A Graph-based and Declarative Approach to a Secure Resource Management in Smart Factories

Alexander Lawall[1] and Thomas Schaller[2]

[1] IU International University of Applied Science,
Juri-Gagarin-Ring 152, 99084 Erfurt, Germany

[2] University of Applied Sciences Hof, Alfons-Goppel-Platz 1, 95028 Hof, Germany

Abstract. The article presents an applied research using the Design Science Research Methodology for securely managing resources of smart factories via a graph-based approach combined with a declarative query language. This query language can be used to find appropriate production facilities that are able to fulfill specific manufacturing tasks. This approach is aimed to solve the problem with the management effort for production facilities using enumeration for naming these facilities for the manufacturing tasks. Thus, the security is ensured by identifying the "current" valid identities (resources). Additionally, the usage of deputy relationships leads to alternative production facilities if resources have a breakdown or have to be serviced which has an effect on the availability.

Keywords: Identity Management · Industry 4.0 · Smart Factory · Organizational Model· Access Control.

1 Industry 4.0, Smart Factories and Resource Management

Production plants as we know them today are actually overthought within the industry 4 initiative. Central production plans for manufacturing big numbers of similar products are replaced by intelligent objects (agents) embedded in self-organizing systems called smart factories [1]. These objects are cyber-physical systems [2] using intelligent sensors for gathering information about the world around them. Similar to multiagent systems [3,4] they are able to react to environmental changes and to generate plans for fulfilling their goals. So, if a customer orders a product at a manufacturing company, an agent responsible for the production of the article is created. This agent knows all about the bill of material and the working plan for the creation of the product. Thus, the agent encompasses the physical and cyber space [5]. On basis of this information, the agent generates a plan how the product can be manufactured. For this he has to communicate with agents representing the resources of the company [6].

DOI:10.5121/ijcis.2022.12301

Additionally, cyber–physical production systems consist of elements and subsystems that connect communications and interactions in production, of machines, processes to manufacturing networks [7]. Cyber-physical production systems summarize the cooperation and response to the assigned tasks. Such systems include i.a. operational sensor networks, smart actuators, information systems, communication protocols and many more [8]. Cyber–physical production systems evolve the traditional automation pyramid [9]. The top-down hierarchy of the traditional automation pyramid proceed from enterprise resource planning, plant management, process control, control (incl. PLCs) to field level [10]. Nowadays, cyber–physical production systems are a connected network with different systems of the traditional automation pyramid hierarchy as nodes. Thus, the layered structure of the automation pyramid is transformed into a dense interconnected network. This network is built with systems from layers of the automation pyramid out of the field, control, supervisory, planning and management level [11,12].

In this article we show a graph-based approach for maintaining the resources (i.a. production systems/machines) of a smart factory and searching for valid ones using a declarative query language. This language can be used by the production agent looking for resources that are able to fulfill manufacturing tasks of the working plan.

2 Graph-based Resource Management (GBRM)

This section describes the meta-model with its entity- and relation-types, an example model for production facilities following this meta-model and the declarative query language resulting in specific resources by traversing the graph-based model.

Graph-based Resource Management (GBRM) consists of a meta-model for describing the resources of an organization and a declarative language. Resources encompass human and automatic agents (please see figure 1: the turners and mills are machines, the assembly line is stuffed with humans.). They are located in one organization, but in federated scenarios, other organizations can make use of them as well (cf. [13]). Therefore, we will have a look at the meta-model, also called organizational meta-model.

2.1 Organizational Meta-Model

The organizational model, depicted in figure 1 follows the organizational meta-model[3] that is formulated in this section. The excerpt of the meta-model consists of the set of entity-types $\mathcal{V}$ and relation-types $\mathcal{R}$.

[3] The basic meta-model is formally described in [14] as an extension of the theory in [15].

The characteristics of entity-types $\mathcal{V} = \mathcal{O} \cup \mathcal{F} \cup \mathcal{A}$ of the meta-model are:

