Formalization and Verification of Reconfigurable Discrete-event System using Model Driven Engineering and Isabelle/HOL

Sohaib Soualah[1,4][a], Yousra Hafidi[1,2,4][b], Mohamed Khalgui[1][c], Allaoua Chaoui[3][d]
and Laid Kahloul[2]

[1]*LISI Laboratory, National Institute of Applied Sciences and Technology, University of Carthage, Tunis 1080, Tunisia*

[2]*LINFI Laboratory, Computer Science Department, Biskra University, Algeria*

[3]*MISC Laboratory, Faculty of NTIC, University Constantine 2 - Abdelhamid Mehri, Constantine, Algeria*

[4]*University of Tunis El Manar, Tunis, Tunisia*

Keywords: Reconfigurable Discrete-event Systems, Model Driven Engineering, Model Transformation, Meta-model Formal Verification, Theorem Prover, Isabelle/HOL.

Abstract: This paper deals with the modelling and verification of reconfigurable discrete event systems using model driven engineering (MDE) and Isabelle/HOL. MDE is a software development methodology followed by engineers. Isabelle/HOL is an interactive/automated theorem prover that combines the functional programming paradigm with high order logic (HOL), which makes it efficient for developing solid formalizations. We are interested in combining these two complementary technologies by mapping elements of MDE into Isabelle/HOL. In this paper, we present a transformation process from Ecore models, to functional data structures, used in proof assistants. This transformation method is based on Model-driven engineering and defined by a set of transformation rules that are described using formal presentations. Furthermore, in order to avoid redundant computations in RDESs, we propose a new algorithm for improved verification. We implement the contributions of this paper using Eclipse environment and Isabelle tool. Finally, we illustrate the proposed approach through FESTO MPS case study.

1 INTRODUCTION

The development of safe systems in industry is considered as an important task because a failure can be critical according to a domain for example: air and railway traffic control (Khalgui et al., 2012), manufacturing systems (Khalgui et al., 2010), real time systems and intelligent control systems (Khalgui et al., 2010). In this context, the main objective of a system is to answer the compromise flexibility vs performance (Hafidi et al., 2019), which means that new developed systems guarantee performance by giving response to customer's needs. Many existing works have been proposed in this perspective of flexibility, which give as result new types of systems. A class of these systems is that of reconfigurable discrete-event systems (RDESs) which are characterized by their discrete nature and their changeable structures. RDESs are affected by their internal as well as external events. An RDES is defined as a hardware or software automation system capable of modifying its internal structure to adapt its answers to its environment changes (Khalgui et al., 2019). We distinguish between two kinds of reconfigurations: static and dynamic (Zhang et al., 2017). The former is applied offline before running the system. The latter is applied automatically at run-time without any interruption. Dynamic reconfigurations can be executed: (1) manually by users, (2) automatically by agents (robot, machine, schedule, etc.), and (3) in a hybrid way which is the combination of manual and automatic reconfigurations. To deal with the safety of reconfigurable discrete event systems, researchers are following many verification approaches. Model checking (Clarke et al., 2018) is one of the most used solutions to validate systems. It presents an automatic verification technique to check functional properties. Model-checking uses mathematical methods to ver-

[a] https://orcid.org/0000-0002-5162-5989
[b] https://orcid.org/0000-0002-3543-6731
[c] https://orcid.org/0000-0001-6311-3588
[d] https://orcid.org/0000-0003-3751-8084

Soualah, S., Hafidi, Y., Khalgui, M., Chaoui, A. and Kahloul, L.
Formalization and Verification of Reconfigurable Discrete-event System using Model Driven Engineering and Isabelle/HOL.
DOI: 10.5220/0009893602500259
In *Proceedings of the 15th International Conference on Software Technologies (ICSOFT 2020)*, pages 250-259
ISBN: 978-989-758-443-5

ify if a property is satisfied in a given system model. If the property is violated, a counter example of the system execution is provided. Authors in (Guellouz et al., 2016) propose an extension to the IEC 61499 (Lewis, 2001) standard called Reconfigurable Function Block, encapsulating several reconfiguration scenarios in one function block. In order to verify the system and to evaluate its performance, authors model it using a class of Petri nets called GR-TNCES (Khlifi et al., 2015). After that, PRISM is used as a model checker to verify the safety of each reconfiguration scenario of the system. In (Zhang et al., 2013), authors propose a new extension of TNCES formalism named reconfigurable net condition/event systems (R-TNCESs). This last allows to deal with reconfiguration and time properties with modular specification in the same formalism. In (Hafidi et al., 2018), a new methodology for formal verification of reconfigurable discrete event control systems (RDECSs) is proposed in order to ensure the correctness of systems. The proposed contribution includes an improved modeling and verification of RDECSs. The main idea is based on the checking of reconfiguration scenarios (inter-verification) and also the checking of the internal behavior of each configuration (intra-verification). All these research works present significant results regarding the verification task of RDESs. However, there has been a luck of researches about the optimization of the verification task considering analysed properties. Actually, the complexity of model checking depends on two parameters: the size of the model, and the number of properties to be verified. For instance, Bounded Model Checking (BMC) is based on a reduction of model checking to satisfiability formulae (Jiang et al., 2016). We propose, in this work a new methodology for the formalization and verification of RDESs using theorem proving Isabelle/HOL to overcome model checking limits. Using such a theorem proving has several advantages. First, it gives a certificate to formal proof when it succeeds. Second, when the verification of the given property fails, it generates a counterexample as a proof to the formula negation, instead of a sequence of states or trees labeled with states, as in traditional model checkers. To the author's best knowledge, this is the first contribution addressing this problem. This paper presents the following contributions:

