Modeling and Verification of B-based Distributed Reconfigurable Control Systems

Raja Oueslati[1], Olfa Mosbahi[2] Mohamed Khalgui[2] and Samir Ben Ahmed[1]
[1]*Faculty of Sciences of Tunis, University of Tunis El Manar, Tunis, Tunisia*
[2]*LISI, INSAT, University of Carthage, Carthage, Tunisia*

Keywords: Distributed Reconfigurable Control Systems, B Method, Modeling, Formal Verification, Multi-agent Architecture, Coordination.

Abstract: The paper deals with the modeling and verification of B based Distributed Reconfigurable Control Systems (DRCS). A distributed multi-agent architecture is developed, where for each system is affected a Reconfiguration Agent to apply a local automatic reconfiguration, and a Coordination Agent is proposed to harmonize between systems when any local reconfiguration is applied in a system. We apply the Distributed Reconfigurable B "DR-B" formalism to define all possible behaviors, to determine reconfiguration functions for each system and to execute the appropriate ones to respond to reconfiguration requests and to switch between the specific distributed configuration at run-time to cope with the coherence of running systems. We verify a DRCS by using the B method. The goal is to guarantee the consistency and the correctness of the abstract specification level. Further more, to avoid combinatorial explosion problem in DRCS, we apply the "Check R-B" tool, to reduce redundant checking of different behaviors sharing similar operations. All the contributions of this work are applied to two benchmark production systems FESTO and EnAS.

1 INTRODUCTION

Nowadays in industry, the development of safe distributed control systems is not a trivial activity because a failure can be critical for the safety of human being e.g. air and railway traffic control (khalgui et al., 2012). In this context, one of the most important challenges is the trade-off between performance and rapid response to market changes and customer needs. One of the most promising directions, where considerable progress has been made, to address these issues is the reconfiguration of Distributed Control Systems (DCS). We distinguish two types of reconfigurations: static (offline) and dynamic (online) (Angelov et al., 2005). The former is applied offline before system cold starts, whereas the latter is applied automatically at run-time. In the latter case, two types exist : manual reconfigurations to be executed by users and automatic (intelligent) reconfigurations to be performed by intelligent agents that can be a physical resource (robot, machine ...) or a logical resource (scheduler), and hybrid reconfigurations which are the combination of manual and automatic reconfigurations.

To deal with the automatic reconfiguration of distributed industrial control systems following the B method, we propose, in this work a new distributed multi-agent architecture. We define two kinds of agents: Reconfiguration Agent (RA) which is assigned to each system to apply a local automatic reconfiguration and a Coordination Agent (CA) which handles the coherence of distributed concurrent reconfigurations of different systems. A "DR-B" formalism is applied to model distributed reconfigurable control systems. It consists of three modules: Behavior, Control and Coordinator. The first defines all possible behaviors of the system, whereas the second is a set of reconfiguration functions applied to change each system from one configuration to another one at run-time by adding or removing some operations in B machines and the third module coordinates between the different systems when applying a reconfiguration scenario by executing the appropriate reconfiguration functions. This reconfiguration scenario is applied as a response to improve the system′s performance, or also to recover and prevent hardware/software errors, or also to adapt its behavior to new requirements according to the environment evolution. After the modeling of the DRCS, the next step is to check the correctness of the DRCS using B method. In order to avoid combinatorial explosion problem, we apply the "Check R-B" tool that was implemented, in (Oueslati et al., 2014), to reduce redundant checking of different behaviors sharing similar operations. To our knowledge, this is the first contribution dealing with

Oueslati R., Mosbahi O., Khalgui M. and Ben Ahmed S..
Modeling and Verification of B-based Distributed Reconfigurable Control Systems.
DOI: 10.5220/0005244601240131
In *Proceedings of the 5th International Conference on Pervasive and Embedded Computing and Communication Systems* (PECCS-2015), pages 124-131
ISBN: 978-989-758-084-0

the B method to dynamically and automatically reconfigure distributed industrial control systems.

The rest of the paper is organized as follows: in the second Section, we present the background in which we introduce B method. In the third Section, we describe the two benchmark production systems FESTO and EnAS to be followed in the paper as running examples to explain our contribution. We define, in the next Section, the Distributed Reconfigurable B "DR-B" formalism that we apply to our system. The distributed multi-agent architecture is proposed, in the fifth Section. In the sixth Section, we present the "Check R-B" tool for DRCS. We finish by a conclusion and the exposition of our future works.

2 BACKGROUND KNOWLEDGE

We present in this section, the well-known B method.

