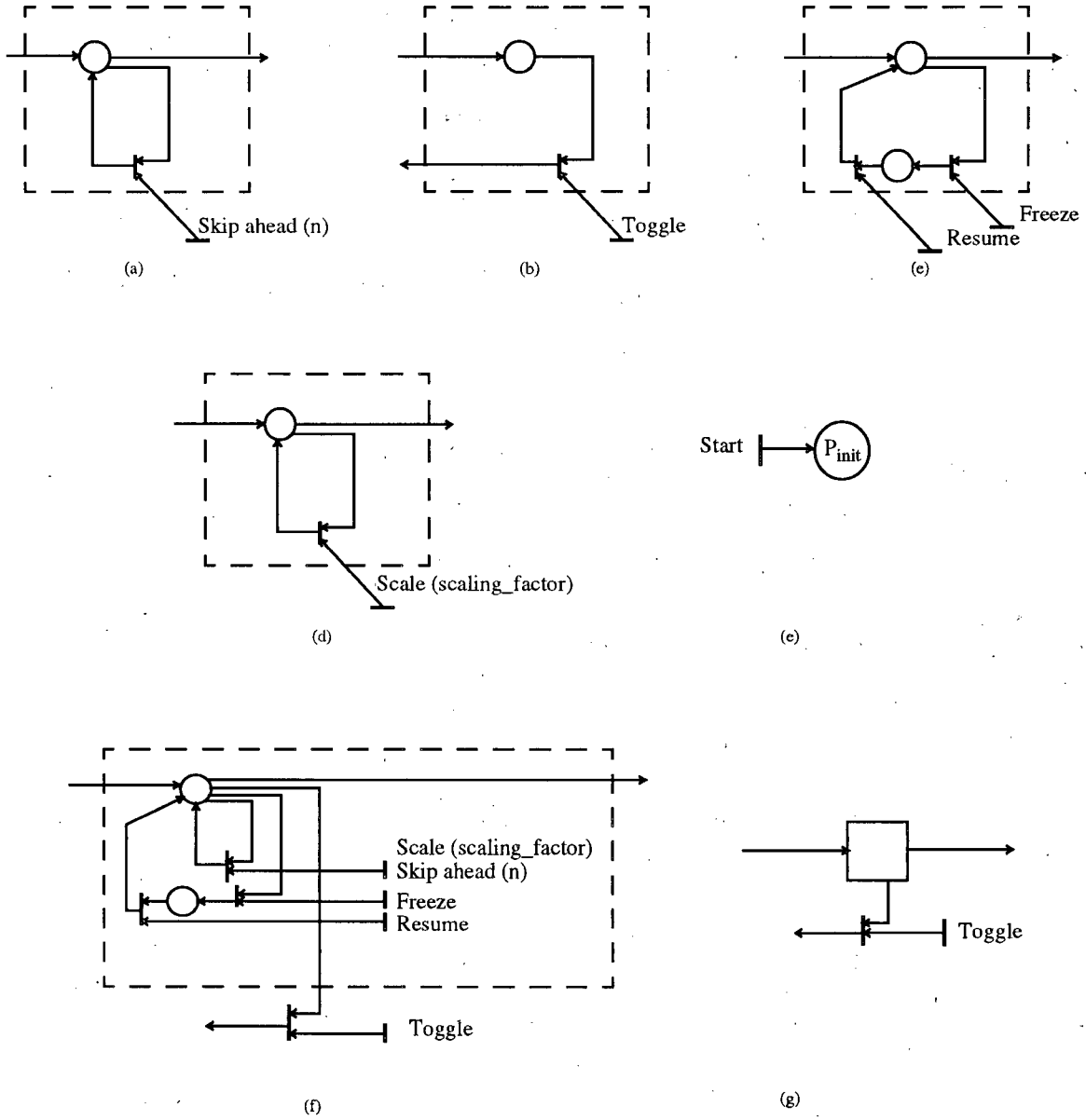
4.3 User Interactions

To support a toolkit approach to describing user interactions, a set of fundamental building blocks, or primitive user interactions, is necessary. One issue is determining which interactions need to be defined as primitives. In the following sections, a set of primitive user interactions is proposed in addition to describing a selection of composite user interactions. The composite interactions illustrate how the primitives may be re-used. The toolkit approach is also useful in the development of tool support, as the code developed to handle the primitive user interactions may be re-used to support the composite interactions.

4.3.1 Primitive User Interactions

The proposed set of primitive user interactions includes skip ahead(n), toggle presentation direction, freeze, resume, scale the presentation speed, and start. These elements are categorized as primitive interactions because they may not be described in terms of other user interactions. The primitive user interactions form a toolkit from which composite user interactions may be defined as required. This toolkit approach is concise and flexible: the basic user interactions are described in one place and reused many times in whatever combination is necessary. A description for each of these primitive user interactions is provided in the following sections. The graphic representation of the primitive user interactions, composite hierarchy, and the replacement hierarchy is provided in Figure 17. The logic necessary to support the primitive user interactions skip ahead(n), toggle, freeze, resume, and scale the presentation speed is provided in Appendix B.

4.3.1.1 Skip Ahead (n). When a skip ahead n interaction occurs, the presentation skips ahead n time units, where the time unit may be defined at a suitable level (e.g. milliseconds,



(a) Skip ahead (n)
(b) Toggle Presentation Playout Direction
(c) Freeze and Resume
(d) Scale Presentation Speed (scaling_factor)
(e) Start
(f) Combined User Interaction Operations
(g)Hierarchy Replacement for Combined User Interaction Operations

Figure 17. Primitive User Interactions

seconds, minutes) for the application. The data streams are advanced in parallel, to avoid the blocking of interstream transitions as found in the DTPN approach, using the TSPN concept of a firing interval. Figure 17 (a) illustrates how to model the skip ahead(n) user interaction for all the data streams. Note that only one interaction is required in the modified skip ahead (n) user interaction, whereas the original DTPN model used two types. The new composite hierarchy is simpler as a result. The logic necessary to control updating the data streams is also simpler than the original DTPN model. This is because the streams are all treated the same way and the blocking of interstream synchronization transitions does not occur when the streams are advanced in parallel (refer to Appendix B).

4.3.1.2 Toggle Presentation Direction. The reverse playout direction user interaction defined in the DTPN approach is generalized to toggling the playout direction. This allows the presentation to switch from the forward to reverse or reverse to forward direction. In each data stream the execution time is retained, the active presentation unit is pre-empted, and the playout direction is toggled. The execution time is updated to the difference between the nominal duration and the retained execution time. This approach to reversing the playout direction is simpler than the original DTPN approach and is not tied to the granularity of a data stream. The presentation is played out in reverse from the point in the presentation at which the user request is made. The user may request to reverse the playout presentation direction and view any part of the presentation in reverse. The partial reverse playout of the first or last presentation units may not be specified in the DTPN model because the approach skips back to the last presentation unit that has been completely played out and begins the presentation from that point. The generalization of the reverse user interaction to a request to toggle the playout

direction is consistent and simple. Figure 17 (b) illustrates how to model the toggle presentation direction user interaction for all the data streams. The logic necessary to control updating the data streams is less complex than the original DTPN model. The simplification is due to the presentation being toggled at one point in the presentation as well as treating all of the data streams the same way (refer to Appendix B).

4.3.1.3 Freeze and Resume. When a freeze interaction occurs, the active presentation unit in each of the data streams is pre-empted and the execution is deferred until a restart user interaction occurs. The value of the execution time is retained before the place is pre-empted. A new "frozen" place is activated and its execution value is set to its normal playout duration, which is a very large positive rational number. When a resume user interaction occurs, the "frozen" place is pre-empted and terminated, and the presentation unit place is re-activated. Figure 17 (c) illustrates how to model the freeze and restart user interactions for all the data streams. The logic necessary to control updating the data streams for the freeze and restart requests is provided in Appendix B. The pseudocode is the same as the original DTPN approach.

4.3.1.4 Scale Presentation Speed (scaling_factor). The speed of the presentation playout may be scaled by a factor to make the playout either faster or slower. When a scale presentation speed interaction occurs, all the presentation units in all the data streams are pre-empted and permanently modified by updating the EFT, nominal playout duration, execution time, and LFT values. Figure 17 (d) illustrates how to model the scale presentation speed user interaction for all the data streams. The logic necessary to control updating the data streams for a scale presentation speed request is provided in Appendix B. After considering

the minor correction to the DTPN model's logic for scaling the presentation speed, the pseudocode for the new model is the same.

4.3.1.5 Start. When a start user interaction occurs, the presentation begins. This simple interaction initializes the start place, and allows the user to decide when to begin the presentation.

4.3.2 Composite User Interactions

The set of user interactions is expanded to provide a flexible set of requests for presentation playout applications. Composite user interactions, including skip back n, restart, terminate, and fast & toggle are built from the simpler, primitive ones discussed in the previous section.

4.3.2.1 Skip Back (n). When a skip back n interaction occurs, the presentation skips back n time units, where the time unit may be defined in a suitable unit e.g. milliseconds, seconds, minutes, for the presentation. In each data stream, the playout presentation direction is toggled using the toggle user interaction. After the direction is toggled, the skip ahead (execution time) user interaction logic is used to move to the correct point in the presentation. Finally, the playout direction is toggled again so that playout may continue in the original direction. The logic necessary to control updating the data streams is described below. Note the re-use of the primitive user interactions "toggle" and "skip ahead (n)" to build up the skip back (n) composite user interaction.

- n

  Description: number of time units to skip back

Scope: seen by all data streams

For each stream
  For the active PU
    toggle presentation direction
    skip ahead (n)
    toggle presentation direction

4.3.2.2 Terminate. When a terminate interaction occurs, the presentation skips ahead to the end. The logic necessary to control updating the data streams is described below. In this case, the primitive user interaction "skip ahead (n)" is used to build the "terminate" composite user interaction.

For each stream
  For the active presentation unit
    skip ahead (very large number)

4.3.2.3 Restart. When a restart interaction occurs, the presentation skips back to the beginning. The logic necessary to control updating the data streams is described below. To build the "restart" composite user interaction, the composite interaction "skip back(n)" is used as the building block.

For each stream
  For the active presentation unit
    skip back (very large number)

4.3.2.4 Toggle & Fast. When a toggle & fast interaction occurs, the presentation speed is scaled to the fast forward factor and the presentation playout direction is changed to the opposite direction. The logic necessary to control updating the data streams is described below. For

this composite interaction, the primitive requests "scale" and "toggle" are used.

For each stream
  For the active presentation unit
    scale presentation speed (fast rate)
    toggle presentation playout direction

4.3.2.5 Continue. When a continue interaction occurs, the presentation progresses. This user interaction is suitable for data that is played out 'until the user requests' to continue. The presentation skips ahead by the execution time left in each active presentation unit. The assumption with this user interaction is that it is used at places in the presentation which remain synchronized when the request is made. This assumption is reasonable for this user interaction.The logic necessary to control updating the data streams is described below. Note that the only primitive user interaction required to build the "continue" interaction is the "skip ahead(n) request.

For each stream
  For the active presentation unit
    skip ahead (E_cur)

4.4 The $TSPN_{UI}$ Approach

The primitive user interactions described in section 4.3.1 are integrated with a modified TSPN model to produce the Time Stream Petri net with User Interaction ($TSPN_{UI}$) model. The modifications are the inclusion of the resource component and the escape arcs. The resource components and escape arcs are included from the DTPN model. The composite hierarchy described in the previous section replaces the presentation units in the original TSPN approach.

