Stress Testing of Distributed Multimedia Software Systems

Jian Zhang[1], Shing-Chi Cheung[2] and Samuel T. Chanson[2]

[1]*Lab. of Computer Sci., Inst. of Software, Chinese Academy of Sciences*
[2]*Dept. of Computer Sci., Hong Kong Univ. of Sci. and Tech.*

Key words: Multimedia applications, stress testing, test case generation, real-time systems, resource modeling, symbolic execution, constraint solving

Abstract:

With the advance in network technologies, the use of distributed multimedia data has become very popular. Distributed multimedia systems are complex and error prone. While there are well established testing techniques for the functionalities of such systems, the technique of stress testing to improve test efficiency has not received much attention. It has been argued that faults are more likely to occur in situations where there is a fierce competition for resources. When the system is heavily loaded, errors like mismanagement of buffers can be easily identified. In this paper, we present several criteria for selecting test cases, and describe two methods for generating test cases which maximize system resource usage. Our approach is based on symbolic execution and constraint solving. An example is given to illustrate the concepts and techniques.

J. Wu et al. (eds.), *Formal Methods for Protocol Engineering and Distributed Systems*

1. INTRODUCTION

Multimedia software systems systems are becoming popular in recent years. In multimedia applications, various kinds of information media are involved, including video, audio, image and text. The architecture of interactive multimedia applications can be roughly divided into two classes: client-server (like remote learning, video-on-demand and product presentations), or peer-to-peer (like video phone and video conferencing).

Distributed multimedia software systems (DMSSs) are complicated and error-prone. They have to be responsive to user events, and they often require timely exchange of large amounts of media data. Furthermore, some of the media objects have to be synchronized (e.g., a piece of text and a speaker's voice reading the text). There can also be very complex user interaction scenarios, as in the case of virtual shopping.

To facilitate the design and implementation of DMSSs, a Distributed Multimedia System Environment (DMSE) was developed at the Hong Kong University of Science and Technology [W97]. In this environment, the programmer can specify the architecture of a DMSS in an object-oriented, textual language. He/she may also describe different views of the system using some graphical notations (including a variant of Petri net and an extended time-line diagram).

A framework for testing DMSS is described in [MCC98], where only generic testing methods are discussed. It is impractical and often infeasible to test a complex software system exhaustively. A system is more likely to misbehave when its resources are heavily utilized. In this paper, we focus on the derivation of test data which drive the system to critical states where almost all available resources are consumed. Experience tells us that in such a situation the system is more vulnerable and some types of errors (like mismanagement of buffers) can be easily identified. Testing is more effective if we concentrate on these cases.

In this paper, we give some criteria for selecting test cases such that the system is tested under more stressful conditions. We also describe two methods for generating such test cases. The basic idea is to execute the system's specification symbolically, and then use constraint solving and optimization techniques to find the timings of events which maximize system resource usage. The paper is organized as follows. In the next section, we briefly describe our specification formalism, with emphasis on a variant of Petri net. Then in Section 3, we discuss test case selection criteria for stress testing of multimedia applications, and present two methods for

automatic test case generation. Finally, our work is compared with other related work, and some observations are made.

2. MODELING OF DMSS

A model for specifying DMSS architectures is given in [W97]. It allows users to author a DMSS by describing different aspects of the system. A major part of the model is an extension of the Petri net, called Temporal Event Petri Nets (TEPNs). An event can denote some action of the user (e.g., EXIT_CLICKED) or a signal from some media object (e.g., EOF which means the end of an object). A TEPN offers a visual representation of system activities and user interactions.

TEPN adopts a discrete time model. That is, the time domain consists of time points which are non-negative integers. We shall use *nt(pl,t)* to denote the number of tokens in place *pl* at time *t*. Place *pl* is active at time *t* if *nt(pl,t)* > 0. A transition is enabled at time *t* if every input place is active at that time. When it fires, the number of tokens at each input place decreases by 1 and the number of tokens at each output place increases by 1. But if transition *v* forms a self-loop (i.e., goes from place *P* to itself), we assume that when *v* fires, the number of tokens in *P* first decreases by 1 and then increases by 1 to get back to the same value.