- Organizational units $\mathcal{O} = \mathcal{O}^i \cup \mathcal{O}^e$ are for example departments, subdivisions, groups, etc.
  - Internal organizational units $\mathcal{O}^i$ are used to model intra-organizational organizational units (e.g. figure 1: *Manufactoring Company*).
  - External organizational units $\mathcal{O}^e$ are entity-types representing organizational units from partner organizations (e.g. figure 1: *Assembly Company*). Thus, inter-organizational federations across company borders are possible.
- Functional units $\mathcal{F} = \mathcal{F}^i \cup \mathcal{F}^e$ are roles respectively job functions in which agents act.
  - Internal functional units $\mathcal{F}^i$ represent intra-orga-nizational functional units like the turners of a manufacturing company (cf. figure 1).
  - External functional units $\mathcal{F}^e$ belong to partner companies.
- Human and automatic resources $\mathcal{A} = \mathcal{A}^i \cup \mathcal{A}^e$ encompass:
  - Internal resources $\mathcal{A}^i$ represent intra-organizational resources (e.g. *Turner 1*, *Mill 4* and *Assembly 1* in figure 1).
  - External resources $\mathcal{A}^e$ are used to specify resources from partner organizations (e.g. *Assembly 1 – Assembly 4* of the *Assembly Company* in figure 1).

The set of relation-types $\mathcal{R} = \mathcal{R}_s \cup \mathcal{R}_o \cup \mathcal{R}_u$ formally defines the interconnections:

- Structural relation-type $r_s \in \mathcal{R}_s$ with

$$r_s \subset \mathcal{O} \times (\mathcal{O} \cup \mathcal{F}) \tag{1}$$

$$r_s \subset \mathcal{F} \times (\mathcal{F} \cup \mathcal{A}) \tag{2}$$

- The sets[4] of organization-specific relation-types $\mathcal{R}_o$ (deputy, supervision and reporting) and user-defined relation-types $\mathcal{R}_u$ with $\forall r_o, r_f, r_a \in \mathcal{R}_o \cup \mathcal{R}_u$:

$$r_o \subset \mathcal{O} \times (\mathcal{O} \cup \mathcal{F} \cup \mathcal{A}) \tag{3}$$

$$r_f \subset \mathcal{F} \times (\mathcal{F} \cup \mathcal{A}) \tag{4}$$

$$r_a \subset \mathcal{A} \times (\mathcal{F} \cup \mathcal{A}) \tag{5}$$

The structural relation-types r_s in formulas (1) and (2) define the structural interconnections between internal and external organizational units, functional units and resources.
The formulas (3), (4) and (5) correspond to organization-specific and user-defined relations that connect internal / external entities, cf. deputy relation between the organizational units *Manufacturing Company* and *Assembly Company* in figure 1.

[4] The validity of organization-specific and user-defined relations can be restricted by constraints, cf. [16].

Hyperedges Lawall formalize in [17] hyperedges related to formula (5), $r_a \subset \mathcal{A} \times (\mathcal{F} \cup \mathcal{A})$. The start entity of the organization-specific and user-defined relations is a resource $a \in \mathcal{A}$ and the target entity is a functional unit or resource. The validity of these relations can be restricted role-dependent using hyperedges. The hyperedge starts with relations $r_a \in \mathcal{R}_o \cup \mathcal{R}_u$ and ends in functional unit $f \in \mathcal{F}$. Thus, the relation r_a is only valid if the start resource a of relation r_a acts in functional unit f. The syntax for role-dependent hyperedges r_{r_a} is described in formula (6).

$$r_{r_a} \subseteq r_a \times \mathcal{F} \tag{6}$$

Formula (4) formalizes relations $r_f \in \mathcal{R}_o \cup \mathcal{R}_u : r_f \subset \mathcal{F} \times (\mathcal{F} \cup \mathcal{A})$ starting in functional unit $f \in \mathcal{F}$ and ending in a functional unit or resource. These relations are not restricted with hyperedges. The acting functional unit f is already contained in relation r_f and as a consequence considered.

The organization-specific and user-defined relations $r_o \in \mathcal{R}_o \cup \mathcal{R}_u : r_o \subset \mathcal{O} \times (\mathcal{O} \cup \mathcal{F} \cup \mathcal{A})$ in formula (3) can also be restricted with hyperedges r_{r_o}. Relations r_o start in an organizational unit $o \in \mathcal{O}$ and end in an organizational unit, a functional unit or resource. The hyperedge r_{r_o} restricts the validity of relation r_o concerning the functional unit $f \in \mathcal{F}$, cf. formula (7).

$$r_{r_o} \subseteq r_o \times \mathcal{F} \tag{7}$$

The semantics of hyperedge r_{r_o} differ from hyperedge r_{r_a}. The relation r_o is only valid for the functional unit f respectively the connected resources. For example, the deputy relation in figure 1 between the *Manufacturing Company* and the *Assembly Company* is only valid for the functional unit *Assembly* – respectively resources *Assembly 1* to *3* – of the *Manufacturing Company*. Thus, this deputyship is not effective for the remaining functional units *Turner* and *Mill* and the connected resources *Turner 1* to *3* and *Mill 1* to *4*.

