International Conference
2nd International Conference on Innovations, Recent Trends and Challenges in Mechatronics, Mechanical Engineering and New High-Tech Products Development
MECAHITECH'10

Bucharest, 23-24 September 2010

A Formal Approach for Safe Controllers Analysis

Paulo Borges
Mechanical Engineering Department, CT2M / University of Minho
Campus of Azurém, 4800-058 Guimarães, Portugal
pborges@gmail.com

José Machado
Mechanical Engineering Department, CT2M / University of Minho
Campus of Azurém, 4800-058 Guimarães, Portugal
jmachado@dem.uminho.pt

Eurico Seabra
Mechanical Engineering Department, CT2M / University of Minho
Campus of Azurém, 4800-058 Guimarães, Portugal
eseabra@dem.uminho.pt

Mário Lima
Mechanical Engineering Department, CT2M / University of Minho
Campus of Azurém, 4800-058 Guimarães, Portugal
mlima@dem.uminho.pt

ABSTRACT

Formal verification of real-time systems software is a complex and hard task, for several reasons. There are multiple works developed in the domain of formal verification of real-time systems behavior by model-checking, and some software tools were developed for this purpose. One of the most complex problems to be solved in the analysis of real-time controllers is the conversion of controllers programming languages in formal languages, for instance finite timed automata, in order to be used as inputs of the existing model-checkers. If the methodology of programming is well developed and known, this task can be improved in order to improve safety and reliability of the obtained controllers. Moreover, most real-time systems (especially embedded systems that we intend to study) are programmed in C language. This paper aims to establish the methodology of creating C code programs, from SFC specification formalism, taking into account the formal verification of desired properties for the system behavior, using the Model-Checking technique and the model-checker UPPAAL.

Keywords: *safe controllers, real-time systems, embedded systems, formal verification, specification formalisms*

INTRODUCTION

A Real-time embedded real-time system is a system that has specific characteristics for specific applications always associated with temporal goals, reliability, safety, size and complexity of the coordination of tasks. These systems can be classified as critical and non-critical, depending of associated specifications, requirements and applications. The distinction between these two types of systems is made according the result that a failure may cause. For instance, some production systems, banking systems are non-critical real time systems. In the case of critical systems, the consequences of a malfunctioning could be, in some cases, devastating for human life or may involve high economic losses. For instance, flight control systems, control systems for nuclear power plants, control of satellites, and others of the same kind, can be considered as critical real-time systems.

These systems are often complex and constantly interact with their environment by receiving input data, processing them in real time and generating outputs. Their behavior is constrained by temporal conditions, sometimes extreme, and they must generate a response in time, previously specified, related with expected or unexpected external behaviors.

Response actions, in real-time embedded systems, follow a programmed sequence of specific activities, with fixed and predetermined periods of time. These periods of time are generally met when the system operates normally without any failures of components. The problem arises when some component fails involving, as consequence, placing the system in safe mode, and can, for instance, in the case of a satellite control, cause the loss of it, due to the delay in response when some unexpected external conditions happen..

According to Stankovic, (1996) [1] "A Real Time System (RTS) is the one in which his correctitude depends not only on the logic of computation, but also the fulfillment of time in delivering results."

According to Shaw (in 2001) [2], a real-time system is composed by two parts: the *control system*, comprising the man-machine interface, and *controlled system*. The control system is responsible for responding to environmental stimuli in time. "It is said reactive because its primary task is to respond or react to signals from the environment." For instance, in an automated factory, the control system consists of a computer and man-machine interfaces that manage it and coordinate the activities at the factory. The interfaces are the network communication between the control system and controlled system. Generally, they are sensors, actuators, receivers of radio signals, among others. The controlled system is the environment that interacts with the computer, for instance, assembly lines and its various parts [1]. It is essential that the plant shall conform to the response time. "

In industrial automation some systems are really critical and some techniques are used to avoid damages. Among these techniques, the most important are Simulation and Formal Verification.

