RA2DL-Pool: New Useful Solution to Handle Security of Reconfigurable Embedded Systems

Farid Adaili, Olfa Mosbahi, Mohamed Khalgui, Samia Bouzefrane

HAL Id: hal-02425168

https://hal.science/hal-02425168

Submitted on 29 Dec 2019

RA2DL-Pool: New Useful Solution to Handle Security of Reconfigurable Embedded Systems

Farid ADAILI[1,2,3], Olfa MOSBAHI[1], Mohamed KHALGUI[1,4] and Samia BOUZEFRANE[3]

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

[2] *Tunisia Polytechnic School, University Of Carthage, Tunisia*

[3] *CEDRIC Laboratory, National Conservatory of Arts and Crafts, France*

[4] *SystemsControl Laboratory, Xidian University, China*

{*farid.adaili, samia.bouzefrane*}*@cnam.fr,* {*olfamosbahi, khalgui.mohamed* }*@gmail.com*

Abstract: The importance of security in software and hardware components becomes the major concern nowadays. This paper focuses on adaptive component-based control systems following the reconfiguration architecture analysis and design language (denoted by RA2DL). Despite of its efficiency, RA2DL component can be compromised due to its lack in terms of security. This paper proposes a new method for modeling the security of RA2DL component, argues for a more comprehensive treatment of an important security aspect with several mechanisms such as authentication and access control. In this paper, we propose a new architecture of RA2DL where pools are containers of sets of RA2DL components characterized by similar properties. The proposed approach is applied to a real case study dealing with Body-Monitoring System (BMS).

1   Introduction

The increasing use of the embedded and reconfiguration technologies in the information systems is not only a concern of major corporations and governments but also an interest of individual users. Due to this wide use, many of these systems manage and store information that is considered sensitive, such as personal or business data. The need to have secured components for each system that contains such information becomes a necessity rather than an option (Mouratidis et al., 2005). The embedded components (MS, 2008) are getting increasingly connected and are more and more involved in networked communications. The users of these components are now able to execute almost all the network/internet applications. These components are also increasingly involved in the transfer of secured data through public networks that need protection from unauthorized access. Thus the security requirements in embedded systems have become critical.

Traditional security research has been focusing on how to provide assurance on confidentiality, integrity, and availability (Clements, 1996). However, with the exception of mobile code protection mechanisms, the focus of past research is not how to develop secured software that is made of components from different sources. Previous research provides necessary infrastructures, but a higher level perspective on how to use them to describe and enforce security, especially for component-based systems, has not received sufficient attention from research communities so far.

The authors propose in (F.Adaili et al., 2015) a new concept of components named RA2DL as a solution for reconfigurable AADL components composed of controller and controlled modules. The first one is a set of reconfiguration functions applied in RA2DL to adapt its execution to any evolution in the environment, described by three reconfiguration forms:

(i) Form 1: Architectural level: modifies the component architecture when particular conditions are met. This is made by adding new algorithms, events and data or removing existing operations in the internal behaviors of the component.

(ii) Form 2: Compositional level: modifies the composition of the internal components (algorithms) for a given architecture.

(iii) Form 3: Data level: changes the values of variables without changing the component algorithms, and the second one is a set of input/output

events, algorithms, and data as represented by reconfiguration modules.

However, securing an RA2DL component is not an easy task. With rapidly advancing hardware/software technologies and ubiquitous use of computerized applications (Ren and Taylor, 2005), modern software is facing challenges that it has not seen before. More and more software is built from existing components which come from different sources. This complicates analysis and composition, even if a dominant decomposition mechanism is available. Additionally more and more software/hardware components are running in a networked environment. These network connections open possibilities for malicious attacks that were not possible in the past. These situations raise new challenges on how to handle security so that to design a component-based architecture that is more resistant to attacks and less vulnerable.

Facing the new challenges for security of reconfigurable RA2DL-based systems, we propose new solutions allowing the required authentification for the access control to components under a set of constraints such as the limitation in memory. These solutions are supported by a new concept called pool which is a container that gathers networked RA2DL under security constraints. The container allows the control of any operation allowing the reconfiguration of RA2DL components as well as the access to local algorithms and data.

The current paper is organized as follows: We discuss in Section 2 the originality of the paper by studying the state of the art. Section 3 describes the background of RA2DL. Section 4 defines the new extension for secured RA2DL components. We expose in Section 5 the case study: Body-Monitoring System (BMS) and how the implementation is performed to secure RA2DL. Section 6 concludes the paper and gives some perspectives as a future work.

2 State of the art

In this section, we present a state of the art of secured component-based design approaches. In (Brereton and Budgen, 2000), the authors present a classification of component-based systems by describing software components as independent units that interact to form a functional system. A component does not need/have to be compiled before it is used. Each component offers services to the rest of the system and adopts a provided interface that specifies the services that other components can use.

