

A Role based Access Control Model for Agent based Control Systems

Michael Drouineaud
Bremen Institute of Safe Systems
Department of Mathematics and Computer Science
University of Bremen
Bibliotheksstr. 1
D-28359 Bremen
Germany
Email: mdruuid@tzi.de

Arndt Lüder
Center Distributed Systems
IAF
Otto-von-Guericke University Magdeburg
Universitätsplatz 2
D 39106 Magdeburg
Germany
Email: arndt.lueder@mb.uni-magdeburg.de

Karsten Sohr
Bremen Institute of Safe Systems
Department of Mathematics and Computer Science
University of Bremen
Bibliotheksstr. 1
D-28359 Bremen
Germany
Email: sohr@informatik.uni-bremen.de

Abstract—The spreading of Ethernet TCP/IP protocol suite based communication in automation systems has raised new questions concerning data consistency and security. This paper introduces temporal-logic RBAC as a method to formalise the security relevant access constraints for an agent based control system. It also offers some examples of such formalised constraints generated from constraints that were originally written down in human language. The authors point out, that temporal-logic RBAC might thus be helpful to cope with the new problems of information security in plant automation.

I. INTRODUCTION

Data consistency and security become more and more important in industrial automation systems. But, in contrast to their importance, both of them are not as integrated in the thinking of automation system designers as it is necessary.

What are the reasons for this? To find a sufficient answer the consideration of some historical facts about automation is necessary.

The industrial automation has been started with control systems providing a closed loop control on a special production process within one machine. These systems were designed in a hard wired manner. The progress of computer systems has lead to the integration of programmable logic controller systems (PLC) in the automation system as central control instance.

The next step was the application of field bus systems as Profibus, Interbus, CAN, and Sercos to mention only a few of the applicable systems, to substitute the parallel cabling between control unit and sensors and actuators by one communication cable.

But with the field bus systems the possibility for a rigorous change of control system design was founded. Now, more

then one control unit could be integrated in an automation system and the system complexity could increase. The first distributed automation systems have been established during the last decade of the 20th century.

Until this moment automation systems have been independent and closed systems with very limited interactions to data manipulation systems outside of the automation system. This situation has changed on the step from the second to the third millennium. At this moment the Ethernet TCP/IP protocol suite based communication systems, mainly developed and maintained for PC based office communication, have found their way into automation systems as communication backbone among PLCs and will more and more be used as main communication system at field level among PLCs and sensors as well actuators. Thereby, this new communication medium will give rise to new challenges for automation especially for distributed automation, distributed intelligence, and the application of office and internet based technologies within automation. But these challenges will also bring new problems about. The use of the Ethernet TCP/IP protocol suite based communication systems in automation systems leads to the integration of this systems in the internet and opens up the automation system with respect to data exchange and data manipulation from outside of the system which was impossible within the Prior-Ethernet era. Now automation system designers and users have to consider the protection of its systems with respect to data consistency and data security. A special problem in this field are the rules of access to the automation system or to parts of the automation system. Its importance and meaning will be described more in detail in one of the

following sections of this paper.

Within this paper problems remaining from the use of internet based technologies in automation will be considered. The technology of interest are agent based control systems.

We will show by some examples the benefits of role based access and its formalised description. Therefore the paper is organised as follows. Within the next section one possible application of agents in automation will be described in more detail. Based on this description in section 3 three fields of data manipulation in automation systems will be given and roles of agents will be named with respect to different kinds of data and system usage. Section 4 will give a foundation of role based access control while section 5 will describe the possible use of formalised descriptions of role based access control to the above mentioned field of agent based control. The paper will terminate with some conclusions.

II. AGENT SYSTEMS IN AUTOMATION

Distributed intelligence in automation leads to the application of a distributed control system based on independent and collaborating control units responsible for controlling parts of the complete system. Thereby parts are not necessary physical system parts but also the creation of products and the acquisition of data.

Concepts for the design of distributed intelligence in automation systems have been considered within different research and development projects. HMS [2] and PABADIS [1], to mention only two of them, are only the most known among a wide ranging variety of projects considering this topic.