A TEPN is an extension of a classical Petri net. A transition may be marked with a triggering condition (i.e., the occurrence of some user interaction). It is firable if it is enabled and the condition is met. For simplicity, we assume that no two user interactions occur at the same time instant. Otherwise, the behavior of the system can be nondeterministic. A transition can also have a timeout value t_{max}. Then the transition fires if it has been enabled for t_{max} time units, or if the triggering condition is met before that time.

A place in a TEPN may be associated with a presentation (e.g., displaying a picture and then some text), which takes a finite amount of time. In such a case, we shall use the same name for a place and its presentation. Each presentation may consist of several media objects (like a paragraph of text, an image and an audio stream). The designer can specify the temporal relationships between the objects using an extended time-line diagram or using some temporal logic formulas (e.g. *A* `ends_with` *B*). A presentation is executed when the corresponding place changes from inactive to active; and it is stopped when the place becomes inactive, even though the presentation is not over. A transition can be associated with some actions

(like the migration of objects). The associated actions are executed when the transition fires.

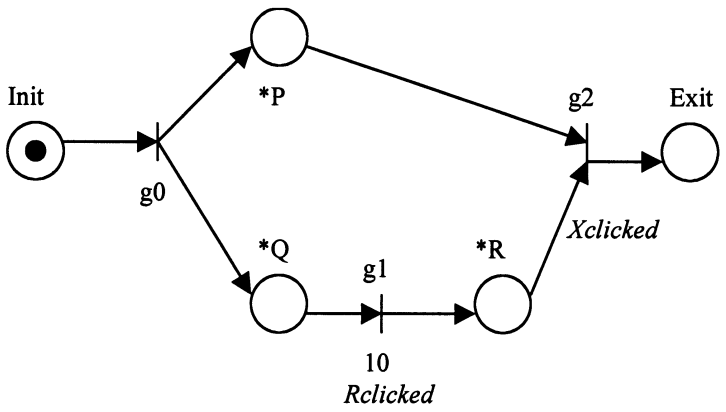

Figure 1. Example TEPN

Throughout this paper, we shall use the simple example in Figure 1 to illustrate the concepts and techniques.

The TEPN has 5 places: *Init*, *P*, *Q*, *R*, *Exit*. Among them, *P*, *Q* and *R* are associated with presentations (indicated by a star beside the name). Suppose there is a media object *A* in *P*, another object *B* in *Q*, two objects *C* and *D* in *R*, and the durations of *A*, *B* and *D* are 5, 12, and 8, respectively. In addition, there is a temporal constraint: *C* during *D*. This means that *C* can only be active when *D* is. The duration of *C* is undetermined. Transition g_1 has a timeout value of 10.

2.1 A Discrete Time Model of TEPN Executions

The TEPN and temporal specifications provide a basis for specifying, analyzing and testing distributed multimedia applications. For timed systems, many analysis problems are undecidable [BD91], [GMMP91]. The reason is that unbounded time values can make the state space infinite. But for testing purposes, we may consider the system's behavior within a finite period of time.

Given a specification, an *execution trace* is defined to be a set of timed events, i.e., { (ev_i, t_i) }. Here each t_i is a non-negative integer denoting a time point, and ev_i denotes an event in the general sense. An event is defined to be the beginning or ending of a presentation, the activation or ending of a media

object, or a user interaction. For a presentation P, we use $P\uparrow$ and $P\downarrow$ to denote its beginning and ending events, respectively, and similarly for media objects. At any time t ($t > 0$), event $P\uparrow$ happens if $nt(P,t-1) = 0$ and $nt(P,t) > 0$; and event $P\downarrow$ happens if $nt(P,t-1) > 0$ and $nt(P,t) = 0$.

