FEASIBLE DYNAMIC RECONFIGURATIONS OF PETRI NETS
Application to a Production System

Mohamed Khalgui[1,2], Olfa Mosbahi[1,2]
[1]*ICTICA, Ariana, Tunisia*

Jiafeng Zhang[2], Zhiwu Li[2]
[2] *Xidian University, Xi'an, Shaanxi, China*

Atef Gharbi
University of Carthage, Tunis, Tunisia

Keywords: Petri Net, Control System, Reconfiguration, Model Checking, Computation Tree Logic.

Abstract: The paper deals with dynamic automatic reconfigurations of Control Systems to be classically modelled by Petri nets. Three different forms can be applied at run-time to reconfigure such systems: Addition/Removal of places, Addition/Removal/Update of transitions or finally the simple change of the initial marking. We define three formal modules allowing reconfigurations of the system's Petri nets: *changer_places* to dynamically change places of the model, *changer_transitions* to dynamically reconfigure transitions, and *changer_marking* to modify the initial markings of places. To guarantee a correct behavior of this architecture according to user requirements, we apply a model checking by using the useful tool SESA for the verification of CTL-based properties of the proposed modules and also of the system. The paper is applied to a Real Benchmark Production System.

1 INTRODUCTION

The new generation of control systems is addressing new criteria as flexibility and agility. To reduce their cost, these systems should be changed and adapted to their environment without disturbances. Several interesting academic and industrial research works have been made in recent years to develop reconfigurable control systems (Gehin and Staroswiecki, 2008). We distinguish in these works two reconfiguration policies: static and dynamic reconfigurations such that static reconfigurations are applied off-line to apply changes before the system's cold start (Angelov et al., 2005), whereas dynamic reconfigurations are dynamically applied at run-time. Two cases exist in the last policy: manual reconfigurations applied by users (Rooker et al., 2007) and automatic reconfigurations applied by Intelligent Agents (Al-Safi and Vyatkin, 2007). We are interested in this paper in automatic reconfigurations of control systems that we model by the formal formalism Net Condition/Event Systems (NCES) which is an extension of Petri nets (Rausch and Hanisch, 1995). In NCES, places classically correspond to control actions to be done by the control system. To move from a place to another, a transition should be fired. There are several conditions to be fulfilled to enable a transition to fire. First of all, all pre-places have to be marked with at least one token. In addition, it may have incoming condition arcs from places and event arcs from other transitions. A transition is enabled by condition signals if all source places of the condition signals are marked by at least one token. The other type of influence on the firing can be described by event signals which come to the transition from some other transitions. We mean in this research paper by an automatic reconfiguration any addition-removal of places, transitions, condition signals or event signals to/from the NCES specifying the control system, and any change in the initial marking. To handle automatic reconfigurations, we define three NCES-based modules such that the first allows the addition-removal of places in/from the system's NCES, the second allows addition-removal of transitions, event signals, condition signals in/from the system, and finally the third allows modifications of the initial marking. To guarantee a correct be-

Khalgui M., Mosbahi O., Zhang J., Li Z. and Gharbi A..
FEASIBLE DYNAMIC RECONFIGURATIONS OF PETRI NETS - Application to a Production System.
DOI: 10.5220/0003448901050110
In *Proceedings of the 6th International Conference on Software and Database Technologies* (ICSOFT-2011), pages 105-110
ISBN: 978-989-8425-77-5

havior of the whole system after any reconfiguration scenario, we apply a model checking by using the tool SESA that allows verifications of properties of system's NCES and also of the proposed modules (Rausch and Hanisch, 1995). We use the temporal logic "'Computation Tree Logic" (denoted by CTL) to specify these properties (Roch, 2000a; Roch, 2000b). The paper is applied to a Benchmark Production System EnAS available in the Research Laboratory of Prof.Dr. Hans-Michael Hanisch at Martin Luther University in Germany. We describe this system in (Mohamed Khalgui, 2008).