The authors in (Ren and Taylor, 2005) present a

treatment of an important security aspect, access control, at the architecture level and modeling of security subject, resource, privilege, safeguard, and policy of architectural constituents. The modeling language, Secure xADL, is based on the existing modular and extensible architecture description language.

In (Cai et al., 2000), the authors propose a QA (Quality Assurance) model for component-based software which covers component requirement analysis, component development, component certification, component customization, and system architecture design, integration, testing and maintenance. An extension of the Component Object Model (COM), Distributed COM (DCOM), is a protocol that enables software components to communicate directly over a network in a reliable, secure, and efficient manner. DCOM is designed for use across multiple network transports, including internet protocols such as HTTP. When a client and its component reside on different machines, DCOM simply replaces the local interprocess communication with a network protocol. Neither the client nor the component is aware of the changes of the physical connections.

In (elena Rugina et al., 2006), Rugina and al. present an iterative dependency-driven approach for dependability modeling using AADL. This approach is a part of a complete framework that allows the generation of dependability analysis and evaluation models from AADL models to support the analysis of software and system architectures in critical application domains.

AADL and OSATE tools can be used to validate the security of systems designed using MILS4 architecture (Hansson et al., 2008). The work in (J. Alves-Foss and Taylor, 2006) uses two mechanisms to modularize or divide and conquer in secure systems: partitions, and separation into layers. The MILS architecture isolates processes in partitions that define a collection of data objects, code, and system resources and can be evaluated separately. Each partition is divided into the following three layers: Separation Kernel Layer, Middleware Service Layer and Application Layer each of which is responsible for its own security domain and nothing else.

In (Jürjens, 2002), the author presents the extension UMLsec of UML that allows to express security relevant information within the diagrams in a system specification. UMLsec is defined as an UML profile using the standard UML extension mechanisms. In particular, the associated constraints give criteria to evaluate the security aspects of a system design by referring to a formal semantic of a simplified fragment of UML.

Note that, no one in all related works deals with

secured reconfigurable components. We propose in this paper a new concept of security of RA2DL components to be named *RA2DL − Pool* that allows: (i) Grouping of RA2DL components that have the same similar properties. (ii) Associating to each *RA2DLPool* a security mechanism like authentication and access control mechanisms.

3 RA2DL Background

We defined in a previous paper (F.Adaili et al., 2015) the concept of RA2DL components as an extension of reconfigurable AADL (Vergnaud et al., 2005) (Architecture Analysis and Design Language). RA2DL as depicted in Figure 1 is composed of controller and controlled modules where the first one is a set of reconfiguration functions applied in AADL, and the second one is a set of input/output events, algorithms, and data. The controlled module is described by the following four modules:

IEM (Input Events Module): This module processes the reconfiguration of input events (IE) stored in the *IEDB* database of input events. It defines and activates at a particular time a subset of events to execute the corresponding algorithms in RA2DL.

OEM (Output Events Module): This module processes the reconfiguration of output events (OE) stored in the *OEDB* database of output events. It defines and activates at a particular time a subset of events to be sent once the corresponding algorithms finish their execution in RA2DL.

ALM (Algorithms module): This module processes the reconfiguration of the active algorithms (addition or removal) at a particular time in order to be coherent with active input and output events of *IEM* and *OEM*. These algorithms are stored in the *ALDB* database of algorithms.

DM (Data Module): This module processes the reconfigurations of *data* in RA2DL in coherence with the rest of modules. It is stored in the *DDB* database of data values.

We focus on three hierarchical reconfiguration levels in RA2DL: **(i) Form 1**: Architectural level: modifies the component architecture when particular conditions are met. This is made by adding new algorithms, events and data or removing existing operations in the internal behaviors of the component. **(ii) Form 2**: Compositional level: modifies the composition of the internal components (algorithms) for a given architecture. **(iii) Form 3**: Data level: changes the values of

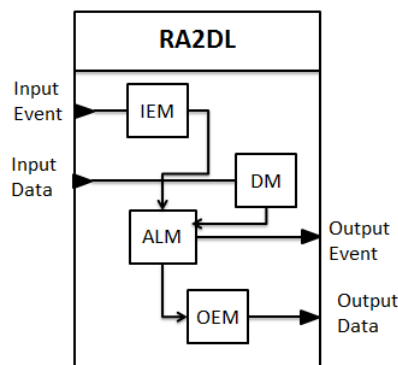variables without changing the component algorithms.

Figure 1: Architecture of an RA2DL component.

In another extension in (Adaili et al., 2015) for enhancing the execution of RA2DL components, a new execution model is proposed which is composed of three layers: **(i) Middleware Reconfiguration level** that handles the input reconfiguration flows, **(ii) Execution Controller level** to control the execution and reconfiguration of RA2DL and **(iii) Middleware Synchronization level** that controls and manages the synchronization of the reconfiguration. Additionally, we proposed a new approach to coordinate several RA2DL components in a distributed architecture based on a coordination matrix.