In an execution trace, more than one events can happen at the same time point, and an event can happen many times. An execution trace should be well-defined. For instance, there should be an event $P\downarrow$ between two consecutive $P\uparrow$ events.

For the TEPN in Figure 1, a possible execution trace is:

$\{ (P\uparrow, 1), (Q\uparrow, 1), (A\uparrow, 1), (B\uparrow, 1), (A\downarrow, 6), (Q\downarrow, 11),$
$(B\downarrow, 11), (R\uparrow, 11), (D\uparrow, 11), (C\uparrow, 13), (Xclicked, 15),$
$(P\downarrow, 15), (R\downarrow, 15), (C\downarrow, 15), (D\downarrow, 15) \}.$

The special places *Init* and *Exit* are omitted for the sake of brevity. In this trace, g_0 fires at time 1, which activates presentations P and Q and their media objects A and B. A ends at time 6 because its duration is 5, and so on. At time 15, the user clicks on X, and g_2 fires. This terminates the presentation R (as well as its media objects C and D). Place P also becomes inactive.

3. STRESS TESTING

Specification-based testing of real-time systems and multimedia applications has been studied by some researchers (see for example, [MMM95], [AS96]). These approaches are based on syntactic features of the specification, such as the structures of formulas.

In what follows, we shall study stress testing [B84], i.e., exercising the system under strenuous conditions. The basic idea is that the system is more susceptible to errors or performance degradation when much resource is being used. For a DMSS, the three most critical resources are CPU cycles, memory and network bandwidth. Heavy communication may cause buffer overflow and packet loss. Errors may also result from insufficient CPU power or programming mistakes. From our experience, mismanagement of buffers is one of the common types of errors in developing multimedia systems. Horn and Girod also reported that when time consuming applications are running simultaneously, buffer overflow and packet loss may happen due to limited CPU capacity [HG97].

3.1 Resource Consumption

The amount of resources consumed by an active media object depends on several factors, like the supporting software/hardware, and the required quality of service (QoS). Based on our observations, we assume that for a particular media type and some fixed QoS, the amount of resources consumed is constant during the object's lifecycle, except for the following two cases:

(1) Immediately after an object is activated, much more resources are needed. This transient period may last for less than a second to a few seconds.

(2) For some media types, random peaks may occur from time to time. For example, when a singer raises her voice in a recital. Such peaks may occur at any time instant. If they occur frequently, we consider the maximum amount of resource usage; otherwise, we consider the average amount.

3.2 Test Case Selection Criteria

As mentioned earlier, we adopt the discrete time model in our analysis. For the purpose of testing, it is reasonable to restrict the time domain to be finite. Let us denote by TT the total testing time. Like an execution trace, a test case is also a (well-defined) set of timed events.

Given a set of execution traces, a trace will be selected as a test case if one of the following conditions is met:

(1) its total resource usage during the period $[0..TT]$ is maximum;
(2) it has the largest number of object beginning events;
(3) there is a large peak resource usage at some time point t ($0 < t \leq TT$).

The second condition is necessary since there is usually a significant increase in the amount of CPU usage when a media object starts or when some user interaction occurs. However the exact amount is unpredictable.

Generally speaking, it is not necessary or practical to produce test data that are strictly optimal in the above sense. Nevertheless, appropriate techniques and tools can help us find good test cases.

We shall be mainly interested in the amount of data required by the active media objects. Usually, the more data a client needs, the more resources the

system consumes. Our analyses will be based on the data rates of the media objects.

For an object with duration d, we assume that the amount of required data is $w(m,q) * d$, where w is the average or maximum amount of data needed per unit time. The parameters m and q denote the media type and the quality of services, respectively. In the following, we simply write wA for the value of the function in case of object A. For example, if A is an audio stream of telephone quality, then wA is about 56 Kbps. If the data are compressed before communication, and we are interested in maximizing the amount of data transfer, then we should divide wA by the compression ratio. For example, instead of 56 Kbps, we use 16 Kbps.

