21

Real-MFG: A Petri Net based model focusing on the integration of schedulability and fault tolerance

Vasilis C. Gerogiannis

Achilles D. Kameas

Panayotis E. Pintelas

Department of Mathematics, Sector of Computational Mathematics and Informatics, University of Patras and Educational Software Development Laboratory, Department of Mathematics, University of Patras
PO BOX 1399, University of Patras, 26500 Patras, HELLAS,
tel: +30-61-997833, +30-61-997313, fax: +30-61-992965, e-mail: {vasilis-kameas-pintelas}@math.upatras.gr

Abstract

Developing real-time software controlled systems for safety critical applications and verifying that they meet functional, fault tolerance and timing requirements are inherently complex processes. In order to meet a variety of (usually) conflicting requirements, it is necessary to follow an integrated approach from the early phases of design. In this paper, an integrated platform called Real-Time Multi Flow Graph (*Real-MFG*), is proposed, which provides encapsulation of both function- and priority- driven scheduling, incorporates constructs of powerful task graphs and integrates the Petri Net (*PN*) formalism with schedulability analysis methods. The model addresses fault tolerance requirements by including the promising imprecise computation paradigm in its semantics.

Keywords

Real-time systems, Petri Net modeling, schedulability analysis, fault-tolerance

1 INTRODUCTION

Petri Net (*PN*) theory (Gerogiannis, 1995 and Gerogiannis, 1996a are two recent surveys) is one of the most popular formal methods that graphically models concurrency, nondeterminism and causal relationships between events that occur in a system. The asynchronous nature inherent in *PNs* implies that there is no measure of time or of the flow of time. The introduction of time into the *PN* model is mainly related to the location and type of time delays, the (absolute or relative) time mode and the type of firing rule (strong or weak firing).

Design oriented schedulability analysis and systematic simulation with *PNs* (i.e., reachability analysis) can be used in conjuction: schedulability analysis provides rigorous results using an analytical approach and a worst-case configuration, while *PN* based simulation provides conclusions without guarantees; it depends on the designer to establish the worst-case scenario (initial marking) and then to determine if timing constraints will be met or not (Gerogiannis, 1996b). However, in many cases (e.g., when a system has access to multiple active resources each of which is

D. Gritzalis (ed.), *Reliability, Quality and Safety of Software-Intensive Systems*

scheduled independently), simulation provides real-time designers with a higher confidence level than schedulability analysis. Apart from the fact that almost all existing schedulability analysis tools/techniques are directly related to the specific phase in which they are involved (Gerogiannis, 1996c contains a more analytical justification), the application of different scheduling algorithms to different resources may lead to a complex behaviour that cannot be predicted through existing schedulability tests (Liu, 1993).

As far as fault tolerance is concerned, conventional techniques usually create temporal and/or spatial redundancy without taking into account any schedulability requirements (Stankovic, 1988). A recently proposed paradigm which tries to manage additional redundancy, while preventing timing faults by decreasing (in an acceptable way) the accuracy of the results, is the imprecise computation model (Liu, 1990). According to this paradigm, a task is composed of two parts: a mandatory part followed by an optional part; the first must complete before the corresponding deadline, while the latter simply refines the accuracy of the result produced by the mandatory part.

This paper focuses on the integration of formal design methods based on high-level timed *PNs* with schedulability analysis techniques. The semantics of a high-level timed *PN* called Real-Time Multi Flow Graph (*Real-MFG*) are given (a first model description had been given in (Gerogiannis, 1996c)). *Real-MFG* extends *IMFG*, a high-level *PN* that can be used for specifying user-system interaction (Kameas, 1995), and addresses fault tolerance requirements by including the promising imprecise computation paradigm in its semantics.

2 REAL-TIME MULTI FLOW GRAPH

Real-MFG is a process-based synchronisation model defined as a high-level Timed *PN*. A real-time application is regarded as a *Real-MFG* that consists of active components called actors (transitions) and passive components called links (places), which accomodate abstract data or control structures (tokens).

Tokens indicate the availability of resources or information, as well as the status of a *Real-MFG* at different phases of its execution. Tokens may indicate also occurrences of certain events which mark the enabling or the completion time of an actor. They can be of several classes; associated with each class there is a corresponding set of arcs. This facilitates tracing the flow of a particular typed token. An actor models a response that must be performed as a consequence of the occurrence of an event (e.g., a resource is available, a timer has expired, an interrupt has occurred). Responses are implemented as tasks and can be decomposed into smaller segments of called actions. Finally, links can model the storage of tokens, their flow among actors, the system conditions, the different events which can occur in the system, the interrelationships between actors, and the type of system resources.