Because of the resource limitations in adaptive systems, satisfying a non-functional requirement such as security requires careful balance and trade-off with other properties and requirements of the system such as performance, memory usage and access rights of the RA2DL. This further emphasizes the fact that security cannot be considered as a feature that is added later to the design of an RA2DL component. It needs to be considered from early stages of development and along with other requirements. In fact, the security by design approach as defined by Arnab (Ray and Cleaveland, 2006) in software engineering ensures that security is addressed at the point of conception to avoid the security vulnerabilities. Considering the characteristics of RA2DL components, major impacts of security features in these systems are based on performance, power consumption, flexibility, maintainability and cost (Kocher et al., 2004). Therefore in the design of RA2DL components, implications of introducing security decisions should be taken into account and analyzed. Several related works do not

provide solutions to develop security of RA2DL components of adaptive embedded systems. The current paper proposes new extended solutions to secure an RA2DL component. However, in this work we want to extend this study by considering a new architecture of secured RA2DL-based pools.

4 New Extension For Secured RA2DL

In this section, we enrich RA2DL by security mechanisms that undergo such a failure to enhance their execution and simulation.

4.1 Motivation: RA2DL-Pool

Security is an aspect that is often neglected in the design of adaptive systems. However, the use of these systems for critical applications such as controlling power plants, vehicular systems control, and medical devices (Salem et al., 2015) makes security considerations even more important. Also because of the operational environment of adaptive systems and the reconfiguration actions applied by an RA2DL component. To allow the required security, we introduce the concept of $RA2DL - Pool$ as a container which is an abstract class that offers different services dealing with security, where each $RA2DL - Pool$ has a level of sensitivity of the information of its RA2DL components. $RA2DL - Pool$ container serves as a general purpose holder of other components. It holds well-defined methods for grouping RA2DL components together. $RA2DL - Pool$ is represented by the following elements:

- **Controller:** it is the crucial part of the pool that contains methods and represents firstly the interface between the user and the pool, and secondly between the pool and the RA2DL components,
- **Tables:** there are three kinds of tables : use table (UT), reconfiguration table (RT) and security table(ST),
- **Database:** is the database containing the sets of RA2DL components,
- **Reconfiguration scenarios:** define the set of reconfiguration scenarios realized in pool or in its RA2DL components. Each scenario will be applied in relation with the three tables (UT, RT and ST),
- **RA2DL:** it is the RA2DL component with its algorithms and input/output ports,

Figure 2 presents the class diagram of $RA2DL - Pool$. An $RA2DL - Pool$ container holds a set of RA2DL components with a set of methods. This set of components has a set of methods that describe how to examine and add or delete components to the $RA2DL - Pool$. It contains the following methods described in Table 1.

4.2 Security Mechanisms for RA2DL

To consolidate the RA2DL technology, we will put a set of security-mechanisms divided into two families:

4.2.1 Authentication Mechanism

This is a critical mechanism where all users must authenticate to access to the reserved services of $RA2DL - Pool$ or RA2DL components. This mechanism is always in relation with the user table (UT), where the columns u are the identifiers of users (id_user) and lines s represent the services ($services\_user$). To implement the authentication mechanism, we use **RADIUS** *(Remote Authentication Dial-In User Service)* developed by Livingston Enterprise (Yoon et al., 2007), which is a networking protocol that provides centralized Authentication, Authorization, and Accounting (AAA) management for users who connect and use a network service. The principle of the authentication of an RA2DL with RADIUS is as follows:

(i) the *Controller* executes a connection request. UT table recovers the identification information, (ii) the *Controller* transmits this information to the target service in RA2DL, (iii) the target component receives the connection request from the *Controller*, controls and returns the configuration information required for the user to provide or deny access, (iv). *Controller* refers to the user an error message if it fails an authentication.