We present a quick state of the art on model checkers in the next section, before specify reconfigurable control systems in Section 3, and propose the reconfiguration modules in Section 4. We apply in Section 5 the model checking of the whole architecture, and conclude the paper in Section 6.

2 STATE OF THE ART: MODEL CHECKERS

Finite state machines (abbr. FSM) are widely used for the modelling of control flow in embedded systems and are amenable to formal analysis like model checking (Clarke et al., 2000; Clarke and Kurshan, 1996; Holzmann, 1997; Vardi and Wolper, 1994; Ma and Tsai, 2008). Two kinds of computational tools have been developed last years for model checking: tools like KRONOS (Daws et al., 1996), UPPAAL (Amnell et al., 2001), HyTech (Henzinger et al., 1997) and SESA (SESA, 2008) which compute sets of reachable states exactly and effectively, whereas emerging tools like CHECKMATE (Chutinan and Krogh, 1999), d/dt (Asarin et al., 2000) and level-sets (Mitchell and Tomlin, 2000) methods approximate sets of reachable states. Several research works have been proposed in recent years to control the verification complexity by applying hierarchical model checking for complex embedded systems. The authors propose in (Alur and Yannakakis, 1998) an approach for verifications of hierarchical (i.e. nested) finite state machines whose states themselves can be other machines. The straightforward way to analyze a hierarchical machine is to flatten and apply a model checking tool on the resulting ordinary FSM, but the authors show in this interesting research work that this flattening can be avoided by developing useful algorithms for verifications of hierarchical machines.

3 SPECIFICATION OF RECONFIGURABLE CONTROL SYSTEMS

Reconfiguration means qualitative changes in structures, functionalities, and algorithms of control systems as responses to qualitative changes of goals of controls, of controlled systems, or of environments the systems behaves within. This could be caused by (partial) failures, breakdowns, or even by human interventions. Let us denote by *Sys* the reconfigurable control system to be modelled by Net Condition/Event Systems $\Sigma(Sys)$ that specify all possible behaviors of the system to be applied after well-defined reconfigurations.

$$\Sigma(Sys) = \{PTN,\ CN,\ WCN,\ I,\ WI,\ EN,\ em\}$$

Where,

$$PTN = \{P_{\Sigma(Sys)}, T_{\Sigma(Sys)}, F_{\Sigma(Sys)}, W_{\Sigma(Sys)}\}$$

We mean by a reconfiguration scenario of $\Sigma(Sys)$ (i) any addition/removal of places, (ii) any addition/removal of transitions, (iii) any addition/removal of condition-event arcs, (iv) any update of marking. The system can be specified by different sub-NCES defining different possible behaviors to be followed under well-defined conditions. Let $\xi(Sys)$ be a sub-NCES that models *Sys* after a well-defined automatic reconfiguration scenario.

$$\xi(Sys) = \{PTN_{\xi(Sys)}, CN_{\xi(Sys)}, WCN, I, WI, EN_{\xi(Sys)}, em\}$$

Where,

$$PTN_{\xi(Sys)} = \{P_{\xi(Sys)}, T_{\xi(Sys)}, F_{\xi(Sys)}, W_{\xi(Sys)}\}$$

Such that,

- $P_{\xi(Sys)} \subseteq P_{\Sigma(Sys)}$,
- $T_{\xi(Sys)} \subseteq T_{\Sigma(Sys)}$,
- $F_{\xi(Sys)} \subseteq F_{\Sigma(Sys)}$.

If $\xi(Sys)$ specifies the system when a particular reconfiguration scenario is applied, the places of $P_{\xi(Sys)}$ (resp, transitions of $T_{\xi(Sys)}$ and arcs of $F_{\xi(Sys)}$) become the only able places of $P_{\Sigma(Sys)}$ to be activated (resp, only able transitions of $T_{\Sigma(Sys)}$ and able arcs of $F_{\Sigma(Sys)}$). The rest of places, transitions and arcs become disable.

Running Example.