2.1 Presentation of B

B is a formal method developed by Abrial to support the software development life cycle from specification to implementation (Abrial, 1996). It is based on Zermelo-Fraenkel set theory and on generalized substitution. Sets are used for data modeling, Generalized Substitutions are used to describe state modification, and the refinement calculus is used to relate models at varying abstraction levels. A machine B is composed of header part allowing the identification of the abstract machine, static part defining observations (sets, variables, constants, etc) of the system and their invariant properties and dynamic part describing operations changing the state of the system.

2.2 Composition in B

Abstract machines can be combined, through the clauses INCLUDES, SEES, IMPORTS and USES to build new specifications (Abrial, 1996). We are interested to the clause INCLUDES which allows a machine to be included in another one with read/write access to the variables of the included machine. A machine M includes a machine M1 means that M has a full access to the constants, sets, variables and operations of M1 and operations of M can be defined by using any M1 operations. It is worth mentioning that at most one operation of the included machine can be called from within an operation of the including machine. In order to avoid an obvious clash, we have the possibility to rename a machine while including it. This is done simply by prefixing, in the clause INCLUDES, the name of the machine we want to rename with a certain identifier by a dot $(x.M1, y.M1)$ as explained in Figure 1.

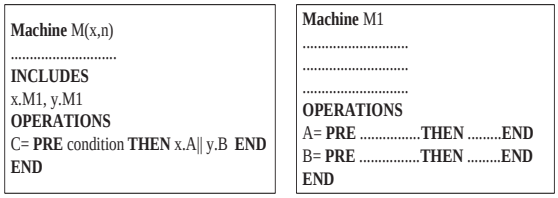

Figure 1: Clause INCLUDES.

3 CASES STUDIES: FESTO AND EnAS

Two benchmark production systems FESTO and EnAS (ref,) are used as intact running examples in this paper, in order to highlight the contributions of our work. They are well documented laboratory systems used by many universities for research and education purposes.

3.1 FESTO System

It consists of three units: Distribution Unit, Test Unit, Processing Unit. The Distribution Unit is formed of a pneumatic feeder and a converter which transmits cylindrical workpieces from a stock to the Test Unit. The Test Unit is composed of a detector, a tester and an elevator. It performs tests on workpieces for height, type of material and color. Workpieces that satisfy these tests are transmitted to the Processing Unit which is composed of a rotating disk, a drill machine and a control machine. The rotating disk is composed of locations to contain and transport workpieces from the input position, to the drilling position, to the control position and finally to the output position. Four production modes are assumed in this paper to be applied in FESTO, depending on the number of workpieces *NP*, as follows:

- **Light1**: If $NP < C1$, Then only *Drill1* is used for drilling workpieces.
- **Light2**: If $NP < C1$, Then only *Drill2* is used for drilling workpieces.
- **Medium**: If $C1 \leq NP < C2$, Then *Drill1* or *Drill2* are used for drilling workpieces.
- **High**: If $NP \geq C2$, Then the two drilling machines are used simultaneously to drill two pieces at the same time.

If both *Drill1* and *Drill2* are broken, the system is completely stopped. We should make FESTO able to switch production modes automatically at run-time

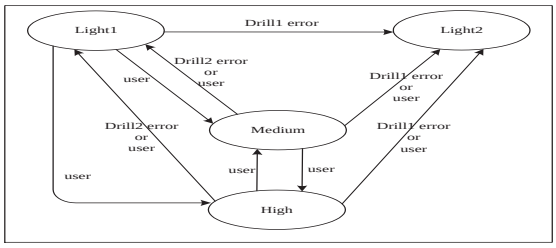

Figure 2: Allowed reconfigurations of FESTO.

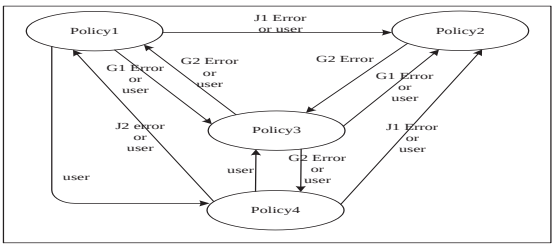

Figure 3: Allowed reconfigurations of EnAS.

according to any changes in the working environment caused by errors or user requirements without a halt. It is assumed that the production modes are interchangeable as shown in Figure 2.

