ZiZo: Modeling, Simulation and Verification of Reconfigurable Real-time Control Tasks Sharing Adaptive Resources
Application to the Medical Project BROS

Mohamed Oussama Ben Salem[1,2], Olfa Mosbahi[1,4], Mohamed Khalgui[1,4] and Georg Frey[3]
[1]*LISI laboratory, INSAT, University of Carthage, Tunis, Tunisia*
[2]*Polytechnic School of Tunisia, University of Carthage, Tunis, Tunisia*
[3]*Saarland University, Chair of Automation, Saarbrücken, Germany*
[4]*eHealth Technologies Consortium, eHTC, Tunis, Tunisia*

Keywords: Distributed Control System, Reconfiguration, Shared Resource, Simulation, Verification, Model Checking, Computer-assisted Surgery.

Abstract: This research paper deals with the modeling, simulation and model checking of reconfigurable discrete-event control systems to be distributed on networked devices. A system is composed of software tasks with shared resources to control physical processes. A reconfiguration scenario is assumed to be a run-time automatic operation that modifies the system's structure by adding or removing tasks or resources according to user requirements in order to adapt the whole architecture to its environment. Nevertheless, a reconfiguration can bring the system to a blocking problem that is sometimes unsafe, or violates real-time properties. We define new Petri Nets-based modeling solutions for both tasks and resources to meet these constraints. These solutions are applied to a real case study named Browser-based Reconfigurable Orthopedic Surgery (abbrev. BROS) to illustrate the paper's contribution. A new Petri Nets-based editor and random-simulator named ZiZo is developed to model and simulate the BROS reconfigurable architecture. It is based also on the model checker SESA to apply an exhaustive CTL-based formal verification of this architecture to ensure safe reconfiguration scenarios of tasks and resources.

1 INTRODUCTION

Robotics is a field that is expanding every day. Robots have left the controlled environment of factories and warehouses they were initially designed for, and are making their way into the highly dynamic and unconstrained world of humans. The excellent geometric accuracy of robots, associated with their ability to integrate several different sources of information, has led to their natural implementation in medicine, more specifically in surgery. The field of robotic surgery is relatively new. The first clinical application of a robot was performed to a neurosurgery in 1985 (Kwoh et al., 1988). Since then, many research centers around the world have developed a multitude of robotic surgical products, tackling new areas such as orthopedics, radiology, urology, cardiothoracic and ophthalmology (Cleary and Nguyen, 2001; Wang et al., 2006; Gomes, 2011). The more the field grows, the more demanding become the end-users. Increasing safety constraints and growing expected flexibility pushed developers to focus on designing robots that are able to fit their environment and shifting users requirements under functional and temporal constraints. This is what we call reconfiguration.

It is in this context that BROS project is being taken. BROS is a reconfigurable robotic platform dedicated to the treatment of elbow's supracondylar fracture. It is capable of running under several operating modes to meet the surgeon's requirements and well-defined constraints. Given the criticality of such a system, checking the safety of BROS becomes crucial. Thus, before starting the implementation, we choose to model the whole system using Reconfigurable Timed Net Condition/Event Systems (R-TNCES) (Zhang et al., 2013), a new formalism extending Petri nets and useful to model such adaptive control systems. Nevertheless, when designing BROS, we face the issue of concurrent access to shared resources, such as the patient's arm and the browser, an image

Ben Salem M., Mosbahi O., Khalgui M. and Frey G..
ZiZo: Modeling, Simulation and Verification of Reconfigurable Real-time Control Tasks Sharing Adaptive Resources - Application to the Medical Project BROS.
DOI: 10.5220/0005181600200031
In *Proceedings of the International Conference on Health Informatics* (HEALTHINF-2015), pages 20-31
ISBN: 978-989-758-068-0
Copyright © 2015 SCITEPRESS (Science and Technology Publications, Lda.)

guidance system. We choose, then, to use a PCP-based new solution for R-TNCES to model reconfigurable shared resources (Salem et al., 2014).

The reconfiguration feature of BROS can bring the latter to a blocking problem that is sometimes unsafe or does not respect real-time properties. We opt, then, for the use of a tool to model and simulate the whole BROS architecture, and, then, apply on it several CTL formulas to check whether the system respects its functional and temporal constraints. However, no one in our community worked on such a tool. Thus, the authors in this paper present a new tool, baptized ZiZo, a R-TNCES modeling and random-simulating software.

The current paper is organized as follows: the next section describes useful preliminaries for the reader to understand our contribution. Section 3 introduces the BROS as project. We expose, in Section 4, the modeling and verification of our robotic platform. Section 5 presents the new tool ZiZo, before finishing the paper in Section 6 with a conclusion and an exposition of our future works.

2 BACKGROUND

We start, in this section, by presenting the formalisms TNCES (Hanisch et al., 1997) and R-TNCES (Zhang et al., 2013), which extend Petri nets for the modeling of adaptive control systems, and the existing tools to model them. We expose, thereafter, two well-known protocols, PIP and PCP, and a PCP-based solution for the management of resource sharing in R-TNCES.

2.1 Modeling Formalisms and Tools

We introduce in this section two formalisms extending Petri nets and which are useful to model distributed reconfigurable control systems. We provide, then, an overview of the existing tools to model Petri Nets.

2.1.1 Timed Net Condition/Event System

A Timed Net Condition/Event System (TNCES) (Hanisch et al., 1997) has a modular structure which may be basic or composite. A basic TNCES is an elementary module extending Petri nets by proposing the new concepts of event and condition signals. A composite TNCES can be composed of basic/composite interconnected modules in a hierarchical form. A TNCES, as shown in Figure 1, is formalized as a tuple in (Zhang et al., 2013) as follows:

$$TNCES = \{PTN; CN; WCN; I; WI; EN; em\} \quad (1)$$

where: (i) $PTN = (P; T; F; K; WF)$ is a classic Place/Transition Net, (ii) $CN \subseteq (P \times T)$ is a set of condition arcs, (iii) $WCN: CN \rightarrow N^+$ defines a weight for each condition arc, (iv) $I \subseteq (P \times T)$ is a set of inhibitor arcs, (v) $WI: I \rightarrow N^+$ defines a weight for each inhibitor arc, (vi) $EN \subseteq (T \times T)$ is a set of event arcs free of circles, (vii) $em : T \rightarrow \{\vee,\wedge\}$ is an event mode for every transition (i.e. if $\wedge$ then the corresponding events have to occur simultaneously before the transition firing, else if $\vee$ then one of them is enough).