In the Benchmark Production System EnAS, only four sub-NCES are possible to specify its behavior when well-defined reconfiguration scenarios are automatically applied at run-time (Figure 1).

- Let $\xi_1(Sys)$ be the first sub-NCES that specifies EnAS when the Second Production Policy is applied such that (EnAS model1 in Figure 1):

$$P_{\xi_1(Sys)} = \{PS1, PS2, PS3, PS4, PS5, PS6, PS9\}$$

The place $PS1$ corresponds to the displacement of an empty tin on the belt to the first Jack station where a piece is put (e.g. the place $PS2$). The tin is displaced thereafter (e.g. place $PS3$) to the second Jack station where a second piece is put before it is closed with a cup (e.g. place $PS4$). The closed tin is displaced thereafter on the belt (e.g. place $PS5$) to the second Gripper station $G2$ for an evacuation to the second storing station $St2$. We note finally that the place $PS9$ defines the number of pieces (e.g. two pieces) to be put in the tin when the Second Production Policy is applied.

- Let $\xi_2(Sys)$ be the second sub-NCES that specifies EnAS when the First Production Policy is applied such that (EnAS model2 in Figure 1):

$$P_{\xi_2(Sys)} = \{PS1, PS2, PS7, PS8, PS10\}$$

The place $PS7$ corresponds to the displacement of a tin containing a piece and closed with a cup from the first Jack station to the first Gripper station (e.g. place $PS8$). We note finally that the place $PS10$ defines the number of pieces (e.g. one piece) to be put in the tin when the First Production Policy is applied.

- Let $\xi_3(Sys)$ be the third sub-NCES that specifies EnAS when the second Jack station is broken such that (EnAS model3 in Figure 1):

$$P_{\xi_3(Sys)} = \{PS1, PS2, PS6, PS10\}$$

The place $PS2$ corresponds to the placement of a piece in a tin to be closed with a cup in the first Jack station. The place $PS6$ corresponds to the removal from the belt to the second Storing Station $St2$.

- Let $\xi_4(Sys)$ be the fourth sub-NCES that specifies EnAS when the first Jack station is broken such that (EnAS model4 in Figure 1):

$$P_{\xi_4(Sys)} = \{PS1, PS3, PS4, PS5, PS6, PS10\}$$

The places $PS1$ and $PS3$ correspond to the displacement of an empty tin on the belt to the second Jack station where a piece and a cup are put (e.g. the place $PS4$). The closed tin is displaced thereafter on the belt (e.g. place $PS5$) to the second Gripper station $G2$ for an evacuation to the second storing station $St2$ (e.g. $PS6$). We note finally that the place $PS10$ defines the number of pieces (e.g. only one piece) to be put in the tin when the first Jack station is broken.

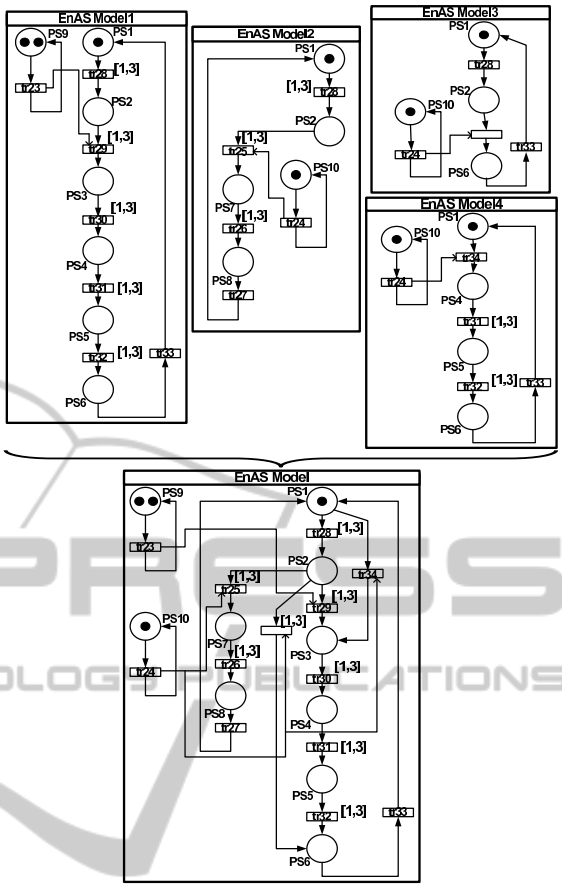

Figure 1: Specification of the Reconfigurable EnAS with NCES.