Formal verification of algorithms has been studied in science in recent years [3] [4], and it has been applied successfully to analyze, for instance, digital circuits and software [5]. In the context of the software verification, the obvious purpose of verification is to verify that the control system satisfies a given set of requirements. These requirements can be formulated considering the control system, the controlled system or both. Several approaches that use formal verification for designing safe controllers can be found in the literature, see for instance [6] [7] [8] [9].They differ concerning the representation of the system and controller, the properties behavior for the system and computational techniques.

This paper aims to propose a methodology for design of real-time embedded systems that are used in computers on board of satellites.

The main goal is to use analysis techniques, used in industrial automation field, well known and well tested for obtaining safe controllers for aerospace systems. In order to accomplish the main goal of this paper, some sections are related. The next section presents some particularities of aerospace systems. Further, it is presented a comparison between industrial systems and aerospace system, in order to illustrate how useful can be using some well structured formalisms (usually used in industrial automation field) to help structuring and obtaining the C program code, usually developed for real-time embedded systems, for aerospace applications. Finally, it is presented, and discussed, a coherent approach to be used for obtaining safe controllers, and further, some conclusions and future work are also presented.

These preliminary studies are presented on the context of a research collaboration project being developed by researchers of CT2M, ALGORITMI and CCTC research centers of University of Minho (Portugal) and the Mechanical Engineering Department of Technological Institute of Aeronautics (Brazil).

SPECIFICITY OF AEROSPACE SYSTEMS

The development of software code for aerospace systems is a hard and complex task that involves a lot of human and financial effort. The reusing of parts of code, for similar applications, is usual and this practice can lead to catastrophic situations because some unexpected events may occur with new applications considering old parts of code.

Several accidents have occurred on satellites due to specification errors or lacks [10]. On June 4, 1996, Ariane 5, on its inaugural flight, crashed 40 seconds after the start of the flight sequence in an altitude of 2700 meters. It was acknowledged in the report that the main cause of the accident was due to complete loss of guidance and attitude information at 37s after starting the engine ignition. The mentioned loss of information was due to specification errors in software development of the inertial reference system. The software had been reused from Ariane 4, and contained parts of code unnecessary for Ariane 5, which were also already unnecessary for Ariane 4.

On April 3, 1999, Titan IV B-32/Centaur TC-14/Milstar-3 was launched from Cape Canaveral [11], whose aim was to put into geostationary orbit. Due to failure in software development the satellite lost attitude control deviating from its orbit by placing an orbit incorrect and unhelpful.

As illustrated above, the software plays an increasingly essential role in aerospace systems. An inadequate development of software may cause catastrophic accidents. The reuse of code in aerospace systems is a reality, a reality which has its advantages and disadvantages. As mentioned above, Ariane 5 has exploded by misspecification in code that had been used in an earlier satellite, with functions that already existed in Ariane 4 but were not required in any of them. Good specifications that include requirements for traceability and reasoning are crucial to the design of complex control systems, especially where part of the code is reused. The specifications must be clear and easily understood by engineers and must allow a fast detection of possible errors or mistakes.

In order to increase the reusability of code, specific information is left out of the specification or, if included, it is identified as aspects in order to change in future applications.

It is also noted that code reusing is possible, only, because most of the satellites often require almost the same functions.

Programming in C

The C programming language, created in the 70s and standardized by ANSI in 1983, is a medium level language, where the code can be low-level and also allows high-level bit manipulation instructions or memory. It is used to program microcontrollers and it is also used in most embedded systems. Although very general language, it is a little flexible language [12], with many maintenance problems and without any graphical structure. Beyond these disadvantages, there is another problem not least important, that is the absence of formal verification techniques as well as the lack of specification methods to structure C programs, even if this exist some Model-Checkers that accept C code as input language [13].

The C language, like others, allows deficient structuring of programs, difficulties in reusing of code and a lack of flexibility in programming, so it is necessary to use different formalisms that help obtaining the code for these control systems.