The firing rules for the $TSPN_{UI}$ are different for the intrastream and the interstream synchronizing transitions. In the intrastream synchronization case, standard Petri net firing rules apply. The intrastream case may be considered as the trivial case for the strong-or (the first and only data event occurs), the weak-and (the last and only data event occurs), or the master (the only data stream is the master stream). Although any of the three transition types may be used in the specification, using the untyped transition provides a modelling check for the developer, especially as the specification evolves. For example, initially a specification may describe a data stream using only intrastream transitions only. After a modification to the specification, an interstream synchronization event is described. If the untyped transition is not updated to a typed transition, the error is obvious. However, if a typed transition is used for the intrastream synchronization events, failing to update the transition type to the desired type may produce an error that is difficult to trace.

The firing rules for the interstream synchronization case use modified TSPN rules. The valid firing interval is re-evaluated by adjusting the EFT, execution time, and LFT by the amount of time the execution time has been reduced by. The nominal duration in the

TSPN model is like the nominal duration time used in the DTPN model, and is not modified. An example of updating the components of the tuple is: If 5 seconds are left to execute when the unit is pre-empted and the execution time is set to zero, then EFT=max(0, EFT -5), execution time=max(0,execution time -5), and LFT= max(0,LFT -5). Then, the dynamic firing interval is re-evaluated with these updated numbers. Note that the rules keep all the time values greater than or equal to zero, unlike in the TSPN approach, where the execution time may become negative. The rules maintain consistency in the $TSPN_{UI}$ approach between the semantics and the formal definition. The calculation of the firing interval for the original TSPN is used in the $TSPN_{UI}$ model. This calculation is reviewed in section 3.3.2.

4.4.1 Formal Definition of the $TSPN_{UI}$

Formally, the $TSPN_{UI}$ approach is defined as a tuple (P, T, W_{pt}, W_{tp}, M(0), Rs, IM, SYN, C) where:

$(P, T, W_{pt}, W_{tp}, M(0))$ defines a marked Petri net;

$Rs : \mathrm{P} \rightarrow (r_1, r_2, r_3, \ldots r_k)$ represents the resources required at each place.

$A \subseteq \{T \times P\} \cup \{P \times T\}$ is a set of arcs;

$B = \{b = (p, t) \in T \times P / W_{tp} \neq 0\}$ is a subset of arcs in the flow direction from the transition to the place;

$C \subseteq P \times T;\ A \cap \mathrm{C} = \varnothing$ C represents a set of escape arcs;

$IM : \mathrm{B} \rightarrow \left(Q^+ \cup \infty\right) \times \left(Q^+ \cup \infty\right) \times \left(Q^+ \cup \infty\right) \times \left(Q^+ \cup \infty\right)$ IM is a function that maps arcs entering transitions into zero or positive rational numbers representing the EFT, the nominal duration, the execution time, and the LFT, respectively;

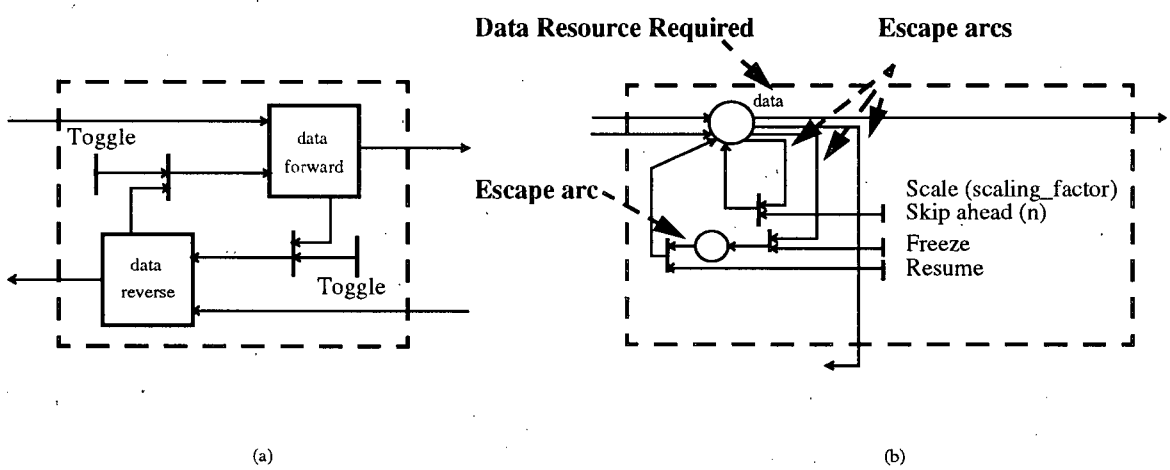
$IM(a_i) = \left(\alpha^s_i, \eta^s_i, \varepsilon^s_i, \beta^s_i\right)$ is the initial, static definition;

$SYN: T \rightarrow$ {or, strong_or, and, weak_and, master, or-master, and-master, weak-master, strong-master} SYN is the typing function that distinguishes between the different firing rules (i.e. the different semantics for interstream synchronization).

4.4.2 Graphic Notation and Example Specification

Two fundamental hierarchies are proposed for the $TSPN_{UI}$ model: "$Data_n$ Forward & Reverse Hierarchy" and the "Data Forward (Reverse) Hierarchy". The data forward hierarchy and the data reverse hierarchy are the same, assuming that the user interactions applicable in the forward direction are also applicable in the reverse direction. If this is not the case, a unique hierarchy is required for each direction. These two hierarchies provide a flexible toolkit for specifying systems (refer to Figure 18 (a) and 18 (b)). The advantages of using these two hierarchies as the basic building blocks are that they may be combined as required with other data and the types of transitions necessary. Describing the synchronization of different data streams in series or in parallel is simple, because each data stream is encapsulated in a diagram. Another advantage of this organization is that the time interval is documented at a higher level in the hierarchy allowing the lower level diagrams to be re-used with different time interval data.

The example worked out in the four multimedia extensions in Chapter 3 is also worked out in the $TSPN_{UI}$ extension. The specification is illustrated in Figures 19-23. Note that diagrams 1.1 and 1.2 are generalized for illustration purposes in Figure 20. The difference between diagram 1.1 and 1.2 is the number of occurrences of the component diagram 1.1.1. Diagram 1.1 has fifteen occurrences while diagram 1.2 only has fourteen. The



(a) $Data_n$ Forward and Reverse Hierarchy

(b) Forward or Reverse Hierarchy

Figure 18. $TSPN_{UI}$ n-Data Stream Hierarchies

maximum depth to the specification is six levels. To assist the reader, the naming convention used in [War85] is used to show the structure of the specification's hierarchy.

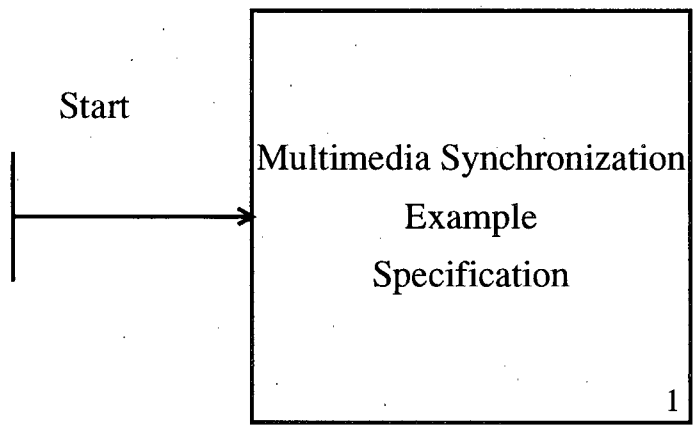


Figure 19. $TSPN_{UI}$ Specification: Diagram 0

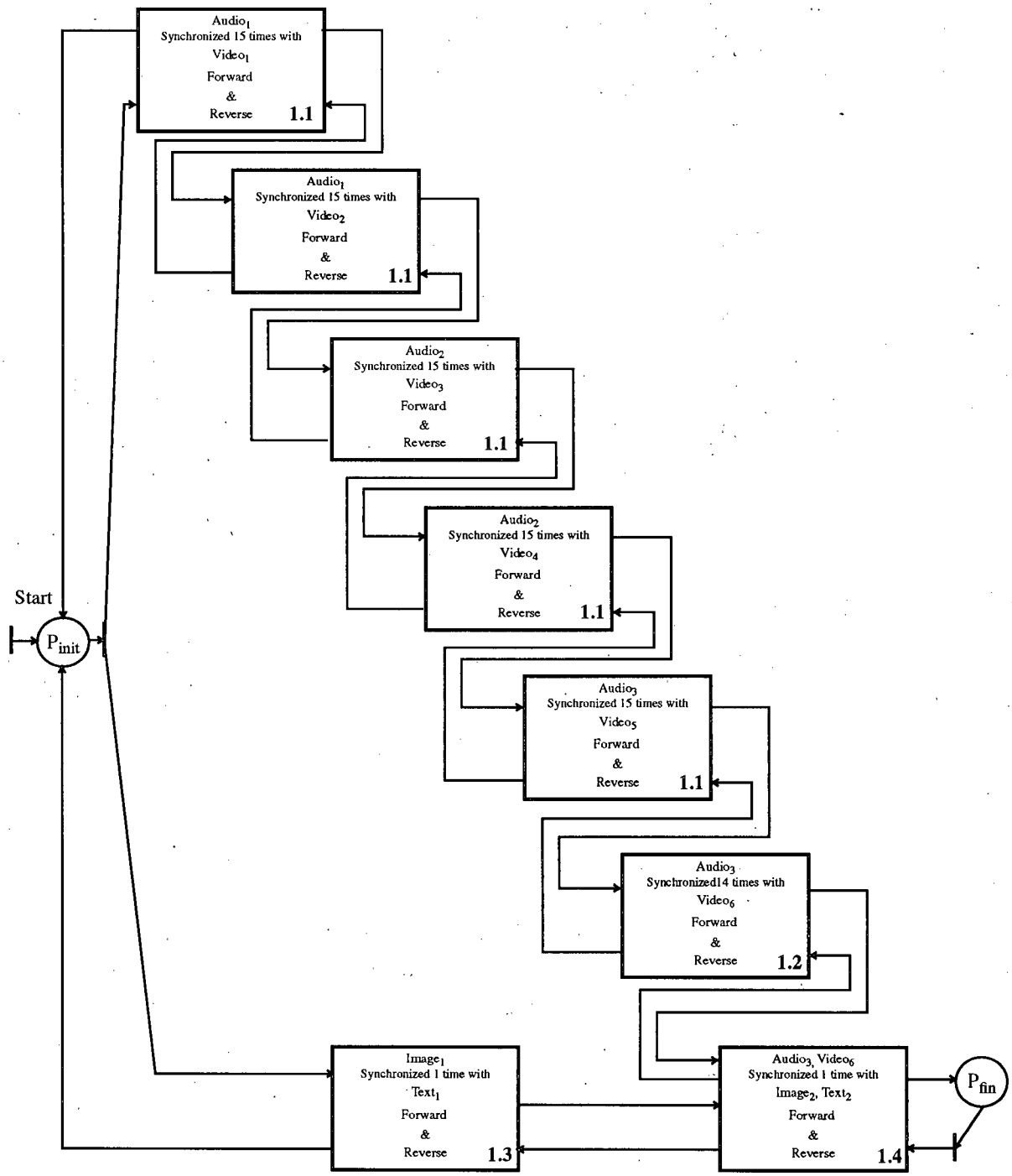
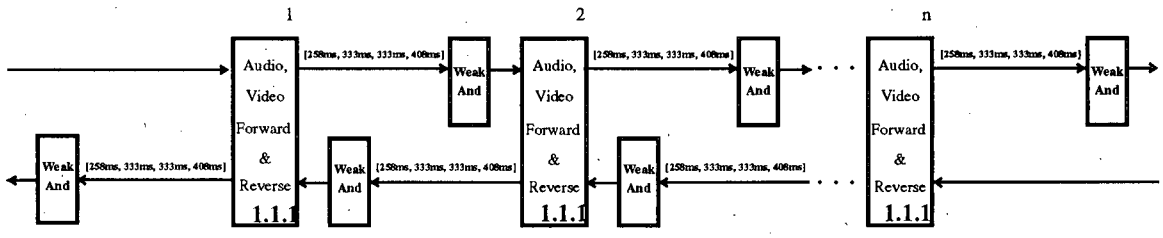