4 RECONFIGURATION OF NET CONDITION/EVENT SYSTEMS

To dynamically reconfigure the NCES $\Sigma(Sys)$, we define nested state machines where states correspond to other state machines. Each state machine forms a module allowing reconfigurations of the system. Three types of modules are distinguished in this paper: the first module called *changer_places* is modelled by a NECS to be denoted by CP in which each place $p = reconfigure(\xi(Sys))$ corresponds to a subset $P_{\xi(Sys)} \subseteq P_{\Sigma(Sys)}$. Therefore each transition in this state machine corresponds to the addition-removal of places in/from the system's specification. ***For each place*** p of CP, we define a particular module called *changer_transitions* and modelled by NCES to be denoted by CT ($CT = transition(p)$) in which each place corresponds to a particular composition of places in the system's specification $\xi(Sys)$. Each transition corresponds therefore to the addition or removal of transitions, event or condition arcs in $\xi(Sys)$ (p =

reconfigure($\xi(Sys)$)). We define finally a third particular type of modules called *changer_marking* modelled by a NCES to be denoted by *CM* in which each place corresponds to a particular marking of $\Sigma(Sys)$. A place of *CM* corresponds to one or more places of a module *changer_transitions* or the whole module *changer_places*.

$$CP = \{PTN_{CP}, CN_{CP}, WCN_{CP}, I_{CP}, WI_{CP}, EN_{CP}, em_{CP}\}$$

$$PTN_{CP} = \{P_{CP}, T_{CP}, F_{CP}, W_{CP}\}$$

$$CT = \{PTN_{CT}, CN_{CT}, WCN_{CT}, I_{CT}, WI_{CT}, EN_{CT}, em_{CT}\}$$

$$PTN_{CT} = \{P_{CT}, T_{CT}, F_{CT}, W_{CT}\}$$

$$CM = \{PTN_{CM}, CN_{CM}, WCN_{CM}, I_{CM}, WI_{CM}, EN_{CM}, em_{CM}\}$$

$$PTN_{CM} = \{P_{CM}, T_{CM}, F_{CM}, W_{CM}\}$$

We denote by $\Delta(CT)$ (resp. $\Delta(CM)$) the set of *CT* (resp. *CM*) modules. The whole control system is characterized by different behaviors such that each one should be executed after a well-defined reconfiguration scenario. Each scenario to be denoted by (p,q,k) ($p \in P_{CP}$, $q \in P_{CT} = transition(p)$ such that $CT \in \Delta(CT)$, and $k \in P_{CM}$ such that $CM \in \Delta(CM)$) is executed when the corresponding place p is active in *CP*, the place q is active in *CT* and finally the place k is active in the module *CM*. We denote by $Behavior_{p,q,k}(Sys)$ the sub-NCES of $\Sigma(Sys)$ that can implement *Sys* when the reconfiguration scenario (p,q,k) should be automatically applied. We synchronize the modules *CP*, *CT* and *CM* by event signals as follows: For each scenario (p,q,k),

- $\forall t1 \in^{\bullet} p$ and $t2 \in^{\bullet} q, \exists ev1 \in (t1,t2)$,
- $\forall t2 \in^{\bullet} q$ and $t3 \in^{\bullet} k, \exists ev2 \in (t2,t3)$.