3.3 Maximizing Total Resource Usage

Obviously, if there are many time points, the number of possible execution traces can be very large. A better way is to use *symbolic execution traces*. Such a trace is also a set of timed events $\{(ev_i, t_i)\}$. But the t_i's can be either integer variables or specific integers. To avoid confusion, we shall call an execution trace a *concrete execution trace* if every t_i is a constant.

As mentioned earlier, a place can be active more than once during an execution. In a given execution trace, we use bP_i and eP_i to denote the i'th beginning and ending times of place P, respectively. Moreover, we define $fP_i = \min(eP_i, TT)$, where TT is the total testing time. (The letter i can be omitted if each place is active only once during the execution.) In the example in Section 2, we have, $bQ = 1$, $eQ = 11$ and so on. If $TT = 10$, then $fQ = 10$.

Given a specification, we can derive a set of finite symbolic execution traces. This is done by choosing sequences of transition firings. For each sequence, we record its associated conditions, and then determine the values of the variables such that some resource usage criterion is satisfied.

Let us illustrate the ideas using the previous example.

For the TEPN in Figure 1, we can see that there is only one firing sequence, namely, g_0 fires first, then g_1 and then g_2. We assume that the timings of the firings are as follows:

$$1 = bP = bQ; \quad 1 + x = eQ = bR; \quad 1 + x + y = eP = eR.$$

where $1 \le x \le 10$ and $1 \le y$.

For a media object O and its j'th activation, let bO_j and eO_j denote the *scheduled* beginning and ending times, respectively. Again, the subscript j can be omitted if the object is active only once. In the present example, we can derive the following equations and inequalities from the temporal specification of the presentations:

$$eA = bA + 5, \ ..., \ bC > bD, \ \ eC < eD.$$

Note that the object may end earlier than the scheduled time if the place becomes inactive (due to some user events) before that time. It is also possible that some objects are not activated at all. The actual running time of object A, denoted by rA, can be defined as:

$$rA = \begin{cases} 0, & \text{if } fP \le bA; \\ fP - bA, & \text{if } bA < fP \le eA; \\ eA - bA, & \text{if } fP > eA. \end{cases}$$

The variables rB, rC and rD can be defined in a similar way. Let wA (wB, wC, wD) denote the amount of data needed for object A (B, C, D) per unit time. Our goal is to maximize ($wA * rA + wB * rB + wC * rC + wD * rD$) under the temporal constraints. This represents maximal resource utilization and contention.

Given the above constraints and objective function, we can find solutions using constraint solving and optimization techniques. Suppose $TT = 15$, $wA = 100$, $wB = 1400$, $wC = 1800$, $wD = 64$. The optimal solution is: $x = 7$, $y = 7$. That is, object B runs for 7 time units, then the event *Rclicked* happens, and then place R is active for 7 time units. If we change the value of wC to 800, then the solution becomes: $x = 10$, $y = 4$. In this case, object B runs for 10 time units when transition g_1 times out. The rationale is to select the timing of events in a way which makes communication intensive objects run as long as possible, so as to maximize the total amount of data usage while satisfying the timing constraints of the application.

If we adopt the second criterion in Section 3.2, we need only change the objective function to the number of all object beginning events (before TT). This can be obtained by summing up the following quantities (over all activations of each object A):

$$nA_i = 1, \text{ if } bA_i < fP_i; \ 0, \text{ otherwise.}$$

Here P is the presentation which contains A.

We can see that even though the TEPN's control structure is very simple, several different test sequences are possible, depending on the resource usage information and on the total testing time. For more complicated systems, it is hard to determine the best (or even near optimal) test data without automated tool support. We have implemented a tool for generating the constraints. The tool NCL [Zh98] is used to obtain the solutions.