Within the HMS project and its associated research activities a control concept based on the application of agents responsible for control and use of production resources, agents responsible for the creation of products and agents responsible for establishing associations among the first mentioned agent types to realise the production process in the controlled system was established. Of course, this is a limited view of the complexity of the designed architecture but for the purposes of this paper it is detailed enough. More information on the results of HMS and its associated research activities can be found in [10].

Within the PABADIS Project a multi agent system consisting of agents responsible for product creation, agents responsible for the control of production resources, and agents for general data acquisition purposes has been designed. More information on the PABADIS results can be found in [6], [5].

Both architectures have in common, that agents with different purposes will access agents responsible for the control of production resources. Thereby, the access rights of the agents for data access and process access has to be distinguished as shown in figure 1 and in addition to that within both classes the agents have to be discriminated with respect to the access to different data sets as well as to different processes.

III. ACCESS ROLES IN AUTOMATION

As shown in the last section in agent based automation two main types of resource access have to be considered.

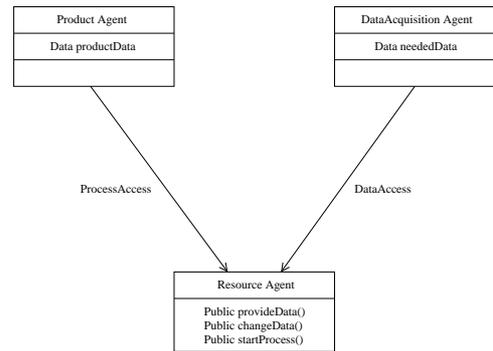


Fig. 1. Agent based access schema

These are the access to data and the access to process. Within the data access the read only access and the read and write access need to be distinguished by reasons of different roles of access as shown in Figure 1 by the functions *provideData()* and *changeData()* of the resource.

The access to the resource is also based on different roles with different access rights. Within agent based automation systems there are the following roles:

- Agent with read only data access to a special data set
- Agents with read and write data access to a special data set
- Agents with process access

In addition, it has to be distinguished between agents which were created within the automation system and which will never leave the system and agents which come from outside the automation system. The access rights of external agents are more limited than for internal agents. Thereby an inheritance relation of agent roles is given.

In addition to the access rights also access constraints are given with respect to the different roles.

- Internal agents are prior to external agents with respect to each access independent of data or process access.
- At any moment only one agent can access a process on a resource.
- If an agent has accessed a process on a resource it has to release the access before an other agent can access the resource.
- At any moment only one agent can access a special data set for read and write access on a resource.
- If an agent has accessed a special data set of a resource for read and write access it has to release the access before an other agent can access the resource.

IV. RBAC

Role-based access control (RBAC) has received considerable attention as a promising alternative to traditional discretionary and mandatory access control. One reason for this increasing interest was an extensive field study carried out by the National Institute of Standards and Technology (NIST) which pointed out that in practice permissions are assigned to users according to their roles/functions in the

organization (governmental or commercial) [3]. In addition, the explicit representation of roles greatly simplifies the security management and makes possible to use well-known and time-honored security principles like separation of duty and least privilege [12]. Furthermore, an RBAC standard has been proposed [4], which is based on the RBAC96 model introduced by Sandhu et al. [12].

The RBAC96 model (adapted for our scenario) has the following components (see figure 2):

Agents – set of agents, Roles – set of roles, P – set of permissions

$UA \subseteq Agents \times Roles$ (user Assignment)

$RH \subseteq Roles \times Roles$ is a partial order also called the role hierarchy or role dominance relation written as \leq .

$PA \subseteq Roles \times P$ (permission assignment)

P is the set of ordered pairs of operations and resources. The relation PA assigns to each role a subset of P. So PA determines for each role the operation(s) it may execute and the resource(s) to which the operation in question is applicable for the given role. Thus any agent having assumed this role can apply an operation to a resource if the corresponding ordered pair is an element of the subset assigned by PA to the role.