Time intervals are assigned to the pre-transition flow arcs, which imposes time constrains to the firing of the transition. They are formalized as follows:

$$DC=\{DR, DL, D_0\} \quad (2)$$

where: (i) DR is a set of delay time that represents the set of minimum times that the token should spend at a particular place before the transition can be fired, (ii) DL is the final set of limitation time that defines the maximum time that a place may hold a token (if all the other conditions for transition firing are met), (iii) D_0 is the initial set of the clocks associated with the places.

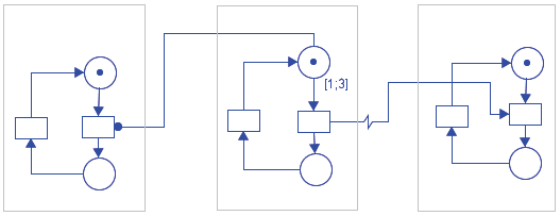

Figure 1: Example of a TNCES.

2.1.2 Reconfigurable Timed Net Condition/Event System

An R-TNCES, as defined in (Zhang et al., 2013), is a structure $RTN =(B, R)$, where R is the control module consisting of a set of reconfiguration functions $R = \{r_1,...,r_m\}$ and B is the behavior module that is a union of multi TNCESs, represented as

$$B =(P, T, F, W, CN, EN, DC, V, Z) \quad (3)$$

where: (i) P (respectively, T) is a superset of places (respectively, transitions), (ii) $F \subseteq (P \times T) \cup (T \times P)$ is a superset of flow arcs, (iii) $W: (P \times T) \cup (T \times P) \rightarrow \{0, 1\}$ maps a weight to a flow arc, $W(x, y) > 0$ if $(x, y) \in F$, and $W(x, y)=0$ otherwise, where $x, y \in P \cup T$, (iv) $CN \subseteq (P \times T)$ (respectively, $EN \subseteq (T \times$

T)) is a superset of condition signals (respectively, event signals), (v) $DC: F \cap (P \times T) \rightarrow \{[l1, h1],...,[l_{|F \cap (P \times T)|}, h_{|F \cap (P \times T)|}]\}$ is a superset of time constraints on output arcs, where $\forall i \in [1, |F \cap (P \times T)|]$, $l_i, h_i \in N$, and $l_i < h_i$, (vi) $V : T \rightarrow \{\vee, \wedge\}$ maps an event-processing mode (AND or OR) for every transition, (vii) $Z_0 = (M_0, D_0)$, where $M_0: P \rightarrow \{0, 1\}$ is the initial marking and $D_0: P \rightarrow \{0\}$ is the initial clock position.

2.1.3 Existing Tools

Several tools already exist to model and/or simulate Petri nets and their extensions. For example, CPN tools is a software for editing, simulating and analyzing Colored Petri Nets. It features a fast simulator that efficiently handles both untimed and timed nets. Full and partial state spaces can be generated and analyzed, and a standard state space report contains information such as boundedness properties and liveness properties (Ratzer et al., 2003). Petri.NET is another tool which allows modeling, simulation and real-time implementation of static and dynamic Petri nets. The results of a Petri net model simulation are presented to the user in the form of a token game and in the graphical form showing diagrams of a state vector (Genter et al., 2007). Nevertheless, neither CPN tools nor Petri.NET can support R-TNCES with their condition and event signals. The VisualVerification (ViVe) toolset is a tool chain for automatic verification of distributed control systems. It allows creation and modification of model components in modelling language of Net Condition/Event Systems (NCES) (Suender et al., 2011). But, it does not deal with the notion of time in NCES and the reconfiguration feature they may have. The TNCES-Editor, developed at the Martin Luther University Halle-Wittenberg, allows the graphical modeling of all NCES based subtypes, including R-TNCES (Dubinin et al., 2006). To support interpretation and reachable state analysis, the TNCES-Editor offers an optional labeling of transitions. The whole net structure including the labels will be stored in a special file format (*.pnt) which can be used as an import file for the model-checker SESA. However, TNCES-Editor doesn't feature the simulation of a built R-TNCES, nor highlights the reconfiguration aspect of a DRCS.

2.2 Resource Sharing Protocols

This section presents the two protocols PIP and PCP. It introduced, then, the specification and modeling of the PCP-based solution for dynamic resource sharing in R-TNCES.

2.2.1 Priority Inheritance Protocol

Priority Inheritance Protocol (PIP) in real-time computing is a solution to avoid the problems of priority inversion. As introduced in (Sha et al., 1990), the basic PIP prevents any blocking of higher priority tasks by lower ones. In fact, if a lower priority task blocks a higher one, then it should execute its critical section with the priority level of the higher priority task that it blocks. In this case, we say that the lower priority task inherits the priority of the higher priority task. In PIP, the maximum blocking time (due to a lower priority task) is equal to the length of one critical section and the blocking can occur at most one time for each lock.

A PIP operation is summed up in (Sha et al., 1990) as follows: Let us assume a system to be composed of a set of tasks $\{T_1,...,T_n\}$ such that (i) T_i and T_k share a resource R ($i \in (1,n)$ and $k \in (1,n)$), (ii) Priority of T_i is lower than T_k's. Then, if T_k is blocked on a semaphore that corresponds to a resource in use by T_i, then T_i immediately inherits the priority of T_k in order to unblock it as soon as possible.

2.2.2 Priority Ceiling Protocol

The Priority Ceiling Protocol (PCP) (Goodenough and Sha, 1988) in real-time computing is a synchronization protocol for shared resources to avoid unbounded priority inversion and mutual deadlock due to wrong nesting of critical sections. In this protocol, each resource R is assigned a priority ceiling Cl(R), which is equal to the highest priority of the tasks that may lock it. A task can acquire a resource only if the resource is free and has a higher priority than the priority ceiling of the rest resources in lock by other tasks.