3.4 Maximizing Peak Usage

A multimedia software system can be considered to be in a critical state if several media objects are active simultaneously and they require a large amount of data. To find a time point at which the resource utilization is maximized, we first analyze the object instances to see which of them can be active at the same time. A pair of instances A_i and B_j can not be active simultaneously, if $eB_j \leq bA_i$ or $eA_i \leq bB_j$. Obviously, for any pair of integers i, j ($i \neq j$), it is impossible that A_i and A_j are active simultaneously.

The above information can be depicted as a graph, which we call the *object instance relation graph*. Each vertex represents an active instance of some media object. A vertex is associated with a weight, which represents the amount of resource consumption per unit time. An edge connects two vertices if the two object instances can be active at the same time. What needs to be solved is an extended maximal clique problem. (Some additional constraints have to be satisfied.) That is, find a subset of the vertices such that every pair of vertices are connected and the sum of the weights is maximum.

To represent the clique, we associate a set of binary variables with the vertices. For example, $xA_i = 1$ means that A_i is included in the clique. We study the satisfiability of the following constraints:

(1) $(xA_i = 0)$ or $(xB_j = 0)$, if there is no edge between vertex A_i and B_j;
(2) $(xA_i = 1) \rightarrow (bA_i < t < eA_i)$;
(3) $(xA_i = 1) \rightarrow (t < eP_i)$;
(4) $t < TT$;
(5) the constraints characterizing the symbolic execution trace.

Here P is the presentation which contains A. The objective function to be maximized is the summation of $(xA_i * wA)$ over all object instances A_i. If the problem has a solution, we obtain a concrete execution trace which uses maximal amount of data at time point t.

Let us consider again the TEPN in Figure 1. Suppose that presentation P consists of 3 objects A, E and F, with the constraints A `begins_with` F, and E `ends_with` F. The durations of the objects and resource consumption weights are given in Table 1. We note that the objects A and E can not be active simultaneously, neither can the presentations Q and R. So we get the object instance relation graph as shown in Figure 2.

Table 1. Durations and Weights of Objects in the Example

	A	*B*	*C*	*D*	*E*	*F*
duration	5	12	6	8	6	13
weight	100	1400	1800	64	150	56

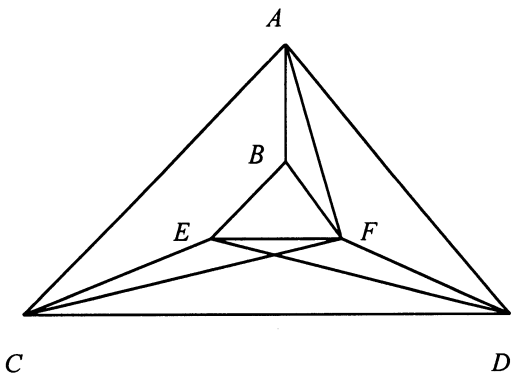

Figure 2. Object Instance Relation Graph of the Example

This optimization problem has a solution, which can be found by NCL in less than 1 second on a SPARCstation 5. The maximal clique consists of the media objects C, D, E and F. Their beginning/ending times are: $bC = 3$, $eC = 9$, $bD = 2$, $eD = 10$, $bE = 8$, $eE = 14$, $bF = 1$, $eF = 14$. At time point $t = 8$, the total amount of data usage is maximized at 2070.

Summary Given a specification and resource consumption information about each media object, we can prepare test data for stress testing in one of the following ways:

(1) Arbitrarily generate many concrete execution traces. Then pick from them the best ones using the test selection criteria given in Section 3.2.
(2) Generate some symbolic execution traces, and then use constraint satisfaction and optimization techniques to determine concrete traces with the largest amount of data usage.

The total testing time TT can be varied as needed.