In the scenario we will consider the roles are limited only w. r. t. operations, i. e. if an agent has assumed a role it can apply any admissible operation to any resource (agent). From now on we will just use the term “resources”, although the actual objects are resource agents. The term “objects” as such is not used, because there are no other possible objects than resources. Furthermore we omit the session concept, which is a part of the RBAC96 model, because in our scenario at any moment each agent can assume at most one role.

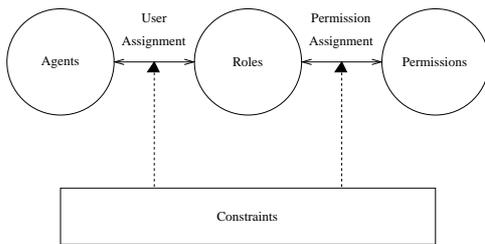


Fig. 2. RBAC for agent based automation

V. ACCESS CONSTRAINTS

As already mentioned before the security policy for our scenario, i. e. the rules and limitations imposed on agents with respect to actions and behaviour to ensure data security and consistency, depends on some dynamic constraints, which cannot be expressed by the RBAC96 model itself, because it offers no formalism for expressing such constraints. To straighten out this shortcoming Till Mossakowski, Michael Drouineaud and Karsten Sohr proposed temporal-logic RBAC as formalism (see [11]). Temporal-logic RBAC is based on

temporal first-order logic, a logic that has been intensively studied in the literature [7], [8] and comes with standard tools, e.g. [9]. A temporal first-order signature consists of a set of sorts, a set of function symbols and a set of predicate symbols (each symbol coming with a string of argument sorts and, for function symbols, a result sort). Function and predicate symbols are partitioned into *rigid* and *flexible* symbols: the former do not change over time, while the latter may vary. Models live over discrete time, indexed by the natural numbers as time steps. They interpret the sorts with (time-independent) carrier sets, rigid function and predicate symbols with time-independent functions and predicates of appropriate types, and flexible function and predicate symbols with families of functions and predicates, where the families are indexed by natural numbers.

Sentences are the usual first-order sentences built from equations, predicate applications and logical connectives and quantifiers \forall , \exists . Additionally, we have the modalities \square (always in the future), \diamond (sometimes in the future) and \bigcirc (in the next step). Satisfaction is defined inductively for a given time step, where the modalities allow referring to other time steps. A sentence is satisfied in a model if it is satisfied in the time step zero.

sorts $Agents, Roles, Operations, Resources$

rigid preds $PA: Roles \times Operations \times Resources$
 $UA: Agents \times Roles$

flexible preds $AUTH: Agents \times Operations \times Resources$
 $EXEC: Agents \times Operations \times Resources$
 $ACTIVE_FOR: Agents \times Roles$
 $ACCESS: Agents \times Roles \times Resources$
 $LOCK: Roles \times Resources$
 $REL_AC: Agents \times Resources$

forall $a : Agents; r : Roles; op : Operations;$
 $res : Resources;$

- $\square((\diamond ACTIVE_FOR(a, r)) \Rightarrow UA(a, r))$
- $\square(AUTH(a, op, res) \Rightarrow (\exists r : Roles. (UA(a, r) \wedge PA(r, op, res) \wedge ACTIVE_FOR(a, r))))$
- $\square(EXEC(a, op, res) \Rightarrow AUTH(a, op, res))$

The rigid predicates PA (permission assignment) and UA (user assignment) describe the corresponding RBAC relations (see [12]). PA is true for a triple (r, op, res) if and only if the role r is entitled to apply the operation op to the resource res . UA is true for a pair (a, r) if and only if the role r is assigned to the agent a . The flexible predicate $AUTH$ is true for a triple (a, op, res) if and only if the agent a is authorized to execute the operation op on the resource res (in this moment). Analogously the flexible predicate $EXEC$ is true for a triple (a, op, res) if and only if the agent a actually executes the operation op on the resource res in this moment. The flexible predicate $ACTIVE_FOR$ is true for a pair (a, r) if and only if the agent a actually assumes the role r (in this moment). So while UA indicates the possession of a role $ACTIVE_FOR$ stands for their actual use (compare

first of the preceding statements). The value true of the flexible predicate *ACCESS* for a triple (a,r,res) expresses, that the agent a in the role r controls the resource res, i. e. it is authorized to execute operations on res within the limitations of *PA* (compare second and third of the preceding statements). The flexible predicate *LOCK* is true for a pair (r,res) if and only if the resource res is locked for the role r. The flexible predicate *REL_AC* indicates, that an agent a is going to end its access to the resource res in the role r, i. e. it is going to cease controlling the resource res in role r. Before we can express the constraints for the agents as statements, we must first name roles and operations.