Let us assume a system to be composed of the tasks T_1, T_2, T_3 and T_4 (having respectively the increasing priorities 1, 2, 3 and 4) and two resources R and Q: R can be used by T_1 and T_2 and Q by T_1 and T_4. Then, Cl(R)=2 and Cl(Q)=4. Thus, T_2 is blocked if it tries to block R which is free when Q is locked. PCP brought improvements from PIP since it gives guarantees that there is no deadlock and each task in blocked at most the duration of one critical section. However, there is a downside when using PCP; there is more run-time overhead than PIP.

2.2.3 PCP-based Solution for Resource Sharing in R-TNCES

We aim in this section to check the safety of each reconfiguration scenario by enriching the Reconfigurable Timed Net Condition/Event System (R-TNCES) with the PCP protocol. We propose, then, to use new patterns introduced in (Salem et al., 2014) to model reconfigurable discrete event systems according to R-TNCES by using PCP. This contribution is original since R-TNCES is an original formalism for reconfigurable systems, but lacks of useful mechanisms to manage reconfigurable shared resources.

a. Formalization

We present in this section the formalization of Distributed Reconfigurable Control Systems (DRCS) sharing resources.

DRCS: The authors in (Salem et al., 2014) assume a DRCS D to be composed of n_1 networked reconfigurable sub-systems sharing n_2 resources. They extend the formalization of DRCS in (Zhang et al., 2013) by adding the new set of resources as follows:

$$D = (\sum R\text{-}TNCES, \varpi, \sum M, \sum R) \quad (4)$$

where: (i) ∑R-TNCES is a set of n_1 R-TNCES, (ii) ϖ a virtual coordinator handling $\sum M$, a set of Judgment Matrices, (iii) $\sum R$, a set of n_2 shared resources.

Shared Resources: On the basis of PCP's definition and the flexibility expected from the DRCS, a resource R is defined as follows :

$$R = (Rec, S, Cl) \quad (5)$$

where: (i) Rec (Reconfiguration) indicates whether R is added to the system / $Rec \in \{added, !added\}$, (ii) S indicates the state of R / $S \in \{free, hold\ by\ a\ task_i\}$, (iii) Cl is used for the ceiling of R.

Control Tasks: Based on the expected reconfiguration of the system, the authors in (Salem et al., 2014) defines a task T by:

$$T = (Rec, S) \quad (6)$$

where: (i) *Rec* (Reconfiguration) indicates whether T is added to the system / R ∈ { *added, !added* }, (ii) S indicates the state of T / $S \in$ { *idle, execute, wait,* $P(R_i)$, $V(R_i)$ } and $P(R_i)$ means locking R_i and $V(R_i)$ unlocking it.

b. Modeling

The authors in (Salem et al., 2014) proposes new solutions to introduce PCP in R-TNCES to avoid any blocking problem after reconfiguration scenarios. An R-TNCES model is proposed for each resource of ∑*R* and task of ∑*R-TNCES*.

Shared Resources: Each shared resource is modeled by a R-TNCES as shown in Figure 2. The latter is composed of three TNCES modeling the resource's reconfiguration (Rec), state (S) and ceiling (Cl). Here is the modeling of a resource R:

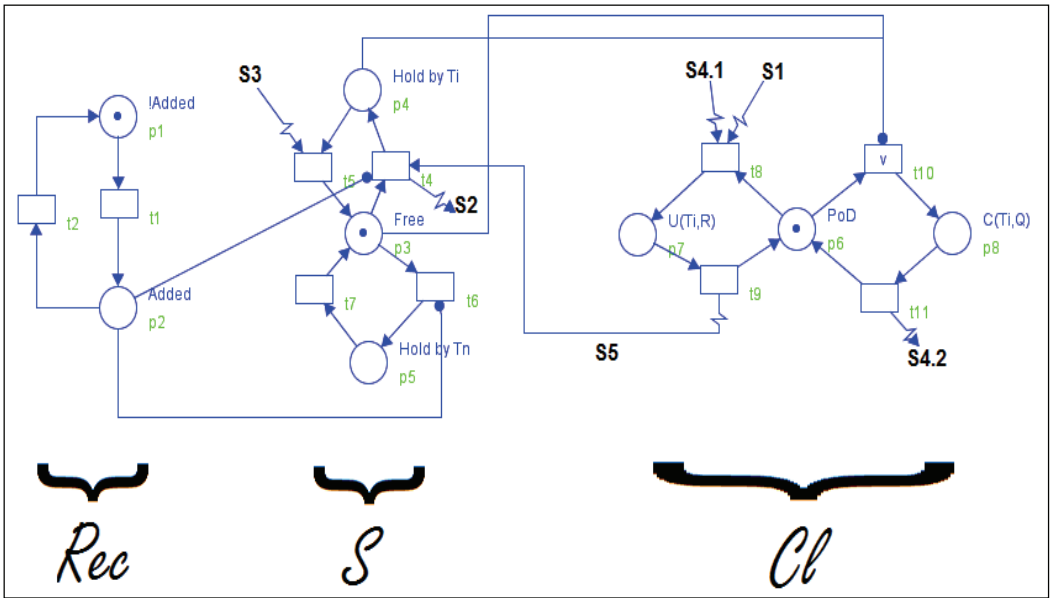

Figure 2: A shared resource's modeling.

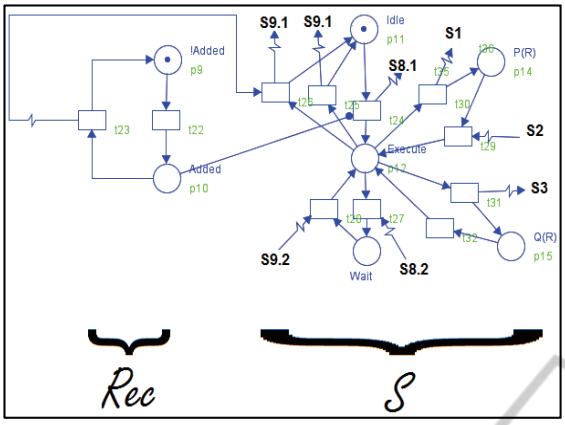

Figure 3: A task's modeling.