2.1 Actors

An actor is described by:

1. Its *name* and a set of *input* and *output* links that make up its interface part.
2. A set of *rules* that make up its behavioural part (the left-hand side of each rule forms a precondition on the actor input links, while the right-hand side part forms a postcondition on its output links).
3. Its *type* that describes the semantics of its firing. The following actor types are provided:

- *Context*: they represent responses that lead to the achievement of a system goal. They have an internal structure that represents how a response is decomposed into action actors to accomplishing a system goal.
- *Action*: an action actor corresponds to a direct representation of a single action of a response.
- *Library*: they are specific system-provided actors that describe the exact way responses are decomposed into actions in order to accomplish a system goal. Library Actors include Chain,

Loop, OR, Select, AND, Parallel-Min and Parallel-Max, Mandatory-Optional actors (characteristics met in AND/OR and imprecise workload models (Gillies, 1990; Liu, 1990) and in data-flow graphs (Papadimitriou, 1992)).

4. A set of *functional parameters* which describe the actor functionality. These include *weight* or *priority* which denotes the importance of actor firing (priority-driven scheduling (Lehoczky, 1991), *importance function* which represents the importance of an actor over time and used to specify multi-level scheduling requirements (function-driven scheduling (Strayer, 1992), *computation function* that represents the internal actor functionality, *preemptability* (the firing of an preemptable actor may be suspended and latter resumed), *laxity type* and *laxity function* which indicate the relative cost of exceeding an actor's deadline but still completing firing (Liu, 1990; Liu, 1993), etc.
5. A set of *temporal parameters* which define the timing requirements associated with the actor (Liu, 1990; Tsai, 1995). These include *minimum timing constraint* (or *ready time*) and *maximum timing constraint* (or *deadline*), *maximum execution time* (or *maximum firing duration*), *mandatory and optional execution parts, average execution time* (or *average firing duration*), *period* (in case of output *actors* of a periodic event link), *minimum separation* (in case of output actors of a *non periodic* event link), *phase*, etc.

The description of a real-time application usually starts with a set of first-level context actors, which represent the basic structure of the application tasks. These are decomposed into action actors, which are used to represent in detail the real-time tasks that constitute the application.

2.2 Links

A link is described by:

1. Its *name* and a set of *input* and *output* actors that make up its interface part. These actors provide, consume, destroy or somehow use the tokens it stores.
2. Its *type* that makes up its behavioural part, since it explicitly defines the link role. The following link types are supported: *Context links,* which represent the context of real-time responses/actions, *event links,* which represent abstractions of application events, *resource links,* which represent abstractions of the various objects available for use by the application responses/actions, *condition links*, which represent global system conditions (system condition links) and possible interrelationships between actors (compound condition links), and *data links*, which model the data flow (the input and output data of a computation).
3. A set of *temporal parameters* which define the link timing behaviour and can be used to conduct analysis based on relative time mode. This set includes *maximum (minimum) timing constraint* and *arrival pattern* (only for event links), *maximum (minimum) (de-) acquisition time* and *context switch time* (only for resource links) (Liu, 1993), etc.

A resource link is described by the following parameters: *access type (exclusive or shared), resource type* (*Read only, Normal*) and *preemptability.* Event links are categorized according to their origin to (Klein, 1993): *Environmental, Internal* and *Timed.* In addition, event links are described by their arrival pattern and categorized to *periodic, irregular, bounded, bursty* and *unbounded.*

It should be noted that not all *Real-MFG* semantics are mandatory; the model formalism incorporates concepts from several paradigms, but it depends on the designer to select which attributes are applicable to his/her application.

2.3 State transition

Real-MFG uses the *PN* enabling and firing rules to represent state transition, modified as follows:

- An actor is said to be *active* if its input context link contains a token, while an active actor is said to be *enabled* if each of its input links has at least one token.
- An enabled actor is *firable* during the time interval from its *ready time* to its *deadline*, and a firable actor may fire only if its maximum firing duration is less than or equal to the time interval between its deadline and its ready time.
- In general, the firing of a firable actor will consume all tokens from its input links and will fill all its output links with tokens.