4. RELATED WORK

Specification and Testing of DMSS

Several authors have extended Petri nets to specify distributed multimedia applications. Most notably, Little and Ghafoor proposed Object Composition Petri Nets (OCPN) [LG90]. It was extended later into XOCPN [WQG94]. Sénac *et al.* proposed the so-called Hierarchical Time Stream Petri Nets (HTSPN) [SSW95].

Considerable amount of work exists on conformance testing of protocols and distributed systems (see for example [S93]). Most of the test data generation methods are based on graph models of protocols. Testing of multimedia systems is more complicated because the exact timings of events are very important.

Some researchers have attempted to adapt protocol testing methods to multimedia system testing. Ates and Sarikaya [AS96] discussed an approach based on timed automata. A finite state machine (FSM) is derived from the automata, and then timed traces are obtained from the FSM. Each timed trace serves as a test case, which is a sequence of transitions from the initial state to some final state. A complex system may have a large number of timed traces. In our approach, only the most significant test cases are considered. These test cases drive the system to states where potential bugs can be revealed with high probability. It is possible to add some criteria to the approach of [AS96], for example, each transition is visited only once. But this can lead to inaccuracy. Sometimes a transition fires more than once before the system enters a critical state.

Modeling and Analysis of Real-time Systems

Real-time systems have been extensively studied. Various formalisms have been proposed to model and analyze them. They include, among others, time Petri nets [BD91], timed automata [AD94], I/O automata[LSGL95], timed CSP [ABSS96], Duration Calculus [Zcc93], Modechart [JM94].

Roughly speaking, there are three classes of techniques for studying the behaviors of real-time systems based on formal specifications:

(1) Simulation. The specification is executed. Typical executions may reveal certain properties of the system. But in most cases, there are many possible executions.

(2) Searching. This includes reachability analysis of timed Petri nets [BD91], [GMP94] and temporal logic model checking [LPW96], [YMW97]. Search-based tools are attractive in that they are automatic. But usually they have difficulty with infinite or very large state spaces. While model checking procedures are quite effective for untimed systems, verifying properties of timed systems is much more difficult in general.
(3) Theorem proving. The user develops a theory about the system based on some logic, like first-order logic or temporal logic. System properties are expressed as formulas and their truth is to be proved in the logic. Constructing nontrivial proofs tends to be arduous, although some steps can be mechanized as done in provers like the Larch Prover [LSGL95] and PVS [SS94], [AH96].

While reachability analysis and symbolic execution are well-known techniques for time Petri nets, they have not been used with resource consumption information.

The Petri net is a natural formalism for describing asynchronous systems. In contrast, many model checking tools are intended for synchronous systems. For example, the UPPAAL tool [LPW96] is designed to verify networks of timed automata. In such a model, the basic communication mechanism is through send/receive primitives which synchronize *two* automata. Yang *et al.* [YMW97] introduce a logic called SREL for specifying synchronous systems, and described an extension of McMillan's SMV tool to verify Modecharts. These approaches still have some limitations. It is hard to specify a requirement such as the following: The meeting begins when all the 3 participants are ready.

Test Data Generation and Constraint Solving

In general, automatic test data generation is an undecidable problem. However, in certain cases, constraint solving methods can be used effectively in protocol testing [CA90], [CZ93]. A test case derivation algorithm was given in [HB94] for a subset of LOTOS called P-LOTOS. The only data types in this subset are integers and Booleans, and the operations on integers are restricted to addition, subtraction and comparison. None of them considered timing or resource consumption.

Mandrioli *et al.* [MMM95] described how to generate functional test cases for real time systems given their temporal logic specifications. The test generation criteria are based on the structure of formulas in the specification. They used an interactive tool. In contrast, constraint solving tools are automatic and highly efficient. In this paper, we are mainly interested in the

resources consumed by the media objects. We believe that for distributed multimedia systems, it is more effective to focus on those system states where large amounts of resources are being used.