Control Tasks: The authors in [6] model each task T by an R-TNCES to be composed of two TNCESs as shown in Figure 3: the first one is illustrating its reconfiguration (Rec), the second its state (S).

3 BROS AS AN ORIGINAL PROJECT

We present in this section the project's motivations and BROS's architecture and operating modes. We expose, thereafter, the problem that may be faced during running of the platform.

3.1 Motivations

Supracondylar fractures of the humerus (or SCH) are a common pediatric elbow injury. They account for 18% of all pediatric fractures and 75% of all elbow fractures ((Brubacher and Dodds, 2008; Cheng et al., 1999; Landin and Danielsson, 1986). They mainly occur during the first decade of life and are more common among boys (Landin, 1983).

The current treatment of SCH fracture may actually lead to many complications. The neurological ones consist in damages caused to the median nerve during the reduction of the fracture or during the open procedure. For example, there were in (Gosens and Bongers, 2003) 23 nerve injuries in 189 patients with closed reduction and percutaneous pinning. 10 of them (9 ulnar nerves and one median nerve) were caused by the reduction of the fracture or the percutaneous pinning. The study in (Gosens and Bongers, 2003) also reports some vascular complications, mostly consisting in the disruption of the brachial artery. Some of them happened during the surgical intervention. Others complications may also occur, like an inadequate reduction (Baumann's angle >10°) of the fracture revealed when reviewing the postoperative X-ray. Repeated percutaneous pinning after satisfactory reduction was performed. All those complications are principally caused by the "blind" pinning the surgeons perform (Flynn, 1993; Flynn et al., 1974). Even though they are usually using an image intensifier, the medical staff can't guess in advance the trajectory the pin will follow. Images are actually taken once the pin is inserted, which may cause the previously mentioned complications.

Other inconvenient of the current treatment technique is the recurrent medical staff exposure to radiations when using the fluoroscopic C-arm (Livyatan et al., 2002; Clein, 1954). These X-ray Radiations are harmful, and fluoroscopic examinations usually involve higher radiation doses than simple radiography.

Considering these constraints and issues, a new project, baptized BROS, has been launched to remedy these problems. This work is carried out within a MOBIDOC PhD of the PASRI program, EU-funded and administered by ANPR in Tunisia.

3.2 Architecture

BROS is a robotic platform dedicated to humeral supracondylar fracture treatment. It is able to reduce fractures, block the arm and fix the elbow bone's fragments by pinning. It also offers a navigation function to follow the pins' progression into the fractured elbow.

BROS is composed of a Browser (BW), a Control Unit (UC), a Middleware (MW), 2 Pining Robotic Arms (P-BROS1 and P-BROS2) and 2 Blocking and Reducing arms (B-BROS1 and B-BROS2). The said components are detailed hereafter.

a. Browser

The browser, which is a Medtronics's product and called FluoroNav, is a combination of specialized surgical hardware and image guidance software designed for use with a StealthStation Treatment Guidance System. Together, these products enable a surgeon to track the position of a surgical instrument in the operating room and continuously update this position within one or more still-frame fluoroscopic images.

b. Control Unit

The Control Unit (CU) ensures the smooth running

of the surgery and its functional safety. It asks the supracondylar fracture's type to the middleware, and then computes, according to it, the different coordinates necessary to specify the robotic arms' behaviors concerning the fracture's reduction, blocking the arm and performing pinning. The surgeon monitors the intervention progress thanks to a dashboard installed on the CU.

c. Middleware

The middleware (MW) is a mediator between the CU and the BW. It is an intelligent component that provides several features of real-time monitoring and decision making. The middleware contains several modules: (i) an image processing module to determine the fracture's type, (ii) a database containing a range of supracondylar fracture images classified according to their type, (iii) a learning module to enrich the database with new images acquired during each intervention, (iv) a controller, (v) a verification module, (vi) a communication module with the CU.

d. Pining Robotic Arms

The two pining robotic arms, P-BROS1 and P-BROS2, insert two parallel Kirschner wires according to Judet technique (Judet, 1953) to fix the fractured elbow's fragments. To insure an optimal postoperative stability, BROS respects the formula $S = \frac{B}{D} > 0.22$, where S is the stability threshold, B the distance separating the two wires and D the humeral palette's width (Smida et al., 2007).

e. Blocking and Reducing Robotic Arms

B-BROS1 blocks the arm at the humerus to prepare it to the fracture reduction. B-BROS2 performs then a closed reduction to the fractured elbow before blocking it once the reduction is properly completed.

3.3 Reconfiguration and Operating Modes

Reconfiguration is an important feature of BROS. It is designed to be able to operate in different modes. The surgeon can actually decide to manually do a task if BROS does not succeed to automatically perform it, whether it is facture reduction, blocking the arm or pinning the elbow. Thus, five different operating modes are designed and detailed below.

Automatic Mode (AM): The whole surgery is performed by BROS. The surgeon oversees the operation running.

Semi-Automatic Mode (SAM): The surgeon reduces the fracture. BROS performs the remaining tasks.

Degraded Mode for Pinning (DMP): BROS only realizes the pinning. It's to the surgeon to insure the rest of the intervention.

Degraded Mode for Blocking (DMB): BROS only blocks the fractured limb. The remaining tasks are manually done by the surgeon.

Basic Mode (BM): The whole intervention is manually performed. BROS provides navigation function using the middleware that checks in real-time the smooth running of the operation.

Table 2 summarizes the operating modes description.

Table 2: BROS's operating modes.