ops *int_provideData, int_changeData, int_startProcess, ext_provideData, ext_changeData, ext_startProcess*
roles *INT_Roles={int_ro,int_rw,int_pac}*
 EXT_Roles={ext_ro,ext_rw,ext_pac}
rigid op *corr : EXT_Roles → INT_Roles*

The rigid predicate *PA* is defined as follows:

forall *res : Resources;*
PA(int_ro, int_provideData, res)
PA(int_rw, int_changeData, res)
PA(int_pac, int_startProcess, res)
PA(ext_ro, ext_provideData, res)
PA(ext_rw, ext_changeData, res)
PA(ext_pac, ext_startProcess, res)

The rigid operation *corr* assigns external roles to the corresponding internal roles. We call *corr* rigid operation, because it is not at all influenced by time, as the following defining equations show:

int_ro = corr(ext_ro)
int_rw = corr(ext_rw)
int_pac = corr(ext_pac)

So internal roles *r* and external roles \hat{r} with $r = corr(\hat{r})$ are called corresponding roles. By now we have gathered the means to express some of the previously described constraints. One of the primary constraints is the exclusiveness of internal and external roles. With the given formalism we can express this as follows:

forall *a : Agents; r : INT_Roles;*
UA(a, r) ⇒ ¬(∃ $\hat{r} : EXT_Roles. UA(a, \hat{r})$)

The reader may notice, that this constraint needs no \Box modality, because here only rigid predicates (*UA*) are involved. Another constraint is, that any agent can assume at most one internal resp. external role at any moment:

forall *a : Agents; r₁, r₂ : INT_Roles; $\hat{r}_1, \hat{r}_2 : EXT_Roles;$*

- $\Box((ACTIVE_FOR(a, r_1) \wedge r_1 \neq r_2) \Rightarrow \neg ACTIVE_FOR(a, r_2))$
- $\Box((ACTIVE_FOR(a, \hat{r}_1) \wedge \hat{r}_1 \neq \hat{r}_2) \Rightarrow \neg ACTIVE_FOR(a, \hat{r}_2))$

Furthermore according to the given constraints, it is not possible, that an agent in an external role and another agent in the corresponding internal role control the same resource at the same time (see constraints 2 and 4 given in section III on page 2):

forall *a₁, a₂ : Agents; $\hat{r} : EXT_Roles; res : Resources;$*

- $\Box(ACCESS(a_1, \hat{r}, res) \Rightarrow \neg ACCESS(a_2, corr(\hat{r}), res))$
- $\Box(ACCESS(a_1, corr(\hat{r}), res) \Rightarrow \neg ACCESS(a_2, \hat{r}, res))$

Finally at any moment at most one internal resp. external agent can control a resource in a given role:

forall *a₁, a₂ : Agents; r₁, r₂ : INT_Roles;*
 $\hat{r}_1, \hat{r}_2 : EXT_Roles; res : Resources;$

- $\Box((ACCESS(a_1, r_1, res) \wedge ACCESS(a_2, r_2, res) \wedge r_1 = r_2) \Rightarrow a_1 = a_2)$
- $\Box((ACCESS(a_1, \hat{r}_1, res) \wedge ACCESS(a_2, \hat{r}_2, res) \wedge \hat{r}_1 = \hat{r}_2) \Rightarrow a_1 = a_2)$

Thus for every resource it is ensured, that at any moment for each external role no more than one agent can exercise control. This holds also for corresponding internal roles. So the constraints 2 and 4 given in section III on page 2 hold. The following equivalences for *LOCK* prevent simultaneous access to the same resource in corresponding roles:

forall *$\hat{r} : EXT_Roles; res : Resources;$*
 $\Box(LOCK(\hat{r}, res) \Leftrightarrow LOCK(corr(\hat{r}), res))$