Figure 20. $TSPN_{UI}$ Specification: Diagram 1



Diagrams 1.1, 1.2

1.1 n = 15, 1.2 n = 14

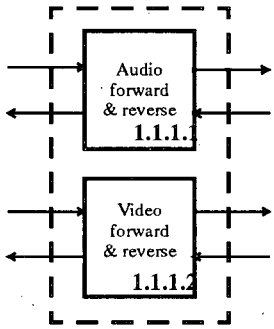


Diagram 1.1.1

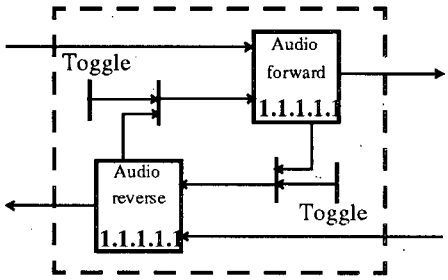
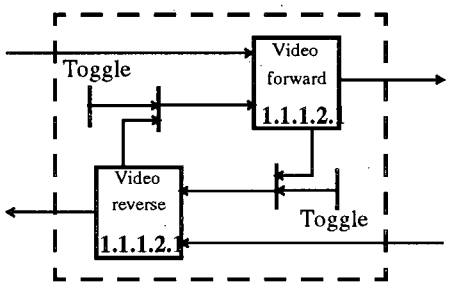


Diagram 1.1.1.1



Diagrams 1.1.1.2

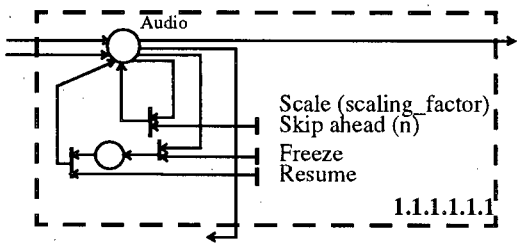


Diagram 1.1.1.1.1

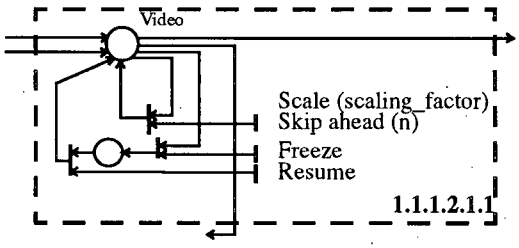


Diagram 1.1.1.2.1

Figure 21. $TSPN_{UI}$ Specification: Diagrams 1.1.n

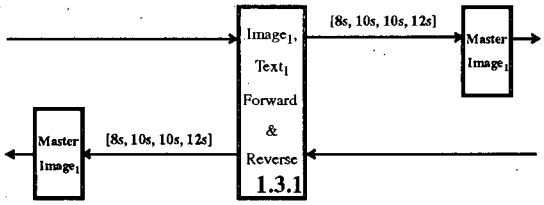


Diagram 1.3

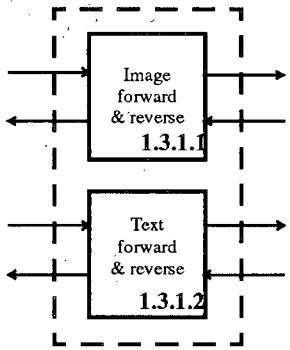
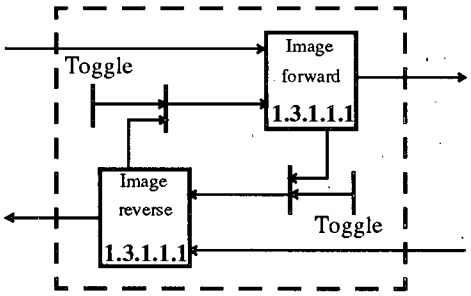


Diagram 1.3.1



Diagrams 1.3.1.1

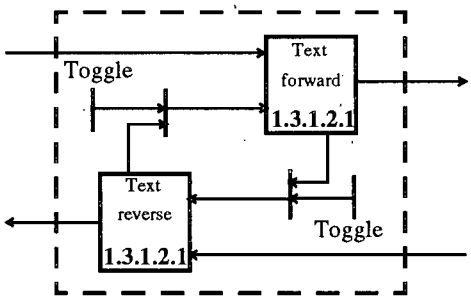


Diagram 1.3.1.2

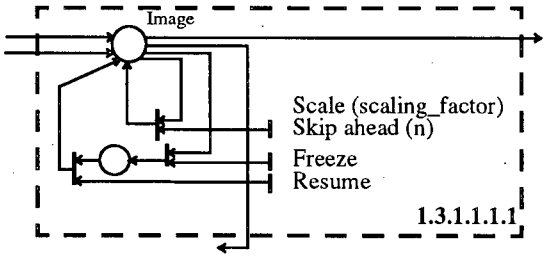


Diagram 1.3.1.1.1

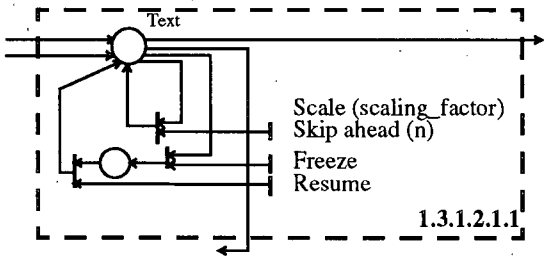


Diagram 1.3.1.2.1

Figure 22. TSPNUI Specification: Diagrams 1.3.n

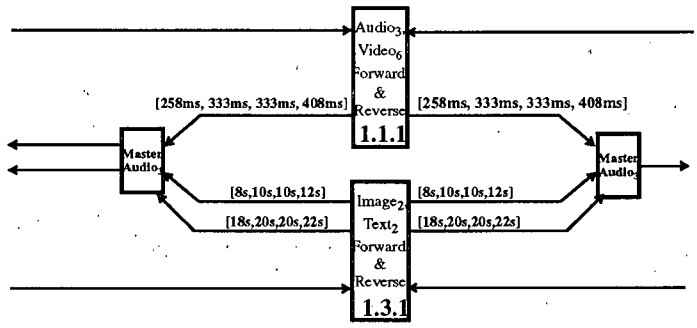


Diagram 1.4

Figure 23. $TSPN_{UI}$ Specification: Diagrams 1.4

4.5 Discussion

The specification work on the four multimedia Petri net models summarized in Table 8 is augmented in Table 9 with the results of the $TSPN_{UI}$ model. The $TSPN_{UI}$ specification required on the order of 10^2 components and all of the requirements in the example specification are supported.

Requirement	Synchronization Specification Models					
	OCPN	DTPN	TSPN	XOCPN	$TSPN_{UI}$	Natural Language
Pre-orchestrated Source Data	yes	yes	yes	yes	yes	yes
Source data time dependency	implicit	implicit	implicit	implicit	implicit	implicit
Source data type of medium	yes	yes	yes	yes	yes	yes
Multiple source sites	yes	yes	yes	yes	yes	yes
Duration of Presentation	yes	yes	yes	yes	yes	yes
Single destination site	yes	yes	yes	yes	yes	yes
Jitter	no	no	yes	no	yes	yes
User Interaction	no	partial	no	no	yes	yes
Level of granularity	too high	too high	ade-quate	adequate	adequate	adequate
Total number of symbols	86	299	1117	19332	721	95

Table 9: Augmented Summary of Specification Approaches

In comparison to the other Petri net models, the $TSPN_{UI}$ model offers a number of advantages. The $TSPN_{UI}$ model:

- defines a flexible set of primitive user interactions. The primitive interactions may be used as atomic request and as building blocks for composite user actions. In turn, the composite interaction may be used to build additional composite user requests.
- uses a consistent definition for the tuple components. Time is always represented as zero, a positive rational number, or unbounded (infinity). In contrast, the execution time is allowed to be updated to a negative value in the TSPN model, although this is not consistent with the formal definition of the tuple and is not useful in the application area.
- The execution time is defined on the arc to keep it coupled with the nominal playout duration time.
- supports pre-emption to allow the specification of user interactions which are independent of the granularity of the specification. The OCPN, XOCPN, and TSPN models are not pre-emptive, and therefore user interactions must wait until the playout of the current presentation units are complete. The DTPN model is pre-emptive, but the skip ahead and reverse user interactions are dependent on the granularity of the primary stream's presentation units.
- The $TSPN_{UI}$ specification is 0.65 times the size of the specification in the original TSPN approach. The decrease in complexity is due to the re-use of hierarchy diagrams which removes repetition from the specification. In comparison to the natu-

ral language approach, however, the $TSPN_{UI}$ specification is 7.59 times the size. This increase in complexity may be considered the cost of formally specifying the example.

4.6 Conclusions

The contribution of this chapter is the development of a new Petri net model which may be used to model all of the requirements in the example problem. The model is called the $TSPN_{UI}$ model to reflect the original basis of the model (TSPN) and what the extension allows the developer to specify that could not be done before using Petri nets (user interactions). The user interactions in the new model are not constrained by the granularity of the presentation units, any point of the presentation may be played in reverse, and the specification may be toggled in either direction. These features correct several deficiencies in the original DTPN model. The new model is developed to provide a toolkit approach for the developer in both building and re-using hierarchy diagrams and in re-using primitive user interactions to build more complex, composite user interactions. This provides the developer with a flexible approach that may be tailored for the application being developed. The logic necessary to control the model is straightforward, as all data streams are treated the same way and re-synchronization of data streams is considered in parallel. This makes tool support development simpler for the model.

5 Visualization Tool

5.1 Introduction

One means of verifying the correctness of a specification is to visualize a simulation of it [LOT89, Edb92]. A tool is developed to visualize $TSPN_{UI}$ specifications. The purpose of the tool is to aid the user in discovering and correcting errors that occur during the simulation of a $TSPN_{UI}$ specification. The tool supports the use of "what if" scenarios: the type of transitions; values for arc time intervals; and the scripting user interactions may be varied. In addition, the tool allows the user to build a specification out of re-used, smaller components. The design of the tool, results of "what if" scenarios, and a discussion of the costs and benefits of using the tool are discussed in the following sections. The "uses" diagrams for the design are available in Appendix C.

5.2 Tool Design

The Visualization Tool is developed for use in the MATLAB environment [MAT94]. The tool MATLAB is used because it is a stable, robust product which is useful in Petri net simulations [Rez95]. The main features of the tool and issues in the development of the tool are discussed in the following sections.

5.2.1 Tool Features

The main features of the tool include the re-use of hierarchy diagrams, scripting capability for user interaction sequences, easy modification of transition types and arc time intervals, and the display of the enabled transition progression.

5.2.1.1 Re-use of hierarchy diagrams

This feature allows the developer to describe small sections of the specification and ask the tool to combine these components both in series and in parallel. The advantage of this is that smaller matrices are easier to review and correct than larger ones. For example, a developer is likely to be able to review and correct a single matrix of size [10,10]. However, reviewing and finding errors in the matrix of size [100,100] is very difficult. The composition matrix may be built, for example, by combining ten of the small matrices in series. Combining matrices is convenient in multimedia presentation scenarios with a significant amount of repetition in the specification. For example, in the example used throughout this thesis, there is a great deal of reuse in the presentation of the audio and video streams. These re-used modules are only defined one time and reused many times. This reduces the possibility for introducing errors, makes correcting errors simpler since the errors are isolated, and reduces the amount of data the user must input. The addition of the start place and transition as well as the final place

is also performed by the tool. The tool has the general ability to process any Petri net model built by the user.