4.2.2 Access Control Mechanism

This mechanism comes just after authentication to control the access to the RA2DL components. Two tables are used in this case: security and reconfiguration tables. The first one is the security table ST which contains in lines (p) all the user privileges ($privilege\_user$) and in columns (u) the (id_user). The second one is the reconfiguration table (RT) that contains in lines (r) reconfigurations identifiers (id_reconf) and in
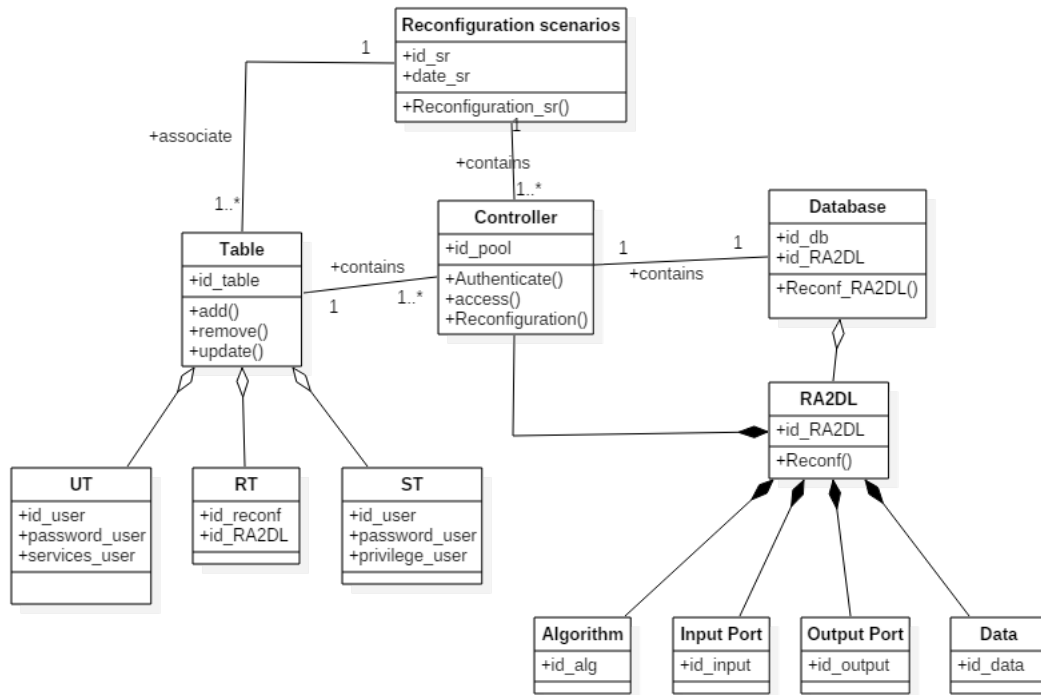
Figure 2: Class diagram of RA2DL-Pool.

columns (*c*) the identifiers of RA2DL components (*id_RA2DL*). This mechanism may be represented by a triplet (S, C, M_{sc}) where S denotes the service, C denotes the RA2DL component (or RA2DL-pool) and M_{sc} that maps each pair (C and S) to a set of access rights.

Figure 3 presents the sequencing of the interaction between the RA2DL components and the RA2DL-Pool. The main goal is to show this interaction and how to apply authentication and access control mechanisms.

Figure 4 highlights the activity of these two mechanisms and tests in order to achieve a secure RA2DL component.

4.3 Architecture of Secured RA2DL-based Pools

We present in Figure 5 the class diagram of the secured RA2DL-based pool. This diagram represents the architecture of RA2DL-based pools with the static aspect of the relation between the RA2DL components and the pool. It does not provide any information about its behavior. The architecture of secured RA2DL-based pools is composed of the following distinct classes: (i)
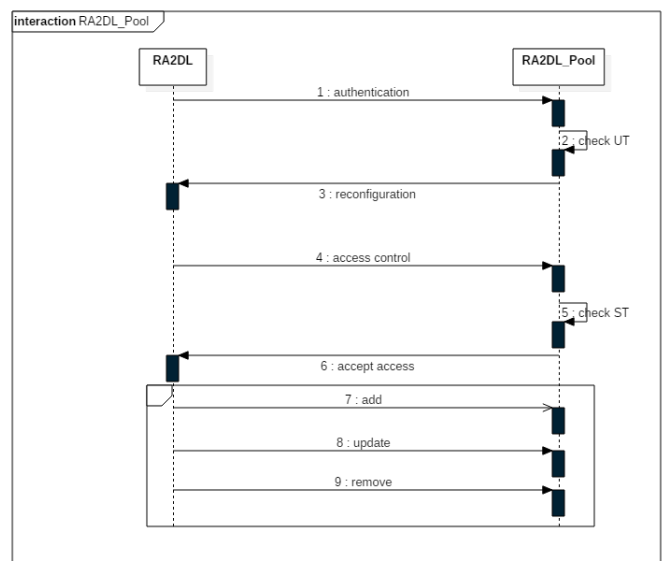

Figure 3: Sequence diagram.

RA2DL : The main class of the architecture, the component concerned by the security concept, (ii) *RA2DL − Pool*: It is the container of RA2DL components, (iii) *Security*: Is an association between RA2DL and RA2DL-Pool which repre-

Table 1: RA2DL-pool Methods.

| Method	Description
getRA2DL ()	returns the number of components within the $RA2DL - Pool$
Component-getRA2DL(int position)	returns the component at the specific position
Component⏋ getRA2DL ()	returns an array of all the RA2DL components held within the container
RA2DL-add (Component RA2DL, int position)	adds RA2DL component to the container $RA2DL - Pool$ at position
add (Component RA2DL, RA2DL constraints)	necessary for layouts that require additional information
public void remove (int index)	deletes the RA2DL component at position index from the $RA2DL - Pool$
remove (RA2DL component)	deletes the RA2DL component from the container $RA2DL - Pool$
removeAll ()	removes all RA2DL components from the $RA2DL - Pool$ container
boolean isAncestorOf (RA2DL)	checks to see if the RA2DL component is a parent of container
addContainerListener (pool)	registers listener as a controller of RA2DL-Pool
removeContainerListener (pool)	removes listener as an interested listener of RA2DL-Pool
processEvent(RA2DLEvent e)	receives all RA2DL events with this $RA2DL - Pool$ container as its target
addNotify ()	creates the peer of all the components within it
removeNotify ()	destroys the peer of RA2DL components contained within it
Insetsgetinsets()	gets the containers current insets
list()	useful method to find out what is inside a container
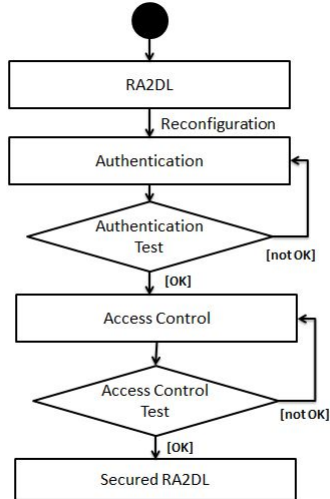
Figure 4: Activity diagram.