Programming languages are based on sets of algebraic expressions such as those resulting from the combination of problems of combinatorial or sequential nature. But, in industrial automation, it is intended the obtaining of algebraic expressions resulting from the translation of formal models defined when using rigorous and well known formalisms such as, for instance, the SFC [14], statecharts [15] or the Networks Petri [16] among others.

INDUSTRIAL CONTROLLERS *VERSUS* AEROSPACE CONTROLLERS

The Programmable Logic Controllers (PLCs) are increasingly used in several application areas, especially in critical safety areas.

Like an embedded system, a PLC is programmed with a specific language. Embedded systems and PLCs have some similarity, both are programmable logic controllers, react with the external inputs and generate outputs according an internal program. They are also based on technologies of microprocessors, programmable logic with their limitations, such as limiting the frequency of internal operation. Based on these principles, some authors [17] begin to treat embedded systems like PLCs enjoying the available support tools to industrial automation systems, mainly some used formalisms.

Taking into account aerospace systems, where the controllers' programming language is C, one of the proposed challenges is the use of mathematical formalisms to support obtaining the C code. The first step is to choose a formalism and then to develop translation techniques from this formalism to C code. With the systematic methodology of translation of a formalism to C code, the reutilization of small parts of code and the organization of the program would be considerably improved.

Some authors [18] tried, before, to use formalisms from the industrial automation field in order to develop some techniques of translation of these formalisms to the C programming language code. The translation that they have developed is from Sequential Function Chart - SFC [14] to the C programming language. This translation is based on mathematical concepts and eliminates some possible human mistakes when compared to doing this translation, without any associated methodology. One of the lacks of this work seems to be not considering the behavior of the controller device where the code is going to be introduced. It is not, only, necessary to translate the formalism, but to consider, too, the behavior of the controller where the code will be implemented.

USED FORMALISM

The task of design of an automation system obeys to different rules and is independent of the used formalism. A possible methodology for designing an automation system is illustrated in Figure 1..

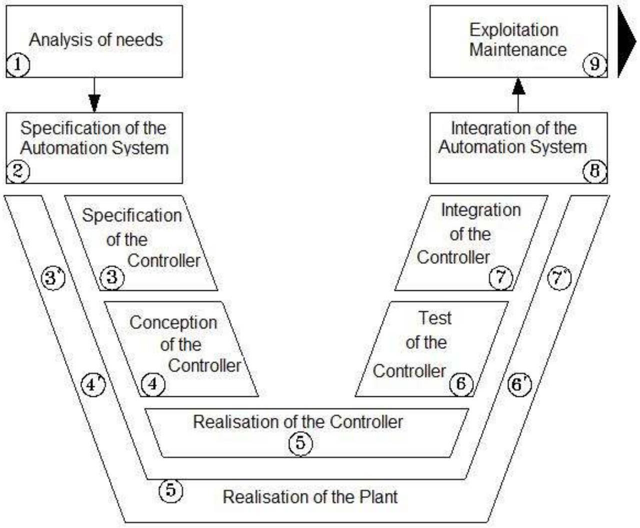

Analysis of needs
1
Specification of the Automation System
2
Specification of the Controller
3
3'
Conception of the Controller
4
4'
Realisation of the Controller
5
5
Realisation of the Plant
Test of the Controller
6
6'
Integration of the Controller
7
7'
Integration of the Automation System
8
Exploitation Maintenance
9

Figure 1: Phases of the design, implementation and verification of a command system for Industrial Automation

A total reflection of needs (production goals, objectives of automation,…) allows a precise definition of the functional specifications of an automation system. It is possible to make the parallel study of the controller and the physical plant. Each of these studies leads to the existence of the specification, realization, implementation, testing and integration of the system.

The integration of all steps of this study, in a coordinated way, and final testing of the set, allowing the automation system enter in the phase of exploitation.