Self-references are used to define recursive relations on entities. It simplifies the interconnection of entities using organization-specific and user-defined relations. The deputy self-reference of the functional unit *Assembly*, as depicted in figure 1, indicates a deputyship between the subordinate resources *Assembly 1* to *3* of the *Manufacturing Company*. Otherwise, all resources have to be interconnected in particular, cf. [18].

2.2 Organizational Model

Figure 1 depicts an excerpt of the organizational model of a manufacturing company producing wooden chairs. The model encompasses the internal organizational unit *Manufacturing Company* with the subordinate internal functional units *Turner*, *Mill* and *Assembly*. The internal resources *Turner 1* to *Turner 3* are related to *Turner*, *Mill 1* to *Mill 4* belong to *Mill* and *Assembly 1* to *Assembly 4* are part of the internal functional unit *Assembly*.

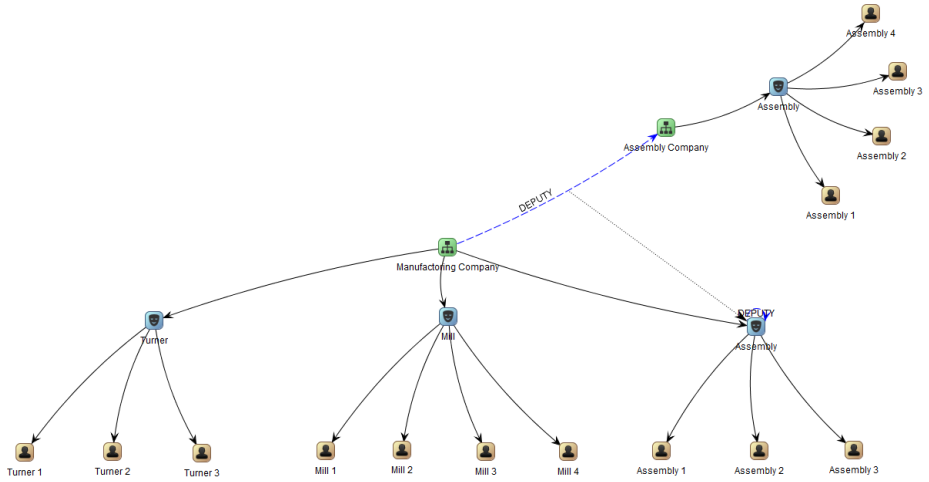

Fig. 1. Federation between Partner Companies

Additionally, there exists an external[5] partner company (assembly company) that can take over assembly tasks on load peaks or cyber attacks against the supply chain [19]. The federation between the manufacturing company and the assembly company extends the aforementioned organizational model. The external organizational unit *Assembly Company* encompasses the external functional unit *Assembly* in conjunction with the external resources *Assembly 1* to *Assembly 4*.

The organization-specific deputy relation between the *Manufacturing Company* and the *Assembly Company* follows formula (3). In addition, the relation is restricted with an hyperedge that ends in the internal functional unit *Assembly*, cf. formula (7). The self-reference of this entity indicates a mutually deputyship between the internal *Assemply* resources.

The entities of the organizational model contain attributes. Attributes can be of any kind (e.g. workpieceLength). Table 1 describes an excerpt of attributes in conjunction with their values. These attribute-value pairs reflect the characteristics of concrete resources.

2.3 Declarative Query Language

In this chapter we describe a query language that in addition to the organization model can be used to find specific resources based on relations and attributes.

A query language expression can be based on entities, relations and attributes that are included in the organizational model. The entities are concrete organiza-

[5] The automated propagation of model elements (entities, relations and attributes) to partner organizations is described in [13].

Resource	Attribute	Value
Turner 1	workpieceLength	30
	workpieceKind	octagonal
	...	...
Turner 2	workpieceLength	60
	workpieceKind	octagonal
	...	...
Turner 3	workpieceLength	40
	workpieceKind	square
	...	...
Mill 1	maxLength	100cm
	maxWidth	100cm
	maxHeight	20 cm
Mill 2	....	...

Table 1. Attributes of Resources

tional units, functional units and resources. Relations are concrete organization-specific – deputy, supervisor and reporter – and user-defined relations. Attributes are similar to user-defined relations. Their types are freely definable as needed in the organizational model. The concrete relations and attributes can be referred by language expressions.

The following syntax is described in [17]. It is repeated to give an overview concerning the power of the query language. The language expressions used in this contribution do not make use of all production rules.