	Reduction	Blocking	Pinning	Unblocking
AM	Robotized	Robotized	Robotized	Robotized
SAM	Manual	Robotized	Robotized	Robotized
DMP	Manual	Manual	Robotized	Robotized
DMB	Manual	Robotized	Manual	Robotized
BM	Manual	Manual	Manual	Manual

3.4 Shared Resources and Issues

BROS is a distributed system composed of several entities: the browser, the control unit, the middleware, the two blocking and reducing arms and the two pinning arms. The said entities may be represented by several sharing resource processes. The most relevant resources in our system are the browser and the patient's arm. The first is solicited by the robotic arms, the MW and the surgeon (when a manual reduction or pining is performed) to update the image on the screen. As to the fractured arm, it may be used by whether the surgeon or the robotic arms.

Applying a reconfiguration scenario on BROS by switching from one operating mode to another may actually lead to a deadlock because of concurrent access to shared resources and which is usually unsafe. This is what happens for example when switching from AM to SAM. To illustrate this

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS

situation, we represent B-BROS1, B-BROS2 and the surgeon by three processes with increasing priorities (B-BROS1 < B-BROS2 < the surgeon). This is due to the fact that human intervention takes precedence over the robotic one, and B-BROS2 has one more function than B-BROS1, which is the fracture reduction.

As illustrated in Figure 4, B-BROS1 starts by locking the patient's arm to block it and frees it once the blocking is achieved (P(A) and V(A) respectively stand for locking and freeing the fractured limb). B-BROS2 locks it to reduce the fracture, and then locks the browser (P(BW) and V(BW) respectively stand for locking and freeing the browser) time to update the image displayed on the screen. Once the blocking is done, B-BROS2 frees the browser. The two robotic arms will successively use the patient's arm to unlock it at the end of the intervention.

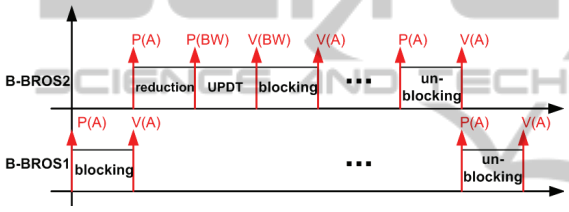

Figure 4: Behaviour of B-BROS1 and B-BROS2 in AM.

When the surgeons judges the fracture reduction performed by B-BROS2 as unsatisfying, he can decide to manually do it, and, thus, switches the system from AM to SAM. However, when trying to use the patient's arm, he finds it already locked by B-BROS2 and a deadlock occurs as illustrated in Figure 5.

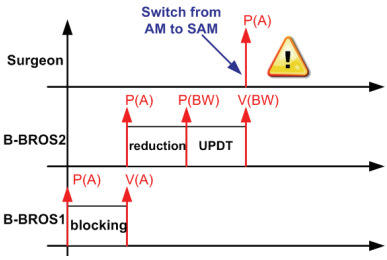

Figure 5: Behavior of the surgeon, B-BROS1 and B-BROS2 in SAM.

This kind of problem due to concurrent access to shared resources after a reconfiguration scenario was treated and solved in a recently accepted work (Salem et al., 2014). The solution is to apply PCP to synchronize sharing resources. Thus, the deadlock in the system is avoided as illustrated in Figure 6.

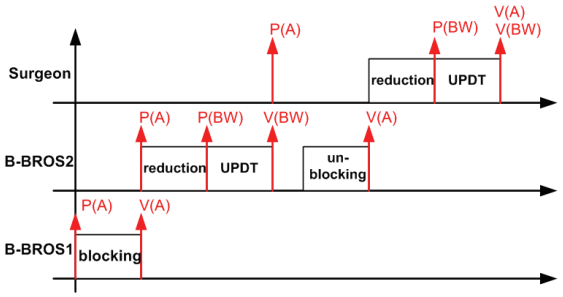

Figure 6: Behavior of the surgeon, B-BROS1 and B-BROS2 in SAM when using PCP.

We see, according to this example, that problems may be faced in reconfigurable control systems, and in this case with BROS, when dealing with concurrent processes that share resources. Thus, we propose to use the contribution made in (Salem et al., 2014) and presented in section 2.2.3 to model and verify reconfigurable tasks and resources to check the safety of any reconfiguration scenario that may be applied on the system.

4 MODELING AND VERIFICATION OF BROS

Since R-TNCES is a useful formalism to model reconfigurable systems, we aim in this section to define BROS's modeling in order to check the system's safety after any reconfiguration scenario.

4.1 Modeling

We continue, in this section, by modeling the BROS using R-TNCES and tasks and shared resources' modeling we previously proposed (in section 2.2.3). Let's remember that, as mentioned in section 3.4, the Surgeon has a higher priority than B-BROS2 (priority(Surgeon)=3 and priority(B-BROS2)=2) and B-BROS2 takes precedence over B-BROS1 whose priority equals to 1. The patient's arm (A) is shared by the three processes, whereas the browser (BW) is shared by B-BROS2 and the Surgeon in this case. Thus, Cl(A)=Cl(BW)=priority(Surgeon)=3. We start by modeling the three tasks as following in Figure 7.

We model in Figure 8 the two shared resources BW and A whose ceilings are equal to 3. Let's remember that BW is shared by two processes (Surgeon and B-BROS2), whilst A is shared by the three.