In an analysis of the existing formalisms for modeling the desired behavior for aerospace systems there are some, more or less used, by their nature: from the finite automata [20], Petri nets [16], SFC [14] Statecharts [15] it is concluded that any of these formalisms can be used in specific specification of the behavior of these systems. The choice of a formalism, in this context, is not important; the most important is to use one of them, in a correct manner. For example, the behaviors that can be modeled by finite automata or Petri nets are the same (if we choose the adequate class of automata), just changing the complexity and comprehensibility of the obtained model. Moreover, the obtained model can be more or less compact. The factor "time", very important in the analysis of real-time systems, can also be considered. The choice will lie with the formalism or formalisms that are better adapted, for instance, for the application of simulation techniques and formal verification, which is intended for obtaining safe controllers.

The finite automata are widely used for modeling and formal verification of safe systems, widely desirable for the software design which is important for any process control. In general, the software is closely connected with the system that must be controlled. Verification of this is done by building an abstract model of the system, and then checked whether it fulfills the desired requirements. Because there are many model-checkers that accept a state model as input, the tasks of verification would be facilitated if a controller is modeled with finite automata, but the major problem of using finite automata is the complexity of models that often are needed to describe more complex behaviors. Thus, the choice must lie in a formalism with a greater capacity for abstraction, which is graphical and easily understood and with an abstraction degree sufficiently close to implementation.

Given the characteristics listed above, Petri Nets, Statecharts and SFC would be good choices. We believe that SFC formalism may be a slight advantage due to intuitive graphical interpretation and normalization, and it is also possible to consider and model the time [14]. In addition, there are some consolidated works dealing with translation of this formalism to C programming language [17]. Also, some consolidated works of translation of this formalism [21] [22] for the model-checker UPPAAL [23] are available, specially developed and designed for checking real-time systems. Thus, if developed a specification for these systems entirely in SFC and if is subsequently simulated and formally verified with UPPAAL model-checker and, further, if the formalism is systematically translated to C language, we can say that our software is reliable and safe. This exist also the possibility of verifying that software developed in C language, using model-checkers that accept C language as input [17].

SYSTEMATIC METHODOLOGY FOR DESIGNING AEROSPACE SYSTEMS

The proposed methodology is divided in two main steps and has, as main goal, to obtain safe C program code from a SFC specification. In the first step (figure 2), some tools and techniques are used in order to assure the quality of the SFC specification and – after being sure that the obtained specification satisfies the intended behaviors for the system - on the second step (figure 3) the goal is to translate (in a systematic way) the safe SFC specification to C code.