5.2.1.2 Scripting Capability

This feature allows the sequences of user interactions to be scripted in one file. This is convenient during the development of the specification when the specification is simulated repeatedly. The user interaction may be specified to occur at a specific time from the start of the presentation, as observed by the user of the system. Again, the developer only needs to describe the sequence of interactions in a single location. The primitive user interactions skip ahead (n), toggle presentation direction, freeze, resume, scale presentation speed(n), and start are supported by the Visualization Tool. Composite user interactions including skip back(n), restart, and terminate have also been built to support the requirements in the example used throughout this thesis. These composite interactions are implemented using the code developed for the primitive user interactions.

5.2.1.3 Easy modification of transition types and arc time intervals

These features allows the transition types and the arc times for the intervals to be modified easily. The transition types for each module are defined in a single data structure as are the arc time intervals. The three basic transition types are supported in the Visualization Tool: strong-or, weak-and, and master.

5.2.2 Design Issues

Design issues such as the large model size and the scope of developing a complete Visualization Tool are addressed in the following sections.

5.2.2.1 Model Size

An interesting problem in the design of the Visualization Tool is how to minimize the size of the model being manipulated by the tool at any time. As the tool must search through these matrices, smaller matrices lead to better performance of the tool. The solution to this problem is to treat the forward model and the reverse model separately. When progressing the specification in the forward direction, only the forward model is manipulated. Similarly, when progressing the model in the reverse direction, only the reverse model is manipulated. When a toggle user interaction request is made, the processing to switch, for example, from the forward state to the reverse state, is similar to a context switch. The marking of the current state in the forward direction is updated, the forward state is saved, the reverse state is restored, and the reverse direction marking is updated.

5.2.2.2 Development of Complete Visualization Tool

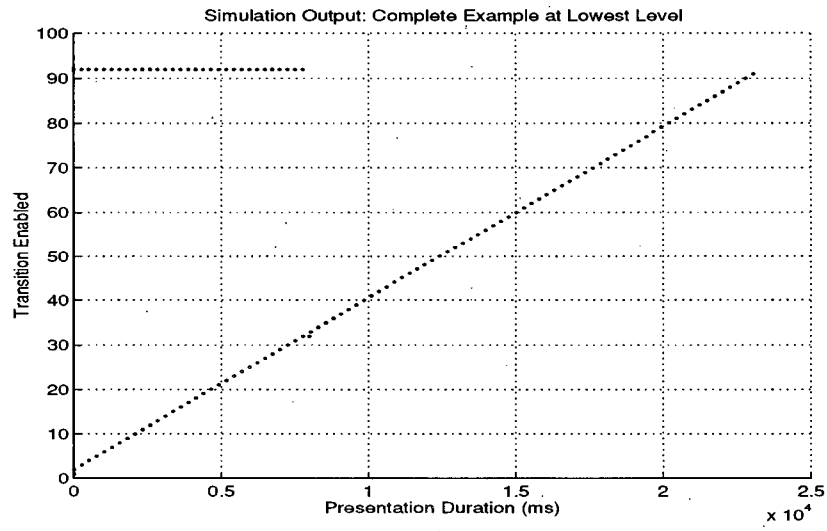
The development of a complete Visualization Tool is a project that requires a significant time investment. A phased development plan is suggested to implement the tool. The first phase includes the development of an engine which may be used to simulate the specification and support the main features described above. The input and output for this phase are simple, and the primary goal is a correct implementation. The second phase includes the development of a graphic user interface which would make the input of the data simpler and the output of the data look like a graphic Petri net model progressing. The third phase includes re-implementing the tool in a compiled language to improve performance. The first phase of the development is completed for the thesis.

5.3 *What if* Scenarios

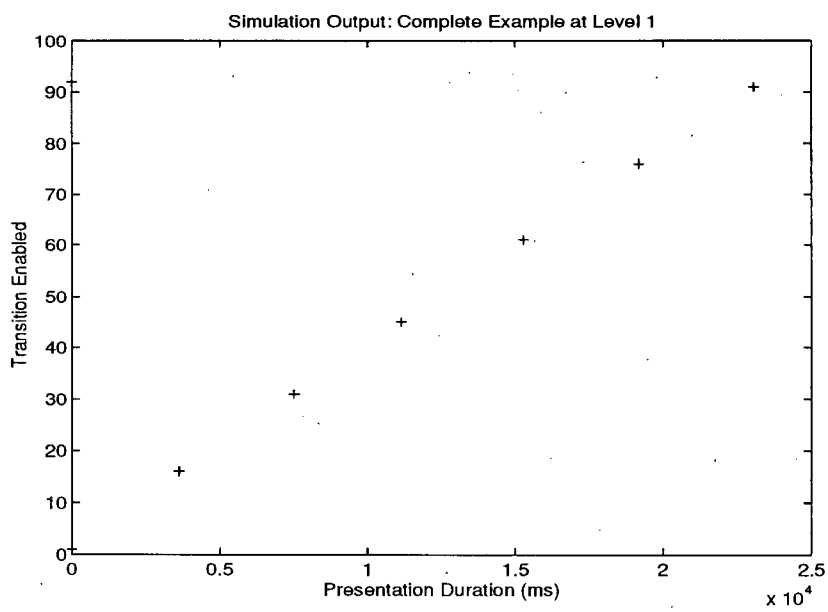
The "what if" scenarios are used to visualize the original example specification and several variations. The natural language specification for the example is available in Appendix A and its $TSPN_{UI}$ specification is available in section 4.4. The output available from the tool is both textual and a plotted graph. The textual information provides detailed information about the simulation including: the marking pattern progression, the enabled transition progression, the presentation duration (as observed by the user), execution time for the places, and the user interactions and firing of transitions that occur. The plotted graph summarizes the progression of the enabled transitions. Progressively more complicated examples of using the Visualization Tool are available in [Coo95].

5.3.1 *What if* Scenario Set 1

This scenario simulation run is of the example used throughout the thesis. specified in the $TSPN_{UI}$ model. No user interactions are scripted for this scenario, other than "start". The purpose of this simulation run is to provide a simple scenario to illustrate the output of the tool (refer to Figures 19 and 20). The text output follows the progression of the specification through the start user interaction and two transitions firing. The graphic output illustrate the progression of the specification at two levels. Figure 19 (a) describes the output at the lowest level of the specification i.e. as if the hierarchy of the specification is removed and the specification is observed at each transition. The plot indicates the points in the presentation duration that the transition is enabled. As this level of detail may not be needed at all times by the developer, a higher level of abstraction is available in Figure 19 (b). This



(a)



(b)

a. Complete Example at Lowest Level, No User Interactions

b. Complete Example at Moderate Level, No User Interactions

Figure 24. Scenario 1: Graphic Output for the Simulation of the Example Specification

Sep 6 1995 15:29:59 **example.out** Page 1

```
1    ****************************************************************
2
3      User Interaction Start Request Made
4
5    ****************************************************************
6
7      The composite firing interval is:
8         0     0
9
10     The presentation duration is:
11        0
12
13     The global clock is:
14        0
15
16
17   ****************************************************************
18
19     Transition Fired!
20
21   ****************************************************************
22
23     The transition that fired was:
24        1
25
26     The execution time for each of the places enabling the transition was:
27   Place
28        1
29
30   Execution Time
31        0
32
33     The new marking is:
34     Columns 1 through 12
35        0     1     0     0     0     0     0     0     0     0     0     0
36     Columns 13 through 24
37        0     0     0     0     1     0     0     0     0     0     0     0
38     Columns 25 through 36
39        0     0     0     0     0     0     0     0     0     0     0     0
40     Columns 37 through 48
41        0     0     0     0     0     0     0     0     0     0     0     0
42     Columns 49 through 60
43        0     0     0     0     0     0     0     0     0     0     0     0
44     Columns 61 through 72
45        0     0     0     0     0     0     0     0     0     0     0     0
46     Columns 73 through 84
47        0     0     0     0     0     0     0     0     0     0     0     0
48     Columns 85 through 96
49        0     0     0     0     0     0     0     0     0     0     0     0
50     Columns 97 through 108
51        0     0     0     0     0     0     0     0     0     0     0     0
52     Columns 109 through 120
53        0     0     0     0     0     0     0     0     0     0     0     0
54     Columns 121 through 132
55        0     0     0     0     0     0     0     0     0     0     0     0
56     Columns 133 through 144
57        0     0     0     0     0     0     0     0     0     0     0     0
58     Columns 145 through 156
59        0     0     0     0     0     0     0     0     0     0     0     0
60     Columns 157 through 168
61        0     0     0     0     0     0     0     0     0     0     0     0
62     Columns 169 through 180
63        0     0     0     0     0     0     0     0     0     0     0     0
64     Columns 181 through 188
65        0     1     1     0     1     0     1     0
66
67
68     The enabled transitions are:
69     Columns 1 through 12
70        0     1     0     0     0     0     0     0     0     0     0     0
71     Columns 13 through 24
72        0     0     0     0     0     0     0     0     0     0     0     0
73     Columns 25 through 36
74        0     0     0     0     0     0     0     0     0     0     0     0
75     Columns 37 through 48
76        0     0     0     0     0     0     0     0     0     0     0     0
77     Columns 49 through 60
78        0     0     0     0     0     0     0     0     0     0     0     0
79     Columns 61 through 72
80        0     0     0     0     0     0     0     0     0     0     0     0
81     Columns 73 through 84
82        0     0     0     0     0     0     0     0     0     0     0     0
83     Columns 85 through 92
84        0     0     0     0     0     0     0     1
85
86
87     The composite firing interval is:
88      258   408
89
90     The presentation duration is:
91        0
```

Sep 6 1995 15:29:59 **example.out** Page 2

```
92
93
94   ****************************************************************
95
96     Transition Fired!
97
98   ****************************************************************
99
100    The transition that fired was:
101       2
102
103    The execution time for each of the places enabling the transition was:
104  Place
105       2
106
107  Execution Time
108     259
109
110  Place
111      17
112
113  Execution Time
114     259
115
116    The new marking is:
117    Columns 1 through 12
118       0     0     1     0     0     0     0     0     0     0     0     0
119    Columns 13 through 24
120       0     0     0     0     0     1     0     0     0     0     0     0
121    Columns 25 through 36
122       0     0     0     0     0     0     0     0     0     0     0     0
123    Columns 37 through 48
124       0     0     0     0     0     0     0     0     0     0     0     0
125    Columns 49 through 60
126       0     0     0     0     0     0     0     0     0     0     0     0
127    Columns 61 through 72
128       0     0     0     0     0     0     0     0     0     0     0     0
129    Columns 73 through 84
130       0     0     0     0     0     0     0     0     0     0     0     0
131    Columns 85 through 96
132       0     0     0     0     0     0     0     0     0     0     0     0
133    Columns 97 through 108
134       0     0     0     0     0     0     0     0     0     0     0     0
135    Columns 109 through 120
136       0     0     0     0     0     0     0     0     0     0     0     0
137    Columns 121 through 132
138       0     0     0     0     0     0     0     0     0     0     0     0
139    Columns 133 through 144
140       0     0     0     0     0     0     0     0     0     0     0     0
141    Columns 145 through 156
142       0     0     0     0     0     0     0     0     0     0     0     0
143    Columns 157 through 168
144       0     0     0     0     0     0     0     0     0     0     0     0
145    Columns 169 through 180
146       0     0     0     0     0     0     0     0     0     0     0     0
147    Columns 181 through 188
148       0     1     1     0     1     0     1     0
149
150
151    The enabled transitions are:
152    Columns 1 through 12
153       0     0     1     0     0     0     0     0     0     0     0     0
154    Columns 13 through 24
155       0     0     0     0     0     0     0     0     0     0     0     0
156    Columns 25 through 36
157       0     0     0     0     0     0     0     0     0     0     0     0
158    Columns 37 through 48
159       0     0     0     0     0     0     0     0     0     0     0     0
160    Columns 49 through 60
161       0     0     0     0     0     0     0     0     0     0     0     0
162    Columns 61 through 72
163       0     0     0     0     0     0     0     0     0     0     0     0
164    Columns 73 through 84
165       0     0     0     0     0     0     0     0     0     0     0     0
166    Columns 85 through 92
167       0     0     0     0     0     0     0     1
168
169
170    The composite firing interval is:
171     258   408
172
173
174    The presentation duration is:
175     259
```

1

Figure 25. Scenario 1: Textual Output for the Simulation of the Example Specification

moderate level follows the progression of the specification at the hierarchy at the second level of the specification. The plot indicates when the last transition in a hierarchy first becomes enabled.