Resource Modeling and Load Testing

The traditional method of representing resource consumption in Petri nets is by way of the number of tokens [M89], [YSS94]. When a transition fires, n ($n \geq 1$) tokens may be deleted from or added to the related places, where the value of n depends on how much resource is needed. This is not adequate for multimedia applications since a presentation (which is associated with a place in the net) consists of several types of media objects, and different objects may consume different amounts of resources.

Avritzer and Weyuker [AW94] discussed load testing of software based on some operational profile. They used Markov chains, and their purpose is to examine the *most probable* system states more thoroughly. Yang and Pollock [YP96] described an algorithm for analyzing sequential programs to identify load sensitive parts. But no method was given to find input data to exercise these parts.

5. CONCLUDING REMARKS

In this paper, stress testing of distributed multimedia systems is studied. A multimedia application is specified by a temporal event Petri net (TEPN) together with some time-line diagrams (or the corresponding temporal formulas). We define execution traces of a specification assuming a discrete time model. From such a trace one can derive a test case of the application. When the media objects' resource consumption data are available, we may use some criteria to select the test cases which drive the application to stressful states.

The focus of the paper is on automatic generation of test cases for stress testing. Essentially our approach is to generate some symbolic execution traces first, and then use constraint satisfaction and optimization techniques to determine the timings of events. We give two methods for generating the constraints, which correspond to different test case selection criteria. One appealing feature of our approach is that the method is easy to automate. Another is that the generated test cases are executable. Preliminary experimental results are quite encouraging. Using a state-of-the-art constraint solver, we are able to determine the timings of user interactions

such that the test criteria are satisfied. On the examples we tried, the test case generation times are reasonable.

Of course, some assumptions are needed in our work, and our model is only an approximation of the real world. For instance, in the model, it is assumed that the firing of every transition takes the same amount of time (i.e., 1 time unit). In reality the transition times may be different from each other, depending on such factors as the distance between a client and a server. Moreover, we assume that the amount of data transfer is constant for a given media object. In practice the bit rates of audio and video objects may vary for different compression techniques. Nevertheless, the results of our analysis can serve as a guideline for test engineers. In most cases, audio and video streams consume much more resources than other media objects.

ACKNOWLEDGEMENTS

This work is partially supported by RGC grant HKUST6088/97E. The paper was written during the first author's visit to the Hong Kong University of Science and Technology, sponsored by the Croucher Foundation. He is also supported in part by the NSF of China.

REFERENCES

[AD94] R. Alur and D.L. Dill. A theory of timed automata. *Theoretical Computer Science* 126(2): 183-235, 1994.

[AH96] M. Archer and C. Heitmeyer. Mechanical verification of timed automata: a case study. *Proc. IEEE Real-time Technology and Applications Symp.*, 192-203, 1996.

[ABSS96] A.F. Ates *et al.* Using timed CSP for specification, verification and simulation of multimedia synchronization. *IEEE J. on Selected Areas in Comm.* 14(1): 126-137, 1996.

[AS96] A.F. Ates and B. Sarikaya. Test sequence generation and timed testing. *Computer Networks and ISDN Systems* 29(1): 107-131, 1996.

[AW94] A. Avritzer and E.J. Weyuker. Generating test suites for software load testing. *Proc. Int'l Symp. on Software Testing and Analysis* (*ISSTA*), 44-57, 1994.

[B84] B. Beizer. *Software System Testing and Quality Assurance*, Van Nostrand Reinhold, New York, 1984.

[BD91] B. Berthomieu and M. Diaz. Modeling and verification of time dependent systems using Time Petri Nets. *IEEE Trans. on Softw. Eng.* 17(3): 259-273, 1991.

[CZ93] S.T. Chanson and J. Zhu. A unified approach to protocol test sequence generation. *Proc. IEEE INFOCOM'93*, 106-114, 1993.

[CA90] W. Chun and P.D. Amer. Test case generation for protocols specified in Estelle. *Formal Description Techniques III* (*FORTE'90*), 191-206.