3.2 EnAS System

EnAS transports workpieces from FESTO into storing stations. The workpieces shall be placed inside tins to close with caps afterwards. The EnAS system is mainly composed of a belt, two Jack stations (*J1* and *J2*) and two Gripper stations (*G1* and *G2*). The Jack stations place new drilled workpieces from FESTO and close tins with caps, whereas the Gripper stations remove charged tins from the belt into storing stations (*ST1* and *ST2*). Initially, the belt moves a particular pallet containing a tin and a cap into the first Jack station *J1*. Four production modes are assumed in this paper to be applied in EnAS, depending on the number of drilled workpieces *nbpieces*, tins and caps *nb(tins+caps)*, as follows:

- **Policy1**: If *nbpieces/nb(tins+caps)* $<C1$, Then *J1* places and closes, *G1* removes into *St1*.
- **Policy2**: If *nbpieces/nb(tins+caps)* $<C1$, Then *J1* places, *J2* closes, *G2* removes into *St2*.
- **Policy3**: If $C1 \leq$ *nbpieces/nb(tins+caps)* $<C2$, Then *J1* places and closes, *G2* removes into *St2* or *J1* places, *J2* closes, *G1* removes into *St1*.
- **Policy4**: If *nbpieces/nb(tins+caps)* $\geq C2$, Then *J1* places, *J2* places and closes, *G2* removes the tin (with two pieces) into *St2*.

The system is completely stopped if both *J1* and *J2* are broken. We should make EnAS able to switch policies automatically at run-time according to any changes in working environment caused by errors or user requirements without a halt. It is assumed that policies are interchangeable as shown in Figure 3.

The two systems FESTO and EnAS are linked to coordinate their work. We define in Table 1 the allowed compositions of behavior modes of the two systems. To guarantee the correctness and safeness of the whole system when a local reconfiguration is allowed to be applied in one of them, then the other should have a proper reaction as a response to the planed reconfiguration. The behavior modes Light1, Light2 and Medium of FESTO can cohere with Policy1, Policy2 and Policy3 of EnAS and High of FESTO requires Policy4 of EnAS and vise versa.

Table 1: Allowed coordinations of FESTO and EnAS.

(FESTO, EnAS)	(FESTO,EnAS)
(Light1,Policy1)	(Light2,Policy3)
(Light1,Policy2)	(Medium,Policy3)
(Light1,Policy3)	(Medium,Policy1)
(Light2,Policy1)	(Medium,Policy2)
(Light2,Policy2)	(High,Policy4)

4 PRESENTATION OF DR-B FORMALISM

In this section, we propose the "DR-B" formalism to model DRCS following the B method. A "DR-B" consists of a behavior module which is the union of all system configurations, a control module formed by a set of reconfiguration functions handling automatic transformations between specific configurations in the behavior module and a coordinator module that manages appropriate reconfiguration functions to switch between distributed configuration without any disturbance. For a DRCS, finite behavior modes (configurations) can be performed and the time cost for the reconfiguration of the control systems should be as short as possible to guarantee the instantaneity, the validity, and more importantly the safety. Each configuration model is called a B machine in this paper.

Definition 1. A DCS is composed of *n* systems as follows $DCS = \{sys_1, sys_2, ..., sys_n\}$ and each one can perform behavior modes as follows $sys_1 = \{x, x', ...\}$, $sys_2 = \{y, y', ...\}$ ... and $sys_n = \{z, z', ...\}$. The set of allowed distributed configurations of the *n* systems is defined according to the coherence between the *n* systems behavior modes as follows:

$$SET_{allowedcoordinations} = \{(x, y, ..., z), (x', y', ..., z'), ...\}$$

Example 1. The set of allowed coordinators $SET_{allowedcoordinators}$ of the two systems FESTO and EnAS is presented in Table 1.

Definition 2. A "DR-B" formalism of the *n* systems of DCS is a structure defined as follows:

$$\text{DR-B}=(\cup(\beta_{sys_i}, R_{sys_i}), Coordinator)$$

where:$\cup(\beta_{sys_i}, R_{sys_i})$ is the union of the behavior and control module of each system sys_i and *Coordinator* is a Coordinator module of the DCS.

Definition 3. Behavior Module. The behavior module of a system β_{sys_i} is the union of *m* configurations of sys_i, represented as follows:

$$\beta_{sys_i} = \{M_0, M_1, ..., M_i, ...M_m\}$$

Where:(i) M_0 is the initial B machine corresponding to the first configuration, (ii) M_i is the machine represented by the following tuple:

$$M_i = (C, S, Const, P, V, I, Init, Op)$$

Where:(i) *C*:the system constraints, (ii) *S*:the sets, (iii) *Const*:the constants, (iv) *P*:the properties constants , (v) *V*:the variables, (vi) *I*:the invariants, (vii) *Init*:the initialization of variables and (viii) *Op*:the operations.