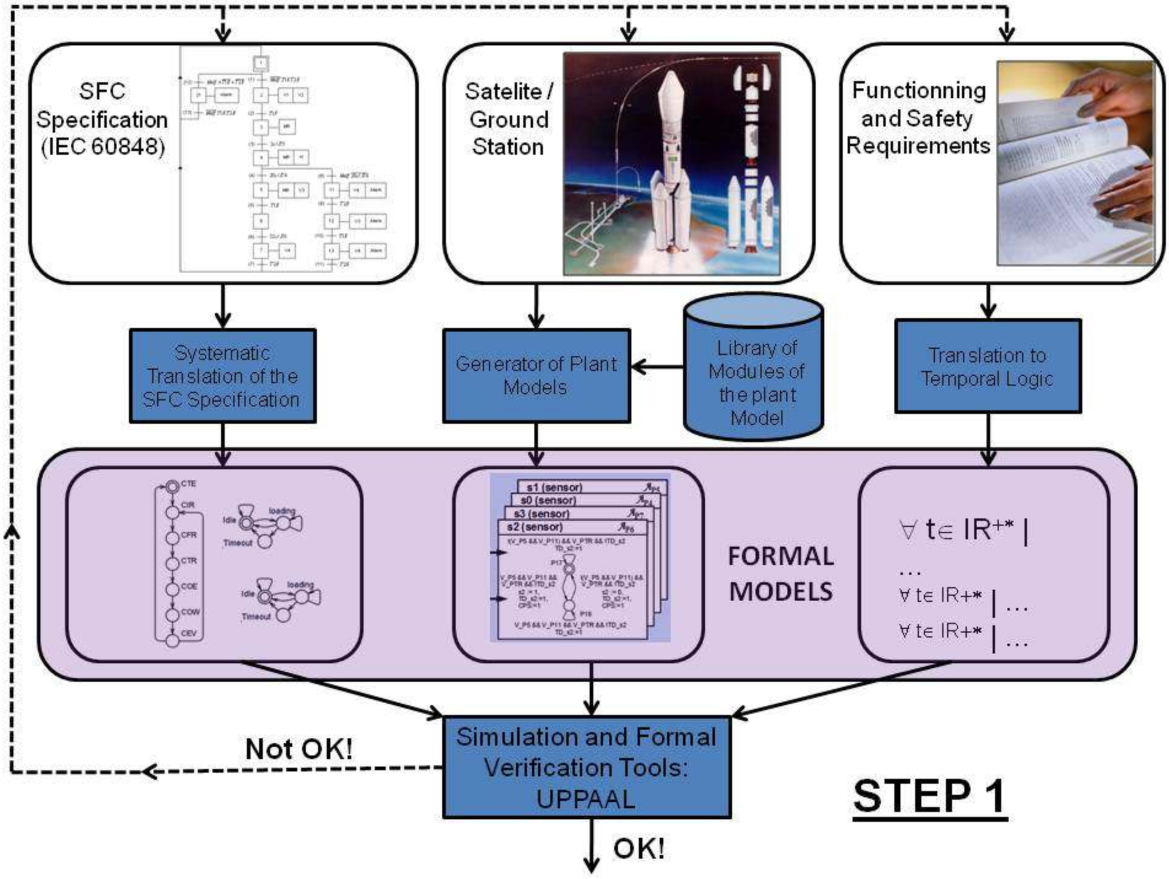
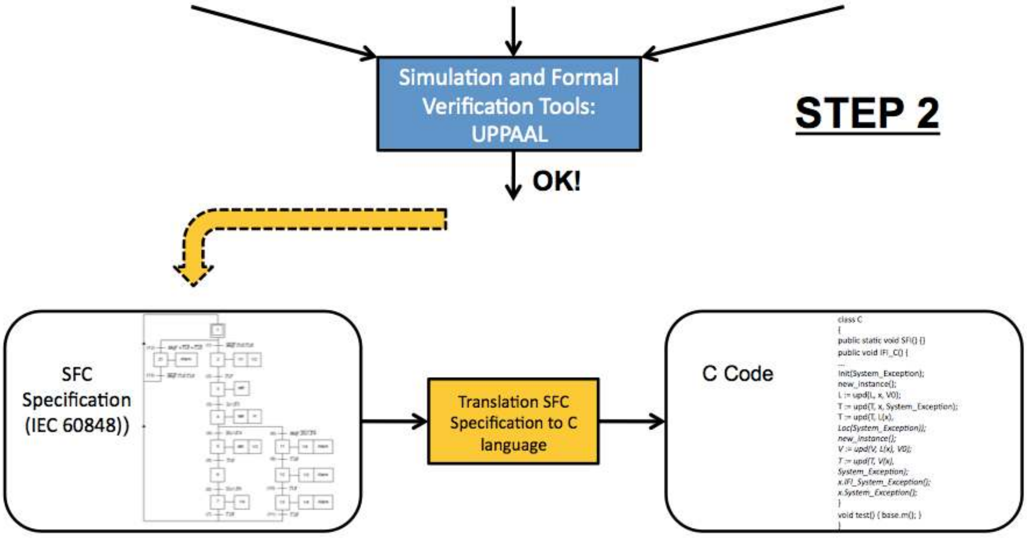

SFC Specification (IEC 60848)
Satelite / Ground Station
Functionning and Safety Requirements
Systematic Translation of the SFC Specification
Generator of Plant Models
Library of Modules of the plant Model
Translation to Temporal Logic
FORMAL MODELS
∀ t∈ IR+* |
∀ t∈ IR+* | …
∀ t∈ IR+* | …
Simulation and Formal Verification Tools: UPPAAL
Not OK!
OK!
STEP 1