- We define a Meta-Model to model RDESs using MDE. Which is part of the evolution by advocating the systematic use of models to facilitate understanding of a complex system and to automate some of the development processes followed by engineers.
- We propose a formalisation of RDESs in Isabelle, which is equivalent to this Meta-model.
- We establish the link between MDE and Isabelle by defining reconfiguration rules to allow automatic generation of system in Isabelle.
- In order to avoid redundant computations, we propose a new algorithm for improved verification.

The remainder of this paper is organized as follows, Section II presents background about Model Driven Engineering (MDE), and Isabelle/HOL. Sections III and IV involve details about the proposed approach. Section V presents the new Algorithm of improved verification. Section VI describes an application of proposed contributions on a real case study: FESTO benchmark system. Section VII illustrates performance evaluation of the suggested approach. Finally, Section VIII concludes this paper and highlights some perspectives of the work.

2 BACKGROUND

In this section, we present details about Isabelle/HOL theorem proving, and Model Driven Engineering.

2.1 Isabelle/HOL

Isabelle/HOL is an interactive/automated theorem prover that combines the functional programming paradigm with high order logic (HOL), which makes it efficient for developing solid formalizations (Meghzili et al., 2017). Using Isabelle/HOL, we can formalize a system and prove its properties (i.e., formalize systems, formulating lemmas and theorems on them) (Ali et al., 2007). Isabelle/HOL has a high degree of credibility for created proofs because it allows us to prove every step, and therefore the whole proof is correct. Isabelle has several methods, to describe data structures. In the following, we show the main Isabelle concepts used in this paper.

- The theory: The main concept enveloping all elements used to write a program in Isabelle/HOL.
- Types bool, nat and list: These are the most important predefined types. Although the lists are already predefined, and can define their own type.
- Types synonym: Synonym types are abbreviations for existing types.
- Function: In most cases, defining a recursive function is as simple as other definitions.
- Record: A record in Isabelle is an element enveloping more than one type, to define another type.
- Lemma: is used to prove a function or properties.

2.2 Model Driven Engineering

Model Driven Engineering (MDE) is a software development methodology followed by engineers, where meta-models are the central elements. A meta-model precisely defines concepts handled in the models as well as the relationships between these concepts. A model is a description, a specification of a system. A model transformation describes the switch from a source model to a target one. MDE is a general and open approach following the proposal of the MDA (Model Driven Architecture) standard proposed by the OMG in 2000 (Djeddai et al., 2012). We apply this method to describe, specify systems, then to define a generic transformation processe from Ecore models (As detailed in the next subsection) to Isabelle/HOL. Figure 1 shows an overview of our transformation. For the translation, we propose Ecore meta-model of the system. This meta-model is the constructor of source model of our transformation. We also define Isabelle/HOL meta-model to be the constructor of target model. To perform this transformation, we define a set of transformation rules (detailed in Section. 4) that maps components of the instance of Ecore meta-model to those instances of Isabelle/HOL meta-models.

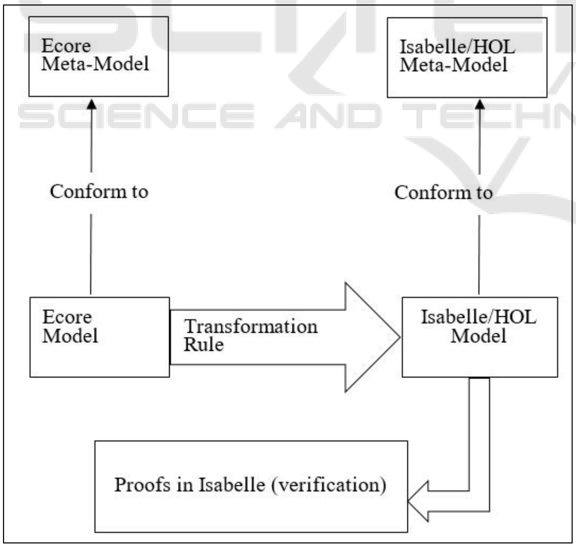

Figure 1: An overview of our transformation.

- **Ecore Meta-model**

In this paper we use a subset of the Ecore meta-model. This subset essentially contains the elements we needed for translation from Ecore model to Isabelle/HOL. It is important to note that this subset of Ecore allows us to define basic models validated by Ecore. A subset of the Ecore meta-model consists of:

- The EPackage gathers all Eclasses and Edatatypes via EClassifers. It is the root element of the Ecore models.
- The EClass is the element that represents the UML class in Ecore. Eclasses define the structure of the objects that make up the instances of the model. It contains EAttributes and EOperations.
- EReferences represent an entire/partial relationship called ≪value aggregation≫ in UML.