5.3.2 *What if* Scenario Set 2

This scenario is of a modified version of the example $TSPN_{UI}$ specification. In this scenario, an error is introduced into the specification such that one of the transitions is not connected to any place by a non-zero Wpt relation. Figure 25 illustrates the error introduced into the specification. The missing arc relation, on transition 76, is circled. Figure 26 contains the graphed output for the run and includes a zoomed in section on the problem in the specification. The graphed output shows that a large part of the presentation is not specified to play

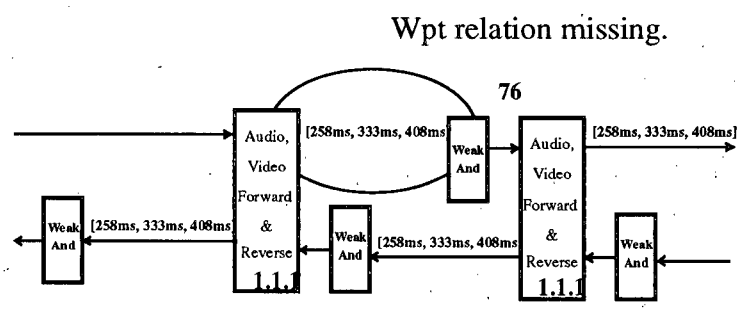
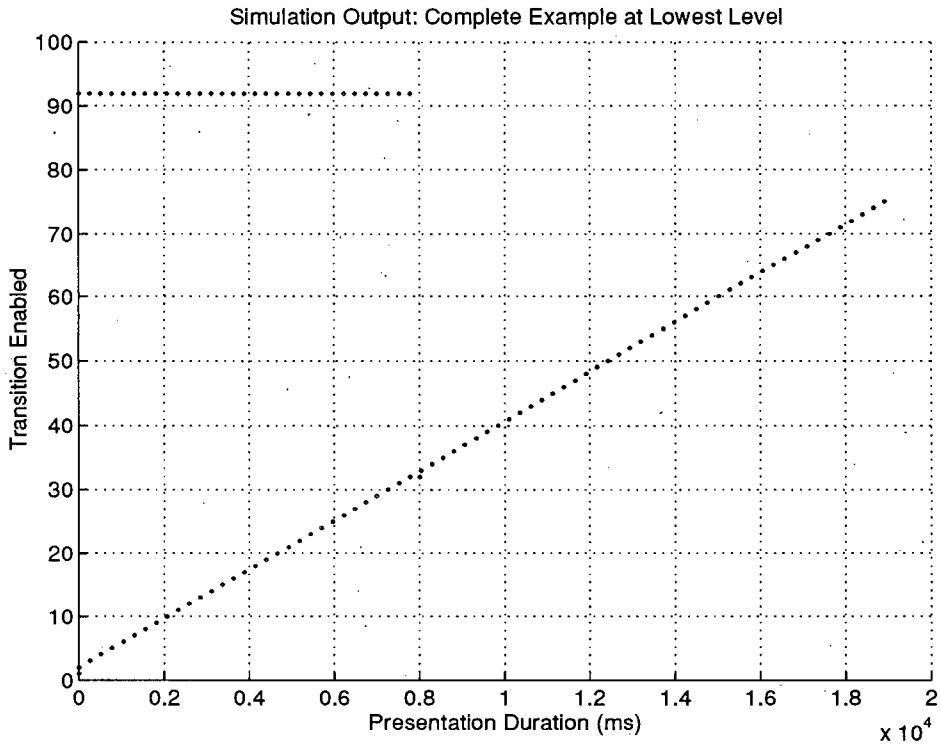
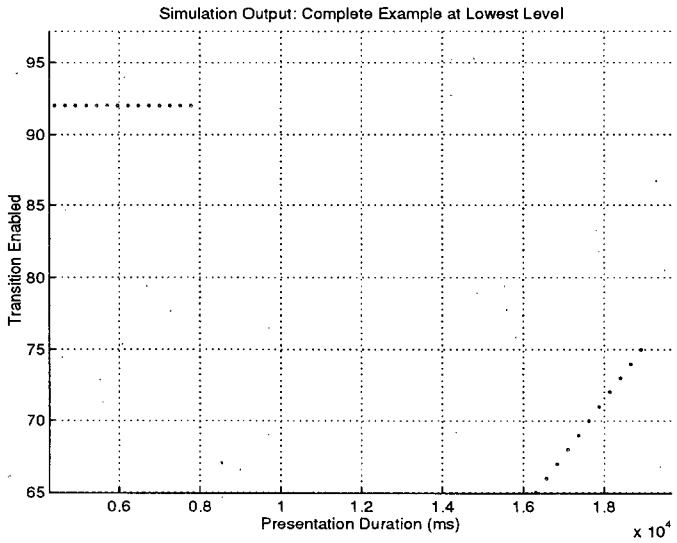


Figure 26. Simulation of time interval modified specification

out, as the transitions 76 through 91 are never enabled. Using the directed nature of the firing, transition,76 is the reasonable place to begin looking for the error.



(a)



(b)

a. Incorrect Example at Lowest Level, No User Interactions
b. Zoomed in Section of Plot from Figure(a).

Figure 27. Graphed Output Showing Transitions That are Never Enabled

5.3.3 *What if* Scenario Set 3

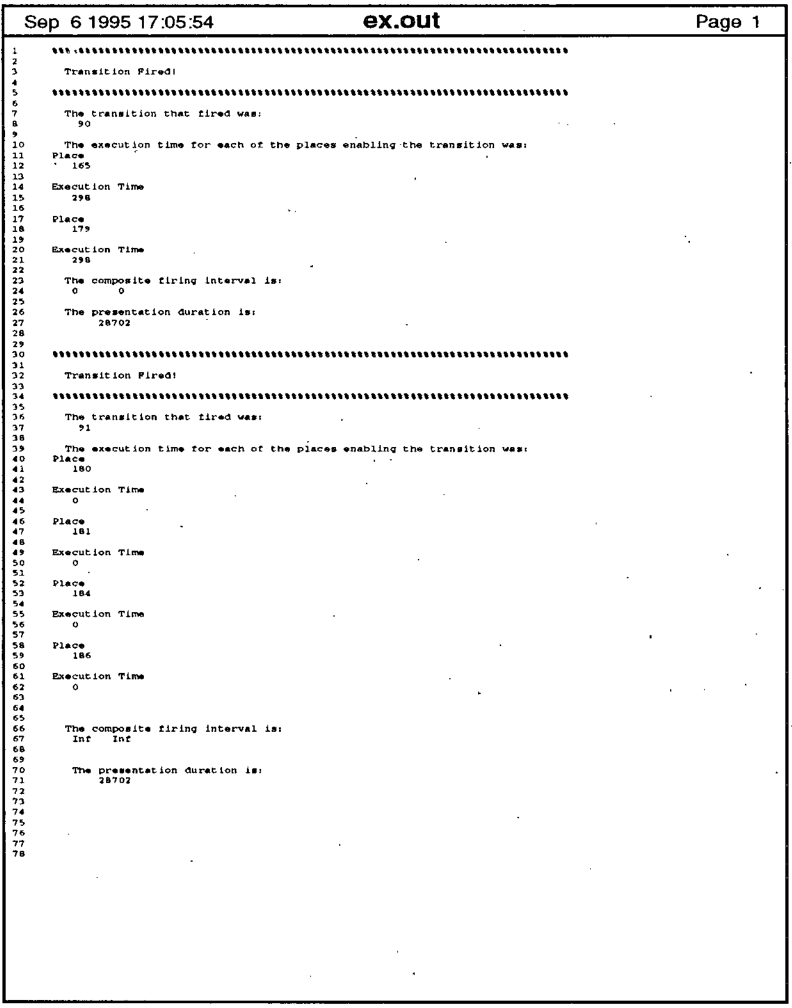
This scenario is of a modified version of the $TSPN_{UI}$ specification. The last typed transition is the specification, originally a Master transition on the Audio 3 stream, is modified to a Weak-and for one run and a Strong-or for a second run. The textual output showing the results during the last transition for the Master and Strong-or type transitions are provided in Figures 27 and 28, respectively. The marking of the places and the enabled transitions output

```
Sep 6 1995 16:50:45                ex.out                          Page 1

1     ****************************************************************
2
3       Transition Fired!
4
5     ****************************************************************
6
7       The transition that fired was:
8         90
9
10      The execution time for each of the places enabling the transition was:
11    Place
12       165
13
14    Execution Time
15       258
16
17    Place
18       179
19
20    Execution Time
21       258
22
23      The composite firing interval is:
24       258    408
25
26      The presentation duration is:
27           23053
28
29
30    ****************************************************************
31
32      Transition Fired!
33
34    ****************************************************************
35
36      The transition that fired was:
37        91
38
39      The execution time for each of the places enabling the transition was:
40    Place
41       180
42
43    Execution Time
44       258
45
46    Place
47       181
48
49    Execution Time
50       258
51
52    Place
53       184
54
55    Execution Time
56       0
57
58    Place
59       186
60
61    Execution Time
62       0
63
64
65
66      The composite firing interval is:
67       Inf    Inf
68
69      The presentation duration is:
70           23311
71
72
73
74
75
76
77
78
```

Figure 28. Simulation of type transition modified specification: Master Audio 3

is removed to emphasize the composite firing interval value in addition to modifying the inter-



Sep 6 1995 17:05:54 **ex.out** Page 1

```
%%%.%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  Transition Fired!

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  The transition that fired was:
    90

  The execution time for each of the places enabling the transition was:
Place
   165

Execution Time
   298

Place
   179

Execution Time
   298

  The composite firing interval is:
   0      0

  The presentation duration is:
       28702


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  Transition Fired!

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  The transition that fired was:
    91

  The execution time for each of the places enabling the transition was:
Place
   180

Execution Time
   0

Place
   181

Execution Time
   0

Place
   184

Execution Time
   0

Place
   186

Execution Time
   0



  The composite firing interval is:
   Inf   Inf


   The presentation duration is:
       28702
```

Figure 29. **Simulation of type transition modified specification Strong-or**

vals on the second image and text presentation objects. For this example, these intervals are set to zero. In the Master type transition case, the audio (and video) data are played out for at least the minimum playout time. In the strong-or type transition case, the audio and video data are not played out at all, as the typing rule has forced the presentation to progress.

5.4 Discussion

The Visualization Tool is a prototype which provides a means of discovering and correcting errors that occur during the simulation of a $TSPN_{UI}$ specification. The tool supports the use of "what if" scenarios: the three basic types of transitions; modifying values for arc time intervals; and scripting user interactions may be modified by the developer. The description and reuse of hierarchies is very useful in applications with significant amounts of repetition.

As the Visualization Tool is a prototype, there are a number of aspects which may be improved including the input and output functionality, performance of the tool, and the development of external documentation (e.g. user guide). In the current version, the developer using the Visualization Tool requires an understanding of the data structures used because the user inputs the matrices directly. A data input interface which abstracts the underlying data structures to the developer would be useful. With this feature, the data structures may be created, modified, and deleted in a specification through a graphic interface. The output of the tool also needs to be enhanced so that the results are easier to interpret. For example, a graphic depiction that looks like a Petri net model progressing would be useful. The MATLAB tool SIMULINK may be useful for developing the input and output graphics [MAT94].