Definition 4. Control Module. The control module of a system R_{sys_i} is a set of reconfiguration functions $R_{sys_i} = \{r_i,..., r_m\}$ allowing automatic transformations between configurations. A reconfiguration function of a system $r_{sys_i(x,x')}$ is a structure changing the system from a configuration *x* to another one *x'* defined as follows $r_{sys_i(x,x')} = (Cond_{sys_i(x,x')}, S_{sys_i(x,x')})$, where:(i) $Cond_{sys_i(x,x')} \in \{\text{True, False}\}$:the pre-condition of $r_{sys_i(x,x')}$, (ii) $S_{sys_i(x,x')}$:$(^{\bullet}M) \rightarrow (M^{\bullet})$ is the structure modification instruction where $(^{\bullet}M)$ denotes the machine M_i before the application of $r_{sys_i(x,x')}$ and $(M^{\bullet})$ denotes the target machine M_j after the reconfiguration function $r_{sys_i(x,x')}$ is applied. The structure $S_{sys_i(x,x')}$ models the transformation from a M_i to another M_j machine, when we apply a reconfiguration scenario. If $Cond_{sys_i(x,x')} = True$, $r_{sys_i(x,x')}$ is executable, otherwise it cannot be executed. The structure modification instruction $S_{sys_i(x,x')}$ guides the system transformation from $(^{\bullet}M)$ to $(M^{\bullet})$, including the addition /removal of operations from a source M_i, to obtain a target M_j machine. The pre-condition of a reconfiguration function means specific external instructions and gusty functioning failures.

Definition 5. Coordinator Module. The coordinator module is a set of distributed reconfiguration functions $RD_{sys_i} = \{rd_i,..., rd_m\}$. A distributed reconfiguration function *rd* which allows the system to apply dynamic reconfigurations at run-time from the current distributed configuration $(x, y, ..., z)$ to the target distributed configuration $(x', y', ..., z')$, is a structure described as follows:

$rd_{(x,y,...,z),(x',y',...,z')} = (Cond_{(x,y,...,z),(x',y',...,z')}, (r_{sys_1(x,x')} \wedge r_{sys_2(y,y')} ... \wedge r_{sys_n(z,z')}))$,

where: (i) $Cond_{(x,y,...,z),(x',y',...,z')} \in \{\text{True, False}\}$: the pre-condition of $rd_{(x,y,...,z),(x',y',...,z')}$, (ii) $(r_{sys_1(x,x')} \wedge r_{sys_2(y,y')} ... \wedge r_{sys_n(z,z')})$: the reconfiguration functions of the systems sys_1, sys_2 ... and sys_n, respectively.

$Cond_{(x,y,...,z),(x',y',...,z')}$ is *True* if the system can switch from the current distributed configuration $(x, y, ..., z)$ to the target distributed configuration $(x', y', ..., z')$ then $r_{sys_1(x,x')}$, $r_{sys_2(y,y')}$, ... and $r_{sys_n(z,z')}$ are executable, otherwise they cannot be executed.

Definition 6. A *DRCS=(*$\cup sys_i$*, Coordinator)* where $\cup sys_i$ represents the *n* systems composing DCS and *Coordinator* denotes the coordinator of the appropriate reconfiguration functions of the *n* systems. When a reconfiguration scenario of a running system is allowed, the coordinator should make a decision and provide an optimal solution for all the other running systems in the environment.

5 DISTRIBUTED RECONFIGURABLE B CONTROL SYSTEMS

In this section, we propose a multi-Agent distributed architecture for DRCS following B method. We define two kinds of agents: Coordination Agent (CA) and Reconfiguration Agent (RA). The RA of each system is represented by the control module R_{sys_i} of the "DR-B" formalism. The role of any RA is to apply dynamic reconfigurations on the system. The execution of a reconfiguration changes the system behavior at run-time from a valid configuration to another one according to well-defined conditions to adapt it to its environment. Any uncontrolled automatic reconfiguration applied in a system can lead to critical problems, serious disturbances in others. Therefore, CA is defined to cope with the coordination of the running systems that handle the coherence of distributed reconfigurations between the different RAs. When a reconfiguration scenario is allowed, the coordinator should provide an optimal solution for all the other running systems in the environment such that the safety and the correctness of the whole system are guaranteed all along. In order to manage the coordination between RAs, we define the CA represented by an abstract B machine which maintains safe reconfiguration scenarios that can be applied by the different RAs.