3 METHODOLOGY

In this section, we propose the "Meta-Model" to model RDES before transforming it in Isabelle/HOL. A "Meta-Model" consists of the elements and relations used to describe: (1) behavior of system which is all system configurations, and (2) Reconfigurations rules allowing automatic transformations between configurations.

3.1 ECORE Meta-model

Definition 1: An RDES is composed of n Units as follows: $RDES = Unit1, Unit2, ..., Unit\ n$ and each subset can perform behavior modes as follows $RDESmode$= ($mode_1$: $Unit1, \ldots$ $Unit\ i$),($mode_2$: $Unit2, \ldots Unit\ j$),...,($mode_n$: $Unit\ i, \ldots\ \ldots\ .Unit\ jj$). The set of allowed configurations of the RDES is defined according to the communications between the $n\ units$. Using reconfiguration rules switching automatic between configurations.
Definition 2: A RDES is a structure defined as follows: $RDES = (B, RR)$ where: B is the behavior and RR reconfiguration rules of system.
Definition 3: RDES Behavior. The behavior of a system B is the union of m configurations, represented as follows: $B = Conf_0, Conf_1, Conf_2, ..., Conf_i, ...Conf_m$ Where:(1) $conf_0$ is the initial configuration, (2) $Conf_i$ represented by the following tuple:

$Conf_i = (U, L)$

Where:(1) U: the set of units, (2) L: the set of links between units.
Definition 4: RDES Reconfigurations Rules. The reconfigurations rules of a system RR is a set of transformations between configurations $RR = r_i, ..., r_m$ allowing automatic transformations between configurations. A reconfiguration rule of a RDES r_i ($Conf$, $Conf'$) is a structure changing the system from a configuration $Conf$ to another one $Conf'$ defined as follows r_i ($Conf$, $Conf'$) = ($Condition$, $Operation$, $S-Conf$, $D-Conf$), where: (1) $Condition$ {True,

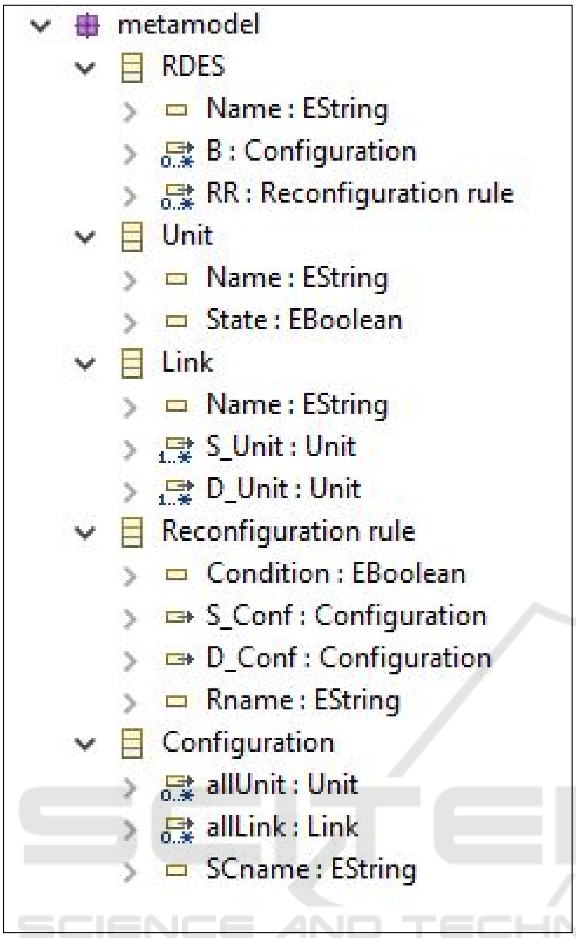

Figure 2: RDES Ecore Meta Model.

False}: the pr-condition of r_i, (2) *Operation* is including the addition/removal of units and links from a source $Conf_i$, to obtain a target $Conf_j$ configuration, (3) $S-Conf$ denotes the configuration $Conf_i$ before the application of r_i and (4) $D-Conf$ denotes the target configuration $Conf_j$ after the reconfiguration rule r_i is applied. The reconfiguration rule r_i for the transformation from a $Conf_i$ to another $Conf_j$ configuration, when we apply a reconfiguration scenario. If $Condition = True$, r_i is executable, otherwise it cannot be executed. The transformation from $Conf_i$ to $Conf_j$. Figure 2 shows RDES Ecore Meta Model.