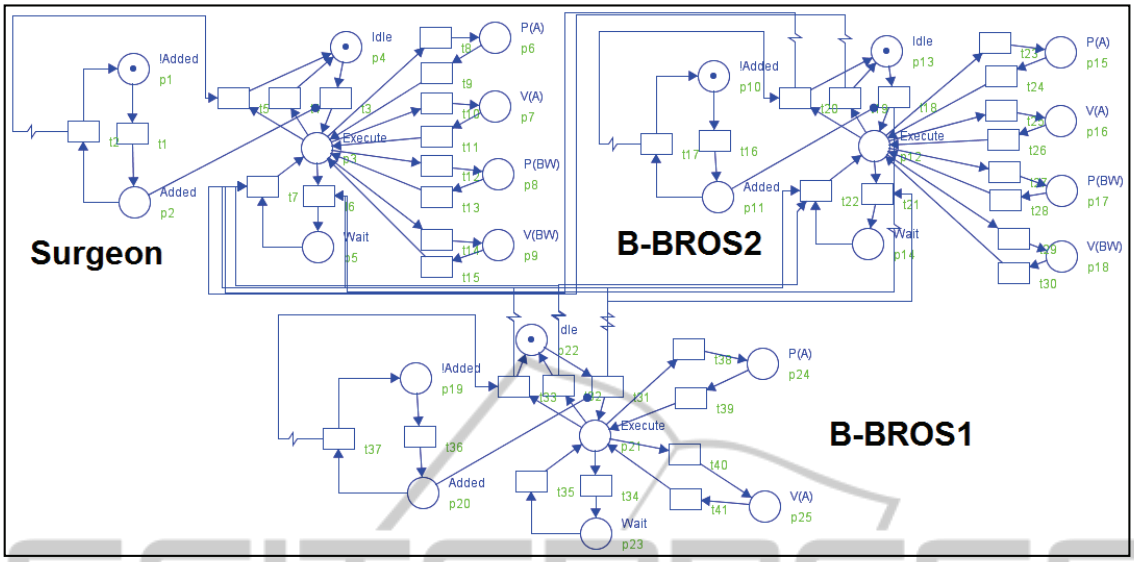

Figure 7: The Surgeon, BROS-1 and BROS2 modeling.

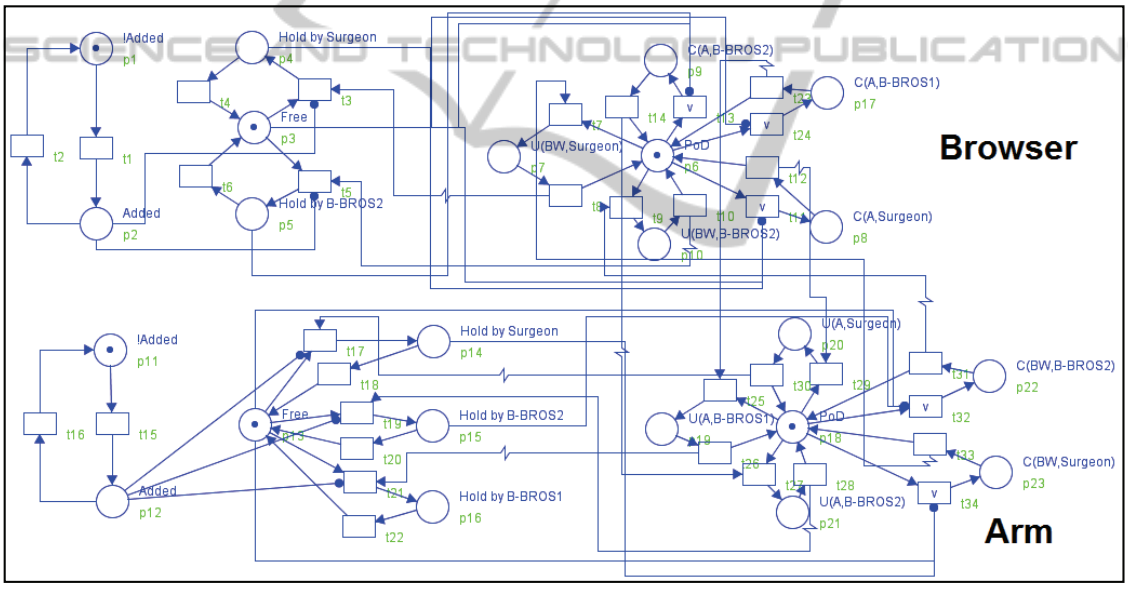

Figure 8: The arm and the browser modeling.

4.2 Verification

Once the R-TNCES model of the DRCS is enriched with PCP, the next step is to verify whether the models meet users' requirements. This means that any reconfiguration scenario dealing with adding/removal of resources does not lead to a blocking situation. Model-checking is a technique for automatically verifying the correctness properties of finite-state systems. Model checking for TNCES and R-TNCES is based on their reachability graphs. SESA (Starke and Roch, 2002) is an effective software environment for the analysis of TNCES, which computes the set of reachable states exactly. Typical properties which can be verified are boundedness of places, liveness of transitions, and reachability of states. In addition, temporal/ functional properties based on Computation Tree Logic (CTL) specified by users can be checked manually. Thus, we check, in this section, the safety of the PCP-based solution to model the concurrent access to reconfigurable shared resources. First, the

following e-CTL formula is applied to check the deadlock-freeness of the system's modeling:

$$AG\ EX\ TRUE \quad (7)$$

This formula is proven to be true by SESA as shown in the screenshot in Figure 9, so there is no deadlocks in our R-TNCES.

```
Parsing formula:
AG EX TRUE

Current model checking options are:
     write a proof
     but only witnesses and counterexamples
     but only for the top level formula
     to the session file
.....................................Reset options? Y/N N
States:         1568
The formula is TRUE.
```

Figure 9: Verification of deadlock-freedom.

We also check the safety property by checking if a given resource may be simultaneously locked by two different tasks. The following CTL formula is checked:

$$EF\ p14\ AND\ p15 \quad (8)$$

where: (i) p14 is the place translating that the resource A is locked by the Surgeon, (ii) p15 means that B-BROS2 locks A. This formula is proven to be false as illustrated in Figure 10.

```
Parsing formula:
(EF p14 AND p15)

Current model checking options are:
     write a proof
     but only witnesses and counterexamples
     but only for the top level formula
     to the session file
.....................................Reset options? Y/N N
States:         6
The formula is FALSE.
```

Figure 10: Verification of the concurrency issue on the patient's arm.

5 NEW ENVIRONMENT: ZIZO

We present in this section the new tool ZiZo and its usefulness in certifying distributed reconfigurable control systems. BROS is the case study.

5.1 Motivations and Originality

ZiZo is a R-TNCES modeling and random-simulating tool written in C# programming language for the Windows platform and developed in LISI laboratory of INSAT and eHTC (Tunisia). Its originality consists in featuring the simulation of a built R-TNCES and highlighting the reconfiguration aspect of a DRCS, which are not offered in any other Petri Nets editor. The main window of ZiZo GUI shown in Figure 11 comprises five dockable frames: Menu Bar, Model Arborescence, Place Properties, the Document Explorer and the Debug Window.