We synchronize in addition the reconfiguration modules and the specification $\Sigma(Sys)$ of the system *Sys* by event signals as follows: For each scenario (p,q,k) such that $Behavior_{p,q,k}(Sys) = \xi(Sys)$,

- $\forall t1 \in^{\bullet} q$, $\exists t2 \in T_{\xi(Sys)}$ such that $\exists ev1 = (t1,t2)$,
- $\forall t3 \in^{\bullet} k$, $\exists t4 \in T_{\xi(Sys)}$ such that $\exists ev2 = (t3,t4)$.

The events $ev1$ and $ev2$ allow applications of reconfiguration scenarios to activate places and/or transitions and/or arcs and/or to change marking in the NCES $\xi(Sys) \in \Sigma(Sys)$.

Running Example.

According to Figure 2, the module *Changer_places* $CP1$ is composed of two places $P1$ and $P2$ that respectively define the Second and the First Production Policy. The transitions $tr1$ and $tr2$ define in this case the addition and removal

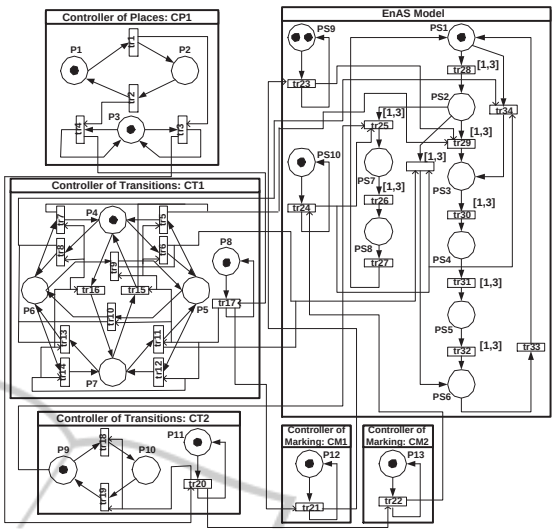

Figure 2: NCES-based Modules for Automatic Reconfigurations of EnAS.

of places in the system's specification. When the transition $tr1$ is fired, we disable the places $PS3$, $PS4$, $PS5$, $PS6$ and $PS9$, and we activate the places $PS7$, $PS8$ and $PS10$. We associate for the place $P1$ the NCES $CT1$ and for the place $P2$ the NCES $CT2$. The place $P4$ of the module $CT1$ corresponds to the execution of the second production policy when $PS1, PS2, PS3, PS4, PS5, PS6, PS9$ are specifying EnAS. The place $P5$ specifies the system when the second Jack station is broken. The place $P6$ corresponds to any problem in the first Jack station. The place $P7$ is reached when the first and the second Jack stations are broken. The place $P9$ of the module $CT2$ defines an execution scenario of EnAS when the first Jack and Gripper stations are used to produce pieces. We note in addition that the places $P12$ is active from the module $CT1$ when we put two pieces in the tin, whereas the place $P13$ is active when only one piece is put in the tin (e.g. it is activated by $CT2$).

5 MODEL CHECKING OF RECONFIGURABLE PETRI-NETS

Once the reconfigurable Petri nets are well-modelled, the next step to be addressed is their verification in order to guarantee a correct behavior of the system after any reconfiguration scenario. We use in this research work the model checker SESA to verify CTL-based properties defined in user requirements. This tool allows the verification of any reactions of reconfiguration modules as well as their synchronization with the