3.2 Isabelle/HOL Meta-model

An RDES consists of: a set of reconfiguration rules to switch from a configuration to another. A configuration is a stable situation that has a certain duration in which a system performs an activity, i.e., system's components are in a specific communication with each other. We denote by *SC*: a configuration, *RR*: a reconfiguration rule, and *Unit*: a component of the system. We propose the Isabelle formalization named *Isa_System* shown in Figure 3. First, we define *SC* using records. *SC* contains: *SCname* is the name of the *SC*, *allUnit* is the set of the units composing the configuration (list of the type *UNIT*), and *allArc* is a list type *Arc* which represents all possible links between units. Type *Arc* is a record contains: *Aname* is the name of the *Arc*, $S-unit$ represents the source unit and $D-unit$ represents the destination unit, and Simple *UNIT* represents action in line operation mode. The type *UNIT* is a record consisting of: *string* the name of the unit and *boolean* represents the state of unit. Reconfiguration rules *RR* is defined by the record *RR*. A reconfiguration rule consists of: the name of the rule, *Pre_Conf* is the precedent configuration, *Next_Conf* is the next configuration, and *c* is the condition and *Op* is the sting represent a name of function apply to perform de next configuration. Finally, we define *FESTO MPS* using *Isa_system* record. A FESTO MPS consists of: *name* of FESTO MPS, a set of *RR* type (reconfiguration rules), and a set of *SC* type (system configurations).

```
record UNIT=
Uname:: "string"
state :: "bool"
record Arc=
Aname:: "string"
S unit :: "UNIT"
D unit :: "UNIT"
record SC=
SCname::"string"
allUnit::"UNIT list"
allArc:: "Arc list"
record RR=
RRname::"string"
Pre_Conf ::" SC"
Next_Conf ::" SC"
c:: "string"
Op:: "string"
record Isa System=
Sname:: "string"
RRs::" RR list"
SCs::" SC list"
```

Figure 3: Isabelle formalization Isa_System.

4 TRANSFORMATION FROM ECORE MODEL TO ISABELLE/HOL

In this section, we present the translation: from RDES models into data structures used in Isabelle/HOL. We detail one by one the different transformation rules. Table 1 shows correspondences between ECORE and Isabelle/HOL elements.

Table 1: Correspondences between Ecore and Isabelle/HOL elements.

Ecore	Isabelle/HOL
U={$Unit$ i, ..., $Unit$ j}	record $UNIT$
L={$Link$ ii, ..., $Link$ jj}	record Arc
B= {$Conf_0$, $Conf_1$, $Conf_2$, ..., $Conf_i$, ..., $Conf_n$}	record SC
RR = {r_i,...,r_m }	record RR
RDES Ecore model	record $IsaSystem$

- **Transformation Process**

The transformation consists of a set of transformation rules. Each rule has the form:

Tr : $Ecore$ $model$ $\rightarrow$ $Isabelle$ $Types$.

The first rule is Instance Epackage To Theory. This rule triggers recursively other transformation rules.

- **Rule Instance EPackage To Theory**

The components of the Ecore models are grouped in EPackages. When we transform Ecore models into isabelle, we transform these packages into theories (Theory). The name of an EPackage gives the name of the Theory. Additional elements nsPrefix and nsURI are specific characteristics of Ecore. They are neither translated nor used in Isabelle. We call the rule by element ($TrMclass()$,$TrMAttribute()$, $TrMReference()$, $TrMgroup()$) to translate type instances Eclassifiers ($Eclass$, $Ereference$, $EAttribute$) contained in the Epackage instance.

Tr (instance EPackage $name$ = p
{ECl_1 ... ECl_n } = create Theory ();
set Name(p);
$TrMcl$ (ECl_i); / $0 \preceq i \preceq n$

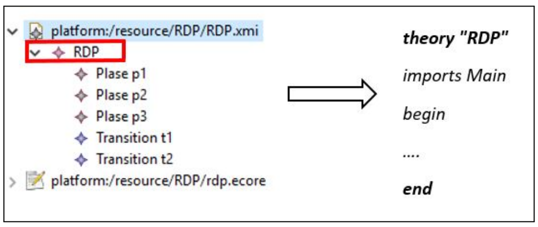

Figure 4: Rule Instance EPackage To Theory.

- **Rule Instance EAttribute to Definition**

If an EAttribute instance is formed of a primitive type(int, bool, string), the transformation generates a new definition. The name and value of this definition are the same of EAttribute instance.

$TrMAttribute$ (instance EAttribute $name$ = a , $value$) =
create Type Definition () ;
set Name(a);
set Value(value)

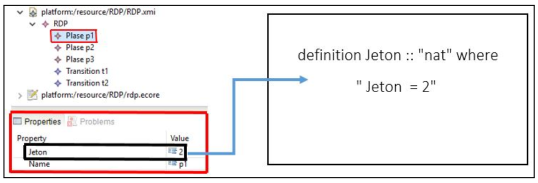

Figure 5: Rule Instance EAttribute To Definition.

- **Rule Instance EClass to Definition**

The simplest case that we can face is how to transform an EClass instance that is independent (i.e., not linked with other EClasses). In this case, the EClass is translated into a definition. The name EClass gives the name of definition constructor. Then, for each instance EStructural feature contained in the instance EClass, we call the appropriate rule: $TrMsf()$ to transform the instances of the Structural Feature x in ($EAttribute$, $EReference$) type. $TrMsf(x)$ is the function that takes as input x and transforms it to Isabelle definition.

$TrMclass$ (instance EClass $name$ = c)
{ESf_i ... $_E Sf_n$ } = if (super Type = vide)
Create Definition();
set Name(c);
$TrMsf$ (ESf_i);/ $0 \preceq i \preceq n$

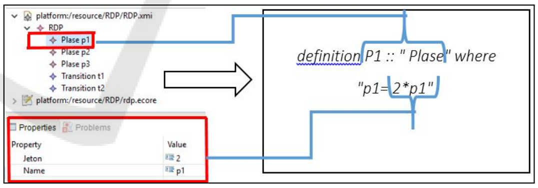

Figure 6: Rule Instance EClass To Definition.