6 APPLICATION TO FESTO AND EnAS

In this section, we apply the proposed formalism to the DCS composed of the two systems FESTO and EnAS in order to explain our contribution. Firstly, we present all the possible configurations of the two systems in order to determine their behavior modules. Secondly, we describe the FESTO and EnAS control modules. Thirdly, we define the coordinator model of the two systems.

6.1 FESTO and EnAS Behavior Modules

According to the fourth production modes, FESTO behavior module β_{FESTO} is composed of eight machines, presented as follows:

$MF1 \triangleq op1; op2; op3; op4$

$MF2 \triangleq op1; op2; op3; op5; op61; op7; op62; op11; op63; op12$

$MF3 \triangleq op1; op2 ; op3; op5; op61; op7$

$MF4 \triangleq op1; op2; op3; op5; op61; op8; op62; op11; op63; op12$

$MF5 \triangleq op1; op2; op3; op5; op61; op9; op62; op11; op63; op12$

$MF6 \triangleq op1; op2; op3; op5; op61; op9$

$MF7 \triangleq op1; op2; op3; op5; op61; op10; op62; op11; op63; op12$

$MF8 \triangleq op1; op2; op3; op5; op61; op10$

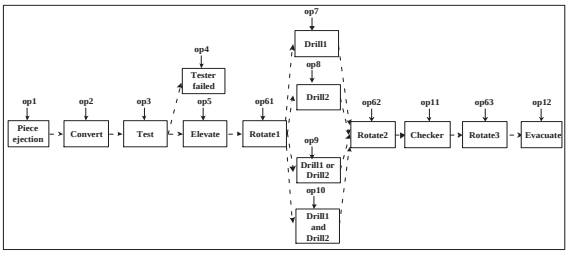

Figure 4: Working process of FESTO.

The default initial production mode Light1 can be described by the combination of **MF1**, **MF2** and **MF3**. In fact, after the execution of *op3*, a workpiece is removed to *op4* or *op5* according to the result of the test unit. Similarly, Light2 is specified by the combination of **MF1** and **MF4**. The combinations of **MF1**, **MF5**, **MF6**, and **MF1**, **MF7**, **MF8** represent respectively the medium and high production modes of the FESTO system. EnAS can perform four types of behavior modes according to the production rate. It's behavior module β_{EnAS} is composed of twelve machines, presented as follows:

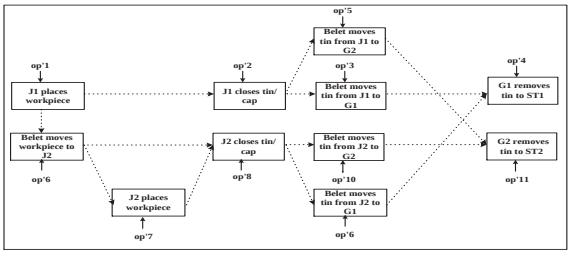

Figure 5: Working process of EnAS.

$ME1 \triangleq op'1 ; op'2 ; op'3 ; op'4$

$ME2 \triangleq op'1 ; op'2$

$ME3 \triangleq op'1 ; op'2 ; op'3$

$ME4 \triangleq op'1 ; op'6 ; op'8 ; op'10 ; op'11$

$ME5 \triangleq op'1 ; op'6 ; op'8 ; op'10$

$ME6 \triangleq op'1 ; op'2 ; op'5 ; op'11$

$ME7 \triangleq op'1 ; op'2 ; op'5$

$ME8 \triangleq op'1 ; op'6 ; op'8 ; op'9; op'4$

$ME9 \triangleq op'1 ; op'6 ; op'8; op'9$

$ME10 \triangleq op'1 ; op'6 ; op'7; op'8 ; op'10; op'11$

$ME11 \triangleq op'1 ; op'6 ; op'7; op'8$

$ME12 \triangleq op'1 ; op'6 ; op'7; op'8 ; op'10$

The default initial production mode Policy1 can be described by the combination of **ME1**, **ME2** and **ME3**. Policy2 is specified by **ME4** and **ME5**. The combinations of **ME6**, **ME7**, **ME8** with **ME9**, and **ME10**, **ME11** with **ME12** represent the Policy3 and Policy4 of the EnAS system, respectively.