Additionally there are the following interactions between the flexible predicates *LOCK* and *ACCESS*:

forall *a : Agents; r : INT_Roles; $\hat{r} : EXT_Roles;$*
 res : Resources;

- $\Box(ACCESS(a, r, res) \Rightarrow (LOCK(r, res) \wedge ACTIVE_FOR(a, r)))$
- $\Box(ACCESS(a, \hat{r}, res) \Rightarrow (LOCK(\hat{r}, res) \wedge ACTIVE_FOR(a, \hat{r})))$

forall *$\hat{r} : EXT_Roles; res : Resources;$*
 $\Box(\neg(\exists a : Agents. ACCESS(a, \hat{r}, res)) \wedge \neg(\exists a : Agents. ACCESS(a, corr(\hat{r}), res))) \Rightarrow (\neg LOCK(\hat{r}, res) \wedge \neg LOCK(corr(\hat{r}), res))$

Considering the equivalences for *LOCK* it is possible to replace $\neg LOCK(\hat{r}, res) \wedge \neg LOCK(corr(\hat{r}), res)$ in the statement above equivalently by $\neg LOCK(\hat{r}, res)$ or $\neg LOCK(corr(\hat{r}), res)$ for corresponding internal roles. These statements assure, that the flexible predicate *LOCK* behaves according to the constraints 2, 3, 4 and 5 given in section III on page 2. So any agent a in some internal role *r* resp. some external role \hat{r} can take control of an arbitrary resource res it does not already control in the same role if and only if *LOCK*(*r*, *res*) resp. *LOCK*(\hat{r} , *res*) is not true. The following two statements describe how the flexible predicate *REL_AC* interacts with the flexible predicates

ACCESS and *LOCK* (see constraints 3 and 5 in section III on page 2):

forall $a : Agents; r : INT_Roles; \hat{r} : EXT_Roles;$
 $res : Resources;$

- $\Box((ACCESS(a, r, res) \wedge REL_AC(a, r, res)) \Rightarrow$
 $\bigcirc(\neg ACCESS(a, r, res) \wedge \neg LOCK(r, res)))$
- $\Box((ACCESS(a, \hat{r}, res) \wedge REL_AC(a, \hat{r}, res)) \Rightarrow$
 $\bigcirc(\neg ACCESS(a, \hat{r}, res) \wedge \neg LOCK(\hat{r}, res)))$

So if $ACCESS(a, r, res)$ resp. $ACCESS(a, \hat{r}, res)$ is true and the agent a intends to release access in role r resp. \hat{r} , i. e. to cease his control of the resource res in role r resp. \hat{r} , then in the next step $ACCESS(a, r, res)$ resp. $ACCESS(a, \hat{r}, res)$ and $LOCK(r, res)$ resp. $LOCK(\hat{r}, res)$ will be not true. Thus in the next moment the agent a in the role r resp. \hat{r} will no longer control the resource res and another agent may take control of the resource in the role r resp. \hat{r} or in corresponding roles (see constraints 3 and 5 in section III on page 2).

It may occur to the reader, that from the strictly theoretic point of view the predicate *LOCK* is not necessary for expressing the constraints. Indeed this is correct. Therefore the predicate *LOCK* may be seen as a step towards a real system. In a real system it would be much harder for an agent intending to take control of a resource in a certain role to check, if the predicate *ACCESS* is true for any other agent in the same or the corresponding role for the resource in question, than just to find out, whether the predicate *LOCK* is true for the desired role and the resource in question.

The constraints regarding the exclusiveness of internal and external roles or the restriction on the number of active roles are classic constraints of information security. The constraints for the predicates *ACCESS* and *LOCK* can also be seen as safety-related constraints (depending on the point of view). This illustrates, that the proposed formalism is flexible enough to cover safety-related matters. It seems even possible to express properties of workflow.