Figure 2: Formal verification of specification SFC, considering models of the physical system: step 1 of the proposed systematic approach for obtaining safe controllers, reliable for aerospace systems

The first step (Figure 2) consists of the formal verification of the SFC specification considering a model of the specification itself. This model is systematically translated to timed automata [21]. This verification must also consider formal models of satellites behavior and ground station. If there is certain, that safe properties must be verified without a plant model (where plant models are not considered), the Liveness properties, must be verified considering plant models [24].The desired behavior properties must be translated into Timed Computation Tree Logic (TCTL) [25].

Until the specification is correct - through successive use of Model-Checker UPPAAL and using simulation and formal verification techniques - the procedures, illustrated in figure 2, must be followed. The specification must be changed as many times as necessary till obtaining a specification that accomplishes all desired behavior properties for the system, proved by formal verification.

After the specification be correct the second step can occur (Figure 3) where a systematic translation of the specification [17] will origin a C code with high levels of reliability and safety.


Simulation and Formal Verification Tools: UPPAAL
STEP 2
OK!
SFC Specification (IEC 60848))
Translation SFC Specification to C language
C Code

```
class C
{
public static void SFI() {}
public void IFI_C() {
...
Init(System_Exception);
new_instance();
L := upd(L, x, V0);
T := upd(T, x, System_Exception);
T := upd(T, L(x),
Loc(System_Exception));
new_instance();
V := upd(V, L(x), V0);
T := upd(T, V(x),
System_Exception);
x.IFI_System_Exception();
x.System_Exception();
}
void test() { base.m(); }
}
```

Figure 3: Translation systematic specification SFC (formally verified the model-checker UPPAAL) to C code: Step 2 of the proposed systematic approach for obtaining safe controllers, reliable aerospace systems

Thus, we think that it is possible to obtain high levels of reliability and safety of these programs. It is certain, however, that a subsequent formal verification of C code, itself, may further increase these levels but, despite the existing work in this area of verification of C code, there are still many difficult tasks related with formal verification of code [26].

The main advantage of doing formal verification on specification and not on the code is the detection of errors or mistakes earlier in the process of design of these controllers. The detection of an error, or mistake, only in the C program can imply high losses of time and, even, to compromise a mission.

CONCLUSIONS AND FUTURE WORK

In this paper it was presented a systematic approach for the design of safe controllers for aerospace systems. This is an on-going work and this approach seems to be promising.

This paper corresponds to the preliminary studies carried out under a research collaboration between the centers CT2M, ALGORITMI and CCTC of University of Minho, Portugal, and Department of Mechanical Engineering Institute of Aeronautical Technology, Brazil, aiming the development and application of some techniques for obtaining real-time embedded controllers reliable and safe. Results of the application of the presented approach are becoming satisfactory, and these results are not here described or specified. They will be, so, in further publications.

The use of well known formalisms and techniques allow us to obtain good results in order to obtain reliable and safe controllers for this specific application.

REFERENCES

[1] Stankovic, John A. “Real-time and embedded systems”, 28 (1).ACM Computing Surveys. 1996

[2] Shaw, Alan C. “Real-time Systems and Software”, (first edition), New York. 2001

[3] Kurshan, R. “Computer-Aided Verification of Coordinating Processes: The Automata-Theoretic Aproach.” Priceton Univ. Press. 1994

[4] Clarke, E., Grumberg, O., and Peled, D., “Model Checking”. MIT Press. 1999

[5] Clarke, E. and Wing, J. “Formal methods: state of the Artandi future directions”. ACM Comp.Surveys. December 1996;28,4.

[6] Tomlin, C., Mitchell, I., Bayen, A., and Oishi, M. “Computational techniques for the verification of hybrid systems.” Proc. of the IEEE. 2003;91,7.