[GMMP91] C. Ghezzi *et al.* A unified high-level Petri net formalism for time-critical systems. *IEEE Trans. on Softw. Eng.* 17(2): 160-172, 1991.

[GMP94] C. Ghezzi, S. Morasca and M. Pezze. Validating timing requirements for time basic net specifications. *J. of Systems and Software* 27(2): 97-117, 1994.

[HB94] T. Higashino and G.v. Bochmann. Automatic analysis and test case derivation for a restricted class of LOTOS expressions with data parameters. *IEEE Trans. on Softw. Eng.* 20(1): 29-42, 1994.

[HG97] U. Horn and B. Girod. Scalable video transmission for the Internet. *Computer Networks and ISDN Systems* 29, 1833-1842, 1997.

[JM94] F. Jahanian and A.K. Mok. Modechart: a specification language for real-time systems. *IEEE Trans. on Software Engineering*, 20(12): 933-947, 1994.

[LPW96] K.G. Larsen, P. Pettersson and Wang Yi. Diagnostic model-checking for real-time systems. *Hybrid Systems III*, LNCS Vol. 1066, Springer, Berlin, 575-586, 1996.

[LG90] T.D.C. Little and A. Ghafoor. Synchronization and storage models for multimedia objects. *IEEE J. on Selected Areas in Comm.* 8(3): 413-427, 1990.

[LSGL95] V. Luchangco *et al.* Verifying timing properties of concurrent algorithms. *Formal Description Techniques VII (FORTE'94)*, 259-273.

[MMM95] D. Mandrioli *et al.* Generating test cases for real-time systems from logic specifications. *ACM Trans. on Computer Systems*, 13(4): 365-398, 1995.

[MCC98] V.B. Misic, S.T. Chanson and S.-C. Cheung. Towards a framework for testing distributed multimedia software systems. *Proc. Int'l Symp. on Software Engineering for Parallel and Distributed Systems*, Kyoto, Japan, 72-81, 1998.

[M89] T. Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4): 541-580, 1989.

[S93] B. Sarikaya. *Principles of Protocol Engineering and Conformance Testing.* Ellis Horwood, New York, NY, USA, 1993.

[SSW95] P. Sénac, P. de Saqui-Sannes and R. Willrich. Hierarchical time stream Petri net: a model for hypermedia systems. *Proc. 16th Int'l Conf. on Application and Theory of Petri Nets*, 451-470, 1995.

[SS94] J. Skakkebaek and N. Shankar. Towards a duration calculus proof assistant in PVS. *Proc. Int'l Symp. on Formal Tech. in Real-Time and Fault-Tolerant Syst.*, 660-679, 1994.

[W97] K.-K. Wong. *Distributed Multimedia Authoring Using Application Structures.* Master Thesis, Hong Kong Univ. of Sci. and Tech. 1997.

[WQG94] M. Woo, N.U. Qazi and A. Ghafoor. A synchronization framework for commu-nication of pre-orchestrated multimedia information. *IEEE Network* 8(1): 52-61, 1994.

[YP96] C.-S.D. Yang and L.L. Pollock. Towards a structural load testing tool. *Proc. Int'l Symp. on Software Testing and Analysis (ISSTA)*, 201-208, 1996.

[YMW97] J. Yang, A.K. Mok and F. Wang. Symbolic model checking for event-driven real-time systems. *ACM Trans. on Prog. Lang. and Syst.* 19(2): 386-412, 1997.

[YSS94] J.C. Yee *et al.* Resource synchronization specification and modeling based on timed stream Petri net. *Proc. of the IASTED/ISMM Int'l Conf. on Distributed Multimedia Systems and Applications*, 171-175, 1994.

[Zcc93] Zhou Chaochen. Duration calculi: an overview. *Proc. Formal Methods in Programming and Their Applications*, 256-266, 1993.

[Zh98] J. Zhou. *The user manual of NCL 1.0.* Loria and Inria-Lorraine, France, 1998.