A state transition can take place only as a consequence of an event (that is, the appearance of a token on an event link) and is represented as an actor firing. An actor fires according to the *weak firing mode* which does not force any enabled actor to fire. This mode both preserves the same firing mode used by *PN* and can be used to model and analyze conflict structures. In addition, a *Real-MFG* can be analyzed based on *absolute* or *relative time modes*, since timing parameters can be associated with either the actors or links. The model maintains an internal structure with all the actors that are active at any moment (the *actor ready list*). This list forms a representation of a system state, and is modified by any event that causes a state transition, since new actors are added to the list, while others are removed. Therefore, *Real-MFG* constitutes an extension with memory of *PN* models.

3 CONCLUSIONS

Real-MFG incorporates advantageous characteristics of various *PN* timed versions, powerful task graphs and features found in real-time paradigms (e.g., relative/absolute time mode, typed enabling/firing rules, weak firing mode, average/fixed time intervals). The model supports multi-level scheduling, priority and function-driven design (by using the weight and importance function concept), incorporates fault tolerance semantics (mandatory and optional actor parts), and allows the designer to specify the application characteristics at any level of detail by using primitives of powerful task graphs. In comparison to implementation-oriented schedulability analysis techniques, *Real-MFG* is advantageous in being completely language independent. Other design-oriented approaches lack formal verification, systematic simulation and visualization support, characteristics inherent in the underlying model formalism (i.e., *PN* based specification) .

Real-MFG provides several design perspectives, such as goal-subgoal structure, task analysis, causation, event flow, data flow, conditions, etc. Analysis can be performed by translating the model to an *ITCPN* (Van der Aalst, 1993) (or to a *TCPN* (Tsai, 1995) and then following reachability analysis. Moreover, *Real-MFG* can be used to represent and analyze complex applications which may include dependent tasks and aperiodic events. In that case, the designer is able to transform the model to a periodic task set in order to apply a schedulability analysis technique, which will determine the feasibility of the timing requirements.

4 REFERENCES

Gerogiannis, V., Diplas, C., Kameas, A. and Pintelas, P. (1995) Petri Nets and High-Level Petri Nets: Formalism, Properties, Analysis and Applications. *Technical Report TR 95-01*, Dept. of Mathematics, Univ. of Patras.

Gerogiannis, V., Kameas, A. and Pintelas, P. (1996a) Comparative Study and Categorization High-Level Petri Nets. *Technical Report TR 96-01*, Dept. of Mathematics, Univ. of Patras.

Gerogiannis, V., Kameas, A. and Pintelas, P. (1996b) On the Integration of High-Level Timed Petri Nets With Schedulability Analysis Methods, in *Proceedings of HERMIS '96 Conf.*, Athens,.

Gerogiannis, V. and Tsoukarellas, M. (1996c) Using SCAN to Analyze the Schedulability of a Real-Time Application, in *Embedded Microprocessing Systems*, (eds. C. Muller Schloer, F. Geerinckx, B. Stanford-Smith and R. van Riet), IOS Press, 344-353.

Gillies, D.W. and Liu, J.W.S. (1990) Scheduling Tasks with AND/OR Precedence Constraints, in *Proceedings of the 2nd IEEE Conference on Parallel and Distributed Processing*, Dallas.

Kameas, A. (1995) A Formal Model for the Specification of Interaction and the Design of Interactive Applications. *PhD thesis*, Dept. of Comp. Engineering & Informatics, Univ. of Patras.

Klein, M.H., Ralya, T., Polak, B., Obenza, R., and Harbour, M.G. (1993) *A Practitioner's Handbook for Real-Time Analysis*. SEI, Carnegie Mellon Univ., Kluwer Academic Publ..

Lehoczky, J.P., Sha, L., Strosnider, J.K. and Tokuda, H. (1991) Fixed Priority Scheduling Theory for Hard Real-Time Systems, in *Foundations of Real-Time Computing: Scheduling and Resource Management*, (eds. A.M. van Tilborg and G.M. Koob), Kluwer Academic Publ., 1-30.

Liu J.W.S. et al. (1990) Scheduling Periodic Jobs That Allow Imprecise Results. *IEEE Transactions on Computers*, **39(9)**, 1156-1174.

Liu J.W.S. et al. (1993) PERTS: A Prototyping Environment for Real-Time Systems. *Technical Report, UIUCDCS-R-93-1802*, Dept. of Computer Science, Univ. of Illinois.