ZiZo is capable of:

- creating several modules within the same model;
- interconnecting modules by input/output condition and event signals;
- randomly simulating the created model to detect any eventual deadlock;
- storing the created model in a special file format (*.pnt);
- loading a created model to edit it and/or simulate it;
- exporting the model to the model-checker SESA.

Since R-TNCES is the more expressive formalism to model adaptive systems because it

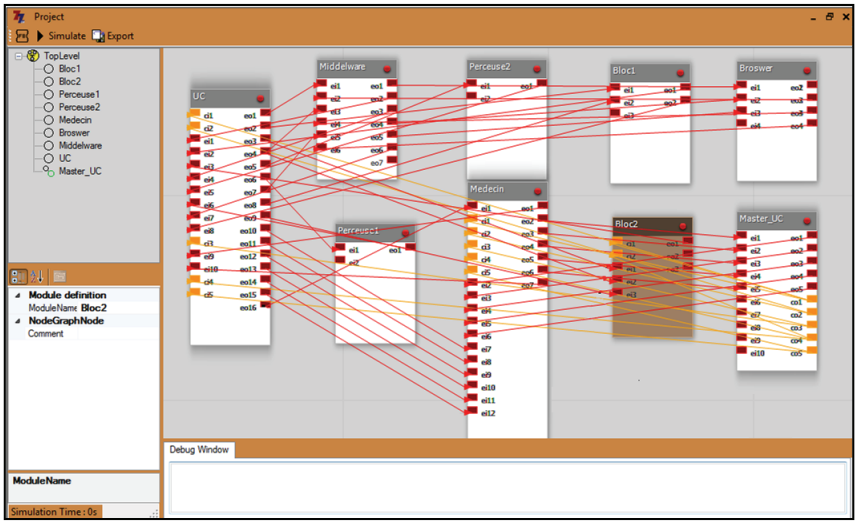

Figure 11: The main window of ZiZo.

addresses all the reconfiguration forms, our community lacks an environment to edit, simulate and check models based on this formalism. The original features of ZiZo are:

- an optimal modeling of reconfigurable distributed real-time tasks sharing adaptive resources;
- simulation of distributed R-TNCES models;
- allows to easily call the model checker SESA for the verification of CTL-based properties.

5.2 Certification of BROS

The purpose of this section is to certify the safety of BROS by checking whether a deadlock may happen at runtime and proving the nonexistence of several potential issues.

a. Simulation

Several researches worked on designing tools to simulate Petri nets-based subtypes. Nevertheless, no one in our community worked on simulating the notion of time and the reconfiguration aspect in Petri nets which are featured by R-TNCES. This makes ZiZo the unique tool offering the ability to model and simulate such formalism. Thus, using ZiZo, we model the whole architecture of BROS with its different modules (UC, MW, BW, B-BROS1, B-BROS2, P-BROS1, P-BROS2 and the surgeon) and shared resources. We obtain an R-TNCES model of 186 places and 283 transitions. Upon definition of the model, ZiZo can simulate it. Simulation can be tracked by selection of a token game. Once simulation is finished, a report is displayed at the debug window.

The obtained report displayed in Figure 12 proves that, after exploring 3057 places by ZiZo, our system is deadlock-free.

```
Simulation finished!
Explored places: 3057
Elapsed time: 1380 seconds
The model is deadlock-free!
```

Figure 12: BROS's simulation report.

b. Verification

After proving in Section 4.2 the non-existence of problems related to concurrent access on BROS's reconfigurable shared resources, we do in this section an exhaustive CTL-based verification to check the existence of several problems that may be faced at BROS's runtime. Thus, we apply several CTL formulas on the model of the whole BROS system, built using ZiZo and then exported to SESA.

Simultaneous Blocking and Pinning: Pinning in the patient's arm while moving it by unblocking it may lead to dramatic consequences. We check, then, whether this two actions may be simultaneously performed by applying the following formula to BROS's model:

$$EF\ p23\ AND\ p43 \quad (9)$$

where: (i) p23 translates unblocking the arm, (ii) p43 pinning it. The formula is found to be false.

Timeout Issue: We check that the whole surgical intervention does not last more than a given definite time. We apply, then, formula 10:

$$EF\ [0,301]\ p23 \quad (10)$$

This formula is also proven to be false.

Intervention Sequence: We have to be sure that BROS complies with the specified logic by performing in order the following actions: reduction, blocking, pinning 1, pinning 2 and unblocking. We apply, therefore, the following CTL formula:

$$AGA\ t18\ X\ AFE\ t25\ X\ AFE\ t40\ X\ AFE\ t74\ X\ AFE\ t111\ X\ TRUE \quad (11)$$

where t18, t25, t40, t74 and t111 are respectively the transitions leading to the places translating reduction, blocking, pinning 1, pinning 2 and unblocking. The formula is proved to be true.

6 CONCLUSION AND PERSPECTIVES

Our work consisted, through this paper, in checking the safety of the surgical robotic system BROS, mainly after different scenarios of addition, removal or update of adaptive shared resources. BROS is a flexible system since it may run under different operating modes: it is reconfigurable. Whence, we chose to model BROS using R-TNCES, the most suitable formalism to model distributed reconfigurable control systems. The concurrent access to shared resources issues were resolved thanks to the PCP-based solution. We simulated BROS's model using our new tool ZiZo to prove the deadlock-freedom and, then, applied several CTL formulas on it which revealed the nonexistence of several issues in BROS. We can now certify that BROS is a safe platform and does not run any risk after any reconfiguration scenario. The next step is to proceed to the real implementation of BROS, using an ABB product, the robotic arm IRB 120 (MIKAELSSON and CURTIS, 2009).

ACKNOWLEDGEMENTS

This research work is carried out within a MOBIDOC PhD thesis of the PASRI program, EU-funded and administered by ANPR (Tunisia). The BROS national project is a collaboration between Tunis Children Hospital of Béchir Hamza (Tunis), eHTC and INSAT (LISI Laboratory) in Tunisia. We thank the medical staff, Prof.Dr.med. Mahmoud SMIDA (Head of Child and Adolescent Orthopedics Service) and Dr.med. Zied JLALIA, for their fruitful collaboration and continuous medical support.