- **Rule Instance EReference to Definition**

In this case we use the same instructions as in the previous rule (Rule Instance EClass to definition), with a simple modification, where the definition constructor name will take the name of EReference Type.

$TrMReference$ (instance EReference $name$ = c
{ESf_1 ... ESf_n}, $eType$ =
Create Definition();
set Name(c);
$TrMsf$ (ESf_i);/ $0 \preceq i \preceq n$

- **The Regrouping Rule**

This rule is made to regroup EReferences of the same type in a single list. Therefore, we need to create a list with the name of EReference, then we add all the definitions of this EReference.

$TrMgroup()$=

List L=Create Liste();

if ($containment = true$ or $UpperBound = *$)

$L.addAll$ (all definitions with the same constructor)

When $Tr()$ has a class instance of reconfiguration rule, it is necessary to add complementary treatment as follows: Create a new Isabelle function with the reconfiguration rule name in order to make the necessary changing in the system such as add/remove links and units according to the rule.

5 VERIFICATION OF RDES

The next step after generating the RDES system in Isabelle/HOL consists of improving the verification by avoiding redundant computations. To this end, we propose an algorithm that treats units and properties to be verified. The main idea is to identify for each configuration, related units that should be checked. A unit should be checked only once in the proposed verification algorithm. Thus, from a configuration to another, only the new unit should be verified. Mainly, the algorithm consists of two steps:

- $Step1$: determines the difference between two sets of units composing source and destination configurations.
- $Step2$: determines the properties that have relation with units selected in $step1$.

The Algorithm of verification (Algorithm 1) takes as input three variables:

- $PListe$: represents the set of all properties to be verified.
- $Conf_i\_Unit_i$ (respectively $Conf_j\_Unit_j$): represents the set of all units composing $configuration_i$ (respectively $configuration_j$). And it gives as result two sets: set of units and set of properties to be verified.

where,

- **difference**($list_1$, $list_2$) is the function that takes as inputs two lists $list_1$, $list_2$, and returns the difference between them;
- **add**(L, i) is the function that adds the item i to the list L.

Algorithm 1: Verification Algorithm.

```
Input: PListe list properties,;
Conf_i_Unit_i, Conf_j_Unit_j: list UNIT;
Output: U,P;
U = difference (Conf_i_Unit_i, Conf_j_Unit_j);
/*U new set to save a result*/;
for Unit u_k ,k = 1..(U.size) do
    for property p_q ,q = 1..(P.size) do
        if exist a relation between: p_q and u_k
         then
            add (P, item);;
            /*P a new set to save a result*/;
        end
    end
end
```

6 APPLICATION TO FESTO

In this section, we apply the proposed approach to the RDES in order to illustrate our contribution. First, we present the production system FESTO as running example. Second, we apply the proposed approach on it.

6.1 Components & Working Process

FESTO consists of three stations: Distribution station, Testing station and Processing station. The Distribution station is formed of a pneumatic feeder and a converter which transmits cylindrical workpieces from a stock to the Test station. The Test Station 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 Station, 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. We assume in this paper that FESTO performs in different production modes by using two drilling machines Driller1 and Driller2, as follows:

- Light1 (respectively Light2): Only Driller1 (respectively Driller2) is activated and used to drill workpieces.
- Medium: Driller1 and Driller2 are activated but used sequentially to drill workpieces (i.e., Driller1 or Driller2 works).
- High: Driller1 and Driller2 are activated and used simultaneously to drill two pieces in the same time.

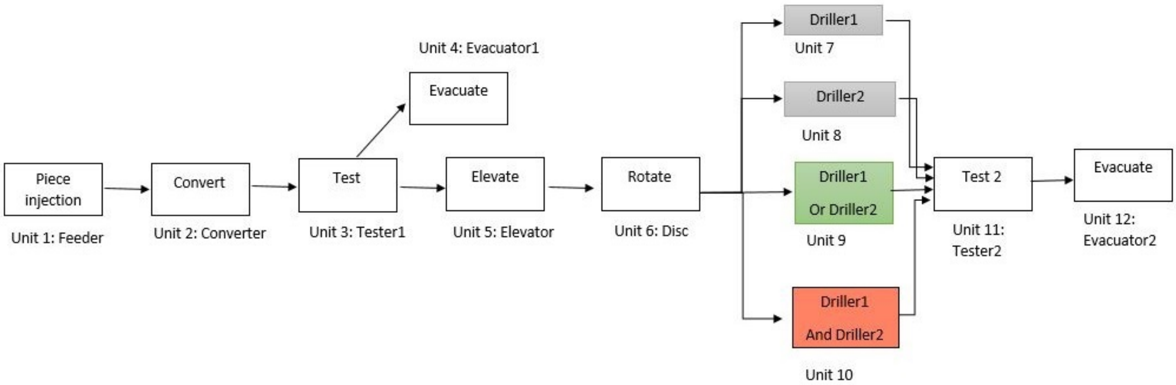

Figure 7: Working process of FESTO.