Papadimitriou, S., Kameas, A., Fitsilis, P. and Pavlides, G. (1992) A new compression technique for tools that use data-flow graphs to model distributed real-time applications, in *Proceedings of 5th Int'l Conf. on Software Engineering & its Applications*, Toulouse, France, 235-244.

Stankovic, J.A. (1988) Real Time Computing Systems: The Next Generation, in *Hard Real-Time Systems Tutorial*, (eds. J. A. Stankovic and K. Ramamritham), IEEE Press, 14-37.

Strayer, W.T. (1992) Function-Driven Scheduling: a General Framework for Expression and Analysis of Scheduling. *PhD thesis*, School of Engineering and Applied Science, Univ. of Virginia.

Tsai, J., Yang, S. and Chang, Y. (1995) Timing Constraint Petri Nets and Their Application to Schedulability Analysis of Real-Time Systems Specification. *IEEE Transactions on Software Engineering*, **21(1)**, 32-49.

Van Der Aalst, W.M.P. (1993) Interval Timed Coloured Petri Nets and their Analysis in *Lecture Notes in Computer Science vol. 691*, (ed. M. A. Marsan), Springer-Verlag, 453-472.

Zhu J. and Lewis T. (1995) Scheduling in Hard Real-Time Applications. *IEEE Software*, **12(3)**, 54-63.

5 BIOGRAPHY

Vasilis C. Gerogiannis received his Diploma in Computer Engineering from the Dept. of Computer Engineering & Informatics at the Univ. of Patras, Hellas, in 1992. Since 1993, he is a postgraduate student with the Dept. of Mathematics at the Univ. of Patras, pursuing a PhD degree in the area of formal design verification techniques for real-time systems. He has participated in the ESPRIT projects OMI/CLEAR, OMI/TOOLS and OMI/ANTI-CRASH. He is a member of the Technical Chamber of Greece since 1993. He joined the Educational Software Development Laboratory (ESD*Lab) in 1993.

Dr. Achilles D. Kameas received his Diploma in Computer Engineering and his PhD in Human-Computer Interaction from the Dept. of Computer Engineering & Informatics at the Univ. of Patras, in 1989 and 1995, respectively. His research interests also include Human Cognition and User modelling, Virtual Reality and Artificial Intelligence, Authoring and Multimedia Systems, and Intelligent Tutoring Systems. He has contributed to a number of publications in those fields, as a result of the ongoing research he participates in. His research experience includes participation in several national and EU research projects, such as GESEM (in the context of EU program COMETT II) and AXE-10 (in the context of national program SYN). He was a teaching assistant with the Dept. of Computer Engineering & Informatics at the Univ. of Patras (1990-1995) and a part-time professor with the Technological Educational Institute of Patras and Messolongi (1995). He is a Voting Member of ACM, SIGCHI and SIGCUE since 1992, a member of IEEE and IEEE Computer Society since 1993, a member of the Society for Machines and Mentality since 1993 and a member of the Technical Chamber of Greece since 1990. He joined the ESD*Lab in 1992.

Dr. Panayotis E. Pintelas received his BSc in Mathematics from the Univ. of Athens, Hellas, in 1971, and his MSc and PhD in Computer Science from the Univ. of Bradford, UK, in 1973 and 1976, respectively. Since 1991, he is an Associate Professor with the Dept. of Mathematics, Univ. of Patras. He was elected Head of the Division of Applied Mathematics and Informatics from 1991 to 1993. In the recent past, he was with the Dept. of Computer Engineering and Informatics, Univ. of Patras, as a Visiting Scientist (1981-1984), Lecturer (1984-1987) and Assistant Professor (1987-1991). His current research interests include Software Specification and Software Design Languages, Project Management, Computer Assisted Training and Education and Software Engineering; in these fields he has numerous publications in international scientific journals and conferences. He is also the author of two textbooks which are included in the curriculum of the Dept. of Computer Engineering and Informatics. He has participated in several National and EU research projects, including COMETT project SEMI (partner), COMETT II project GESEM (project co-ordinator), ESPRIT project DAISY (consultant), SYN (national) project AXE-10 (project co-ordinator). He is a member of the British Computer Society and of the British Association of Chartered Engineers, and a member of the Association for the Development of Computer-Based Instructional Systems. He is in charge of the ESD*Lab, the establishment of which he proposed in 1992.