6.2 FESTO and EnAS Control Modules

In this section, we describe the RAs of the two systems FESTO and EnAS allowing automatic changes between all the system configurations. In the following, the four behavior modes of FESTO are denoted by 1, 2, 3, and 4 corresponding to Light1, Light2, Medium and High, respectively. The four behavior modes of EnAS are denoted by 1, 2, 3 and 4 corresponding to Policy1, Policy2, Policy3 and Policy4, respectively. As shown in Figure 2, the RA of FESTO is represented as follows:

$$R_{FESTO} = \{ r_{FESTO(1,2)} , r_{FESTO(1,3)}, r_{FESTO(1,4)}, r_{FESTO(3,1)} , r_{FESTO(3,2)} , r_{FESTO(3,4)}, r_{FESTO(4,1)} , r_{FESTO(4,2)}, r_{FESTO(4,3)} \}$$

Let us assume that FESTO is in *Light1* production mode when the user requests to change the production to *Medium*. If $Cond_{FESTO(1,3)} = true$, then the

reconfiguration function $r_{FESTO(1,3)}$ is executed automatically to respond to this request. To implement $r_{FESTO(1,3)}$, we execute the structure modification instruction $S_{FESTO(1,3)}$ including the removal of the operation *op7* and the addition of the operation *op9*. The $S_{FESTO(1,3)}$ is presented as follows:

$$S_{FESTO(1,3)} : MF2 \rightarrow MF5$$

After, $S_{FESTO(1,3)}$ is executed, *Drill2* or *Drill1* is used to drill workpieces. FESTO continues to work in the Medium mode. We define in the following, the FESTO controller B machine using the clause INCLUDES for calling the needed FESTO machines, as follows:

```
MACHINE FESTO_Controller_machine(........)
CONSTRAINTS ..............................
INCLUDES
    a1.MF2(.........),a2.MF2(.........), ....
    d1.MF5(.........),d2.MF5(.........),.....
SETS
REQ_FESTO_USER= {No_Req_FESTO, L1, L2, M, H};
VARIABLES req_festo_user
INVARIANT REQ_FESTO_USER: req_festo_user
INITIALISATION req_festo_user:= No_Req_FESTO
OPERATIONS
    MF2_to_MF5= SELECT ...................
        THEN ANY ...WHERE ................
        THEN a1.eject_piece (.........)||
             a2.convert(.........)||
             a3.test_unit (.........) ||
             a4.To_processing_unit (....) ||
             a5.rotate1(.........) ||
             d1.Drill(.........) ||
             a7.rotate2(.........) ||
             a8.Check (.........) ||
             a9.rotate3(.........) ||
             a10.Remove(.........)
        END END; .........................
```

Where: *a1.MF2* (resp. *d1.MF5*) represents the instance of the MF2 machine (resp. the instance of MF5 machine). For example, *a2.convert* (resp. *d1.Drill*) means the call of the operation *convert* (resp. *Drill*) from the instance of MF2 (resp. *MF5*).

There are ten different reconfiguration scenarios that can be applied to EnAS as shown in Figure 3. The control module of EnAS is represented as follows:

$$R_{EnAS}=\{ r_{EnAS(1,2)} , r_{EnAS(1,3)}, r_{EnAS(1,4)}, r_{EnAS(2,3)} , r_{EnAS(3,1)} , r_{EnAS(3,2)}, r_{EnAS(3,4)} , r_{EnAS(4,1)}, r_{EnAS(4,2)}, r_{EnAS(4,3)} \}$$

Let us assume that EnAS is in *Policy1* production mode when the user requests to change the production to *Policy3*. If $Cond_{EnAS(1,3)}$ *=true*, then the reconfiguration function $r_{EnAS(1,3)}$ is executed automatically to respond to this request. To implement $r_{EnAS(1,3)}$, we execute the structure modification instruction $S_{EnAS(1,3)}$ including removing operations *op′3; op′4* and adding operations *op′5; op′11*. The $S_{EnAS(1,3)}$ is presented as follows:

$$S_{EnAS(1,3)} : ME1 \rightarrow ME6$$

After, $S_{EnAS(1,3)}$ is executed, *J1* and *G2* are used to store drilled workpieces. EnAS continues to work in Policy3 mode. We define in the following, the EnAS controller B machine using the clause INCLUDES for calling the needed EnAS machines, as follows:

```
MACHINE EnAS_Controller_Machine(.........)
CONSTRAINTS ..............................
INCLUDES
    h1.ME1(...........),h2.ME1(...........),...
    n3.ME6(...........),n4.ME6(...........),...
SETS
    REQ_USER_ENAS={No_Req_EnAS, P1, P2, P3, P4}
VARIABLES req_enas_user
INVARIANT req_enas_user : REQ_USER_ENAS
INITIALISATION req_enas_user:= No_Req_EnAS
OPERATIONS
     ME1_to_ME6= SELECT ...............
     THEN ANY ......WHERE.............. THEN
     h1.place1(...) ||
     h2.close1(.....)||
     n3.move4(......) ||
     n4.remove2(.......)
 END END; ........................
```

Where: *h1.ME1* (resp. *n1.ME6*) represents the instance of the *ME1* machine (resp. the instance of *ME6* machine). For example, *h1.place1* means the call of the operation *place1* from the instance of *ME1* and *n3.move4* means the call of the operation *move4* from the instance of *ME6*.