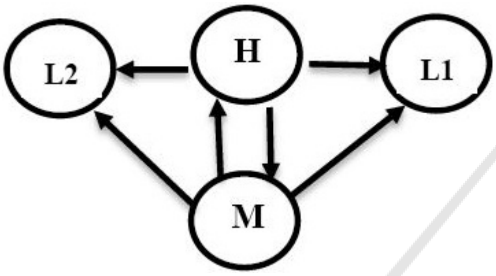

Figure 8: FESTO possible reconfiguration mode.

The system reconfigures in order to avoid any problem caused by a physical fault (i.e., when Driller1 or Driller2 breakdown) or to answer user requirements. The reconfiguration behavior of the studied system loses its usefulness when both machines Driller1 and Driller2 are broken. In the last case, the system totaly stops. Figure 8 describes possible reconfigurations of FESTO..

6.2 Production Lines

Based to the different production modes as shown in Figure 7, FESTO behavior is represented by four production lines such that each line is a list of described as follows: Line1 represents the default production mode Light1. After the work of Unit3, a workpiece is moved to Unit4 or Unit5 according to the result of the test station. Light2 is described by Line2. Line3, line4 represent, respectively the medium and high production modes of the FESTO system.

- **Line1**= $Unit1$; $Unit2$; $Unit3$; $Unit5$; $Unit6$; $Unit7$; $Unit11$; $Unit12$.
- **Line2**= $Unit1$; $Unit2$; $Unit3$; $Unit5$; $Unit6$; $Unit8$; $Unit11$; $Unit12$.
- **Line3**= $Unit1$; $Unit2$; $Unit3$; $Unit5$; $Unit6$; $Unit9$; $Unit11$; $Unit12$.
- **Line4**= $Unit1$; $Unit2$; $Unit3$; $Unit5$; $Unit6$; $Unit10$; $Unit11$; $Unit12$.

We denote by: $L1$, $L2$, M, and H, the four possible system modes: $Light1$, $Light2$, $Medium$, and $High$, respectively. As shown in Figure 8, the set of FESTO RR are described as follow: RR_FESTO=$\{r_1(M,L1)$, $r_2(M,L2)$, $r3(M,H)$, $r_4(H,L1)$, $r4(H,L2)$, $r_5(H,M)\}$

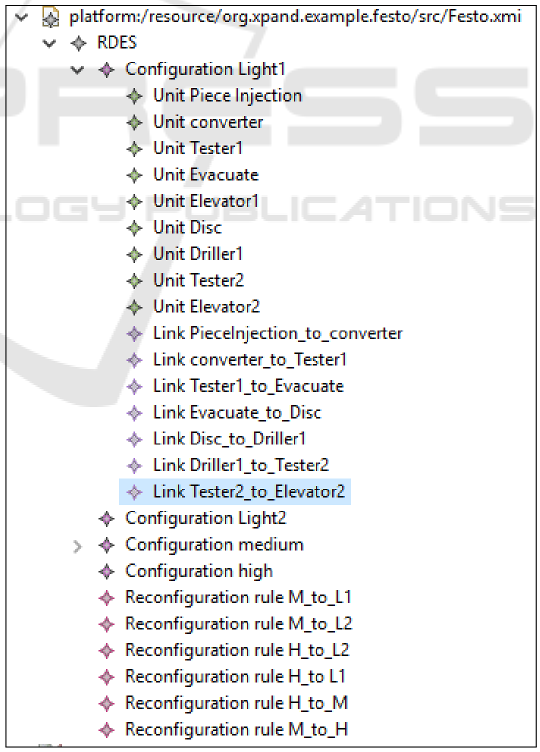

Figure 9: Model Ecore of our system.

6.3 Formalization in Isabelle Tool

Figure 9 presents the Ecore model of our system in Isabelle tool. We apply the above transformation rules to get the following result: the system component (UNIT) shown in Figure 10, all possible links between units(Arc) shown in Figure 11, different configurations (SC), reconfiguration rules (RR) shown in Figure 12, and the complete system (Isa System) is shown in Figure 13.

```
definition Piece Injection::UNIT where " Piece Injection UName
=("Piece Injection"),state=False "
definition converter::UNIT where " converter UName
=("converter"),state=False "
definition Tester1::UNIT where " Tester1UName
=("Tester1"),state=False "
definition Evacuate::UNIT where " Evacuate UName
=("Evacuate"),state=False "
definition Elevator::UNIT where " Elevator UName
=("Elevator"),state=False "
definition Disc::UNIT where " Disc UName
=("Disc"),state=False "
definition Driller1::UNIT where " Driller1UName =("Driller1")
,state=False "
definition Tester2::UNIT where " Tester2UName
=("Tester2"),state=False "
definition Elevator2::UNIT where" Elevator2UName =("
Elevator2"),state=False "
```