[7] Havelund, K., Larsen, K., and A-Skou ”Formal verification of a power controller using the real-time model checker UPPAAL2K.” in Proc.5th AMAST Workshop. 1999;277.

[8] Kapinski, J. and Krogh, B. “A new tool for Verifying computer controlled systems. Conf.” on Computer-Aided Control System Design, IEEE 2002;98.

[9] Stursberg, O., Kowalewski, S., Preussig, J., and Treseler, H. “Block-diagram based modeling and analysis of hybrid processes under discrete control.” J. Europ. des Syst.Automatises. 1998;32,9-10.

[10] Leveson, NG. “The role of software in aerospace accidents.” AIAA Journal of Spacecraft and Rockets (in press). 2003.

[11] Pavlovich, JG, “Report of Formal Investigation of the 30 April 1999 Titan IV B / Centaur TC-14/Milstar-3 (B-32) Space Launch Mishap.” U.S. Air Force. 1999

[12] Gupta, G. “Reliable software construction: a logic programming based methodology.” Proced. fith of IEEE International Symposium on High Assurance Systems Engineering, IEEE. 2000;140.

[13] Clarke, E., Kroening, D., and Lerda, F. ”A tool for checking ANSI-C programs.” In K. Jensen and A. Podelski, editors, cups, 2004, (vol. 2988 of Lecture Notes in Computer Sc IENCE, pages 168-176).Springer. 2004.

[14] EN 2002 – “European Standard 60 848: GRAFCET specification language for sequential function charts.” 2002.

[15] Harel , D. “Statecharts: A visual formalism for complex systems.Science of Computer Programming.” 1987;231.

[16] Meda-Campaña, ME and Lopez-Mellado E. "Incremental synthesis of petri net models for identification of discrete event systems." Proceedings of the 41st IEEE Conference on Decision and Control.. Las Vegas, Nevada USA. December 2002.

[17] Bayo-Puxan, O., Rafecas-Sabaté, J., Gomis-Bellmunt, O., and Berger-Jané, J. "GRAFCET-compiler methodology for C-programmed microcontrollers." Assembly Automation Emerald Group Publishing Limited. 2008;55.

[18] Clarke, E., Kroening, D., Sharygina, N., and Yorav, K. "Predicate abstraction of ANSI-C programs using SAT." (FMSD) Formal Methods in System Design,. 2004;25,105.

[19] Lewis, RW. "Programming Industrial Control Systems Using IEC 1131-3." The Institution of Electrical Engineers. London. Revised edition. 1998;5.

[20] Klein, S. "isttable; fault detection of discrete event systems using an identification approach*."* Doctoral Thesis. University of Kaiserslautern, Kaiserslautern. Juny 2005.

[21] Remelhe, MP, Lohmann, S., Stursberg, 0., and Engell, S. "Algorithmic Verification of Logic Controllers Given the Sequential Function Charts." IEEE International Symposium on Computer Aided Control Systems Design Taipei. Taiwan. 2004.

[22] Stursberg, O., Lohmann, S., and Engell, S. "Improving dependability of logic controllers by algorithmic verification." IFAC World Congress. Czech Republic. 2005;16,1.

[23] Gourcuff, V. "Verification of a timed multitask system with UPPAAL." Memory of DEA Lurpa. ENS Cachan from. 2004.

[24] Machado, J., Denis, B. and Lesage, JJ. "A generic approach to build plant models for DES verification purposes." Proc. of Wodes'2006 - 8 th Workshop on Discrete Event Systems. Ann Arbor, Michigan, USA. July 2006.

[25] Alur, R., Courcoubetis, C. and Dill, DL. "Model-checking in dense real-time." Information and Computation. 1993;104,1.

[26] Ball, T., Podelski, A. and Rajamani, SK. "Boolean and cartesian abstractions for model checking C programs." In T. Margara & W. Yi (eds.)*LNCS*. Springer Berlin Heidelberg. 2001;268.