data of RA2DL.

Figure 5: Architecture of a secured RA2DL-based pool.

sents the security-mechanisms, (iv) $RA2DL - Soft$: It is the software component of RA2DL, (v) $RA2DL - Hard$: It is the hardware component of RA2DL, (vi) $Algorithm$: Is a set of methods to be executed by each RA2DL component, (vii)$Reconfiguration$: Represents all of the reconfiguration scenarios to execute with RA2DL, (viii) $Architecture$: Describes the reconfiguration scenarios that touch on the RA2DL architecture, (ix) $Structure$: Describes the reconfiguration scenarios that touch on the RA2DL composition or structure,(x) $Data$: Describes the reconfiguration scenarios that touch on the RA2DL data, (xi) $EventPort$: Port for input/output event of RA2DL, (xii) $DataPort$: Port for input/output
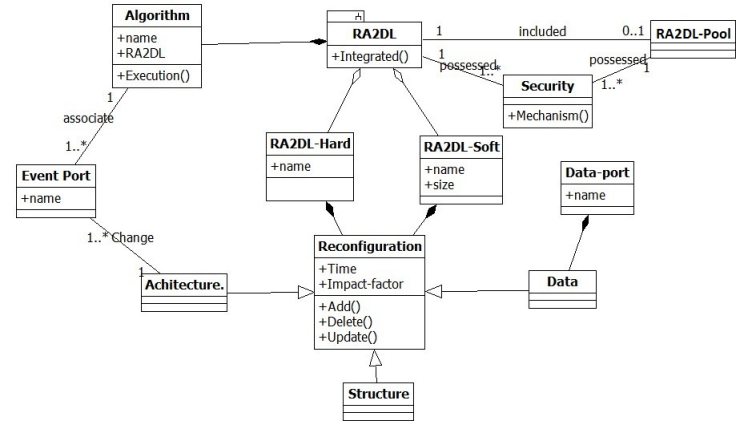
4.4 Modelling and Verification

We propose in this section the modelling and verification of the new architecture of secured RA2DL-based pools by using UPPAAL (Bengtsson et al., 1996). Firstly, we model the pool with its security aspect. Secondly we check a set of properties to ensure the security of the pool.

4.4.1 Modelling of secured RA2DL-based Pool

The modelling of the RA2DL-based pool with the security aspect is described by the state machine presented in Figure 6.
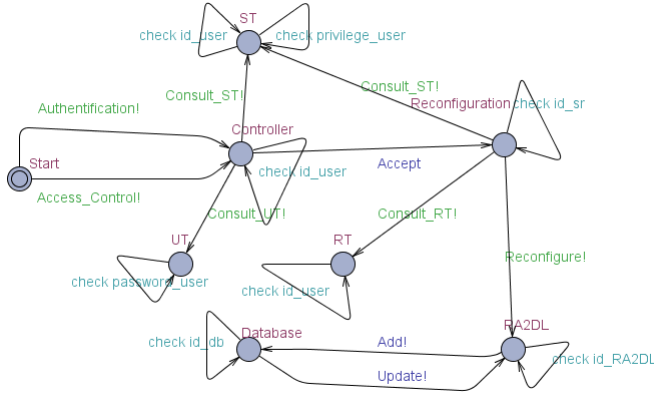
Figure 6: Modelling of Secured RA2DL-Based Pools.

The states of this model are described as follows: *start* to start the querying or the connection of $RA2DL - pool$. *Controller* represents the first contact with the pool, in this state the checking of *id_user* is important after verification of the *password_user* in the table UT. If the authentication is accepted and the password is checked, it can go to the state *Reconfiguration* which represents all the reconfiguration scenarios. After the verification of the following parameters: (i) *id_sr* for the IDs of scenarios, *privilege_user* for the privilege of the user in the table ST and (iii) *id_reconf* for the IDs of the reconfiguration in the table RT. If all of the IDs are accepted, then the user may apply the reconfiguration in the target $RA2DL$ component after checking the *id_RA2DL*. A *database* is associated to this level to facilitate the reconfiguration of the RA2DL components.