6.3 Coordinator Module

In this section, we define the CA of the two systems FESTO and EnAS that executes appropriate reconfiguration functions of RAs to switch between distributed configuration without any disturbance to respond to reconfiguration requests. According to Figure 2 and Figure 3, a state machine is defined as shown in Figure 6, where each state corresponds to a specific distributed configuration and each transition means the system reconfiguration from a distributed configuration to another. In the following, the vector (4,3) means that FESTO is in the High production mode while EnAS is in Policy3 production mode.

There are 58 different reconfiguration scenarios that can be applied to FESTO and EnAS to respond to user requests or occurred fault. The Coordinator Module is represented as follows:

$$RD_{Coordinator}=\{ rd_{(1,1),(1,2)}, rd_{(1,1),(1,3)}, ..., rd_{(3,3),(1,1)}, rd_{(3,3),(1,2)} , rd_{(3,3),(1,3)},..., rd_{(4,3),(3,1)}, rd_{(4,3),(3,2)}, rd_{(4,3),(3,3)} \}$$

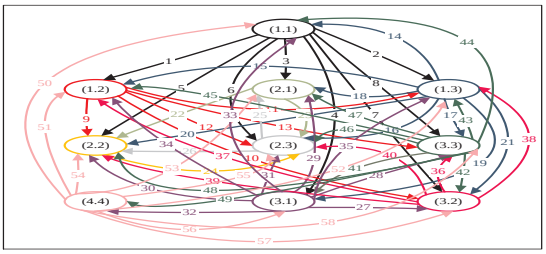

Figure 6: Specification of the Coordinator of FESTO and EnAS.

Let us assume that FESTO is in Light1 when the user requests to change to Medium and EnAS is in Policy1 when the user requests to change Policy3. If $Cond_{(1,1),(3,3)}$= True then the reconfiguration functions $r_{FESTO(1,3)}$ and $r_{EnAS(1,3)}$ are executed automatically to respond to the two requests. We define in the following the CA represented by a B machine including FESTO and EnAS controller machines and taking into account the allowed coordinations of the two systems. Therefore, we use the clause INCLUDES calling the two RAs of DCS defined by *FESTO_Controller_Machine* and *EnAS_Controller_Machine*. The Coordinator machine is as follows:

```
MACHINE Coordinator_Machine
INCLUDES FESTO_Controller_Machine(.........)
         EnAS_Controller_Machine(.........)
OPERATIONS
    L1toM_P1toP3= SELECT
    req_festo_user=M &req_enas_user=P3
    THEN    MF2_to_MF5 || ME1_to_ME6
 END END; ......................
```

Where *MF2_to_MF5* (resp. *ME1_to_E6*) represents the call of the operation of the included machine *FESTO_Controller_Machine* (resp. *EnAS_Controller_Machine*) that switches the system from Light2 to Medium (resp. from Policy1 to Policy3).

The proof obligations of B machines were proved by the B4free prover and all invariants were preserved by operations.

7 VERIFICATION OF DRCS

Once a DRCS model is well established, the next step is the optimal verification to avoid redundant calculation. In (Oueslati et al., 2014), an optimal verification algorithm was developed and a prototyped tool called "Check R-B" is implemented to solve the redundancy problem of the operations and to validate B machines, we can consider it as a module that can be added to B4free. The main idea is to identify for a given distributed configuration, the operations that should be checked. An operation should be checked only once by the B4free prover. So, from one distributed configuration to another, only the new operations should be verified and also old ones that did not respect precedence relationship between them. In this section, we use the same tool to simulate the verification process of DRCS.

Example 2 . Verification of DRCS

As shown in Figure 7, from the distributed configuration (1,1) to (2,3) (resp. (3,3)), only *op8* and *op'5;op'11* (resp. *op9* and *op'5; op'11*) need to be verified, the same operations have not to be checked again. Further more, from the distributed configuration (1,1) to (2,2) (resp. (3,2)), only operations *op8* and *op′6 ; op′8 ; op′10 ; op′11* (resp. *op9* and *op′6 ; op′8 ; op′10 ; op′11*) need to be verified, since the others are similar.