A drawback of the Visualization Tool is its performance. The example simulated processes over thirty thousand samples and requires over ten hours to run on a workstation. The tool is developed in an interpreted language, and requires over twenty five hundred lines of code. Re-implementing the tool in a compiled language would improve the performance.

5.5 Conclusions

This chapter presents a Visualization Tool which is used to simulate a $TSPN_{UI}$ specification. The first phase of the development of the tool is complete and shows how such a tool may be of use to a developer. The purpose of the tool is to aid the developer of a $TSPN_{UI}$ specification to correctly describe the requirements for a mm presentation because an error is less expensive to correct earlier in the development cycle than later. The tool supports re-using hierarchy diagrams and combining them in series or in parallel, *What if* scenarios on the time intervals and transition types, and scripting user interactions. The current output for the tool provides both a textual output and a graphed output. The textual output is suitable for following the progression of the tool in detail. The graphed output is useful for obtaining a high level summary of the progression of the specification.

The following phases of the tool development include adding a graphical user interface, followed by re-implementing the tool in a compiled language. The modifications to the code to support a SIMULINK extension for the graphics are expected to be straightforward because the input and output functions are isolated in their own modules.

6 Conclusions and Future Work

6.1 Conclusions

A set of requirements which impact synchronization are documented for a generic multimedia application. Additional requirements such as where to place images, do not impact the synchronization requirements, and are not included in the discussion. Since the requirements are analyzed for a generic application, not all may be applicable to a specific project. The generic requirements provide a template which may be tailored to a specific application. The requirements which are used may be prioritized. The relationships among the requirements are provided at a high level to assist in maintaining consistency in the specification as it evolves and during the high level design.

The analysis of the specification approaches available provides a means to select an appropriate modelling approach to specify an application with and highlights the need for extensions to existing models. A specification approach may be selected which supports modelling the synchronization requirements, in addition to understanding the risks associated with the selection. Requirements evolve as projects are developed, which may mean the addition of a new requirement. The risk of choosing the approach which only covers the requirements known in the early stages of a project is that a new requirement is added which may not be modelled. For example, if the application does not have any user interactions specified early in the project and TSPN model is selected, significant rework may be required if user interaction requirements are added later.

The analysis of the specification approaches proposed in the literature indicates that no single Petri net model extension may describe all of the synchronization requirements. A Petri

net model extension, $TSPN_{UI}$, is proposed which extends the TSPN approach to allow the specification of user interactions. The $TSPN_{UI}$ model is based on a toolkit approach, which provides the two lowest layers of the hierarchy as building blocks. From these, a specification may be developed by creating higher level hierarchies that are suitable for the application.

The $TSPN_{UI}$ approach provides a number of advantages in specifying a model:

- The complexity associated with the requirement to playout a presentation in either the forward or the reverse direction is hidden in the two lowest layers. The addition or removal of the requirement for presenting in one or both directions is encapsulated in one place.
- With the exception of the start user interaction, the user interactions are hidden in the two lowest layers. The addition, modification, or deletion of a user interaction is encapsulated in one place.
- The time requirements are fed into the reused components at a higher layer. This allows the reuse of a playout presentation unit with different timing requirementsx. If this was not done, each presentation unit with unique time requirements for the earliest firing time, nominal firing time, and latest firing time would need to be described separately.
- The two lowest layers of the hierarchy are combined as needed with different types of transitions. This provides a very flexible approach, as only combinations which are required for a particular system are described.

- A flexible set of primitive user interactions are defined, including toggling the presentation direction, skipping ahead n presentation time units, freezing and resuming the presentation, and scaling the presentation speed. Composite user interactions are built with these primitives. For example, skipping back n presentation time units is composed of a toggle presentation interaction, followed by skipping ahead n presentation units and then with another toggle presentation interaction. The user interactions for a project may be built as required.
- $TSPN_{UI}$ uses a consistent definition (zero, positive rational number, or unbounded) for the tuple components. Time is never a negative value.
- The execution time is defined on the arc to keep it coupled with the nominal playout duration time. Using the arc definition, the specification explicitly describes the jitter and playout durations graphically, as in the TSPN approach.
- Pre-emption allows the specification of user interactions which are not constrained by the granularity of the presentation units. Response times are tied to the granularity of the coarsest stream in the OCPN and TSPN models and to the DTPN's primary stream.
- The logic necessary to control the synchronization is simpler using the simplified approach (treats all streams the same) than the original DTPN approach (primary, secondary).

The Visualization Tool allows the visualization of the specification before the presentation system is developed. The advantage of this is that errors may be seen and corrected

early in the development cycle, when they are the least expensive to correct [Boo91]. Providing two types of output is useful in developing the specification. The plotted output provides a high level summary of how the model progresses. The developer can observe if a particular transition is never enabled, for example. The textual output provides detailed information to help the developer evaluate the correctness of the specification. The tool, however, is a prototype. The performance, user interface, and external documentation for the tool need attention.

6.2 Future Work

There are numerous interesting areas to investigate using the work discussed in this thesis as a basis. Firstly, the example used throughout this thesis may be extended to include additional synchronization requirements such as tolerable transmission error rates, transmission error handling, and multiple destination sites. In order to support the graphic, formal specification of these requirements, the $TSPN_{UI}$ approach may be extended. An interesting area to pursue is the possible reuse of the primitive user interactions described to support QoS induced resynchronization. Instead of an end user causing a resynchronization, a QoS monitor may determine when the QoS is no longer within the tolerances of the presentation application and use the interactions to resynchronize the presentation. If successful, this investigation would produce a model that uses a single, consistent approach to handle resynchronization caused by either the end user or the network. In addition, describing the synchronization requirements among multiple destination sites is an interesting modelling problem. The example may be worked out in the $TSPN_{UI}$ approach to determine if it is a reasonable approach to use or if a more powerful modelling extension is necessary.

The integration of the Gate concept from the FDT LOTOS with the $TSPN_{UI}$ model may be useful in order to formalize the hierarchy diagrams. Similar work has been done integrating the Gate concept with extended finite state machine hierarchies [Ani95]. The use of hierarchies in Petri net modelling is intuitive, but is not part of the formal definition.

Enhancements to the Visualization Tool to include graphic interface for the input and output of the simulations is a useful area to pursue. In addition, analysis functionality for the Petri net models may be useful for developers of specifications. Adding this functionality is

another interesting area of work to pursue. To improve performance, the Visualization Tool may be re-implemented using a compiled language, such as C or C++, as opposed to the intepreted language available in the MATLAB environment.

References

[All83] J. Allen, "Maintaining knowledge about temporal intervals", *Communications of the ACM*, Vol. 26, No. 11, November, pp. 832-843, 1983.

[Ani95] N. Anisimov, A. Kovalenko, and P. Postupalski, "Compositional approach for specification of multimedia objects using petri nets (proposal), Institute for Automation and Control Processes, Far East Branch of the Russian Academy of Sciences, Vladivostok, Russia, 1995.

[ANS84] ANSI/IEEE Std 830-1984, *An American National Standard IEEE Guide to Software Requirements Specifications*, 1984.

[Aud94] Audio FAQ sheet Version 3.09, World Wide Web site: http://www.cis.ohio-state.edu/hypertext/faq/usenet/audio-fmts, August 1,1994.

[Bol88] T. Bolognesi and E. Brinksma, "Introduction to the ISO Specification Language LOTOS", *First International Conference on Formal Description Techniques*, 6-9 September, University of Stirling, 1988.

[Boo91] G. Booch, *Object oriented design with applications*, Benjamin/Cummings Publishing Co., USA, 1991.

[Com95] Compression FAQ sheet, World Wide Web site, http://www.cis.ohio-state.edu/hypertext/faq/usenet/compression-faq/top.html, 1995.

[Coo95] K. Cooper, *$TSPN_{UI}$ Visualization Tool :User Guide,* Draft, Department of Electrical Engineering, The University of British Columbia, Vancouver, Canada, 1995.

[Edb92] Estelle simulator/debugger, Institut National des Telecommunications, Systems and Networks Department, France, 1992.

[Gib91] S. Gibbs, L. Dami, and D. Tsichitzis, "An object-oriented framework for multimedia composition and synchronisation", *Proceedings of the Eurographics Multimedia Workshop*, Stockholm, 1991.

[Gon95a] J. Goncalves, K. Cooper, S. Vuong, and M. Ito, "A classification of multimedia application requirements", 1995 Pacific Workshop on Distributed Multimedia Systems, Honolulu, Hawaii, March 31-April 2, 1995.

[Gon95b] J. Goncalves, K. Cooper, S. Vuong, M. Ito, and S. Stiubiener, "Synchronization of multimedia objects", PSTV'95 IFIP WG 6.1 Fifteenth International Symposium on protocol specification, testing and verification, Warsaw, Poland, 13 - 16 June 1995.

[Har87] D. Harel, A. Pnueli, J. Schmidt, and R. Sherman, "On the Formal Semantics of Statecharts (Extended Abstract)", in Proceedings 2nd IEEE Symposium on Logic in Computing Science , pp. 54-64, 1987.

[Heh90] D. Hehmann, M. Salmony and H. Stuttge, "Transport services for multimedia applications on broadband networks", *Computer Communications*, pp 197-203, May 1990.

[Hin93] M. Hinchey, "Formal methods for system specification an ounce of prevention is worth a pound of cure", *IEEE Potentials*, October, pp 50-52, 1993.

[Hoe92] P. Hoepner, "Synchronizing the presentation of multimedia objects", *Computer Communications*, pp 557-564, Volume 15, No 9, November 1992.

[Hof93] H. Hofmann, *Requirements Engineering A Survey of Methods and Tools,* NR. 93.05, Institut fur Informatik der Univeritat Zurich, March, 1993.

[Iin94] M. Iino, F. Day, and A. Ghafoor, "An Object-Oriented Model for Spatio-Temporal Synchronization of Multimedia Information", Multimedia, 1994 International Conference, pp 110-119, 1994.

[Lit90] T. Little and A. Ghafoor, "Synchronization and storage models for multimedia objects", *IEEE Journal on Selected Areas in Communications*, pp 52-61, Volume 8, No 3, April 1990.

[Lit93] D. Little and A. Ghafoor, "Interval-Based Conceptual Models for Time-Dependent Multimedia Data", *IEEE Transactions on Knowledge and Data Engineering*, pp 551-563, Vol. 5, No 4, August, 1993.

[Log92] L. Logrippo, M. Faci, and M. Haj-Hussein, "An introduction to LOTOS: learning by examples", Computer Networks and ISDN Systems, 23, 325 - 342, 1992.

[LOT89] LOTOS Compiler Version 2.3, The University of Ottawa, Canada1989.

[MAT94] MATLAB Version 4.2c, The MathWorks, Inc., 1994.

[Mer76] P. Merlin and D. Farber, "Recoverability of Communication Protocols - Implications of a Theoretical Study", *IEEE Transactions on Communications*, Vol. 24, No. 9, September, pp 1036-1043, 1976.

[Mur89] T. Murata, "Petri Nets: Properties, Analysis and Applications", *Proceedings of the IEEE*, Vol. 77, No. 4, April, pp 541-580, 1989.

[Mur94] J. Murray and W. van Ryper, *Graphics File Formats*, O'Reilly and Associates, Inc. , USA, 1994.

[Pet81] J. Peterson, *Petri net theory and the modeling of systems*, Prentice-Hall, Inc., New Jersey, 1981.

[Pla92] N. Plat, J. van Katwijk, and H. Toetenel, "Application and benefits of formal methods in software development", *Software Engineering Journal*, September, pp 335-346, 1992.