4.4.2 Verification of Secured RA2DL-based Pool

We propose the following properties in order to verify the security of the RA2DL components.
- **Property 1:** *(Controller[].check id_user)AND (UT[].check passeword_user):* for each connection with the pool, we should check the user authentification by using the UT table,
- **Property 2:** *(Reconfiguration[].check id_sr) AND (RT[].check id_reconf):* before the execution of any reconfiguration scenario, it is important to check if it is registered in the reconfiguration table (RT),
- **Property 3:** *(Reconfiguration[].Reconfigure! ⇒ RA2DL[].check id_RA2DL) AND (ST[].check privilege_user):* this property concerns the verification of the access control mechanism,

- **Property 4:** *RA2DL[].save ⇒ Database[].check id_db*: each RA2DL component should be imperatively saved in a *Database* to facilitate the use of RA2DL components and to minimize the execution time,
- **Property 5:** *(Controller[] AND Reconfiguration[] AND RA2DL[] AND Database[] AND ST[] AND RT[] AND UT[]) not deadlock:* the system is deadlock-free.

The verification of these properties is summarized in Table 2.

Table 2: Verification results.

Property	Result	Time (sec)	Memory (Mo)
Property 1	True	10.52	5.72
Property 2	True	9.12	4.82
Property 3	True	5.32	3.20
Property 4	True	13.25	6.56
Property 5	True	8.23	4.37

5 Case Study and Implementation

We use as a running example in the current paper the body-monitoring system (BMS) to evaluate the paper's contribution.

5.1 Case Study: Body-Monitoring System (BMS)

During the last few years there has been a significant increase in the number and variety of wearable health monitoring devices ranging from simple pulse monitors, activity monitors, and portable Holter monitors, to sophisticated and expensive implantable sensors. The Body-Monitoring System (BMS) (Husemann et al., 2004) is designed as a mobile device that is able to collect measured data and to act according to instructions set by a supervisor. The system consists of a body-monitoring network. In order to recognise the monitored person's state, the monitor unit connects to various body sensors and i/o devices by using either wired or wireless communication technologies. Data from all sensors are collected, stored and analysed at real-time and, according to the analysis, actions may then be performed. A computer is used as an interface to the body-monitoring network, and developed software allow a supervisor to configure the monitor unit for the monitored person, to connect sensors and i/o

devices, define and upload instructions for monitoring and download collected data (Figure 7).

The monitor unit software consists of a communication module (responsible for connecting and controlling sensors, and for gathering and pre-processing measured data), a storage module (for storage of collected data), and a policy interpretation module responsible of controlling the behaviour of the monitor unit according to instructions defined by a supervisor.

To secure this system, we must take into account these steps: (i) make the grouping of RA2DL components according to similar characteristics in RA2DL-Pool. (ii) assign for each RA2DL-pool a security level (depending on the degree of importance of the RA2DL components that they contain). (iii) allocate for each RA2DL-pool a security mechanism.
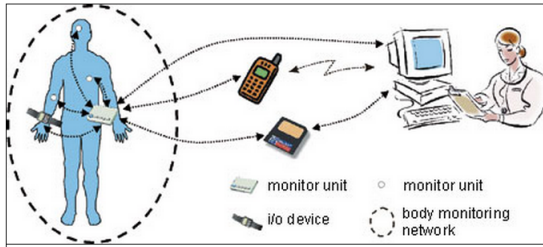

Figure 7: Overview of the Body Monitoring System (Bielikov, 2002).

Running Example: We group the RA2DL components of BMS system in five RA2DL-Pools as shown in Figure 8. (i) **RA2DL-Pool 1:** includes the following RA2DL components: *RA2DL-G* for the Glucose detection, *RA2DL-C* for the chloride detection and *RA2DL-W* for the water detection.(ii) **RA2DL-Pool 2:** includes the following RA2DL components: *RA2DL-L* for the lactate detection and *RA2DL-PH* for the PH detection. (iii) **RA2DL-Pool 3:** includes the following RA2DL components: *RA2DL-DM* for the Diabetes mellitus detection and the *RA2DL-BP* for the Blood pressure. (iv) **RA2DL-Pool 4:** contains the display device which is the component *RA2DL-Mobil*. (v) **RA2DL-Pool 5:** contains the *RA2DL-Soft* for the transmission of data with a protocol until *RA2DL-Mobil*.

5.2 Implementation

We present in this section the tool of the BMS system that we developed in LISI Laboratory at INSAT Institute of University of Carthage in Tunisia and CEDRIC Laboratory at National Conserva-

tory of Arts and Crafts of Paris in France. We assume five pools with their parameters such as the number of RA2DL components in pool, Worst Case Execution Time (WCET), the authentication and the access control mechanisms (Figure 9).

Figure 9: RA2DL-Pools of BMS system.