Of course we need additional flexible predicates and a lot more temporal-logic statements to describe how the access to a resource for waiting agents in certain roles is handled. Basically this is done by assigning any resource a FIFO list for each role. Thus we have 6 FIFO lists for every resource. But giving all the statements needed would definitely exceed the scope of this paper. Furthermore the constraints in question belong rather to the area of safety than information security. The authors intend to make the full amount of temporal-logic statements needed to describe all the constraints for agent based management proposed in this paper (including the handling of access to a resource for waiting agents) available on the world wide web later on.

VI. CONCLUSIONS AND FUTURE WORK

In this paper we have demonstrated, how constraints for a scenario of plant automation can be formalised using temporal-logic RBAC. This process was quite helpful to clarify the constraints that had previously been written down in human language. It would be interesting to apply the same procedure

to other (perhaps even more detailed) scenarios of plant automation. As the authors found out, when examining the agent based management scenario, the temporal-logic statements can easily become rather complex and numerous. This is an additional argument for embedding temporal-logic RBAC into a theorem prover and/or model checker. The authors hope, that this will allow proving the consistency of the formalised constraints respectively the existence of a model for the formalised constraints automatically. Since there is a rich experience in using model checking for similar formalisms in the area of safety the prospect of (at least partially) achieving this aim seems to be good.

We hope, that the reader of this paper has received an impression, how formalising constraints in temporal-logic RBAC can help to improve (information) security in plant automation.

REFERENCES

- [1] PABADIS Consortium, PABADIS project homepage, <http://www.pabadis.org>, 2003.
- [2] HMS Consortium, HMS project homepage, <http://hms.ifw.uni-hannover.de>, 2003.
- [3] D. Ferraiolo, D. Gilbert, and N. Lynch, *An examination of federal and commercial access control policy needs*, Proc. of the NIST-NCSC Nat. (U.S.) Comp. Security Conference, 1993, pp. 107–116.
- [4] David F. Ferraiolo, Ravi Sandhu, Serban Gavrila, D. Richard Kuhn, and Ramaswamy Chandramoli, *Proposed NIST standard for role-based access control*, ACM Transactions on Information and System Security **4** (2001), no. 3, 224–274.
- [5] A. Klostermeyer and A. Lüder, *A re-configurable multi-agent based architecture to enhance the flexibility of job control in single piece production systems in turbulent environments*, Msy'02 Embedded Systems in Mechatronics (Zürich, Switzerland), October 2002, pp. 131–138.
- [6] A. Lüder, J. Peschke, S. Deter, and A. Bratoukhine, *Interaction of software agents and real-time industrial control system within a pabadis system*, 14th Euromicro International Conference on Real-Time Systems (Work in Progress Session) (Vienna, Austria), June 19–21 2002, pp. 21–25.
- [7] Z. Manna and A. Pnueli, *The temporal logic of reactive and concurrent systems, specification*, Springer-Verlag, 1992.
- [8] ———, *Temporal verification of reactive systems: Safety*, Springer-Verlag, New York, 1995.
- [9] Zohar Manna, Nikolaj Björner, Anca Browne, Edward Chang, Michael Colón, Luca de Alfaro, Harish Devarajan, Arjun Kapur, Jaejin Lee, Henny Sipma, and Tomás E. Uribe, *STeP: The stanford temporal prover*, TAPSOFT '95: Theory and Practice of Software Development (Peter D. Mosses, Mogens Nielsen, and Michael I. Schwartzbach, eds.), Lecture Notes in Computer Science, vol. 915, Springer-Verlag, 1995, pp. 793–794.
- [10] D.C. McFarlane and S. Bussmann, *Developments in holonic production planning and control*, Int. Journal of Production Planning and Control **11** (2000), no. 6, 522–536.
- [11] Till Mossakowski, Michael Drouineaud, and Karsten Sohr, *A temporal-logic extension of role-based access control covering dynamic separation of duties*, The paper will be presented at TIME-ICTL 2003. It will be published by IEEE Computer Society Press.
- [12] Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein, and Charles E. Youman, *Role-based access control models*, Computer **29** (1996), no. 2, 38–47.