REFERENCES

Kwoh, Y. S., Hou, J., Jonckheere, E. A., and Hayati, S. (1988). A robot with improved absolute positioning accuracy for ct guided stereotactic brain surgery. *Biomedical Engineering, IEEE Transactions on*, 35(2):153–160.

Cleary, K. and Nguyen, C. (2001). State of the art in surgical robotics: clinical applications and technology challenges. *Computer Aided Surgery*, 6(6):312–328.

Wang, Y., Butner, S. E., and Darzi, A. (2006). The developing market for medical robotics. *PROCEEDINGS-IEEE*, 94(9):1763.

Gomes, P. (2011). Surgical robotics: Reviewing the past, analysing the present, imagining the future. *Robotics and Computer-Integrated Manufacturing*, 27(2):261–266.

Zhang, J., Khalgui, M., Li, Z., Mosbahi, O., and Al-Ahmari, A. M. (2013). R-tnces: A novel formalism for reconfigurable discrete event control systems. *Systems, Man, and Cybernetics: Systems, IEEE Transactions on*, 43(4):757–772.

Salem, M. O. B., Mosbahi, O., and Khalgui, M. (2014). Pcp-based solution for resource sharing in reconfigurable timed net condition/event systems. In *ADECS 2014, Proceedings of the 1st International Workshop on Petri Nets for Adaptive Discrete-Event Control Systems, co-located with 35th International Conference on Application and Theory of Petri Nets and Concurrency (Petri Nets 2014), Tunis, Tunisia, June 24, 2014*., pages 52–67.

Hanisch, H.-M., Thieme, J., Luder, A., and Wienhold, O. (1997). Modeling of plc behavior by means of timed net condition/event systems. In *Emerging Technologies and Factory Automation Proceedings, 1997. ETFA '97., 1997 6th International Conference on*, pages 391–396.

Ratzer, A. V., Wells, L., Lassen, H. M., Laursen, M., Qvortrup, J. F., Stissing, M. S., Westergaard, M., Christensen, S., and Jensen, K. (2003). Cpn tools for editing, simulating, and analysing coloured petri nets. In *Applications and Theory of Petri Nets 2003*, pages 450–462. Springer.

Genter, G., Bogdan, S., Kovacic, Z., and Grubisic, I. (2007). Software tool for modeling, simulation and real-time implementation of petri net-based supervisors. In *Control Applications, 2007. CCA 2007. IEEE International Conference on*, pages 664–669. IEEE.

Suender, C., Vyatkin, V., and Zoitl, A. (2011). Formal validation of downtimeless system evolution in embedded automation controllers. *ACM Transactions on Embedded Control Systems*.

Dubinin, V., Hanisch, H., and Karras, S. Building of reachability graph extractions using a graph rewriting system. In *proceedings of the 7th International Conference of Science and Technology, NITis 2006*.

Sha, L., Rajkumar, R., and Lehoczky, J. P. (1990). Priority inheritance protocols: An approach to real-time synchronization. *Computers, IEEE Transactions on*, 39(9):1175–1185.

Goodenough, J. B. and Sha, L. (1988). *The priority ceiling protocol: A method for minimizing the blocking of high priority Ada tasks*, volume 8. ACM.

Brubacher, J. W. and Dodds, S. D. (2008). Pediatric supracondylar fractures of the distal humerus. *Current reviews in musculoskeletal medicine*, 1(3-4):190–196.

Cheng, J. C., Ng, B., Ying, S., and Lam, P. (1999). A 10-year study of the changes in the pattern and treatment of 6,493 fractures. *Journal of Pediatric Orthopaedics*, 19(3):344–350.

Landin, L. A. and Danielsson, L. G. (1986). Elbow fractures in children: an epidemiological analysis of 589 cases. *Acta Orthopaedica*, 57(4):309–312.

Landin, L. A. (1983). Fracture patterns in children: Analysis of 8,682 fractures with special reference to incidence, etiology and secular changes in a swedish urban population 1950-1979. *Acta Orthopaedica*, 54(S202):3–109.

Gosens, T. and Bongers, K. J. (2003). Neurovascular complications and functional outcome in displaced supracondylar fractures of the humerus in children. *Injury*, 34(4):267–273.

Flynn, J. C. (1993). Displaced supracondylar fracture of the humerus in children: Technique of closed reduction and percutaneous pinning. *Operative Techniques in Orthopaedics*, 3(2):121–127.

Flynn, J. C., Matthews, J. G., Benoit, R. L., et al. (1974). Blind pinning of displaced supracondylar fractures of the humerus in children. *J Bone Joint Surg Am*, 56(2):263–72.

Livyatan, H., Yaniv, Z., and Joskowicz, L. (2002). Robus automatic c-arm calibration for fluoroscopy-based navigation: a practical approach. In *Medical Image Computing and Computer-Assisted InterventionMICCAI 2002*, pages 60–68. Springer.

Clein, N. W. (1954). How safe is x-ray and fluoroscopy for the patient andthe doctor? *The Journal of pediatrics*, 45(3):310–315.

Judet, J. (1953). Traitement des fractures sus-condyliennes transversales de l'humérus chez l'enfant. *Rev Chir Orthop*, 39:199–212.

Smida, M., Smaoui, H., Ben Jlila, T., Saeid, W., Safi, H., Ammar, C., Jalel, C., and Ben Ghachem, M. (2007). Un index de stabilité pour l'embrochage percutané latéral parallèle des fractures supracondyliennes du coude chez l'enfant. *Revue de Chirurgie Orthopédique et Réparatrice de l'Appareil Moteur*, 93(4):404.

Starke, P. H. and Roch, S. (2002). *Analysing signal-net systems*. Professoren des Inst. für Informatik.

MIKAELSSON, P. and CURTIS, M. (2009). Portrait-robot d'un petit prodige: Abb présente son nouveau robot irb 120 et son armoire de commande irc5 compact. *Revue ABB*, (4):39–41.

SCITEPRESS
SCIENCE AND TECHNOLOGY PUBLICATIONS