Figure 10 shows the connectivity test of the different pools according to the authentication mechanisms and also to check the configuration between the various RA2DL components in each pool.
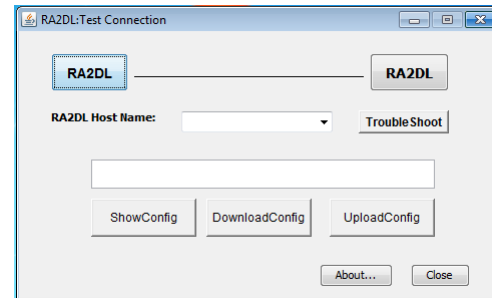

Figure 10: Test of Authentification mechanism.

Running Example: The application of our approach to the BMS case study is illustrated in Table 3, where we give a security level (S.L) for the five pools depending on the sensitivity of the comprising components. In the BMS system, the RA2DL-pool 3 is the most secured and RA2DL-pool 1 is the less secured one. Both security mechanisms are applied to the five pools.

Table 3: Running example.

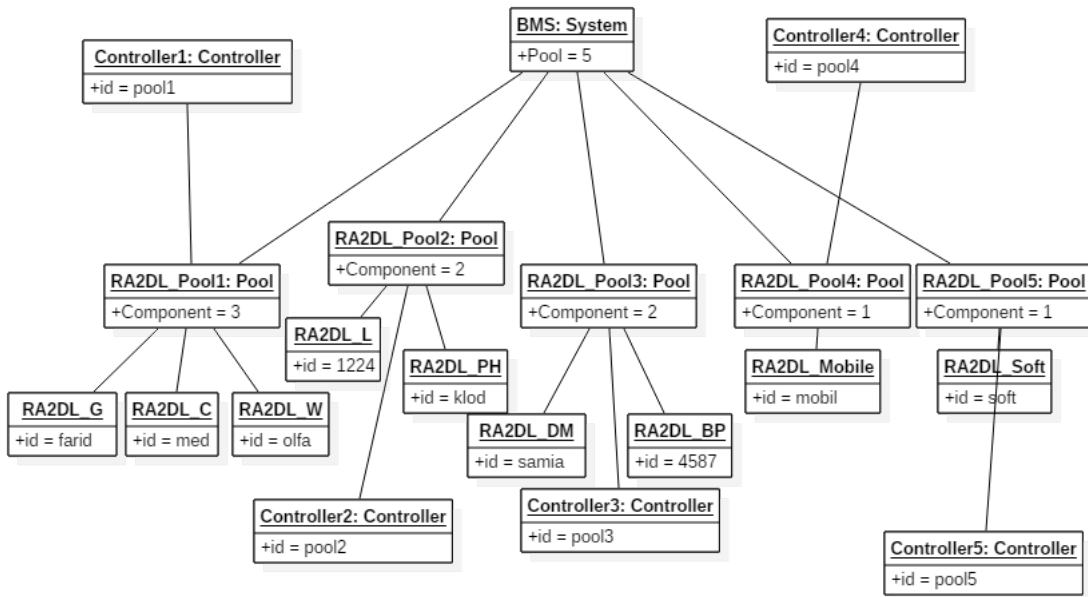|	S.L	Authen.	AccessControl	Security
Pool 1	1	No	Yes	Yes
Pool 2	2	Yes	No	Yes
Pool 3	6	Yes	Yes	Yes
Pool 4	5	Yes	Yes	Yes
Pool 5	5	Yes	Yes	Yes

Figure 8: Object diagram of BMS.

5.3 Evaluation

This step is devoted mainly to test our approach and evaluate the execution time. Ten assessments are applied to the two mechanisms that are focused on two stolen: RA2DL without pool and RA2DL with pool of the BMS system. We show in Figure 11 the results of the evaluation. We are interested in response time gains for secured and not secured RA2DL components.
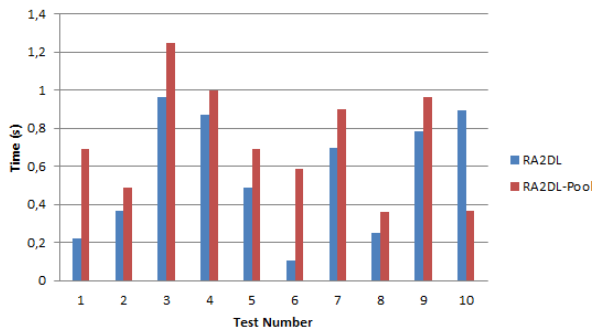

Figure 11: Result of evaluation.

The proposed approach has the following advantages:

(a) Functionality: RA2DL component in $RA2DL - Pool$ are at a functional level much more adaptable and extendable than traditional RA2DL components.

(b) Reusability: A reusability is an important characteristic of a high-quality RA2DL component. Programmers should design and implement RA2DL components in such a way that many different programs can reuse them.

(c) Maintainability: In BMS system a piece of functionality ideally is implemented just once. It is self-evident that this results in easier maintenance of system, which leads to lower cost, and a longer life.