[Pra93] B. Prabhakaran and S. Raghaven, "Synchronization Models for Multimedia Presentation with User Participation", Proceedings of ACM Multimedia 93, June, California, USA., pp 157-166, 1993.

[Ram74] C. Ramchandani, "Analysis of Asynchronous concurrent systems by timed petri nets", MIT MAC-TR-120, 1974.

[Rez95] M. Rezai, *Global Petri Net Modeling of Hybrid Systems and Fault Analysis*, Ph. D. Thesis, The Univeristy of British Columbia, Canada, 1995.

[Sai93] H. Saiedian and M. Awad, "Managing synchronization and time factors in multimedia presentation", *Information and Software Technology*, Vol. 35, No. 11/12 November/December, pp 653-657, 1993.

[Sen94] P. Senac, M. Diaz, and P. de Saqui-Sannes, " Toward a formal specification of multimedia synchronization scenarios", *annals of telecommunications,* Vol. 49, No. 5-6, May-June, pp 297-314, 1994.

[Sif80] J. Sifakis, "Performance evaluation of systems using nets", Lecture Notes in Computer Science, W. Brauer ed., 1980.

[Sto63] R. Stoll, *Set Theory and Logic*, Dover Publications, Inc., New York, 1963.

[Vuo95a] S. Vuong, K. Cooper, and M. Ito, "Petri net models for describing multimedia synchronization requirements", to be published in the 1995 International Conference on Network Protocols, Tokyo, Japan, November 7-10 1995.

[Vuo95b] S. Vuong, J. Goncalves, K. Cooper, M. Ito, and S. Stiubiener, "On modelling multimedia synchronization requirements", IEEE Pacific Rim Conference on communications, computers, and signal processing, Victoria, BC, May 17-19, 1995.

[Wal83] B. Walter, "Timed Petri nets for modeling and analysing protocols with time", Proceedings of the IFIP Conference on Protocol Specification, Testing, and Verification, North Holland, H. Rudin and C. West eds., III, pp 1983.

[War85] P. Ward and S. Mellor, *Structured development for real-time systems*, Yourdon

Press, USA, 1985.

[Woo94] M. Woo, N. Qazi, and A. Ghafoor, "A synchronization framework for communication of pre-orchestrated multimedia information", *IEEE Network*, January/February 1994.

Appendix A. Natural Language Specification

- The presentation is composed of six video sequences (VS), three audio sequences (AS), two images (I), and two text strings (T). The presentation has a total duration of thirty seconds at the single destination site.
- Each of the VS has a playout duration of five seconds.
- Each of the AS has a playout duration of ten seconds.
- Each of the I has a playout duration of ten seconds.
- The maximum allowed jitter is 2 s for each image.
- The first T has a playout duration of ten seconds.
- The second T has a playout duration of twenty seconds.
- The maximum allowed jitter is 2 s for each string.
- The first VS, first AS, first I, and the first T all start to be presented at the same time.
- When the first VS finishes, the second VS starts.
- The second VS and the first AS finish at the same time. When they finish, the third VS and the second AS start at the same time.
- When the third VS finishes, the fourth VS starts.
- The fourth VS and the second AS finish at the same time. When they finish, the fifth VS and the third AS start at the same time.
- When the fifth VS finishes, the sixth VS starts.

- The AS and VS sequences need to be synchronized every one third of a second. The maximum allowed jitter is 75 ms for each synchronization point.
- The first I and the first T finish at the same time. When they finish, the second I and the second T start at the same time.
- The second I finishes half way through the presentation of the second T.
- When the second I finishes, the display area for the image is cleared for the remainder of the presentation.
- The sixth VS, third AS, and the second T all finish being presented at the same time. The sixth VS and the second T remain on the screen until the user indicates they are done with the presentation.
- Each of the data streams originates from a unique site.
- The data sources are all pre-orchestrated data.
- The user is allowed to interact with the presentation in the following ways:
- start

  The user requests to start the presentation. This user interaction is not allowed after a presentation has started being presented.
- skip ahead

  The user requests to skip ahead to the next presentation unit in a data stream. The direction is in the current playout direction. If the presentation is currently being played out in the forward direction, the presentation unit skipped to is the next presentation unit in the forward direction. If the current playout direction is in the reverse

direction, then the presentation unit skipped to is the next presentation unit in the reverse direction. This user interaction is not allowed after a presentation has finished or been terminated. This user interaction is not allowed after a freeze request has been made and before a resume request is made.

- skip back

  The user requests to skip back to the previous presentation unit in a data stream. The direction is in the opposite playout direction to the current playout direction. If the presentation is currently being played out in the forward direction, the presentation unit skipped to is the previous presentation unit in the reverse direction. If the current playout direction is in the reverse direction, then the presentation unit skipped to is the previous presentation unit in the forward direction. This user interaction is not allowed before a presentation has started to be presented.This user interaction is not allowed after a freeze request has been made and before a resume request is made.

- restart

  The user requests to start the presentation from the beginning in the current playout direction. If the presentation is currently being played out in the forward direction, the presentation unit skipped to is the first presentation unit that is played out in the forward direction. If the current playout direction is in the reverse direction, then the presentation unit skipped to is the first presentation unit that is played out in the reverse direction.This user interaction is not allowed before a presentation has started to be presented. This user interaction is not allowed after a freeze request has been made and before a resume request is made.

- terminate

  The user requests to terminate the presentation. After a termination request is requested, the user may only request to start the presentation.This user interaction is not allowed until a presentation has started to be presented.

- freeze

  The user requests to freeze the presentation. The playout of the presentation may resume when the user initiates a resume request. This user interaction is not allowed until a presentation has started to be presented. Once a freeze request is made, the only allowed commands are resume and terminate.

- resume

  The user requests to resume the presentation. The playout direction is the playout direction that was current when the presentation was stopped. The resume request is only allowed after a freeze request has been made.

- toggle current presentation direction

  The user requests to toggle the current presentation direction. If the current presentation direction is forward, the direction is changed to reverse. If the current presentation direction is reverse, the direction is changed to forward. This user interaction is not allowed until a presentation has started to be presented. This user interaction is not allowed after a freeze request has been made and before a resume request is made.

- scale presentation speed

  The user requests to scale the presentation speed of the presentation. The presentation may be scaled either slower or faster relative to the normal playout speed. The scaling

factors allowed are: 1/2, 1, 2, where factors less than one indicate slower presentation speeds, the factor one indicates the normal presentation speed, and a factor greater than one indicate faster presentation speeds. This user interaction is not allowed after a freeze request has been made and before a resume request is made

- When the user interacts with the presentation, all components of the presentation respond to the request. For example, if a freeze request is made, all of the components freeze.
- Each user request requires only one input from the user. For example, if the current playout direction is in the forward direction at normal playout speed, and the user requests to fast reverse the presentation, only a single request is made by the user.

Appendix B. User Interaction Logic

The logic for the primitive user interactions is described for both the DTPN and the TSPNUI models. The user interactions described include skip ahead/skip ahead (n), reverse/ toggle, freeze and resume, and scale presentation speed. For each user interaction, the logic for the DTPN model is presented first followed by the logic for the $TSPN_{UI}$ model.

1 Skip ahead: DTPN Model

- n_stream

  Description: number of time units to skip ahead.

  Scope: available to all data streams for reading, only updated by primary stream.

- n_dep_sec

  Description: current number of time units to skip ahead in dependent secondary stream

  Scope: local to dependent stream.

- E_cur

  Description: number of execution time units left for playout in the current presentation unit.

  Scope: local to presentation unit.

- D_cur

  Description: normal playout presentation duration for the presentation unit.

  Scope: local to presentation unit.

WHILE streams remain to be re-synchronized

```
For the primary stream                          /* primary stream is defined */
    For the active presentation unit
    n_stream = E_cur
    E_cur = 0                                   /*P & T*/
    IF transition is not blocked
        transition fires
    ELSE
        store transition and stream blocked
    END
For each dependent stream not re-synchronized
    For the active presentation unit
    IF E_cur - n_stream <= 0 THEN               /* dependent secondary stream */
        n_dep_sec = n_stream
        WHILE n_dep_sec > E_cur and re-synchronization for stream is not blocked
            n_dep_sec = n_dep_sec - E_cur
            E_cur = 0                           / *P & T */
            IF transition is not blocked
                transition fires
            ELSE
                store transition and stream blocked
            END IF
        END WHILE
        IF re-synchronization is not blocked
```

```
            E_cur = E_cur - n_dep_sec          /* P & TM */
        END IF
    ELSE not dependent secondary stream
    END IF
END WHILE
    For each independent stream
        For the active presentation unit
        IF 0 < E_cur - n_stream THEN            /* independent secondary stream */
            E_cur = E_cur - n_stream            /*P & TM*/
        ELSE stream not independent
        END IF
```

2 Skip ahead (n): $TSPN_{UI}$ Model

- n

  Description: number of time units to skip ahead

  Scope: seen by all data streams

- current_time

  Description: current time for virtual clock

  Scope: seen by all data streams

- current_time_old

  Description: previous time-stamp for virtual clock

  Scope: seen by all data streams

```
Determine which transitions are enabled
WHILE n>0
        current_time_old = current_time
        calculate firing interval for enabled transitions using TSPN rules
        fire an enabled transition
        n = n - (current_time - current_time_old)
        determine which transitions are enabled
END WHILE
```

3 Reverse: DTPN Model

- n_stream

  Description: number of time units to skip back

  Scope: available to all data streams for reading, only updated by primary stream.

- n_dep_sec

  Description: current number of time units to skip back in dependent secondary stream

  Scope: local to dependent stream.

- E_cur

  Description: number of execution time units left for playout in the current presentation unit

  Scope: local to presentation unit

- D_cur

  Description: normal presentation duration for the presentation unit

  Scope: local to presentation unit

```
WHILE streams remain to be re-synchronized
    For the primary stream
        For the active presentation unit
            IF previous presentation unit exists THEN
            n_stream = D_cur - E_cur
            reverse playout direction
            E_cur = 0                                   /* P & T */
```

```
    IF transition is not blocked
        transition fires
    ELSE
        store transition and stream blocked
    END
ELSE can't reverse presentation playout direction


For each dependent stream not re-synchronized
    For the active presentation unit
        IF D_cur < E_cur + n_stream THEN /* dependent secondary stream */
        n_dep_sec = n_stream
        reverse playout direction
        WHILE (n_dep_sec >= (D_cur - E_cur) ) and
            (re-synchronization for stream not blocked)
            n_dep_sec = n_dep_sec - (D_cur - E_cur)
            E_cur = 0                          /* P & T */
            IF transition is not blocked
                transition fires
            ELSE
                store transition and stream blocked
            END IF
        END WHILE
        E_cur = D_cur - n_dep_sec              /* P & TM */
```

```
    ELSE not dependent secondary stream


For each independent stream
    For the active presentation unit
        IF D_cur >= E_cur + n_stream THEN /* independent secondary stream*/
            reverse playout direction
            E_cur = E_cur + n_stream            /*P & TM*/
        ELSE stream not independent
```

4 Toggle: $TSPN_{UI}$ Model

- E_cur

  Description: execution time left for playout in the current presentation unit

  Scope: local to presentation unit

- D_cur

  Description: constant presentation duration for the presentation unit

  Scope: local to presentation unit

- E_retained

  Description: execution time left for playout in the current presentation unit which is retained.

  Scope: local to presentation unit.

For each stream

For the active presentation unit

E_retained = E_cur

E_cur = 0

reverse playout direction

E_cur = D-cur - E_retained

5 Freeze and Resume: DTPN Model

5.1 Freeze

- E_cur

  Description: execution time left for playout in the presentation unit.

  Scope: local to presentation unit.

- E_retained

  Description: execution time left for playout in the presentation unit which is retained.