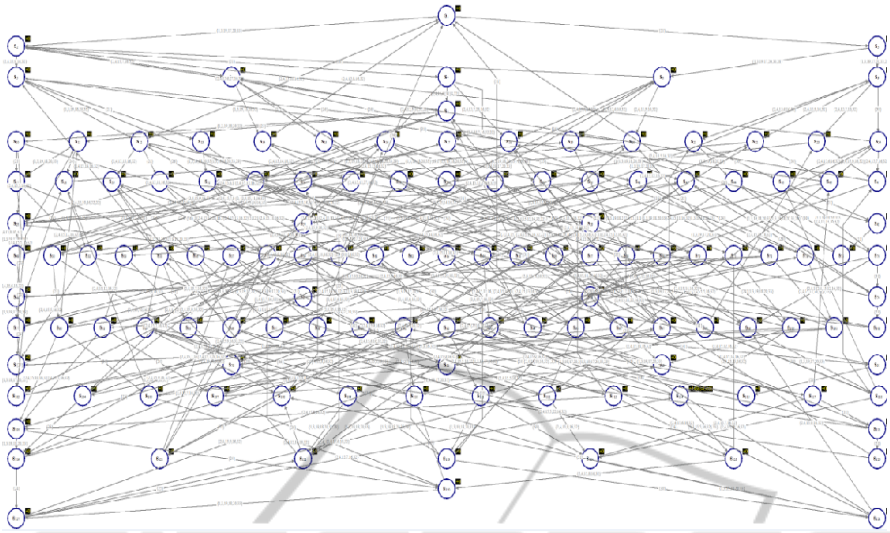

Figure 3: Reachability Graph of the Reconfigurable Architecture.

system's NCES that should be checked too. We show in Figure 3 a reachability graph generated by SESA for the verification of NCES depicted in Figure 2.

Running Example.

In the system EnAS, we check functional properties of the state machines that encode the agent and the system's NCES. We have to check in particular that whenever the transition $tr1$ is fired, then the place $PS7$ should be reached:

$$AGAtr1XPS7$$

This formula is proven to be True by applying this tool. Indeed, when conditions are satisfied to apply the Second Production Policy, the state PS7 should be reached. We have also to check that whenever the transition $tr5$ is fired to apply the second policy, the place $PS5$ should be applied to bring the tin from the first and second Jack stations to the second Gripper station:

$$AGAtr5XPS5$$

This formula is proven to be True. We check also the correct behavior of the system EnAS when the Second Production Policy is applied by verifying the following formula:

$$AGAtr29XAFEtr30XAFEtr31XAFEtr32XTRUE$$

Indeed, whenever the belt is activated to transport a piece to the first Jack station, it is activated again to transport the piece to the second Jack station before reaching the second Gripper station. This formula is proven to be True by SESA. When the Second Production Policy is applied, we check also if the evacuation of a closed tin from the belt can be done in 4 time units. The following formula is proven to be False by SESA:

$$EF[3,4]PS6$$

The following formula is proven to be True:

$$AF[5,6]PS6$$

Indeed the state $PS6$ (e.g. evacuation from the belt) should be reached 5 time units at least after the activation of the place $PS1$.

6 CONCLUSIONS

The paper deals with automatic reconfigurations to dynamically change the behaviors of control systems: it is a New Challenge in Industry. We specify this behavior by Net Condition/Event Systems which is an extension of Petri nets. A reconfiguration scenario is any addition-removal-update of places, transitions, event signals, condition signals, or just the modification of the initial marking. We define formal modules allowing reconfigurations of NCES, where the first module deals with places, the second with transitions and the third with the marking. We apply a model checking for the verification of CTL-based properties in order to guarantee a safe bevahior of this reconfigurable architecture. In future works, we plan the NCES-based modelling and CTL-based verification of communication protocols that allow safe coordinations inside distributed control systems.

ACKNOWLEDGEMENTS

This work was supported in part by the Natural Science Foundation of China under Grant 60773001, the

Fundamental Research Funds for the Central Universities under Grant No. 72103326, the National Research Foundation for the Doctoral Program of Higher Education, the Ministry of Education, P. R. China, under Grant No. 20090203110009, "863" High-tech Research and Development Program of China under Grant No 2008AA04Z109, the Research Fellowship for International Young Scientists, National Natural Science Foundation of China, and Alexander von Humboldt Foundation.

REFERENCES

Al-Safi, Y. and Vyatkin, V. (2007). An ontology-based reconfiguration agent for intelligent mechatronic systems. In *Third International Conference on Industrial Applications of Holonic and Multi-Agent Systems*. Springer-Verlag.