6 Conclusion

This paper deals with new solutions for a required security in adaptive RA2DL control based systems. Firstly, we define a new grouping methodology named RA2DL-Pool which has its own methods for the grouping and security techniques. Secondly, we propose two important mechanisms to control the security in pools. Finally their implementation is applied to a Body-Monitoring system (BMS). The future work will deal with the flexibility in the network that links different devices of RA2DL-based systems. In addition, we will be interested either in the different real-time aspects of RA2DL or in the run-time tests of components once deployed on the target devices.

REFERENCES

Adaili, F., Mosbahi, O., Khalgui, M., and Bouzefrane, S. (2015). New solutions for useful execution models of communicating adaptive ra2dl. In *Intelligent Software Methodologies, Tools and Techniques*, volume 532, pages 87–101. Springer International Publishing.

Bengtsson, J., Larsen, K., Larsson, F., Pettersson, P., and Yi, W. (1996). Uppaal: A tool suite for automatic verification of real-time systems. pages 232–243, Secaucus, NJ, USA. Springer-Verlag New York, Inc.

Bielikov, M. (October 2002). A body-monitoring system with eeg and eog sensors. *Journal of ERCIM News No. 49*.

Brereton, P. and Budgen, D. (2000). Component-based systems: a classification of issues. *Computer*, 33(11):54–62.

Cai, X., Lyu, M., Wong, K.-F., and Ko, R. (2000). Component-based software engineering: technologies, development frameworks, and quality assurance schemes. In *Software Engineering Conference, 2000. APSEC 2000. Proceedings. Seventh Asia-Pacific*, pages 372–379.

Clements, P. C. (1996). A survey of architecture description languages. In *Proceedings of the 8th International Workshop on Software Specification and Design*, IWSSD '96, pages 16–, Washington, DC, USA. IEEE Computer Society.

elena Rugina, A., Kanoun, K., and Kaniche, M. (2006). An architecture-based dependability modeling framework using aadl. In *In 10th IASTED International Conference on Software Engineering and Applications SEA2006*.

F.Adaili, O.Mosbahi, M.Khalgui, and S.Bouzefrane (2015). Ra2dl: New flexible solution for adaptive aadl-based control components. In *Proceedings of the 5th International Conference on Pervasive and Embedded Computing and Communication Systems*, pages 247–258.

Hansson, J., Feiler, P. H., and Morley, J. (2008). Building secure systems using model-based engineering and architectural models. *CrossTalk: The Journal of Defense Software Engineering*, 21(9).

Husemann, D., Steinbugler, R., and Striemer, B. (2004). Body monitoring using local area wireless interfaces. US Patent App. 10/406,865.

J. Alves-Foss, W. S. Harrison, P. O. and Taylor, C. (2006). The mils architecture for high assurance embedded systems. *International Journal of Embedded Systems*.

Jürjens, J. (2002). Umlsec: Extending uml for secure systems development. In *Proceedings of the 5th International Conference on The Unified Modeling Language*, UML '02, pages 412–425, London, UK, UK. Springer-Verlag.

Kocher, P., Lee, R., McGraw, G., and Raghunathan, A. (2004). Security as a new dimension in embedded system design. In *Proceedings of the 41st Annual Design Automation Conference*, DAC '04, pages 753–760, New York, NY, USA. ACM. Moderator-Ravi, Srivaths.

Mouratidis, H., Kolp, M., Faulkner, S., and Giorgini, P. (2005). A secure architectural description language for agent systems. In *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems*, AAMAS '05, pages 578–585, New York, NY, USA. ACM.

MS, A. (2008). Security needs in embedded systems. Cryptology ePrint Archive, Report 2008/198. http://eprint.iacr.org/.

Ray, A. and Cleaveland, R. (2006). A software architectural approach to security by design. In *30th Annual International Computer Software and Applications Conference, COMPSAC 2006, Chicago, Illinois, USA, September 17-21, 2006. Volume 2*, pages 83–86.

Ren, J. and Taylor, R. (2005). A secure software architecture description language. In *Workshop on Software Security Assurance Tools, Techniques, and Metrics*, pages 82–89.

Salem, M. O. B., Mosbahi, O., Khalgui, M., and Frey, G. (2015). Zizo: Modeling, simulation and verification of reconfigurable real-time control tasks sharing adaptive resources - application to the medical project bros. In *Proceedings of the International Conference on Health Informatics*, pages 20–31.

Vergnaud, T., Pautet, L., and Kordon, F. (2005). Using the aadl to describe distributed applications from middleware to software components. In *Reliable Software Technology - Ada-Europe 2005, 10th Ada-Europe International Conference on Reliable Software Technologies, York, UK, June 20-24, 2005, Proceedings*, pages 67–78.

Yoon, E., Lee, W., and Yoo, K. (2007). Secure pap-based RADIUS protocol in wireless networks. In *Advanced Intelligent Computing Theories and Applications. With Aspects of Contemporary Intelligent Computing Techniques, Third International Conference on Intelligent Computing, ICIC 2007, Qingdao, China, August 21-24, 2007. Proceedings*, pages 689–694.