  Scope: local to presentation unit.

```
For all data streams
    For the active presentation unit
        E_retained = E_cur
        E_cur = 0                                   /* P & D */
        activate the "frozen" place
        E_cur = very large
```

5.2 Resume

- E_cur

  Description: execution time left for playout in the presentation unit.

  Scope: local to presentation unit.

- E_retained

  Description: execution time left for playout in the presentation unit which is retained.

Scope: local to presentation unit.

```
For all data streams
    For the active "frozen" unit
        E_cur = 0                                   /* P &T */
        re-activate presentation unit
        E_cur = E_retained
```

6 Freeze and Resume: $TSPN_{UI}$ Model

6.1 Freeze

- E_cur

  Description: execution time left for playout in the current presentation unit.

  Scope: local to presentation unit.

- E_retained

  Description: execution time left for playout in the presentation unit which is retained.

  Scope: local to presentation unit

  For all streams

  For the active presentation unit

  E_retained = E_cur

  E_cur = 0

  activate frozen place

  E_cur = very large

6.2 Resume

- E_cur

  Description: execution time left for playout in the current presentation unit.

  Scope: local to presentation unit.

- E_retained

  Description: execution time left for playout in the presentation unit which is retained.

Scope: local to presentation unit.

For all data stream s

For the active presentation unit

E_cur = 0

re-activate presentation unit

E_cur = E_retained

7 Scale Presentation Speed: DTPN Model

- E_cur

  Description: number of execution time units left for playout in the current presentation unit

  Scope: local to presentation unit

- scaling_factor

  Description: scaling factor to speed up or slow down the presentation

  Scope: seen by all data streams

- E_retained

  Description: execution time left for playout in the presentation unit which is retained.

  Scope: local to presentation unit.

```
For all streams
  For all presentation units
    D = scaling_factor * D                    /* P & PM */
  For all active presentation units
    E_cur = scaling _factor * E_cur
  For all frozen presentation units[1]
    E_retained = scaling _factor * E_retained
```

1. This processing is necessary, however was omitted from the discussion in the original DTPN approach.

8 Scale Presentation Speed (n): $TSPN_{UI}$ Model

- E_cur

  Description: number of execution time units left for playout in the current presentation unit

  Scope: local to presentation unit

- scaling_factor

  Description: scaling factor to speed up or slow down the presentation

  Scope: seen by all data streams

- E_retained

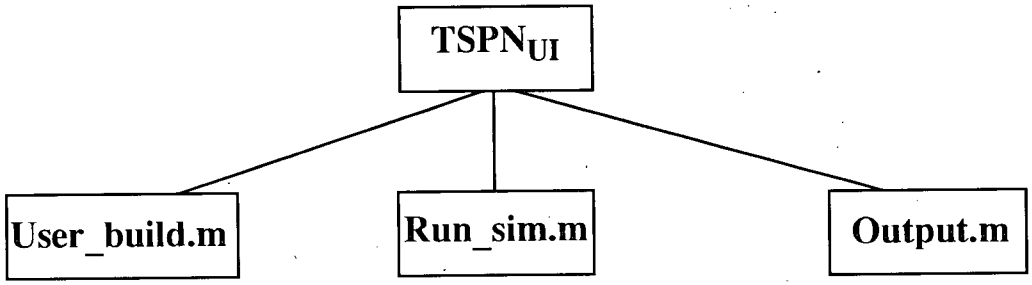
  Description: execution time left for playout in the presentation unit which is retained.

  Scope: local to presentation unit.

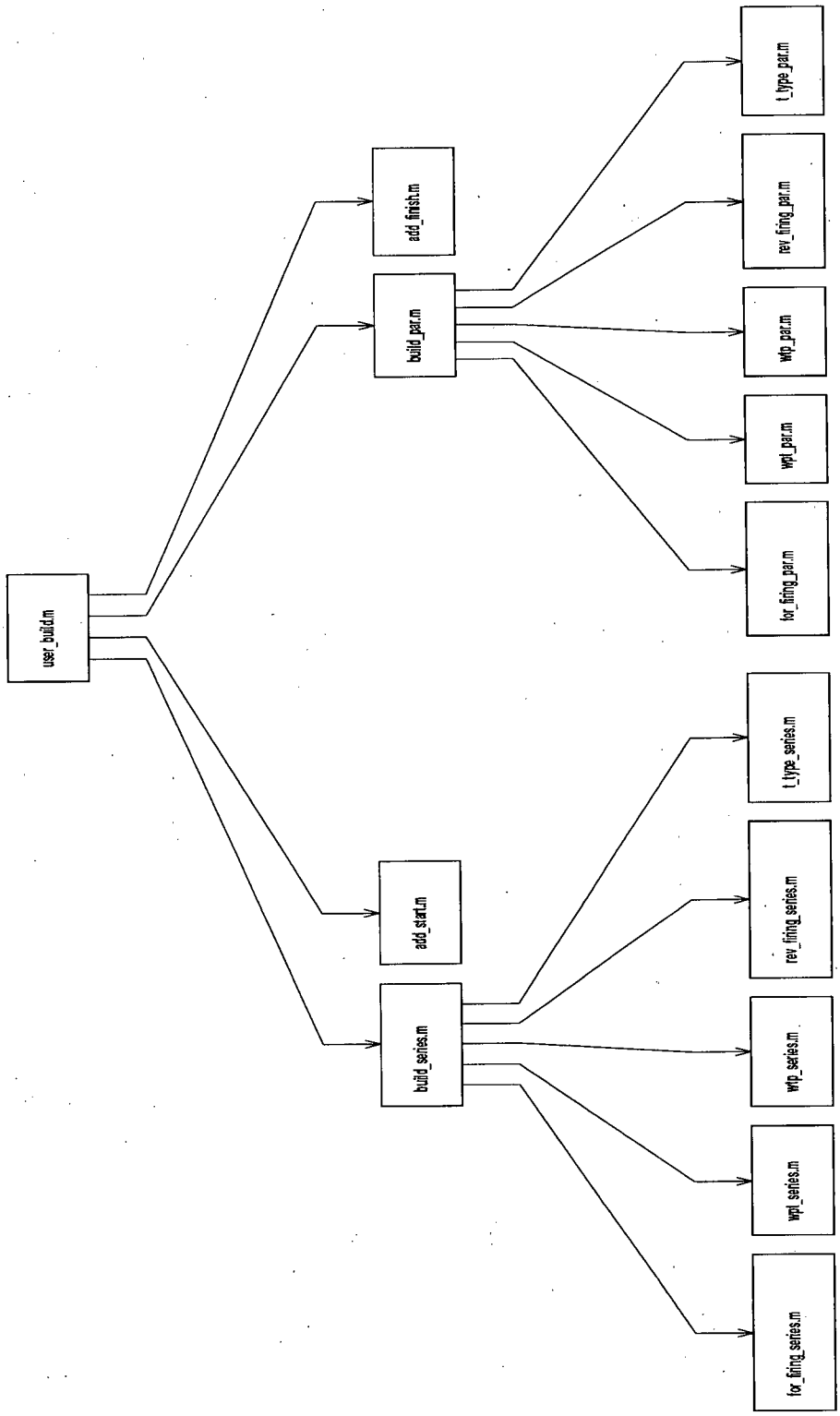
```
For all streams
  For all presentation units
    D = scaling_factor * D
  For all active presentation units
    E_cur = scaling _factor * E_cur
  For all frozen presentation units
    E_retained = scaling _factor * E_retained
```

Appendix C. Uses Diagram (Phase 1)

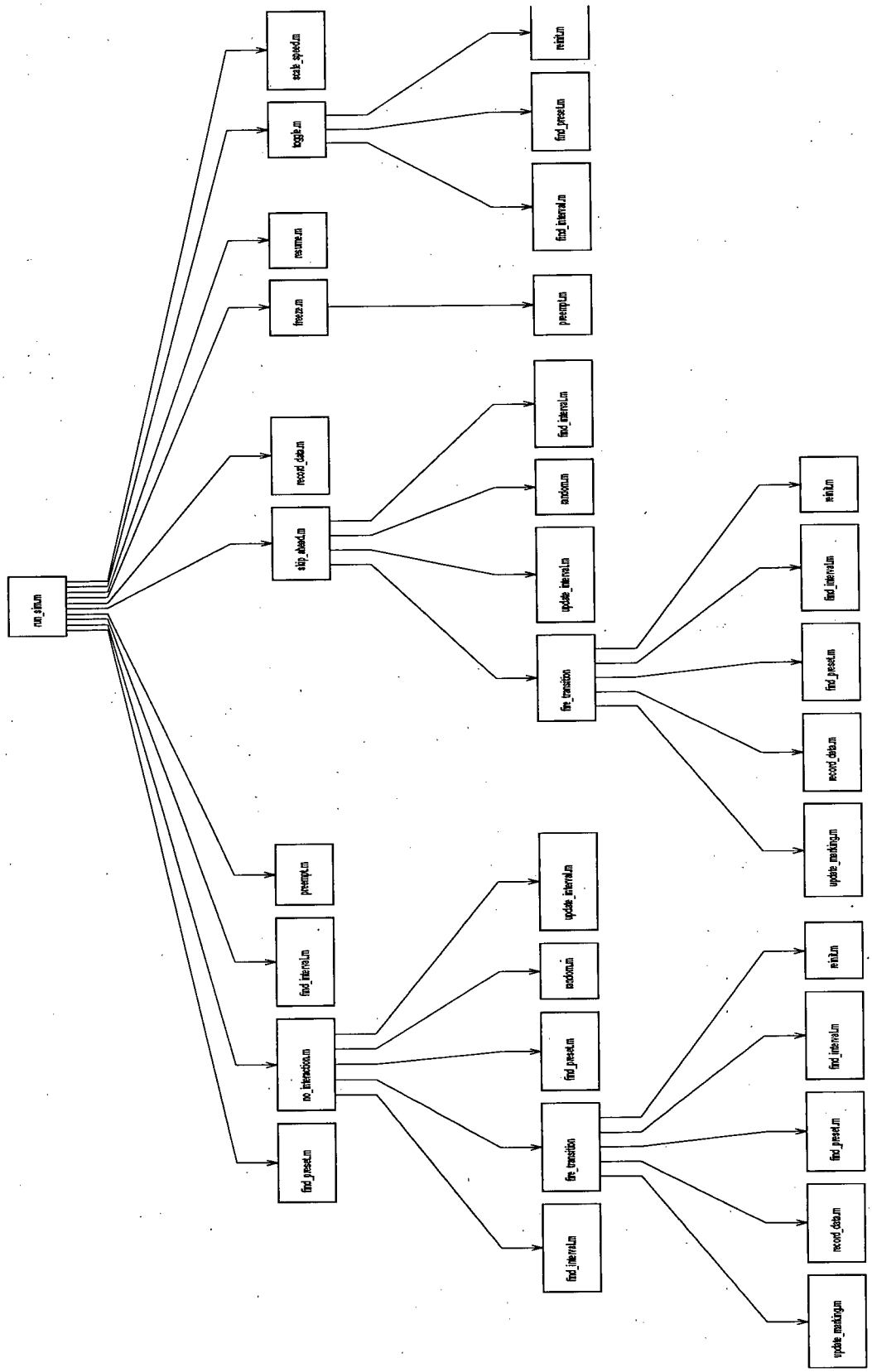
Uses Diagram Level 0



Uses Diagram Level 1.1



Uses Diagram Level 1.2



Appendix D. List of Acronyms

DTPN	Dynamic time Petri net
ERD	Entity relationship diagram
IPP	Interstream pacing point
ISP	Inter-object synchronization point
MM	Multimedia
OCPN	Object composition Petri net
SIU	Synchronization interval unit
TSPN	Time stream Petri net
$TSPN_{UI}$	Time stream Petri net with user interactions
XOCPN	Extended object composition Petri net