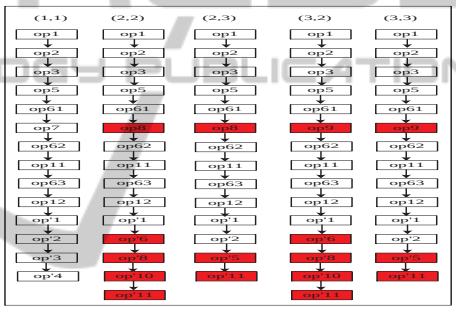

Figure 7: Verification of DRCS:FESTO and EnAS.

Example 3 . Simulation of the configuration (3,3)

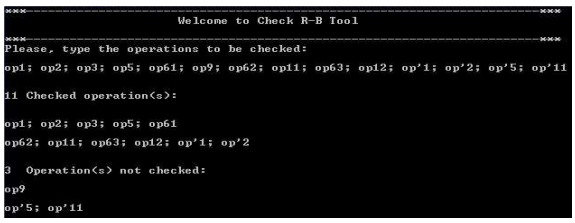

Figure 8: Simulation of the configuration (3,3).

Let us assume that the user introduces operations corresponding to the execution of the distributed configuration (3,3). Let us assume that the tool has verified firstly the distributed configuration (1,1), so a search in a file containing checked machines will be done. If a sequence of operations with precedence relationship already exists, it is not necessary to check it again. Otherwise, it will be forwarded to the prover. As shown in Figure 8, the sequence of operations *(op1; op2; op3; op5; op61)* and *(op62; op11; op63; op12; op'1; op'2)* have already been checked and the operations *(op9)* and *(op′5 ; op′11)* have to be verified.

Figure 9 shows two curves corresponding to the comparison between verification process with and without using “Check R-B”. The values of the abscises axis correspond to the distributed configurations when the system runs two times ((1,1), (2,2), (2,3), (3,2), (1,2), (3,3), (4,4), (3,1), (1,3), (2,1)) in order. The ordinate axis correspond to the number of checked operations. The curve in blue corresponds to the verification without “Check R-B”. The curve in pink corresponds to the optimal verification using “Check R-B”. It is important to note that the number of checked operations decreases gradually until the value zero when we use the *Check R-B* tool as compared to a direct verification without *Check R-B*.

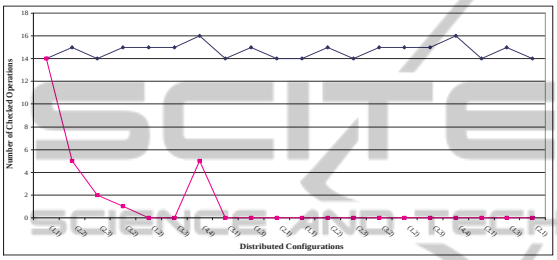

Figure 9: Comparison between verification process with and without using “Check R-B”.

8 CONCLUSION

This paper deals with the modeling and verification of distributed multi-agent reconfigurable control systems following the B method. We affect to each subsystem a CA to apply local automatic reconfigurations, and a CA for any coordination between systems in order to guarantee safe and adequate distributed reconfigurations. We propose a “DR-B” formalism to model DRCS. Further more, to reduce redundant checking of different behaviors sharing similar operations, we applied the “Check R-B” tool.

Different directions can be mentioned as further work. First of all, we plan to generate the C code for each developed B machine. We plan also to develop a graphical tool that allows the efficient modeling and verification of reconfigurable systems with “DR-B”.

REFERENCES

http://aut.informatik.unihalle.de/forschung/testbed/.

Abrial, J.-R. (1996). *The B-Book*. Cambridge University Press.

Angelov, C., Sierszecki, K., and Marian, N. (2005). Design models for reusable and reconfigurable state machines. volume 3824, pages 152–163. 3th International Conference on Embedded and Ubiquitous Computing.

khalgui, M., Mosbahi, O., Hanisch, H., and Li, Z. (2012). A multi-agent architectural solution for coherent distributed reconfigurations of function blocks. *Journal of Intelligent Manufacturing*, 23:2531–2549.

Oueslati, R., Mosbahi, O., Khalgui, M., and Ben Ahmed, S. (2014). New solutions for modeling and verification of b-based reconfigurable control systems. pages 749–757. 11th International Conference on Informatics in Control, Automation and Robotics.