Figure 10: FESTO MPS Units Isabelle formalization.

```
definition Piece Injection_to_converter::Arc where " Piece
Injection_to_converter AName =(" Piece
Injection_to_converter"),S_unit =( Piece Injection) ,D_unit
=( converter) "
definition converter_to_Tester1::Arc where "
converter_to_Tester1AName =("
converter_to_Tester1"),S_unit = (converter),D_unit =
(Tester1)"
definition Tester1_to_Evacuate::Arc where "
Tester1_to_Evacuate AName =("
Tester1_to_Evacuate"),S_unit = (Tester1),D_unit =
(Evacuate)"
definition Arc4 ::Arc where " Evacuate_to_Disc AName =("
Evacuate_to_Disc"),S_unit = (Evacuate),D_unit = (Disc)"
definition Disc_to_Driller1::Arc where " Disc_to_Driller1
AName =(" Disc_to_Driller1"),S_unit = (Disc),D_unit =
(Driller1)"
definition Driller1_to_ Tester2::Arc where " Driller1_to_
Tester2 AName =(" Driller1_to_ Tester2"),S_unit =
(Driller1),D_unit = (Tester2)"
definition Tester2_to_ Elevator2:: Arc where " Tester2_to_
Elevator2 AName =(" Tester2_to_ Elevator2"),S_unit =
(Tester2),D_unit = (Elevator2)"
```

Figure 11: FESTO MPS Arcs Isabelle formalization.

6.4 Verification

As shown in Figure 14, from the mode (H) to (L1) (resp. (L2)), only Driller 1 (resp. Driller 2) need to be checked, the unchanged units do not have to be checked again. Furthermore, from the configuration mode (M) to (L1) (resp. (L2)), only Driller 1 (resp. Driller 2) need to be checked. Let us assume that

```
definition Light1 ::SC where " Light1 CName=("
Light1"),allUnits= [Piece Injection, converter, Tester1,
Evacuate, Elevator1, Disc, Driller1, Tester2, Elevator2],
allArcs=[ Piece Injection_to_ converter,
converter_to_Tester1, Tester1_to_Evacuate,
Evacuate_to_Disc,Disc_to_Driller1, Driller1_to_ Tester2,
Tester2_to_ Elevator2] "
definition M_to L1 ::RR where " M_to L1 RName =(" M_to
L1"),Prec M=("H"),Next M=("M") ,c=("two Driller active") "
definition M_to L2::RR where " M_to L2 RName =(" M_to
L2"),Prec M=("H"),Next M=("L") ,c=("One Driller active") "
definition H_to L1::RR where " H_to L1RName =(" H_to
L1"),Prec M=("M"),Next M=("L") ,c=("One Driller active") "
definition H_to L2::RR where " H_to L2 RName =(" H_to
L2"),Prec M=("M"),Next M=("H") ,c=("Two Driller active") "
definition H_to M::RR where " H_to L2 RName =(" H_to
M"),Prec M=("M"),Next M=("H") ,c=("Two Driller active") "
```

Figure 12: FESTO MPS configurations (SC), reconfiguration rules (RR) Isabelle formalization.

```
definition Festo ::Isa System where "Festo SName =("Festo"),
RRs=[M_to_L1, M_to_L2, H_to_L1, H_to_L2, M_to_H,
H_to_M], SCs=[ Light1, Light2, medium, high]"
```

Figure 13: FESTO MPS Isabelle formalization.

the user wants to switch the system to configuration (H). In this case, the proposed algorithm of verification searches in the set of the already checked units. If a precedence relationship is found, then it is not necessary to check it again. Otherwise, it will be forwarded to the prover. The algorithm already determined the properties to be verified in the configuration (L1), Therefore, it searches in the set of the already checked units about the properties is necessary to verifier.

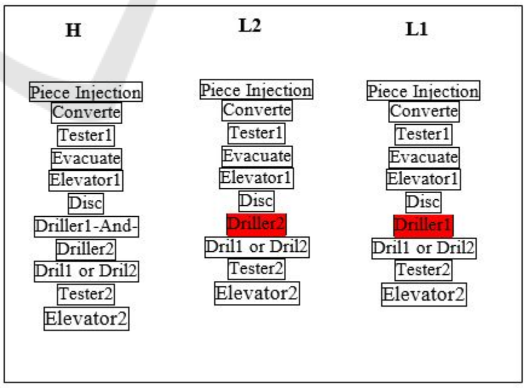

Figure 14: Verification of FESTO MPS.

Let us assume that FESTO is in Medium production mode when the system receives requests to change the production to Light1. If condition r (M, L1) is satisfied, then the reconfiguration rule r(M, L1) is executed automatically to respond to this request. Using Isabelle tool we implement r(M, L1), where we make the modification in the system, including the remove of the unit Driller1-Or-Driller2 and add the unit Unit7. The r (M, L1) means the switching from Driller1 to line2. After, r (M, L1) is executed, the workpieces are drilled by machine Driller1. In the next, we present how to process a request, as follows:

```
Reconfiguration Rule M_to_L1!
If ((M_to_L1).c)then
Operation{
activated (Piece Injection)
activated (converter)
activated (Tester1)
activated (Evacuate)
activated (Elevator1)
activated (Disc)
add (Driller1)
remove (Drill or Dril2)
activated (Tester2)
activated (Elevator2)
}
End If
End
```

Where: activated (Unit i) represents the working of the Unit i. For example, activated (Unit 2) means the execution of the operation convert.

7 PERFORMANCE EVALUATION

Figure 15 shows two curves corresponding to the verification process with and without using our proposed algorithm. The values of the abscises axis correspond to the reconfigurations rules when the system runs two times ((H to L1), (H to L2), (M to L1), (M to L2), (M to H), (H to M))) in order. The ordinate axis correspond to the number of checked units. The curve in blue corresponds to the verification without proposed algorithm. The curve in red corresponds to the optimal verification using proposed algorithm. It is important to note that the number of checked units decreases gradually until the value zero when we use the proposed algorithm. The reduction in the number of units is followed by a reduction in the number of properties.