Alur, R. and Yannakakis, M. (1998). Model checking of hierarchical state machines. In *Sixth ACM Symposium on the Foundations of Software Engineering, pp. 175-188*.

Amnell, T., Behrmann, G., Bengtsson, J., D'Argenio, P. R., David, A., Fehnker, A., Hune, T., Jeannet, B., Larsen, K. G., Mller, M. O., Pettersson, P., Weise, C., and Yi, W. (2001). *Uppaal - Now, Next, and Future*. In Proceedings of Modelling and Verification of Parallel Processes (MOVEP'2k), France. LNCS Tutorial 2067, pages 100-125, F. Cassez, C. Jard, B. Rozoy, and M. Ryan (Eds.).

Angelov, C., Sierszecki, K., and Marian, N. (2005). Design models for reusable and reconfigurable state machines. In *L.T. Yang and All (Eds): EUC 2005, LNCS 3824, pp:152-163. International Federation for Information Processing.*

Asarin, E., Bournez, O., Dang, T., and Maler, O. (2000). Approximate reachability analysis of piecewise-linear dynamical systems. In *Hybrid Systems: Computation and Control, Third International Workshop, LNCS*.

Chutinan, A. and Krogh, B. K. (1999). Verification of polyhedral-invariant hybrid automata using polygonal flow pipe approximations. In *Hybrid Systems: Computation and Control, Second International Workshop, LNCS*.

Clarke, E., Grumberg, O., and Peled, D. (2000). Model checking. In *MIT Press*.

Clarke, E. and Kurshan, R. (1996). Computer-aided verification. In *IEEE Spectrum, 33(6)*.

Daws, C., Olivero, A., Tripakis, S., and Yovine, S. (1996). The tool kronos. In *Hybrid Systems III, Verification and Control, LNCS 1066, Springer-Verlag*.

Gehin, A.-L. and Staroswiecki, M. (2008). Reconfiguration analysis using generic component models. In *IEEE Transactions on Systems, Machine and Cybernetics, Vol.38, N.3*.

Henzinger, T. A., Ho, P., and Womg-Toi, H. (1997). Hytech: the next generation. In *TACAS95: Tools and Algorithms for the Construction and Analysis of Systems, LNCS*.

Holzmann, G. (1997). The model checker spin. In *IEEE Transactions on Software Engineering, 23(5)*.

Ma, L. and Tsai, J. (2008). Formal modeling and analysis of a secure mobile-agent system. In *IEEE Transactions on Systems, Machine and Cybernetics, Vol.38, N.1*.

Mitchell, I. and Tomlin, C. (2000). Level set methods for computation in hybrid systems. In *Hybrid Systems: Computation and Control, Third International Workshop, LNCS*.

Mohamed Khalgui, Martin Hirsch, D. M. H.-M. H. (2008). Reconfiguration of embedded systems. In *International Conference on Informatics in Control, Automation and Robotics ICINCO-ICSO, pages: 157-162*.

Rausch, M. and Hanisch, H.-M. (1995). Net condition/event systems with multiple condition outputs. In *Symposium on Emerging Technologies and Factory Automation. Vol.1, pp.592-600.*

Roch, S. (2000a). Extended computation tree logic. In *Proceedings of the CESP2000 Workshop, number 140in Informatik Berichte, pages225-234, Germany*.

Roch, S. (2000b). Extended computation tree logic: Implementation and application. In *Proceedings of the AWPN2000 Workshop, Germany*.

Rooker, M. N., Sunder, C., Strasser, T., Zoitl, A., Hummer, O., and Ebenhofer, G. (2007). Zero downtime reconfiguration of distributed automation systems : The εcedac approach. In *Third International Conference on Industrial Applications of Holonic and Multi-Agent Systems*. Springer-Verlag.

SESA (2008). Signal/net system analyzer. In *http://www.ece.auckland.ac.nz/ vyatkin/tools/modelchekers.html*.

Vardi, M. and Wolper, P. (1994). Reasoning about infinite computations. In *Information and Computation, 115(1)*.