8 CONCLUSION

This paper deals with the modeling and verification of reconfigurable discrete event systems following the MDE approach and using Isabelle/Hol theorem prover. We define a Meta-Model to model RDESs using MDE, we propose a new type Isabelle equivalence to this Meta-model, and we establish the link

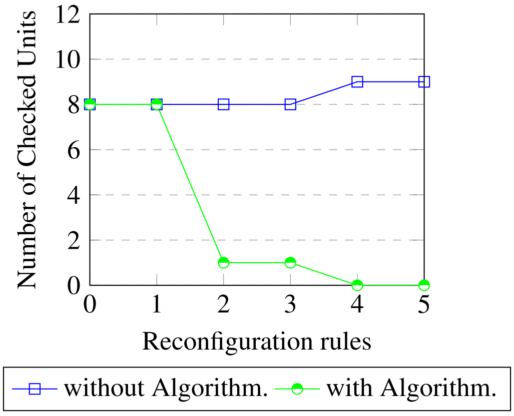

Figure 15: Comparison between verification process with and without using the proposed algorithm.

between MDE and Isabelle by defining a set of reconfigurations rules to allow automatic generation of system in Isabelle. Further more, once the system described in Isabelle, we apply the verification process. In order to avoid redundant computations, we propose a new algorithm for optimal verification. In a future work, we plan to reduce verification time of RDESs by minimizing the number of properties. We plan also to deal the correctness of the transformation itself by describing both metamodels and transformation rules in in Isabelle/HOL, then, use its theorem prover to verify some properties that are preserved by the transformation (Meghzili et al., 2019).

REFERENCES

Ali, T., Nauman, M., and Alam, M. (2007). An accessible formal specification of the uml and ocl meta-model in isabelle/hol. In *2007 IEEE International Multitopic Conference*, pages 1–6. IEEE.

Clarke, E. M., Henzinger, T. A., Veith, H., and Bloem, R. (2018). *Handbook of model checking*, volume 10. Springer.

Djeddai, S., Strecker, M., and Mezghiche, M. (2012). Integrating a formal development for dsls into meta-modeling. In *International Conference on Model and Data Engineering*, pages 55–66. Springer.

Guellouz, S., Benzina, A., Khalgui, M., and Frey, G. (2016). Reconfigurable function blocks: Extension to the standard iec 61499. In *2016 IEEE/ACS 13th International Conference of Computer Systems and Applications (AICCSA)*, pages 1–8. IEEE.

Hafidi, Y., Kahloul, L., Khalgui, M., Li, Z., Alnowibet, K., and Qu, T. (2018). On methodology for the verification of reconfigurable timed net condition/event systems. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*.

Hafidi, Y., Kahloul, L., Khalgui, M., and Ramdani, M.

(2019). New method to reduce verification time of reconfigurable real-time systems using r-tncess formalism. In *International Conference on Evaluation of Novel Approaches to Software Engineering*, pages 246–266. Springer.

Jiang, Y., Liu, J., Dowek, G., and Ji, K. (2016). Sctl: Towards combining model checking and proof checking. *arXiv preprint arXiv:1606.08668*.

Khalgui, M., Mosbahi, O., Hanisch, H.-M., and Li, Z. (2012). Retracted article: A multi-agent architectural solution for coherent distributed reconfigurations of function blocks. *Journal of Intelligent Manufacturing*, 23(6):2531–2549.

Khalgui, M., Mosbahi, O., and Li, Z. (2019). On reconfiguration theory of discrete-event systems: From initial specification until final deployment. *IEEE Access*, 7:18219–18233.

Khalgui, M., Mosbahi, O., Li, Z., and Hanisch, H.-M. (2010). Reconfiguration of distributed embedded-control systems. *IEEE/ASME Transactions on Mechatronics*, 16(4):684–694.

Khlifi, O., Mosbahi, O., Khalgui, M., and Frey, G. (2015). Gr-tnces: New extensions of r-tnces for modelling and verification of flexible systems under energy and memory constraints. In *2015 10th International Joint Conference on Software Technologies (ICSOFT)*, volume 1, pages 1–8. IEEE.

Lewis, R. (2001). *Modelling control systems using IEC 61499: Applying function blocks to distributed systems*. Number 59. Iet.

Meghzili, S., Chaoui, A., Strecker, M., and Kerkouche, E. (2017). On the verification of uml state machine diagrams to colored petri nets transformation using isabelle/hol. In *2017 IEEE International Conference on Information Reuse and Integration (IRI)*, pages 419–426. IEEE.

Meghzili, S., Chaoui, A., Strecker, M., and Kerkouche, E. (2019). Verification of model transformations using isabelle/hol and scala. *Information Systems Frontiers*, 21(1):45–65.

Zhang, J., Frey, G., Al-Ahmari, A., Qu, T., Wu, N., and Li, Z. (2017). Analysis and control of dynamic reconfiguration processes of manufacturing systems. *IEEE Access*, 6:28028–28040.

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. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 43(4):757–772.