The tuple $G = (N, \Sigma, P, S_G)$ of the context-free grammar G for defining queries consists of:

- The set of non-terminals $N = \{start, query, actor, funits, funit, oudef, ounits, ounit, relationTokens, withParams, contextDef, attConstraints, kcv, parameter, kvp, id, string\}$
- The alphabet of terminals $\Sigma = \{$‘a’,‘b’,...,‘z’,‘A’,‘B’, ...,‘Z’,‘ä’,‘ü’,‘ö’,‘Ä’,‘Ü’,‘Ö’, ‘0’,‘1’,...,‘9’,‘_’,‘-’,‘(’,‘)’,‘,’,‘.’, ‘*’,‘=’,‘<’,‘>’$\}$
- The set of production rules P[6]

  $start \rightarrow query \mid query\ logic\ query \mid \varepsilon$
  $query \rightarrow actor | actor$ ‘AS’ $funits$
  $query \rightarrow query$ ‘NOT’ $query$
  $query \rightarrow query$ ‘FALLBACKTO’ $query$
  $query \rightarrow query$ ‘WITH’ $withParams$
  $query \rightarrow funits$ ‘(’$oudef$‘)’
  $query \rightarrow relationTokens$ ‘(’$query$‘)’
  $query \rightarrow$ ‘(’$query\ logic\ query$‘)’
  $query \rightarrow$ ‘(’$query$‘).’$attConstraints$

[6] Meaning of meta-symbols: ? means 0 or 1 and * means 0 to ∞ occurrences.

$actor \rightarrow$ '*' $|id|string$
$funits \rightarrow funit|$ '('$funit\ logic\ funit$')'
$funit \rightarrow$ '*' $|id|string$
$oudef \rightarrow ounit|ounits\ logic\ ounits$
$ounits \rightarrow ounit|$ '('$ounits\ logic\ ounits$')'
$ounit \rightarrow$ '*' $|id|string|ounit$ 'SUBS'
$relationTokens \rightarrow$ ('ALL' | 'ANY')? id ('OF'|'TO')
$withParams \rightarrow contextDef \mid parameter \mid$
$withParams$ ',' $withParams$
$contextDef \rightarrow$ 'CONTEXT=' $context$ (',' $context$)$*$
$attConstraints \rightarrow$ 'ATT.' kcv
$kcv \rightarrow id\ comp\ string|$ '('$kcv\ logic\ kcv$')'
$parameter \rightarrow kvp$ (',' kvp)$*$
$kvp \rightarrow id$ '=' $string$
$logic \rightarrow$ 'AND' | 'OR'
$comp \rightarrow$ ('=' | '<=' | '>=' | '<' | '>' | '!=')
$id \rightarrow$ (['a'-'z','A'-'Z'] | '_' | 'Ä' | 'ä' | 'Ü' | 'ü' | 'Ö' | 'ö') (['a'-'z','A'-'Z'] | 'Ä' | 'ä' | 'Ü' | 'ü' | 'Ö' | 'ö' |[0 − 9]| '_' | '-')$*$
$string \rightarrow$ '"' id '"'

– The set of start symbols $S_G = \{start\}$

The informal semantics of the declarative query language is explained below. The language query expressions explicate the reference to entities, relations and/or attributes:

– The language expression `(Turner AND/OR Mill)(*). ATT. (workpieceLength ≥ "40" AND workpieceKind = "octagonal")` refers to
  1. **Entities**: `(Turner AND/OR Mill)(*)`
     (a) **Functional Units:** *Turner* and *Mill*. The result set of resources is determined by the terminal `AND` respectively `OR`. `AND` means that resources have to be simultaneously assigned to *Turner* and *Mill*. However, `OR` indicates that resources have to be *Turner* and/or *Mill*.
     (b) **Organizational Units:** `*`, arbitrary organizational units. The specification of concrete organizational units is similar to the definition of functional units. `AND` and `OR` are also semantically identical.
  2. **Attributes:** `workpieceLength ≥ "40" AND workpieceKind = "octagonal"`
     After traversing the organizational model for appropriate entities, resulting resources are proven if the requirements of attributes are fulfilled.
– `DEPUTY OF(Assembly(Manufacturing Company).ATT. (workpieceLength ≥ "30"))` refers to
  1. **Entities**: `Assembly(Manufacturing Company)`
     (a) **Functional Unit:** *Assembly*. The intermediate result set of the query contains the internal resources *Assembly 1*, *Assembly 2* and *Assembly 3* and the external resources *Assembly 1* to *Assembly 4*.
     (b) **Organizational Unit:** The entity *Manufacturing Company* induces a reduction of the intermediate result set to the internal resources *Assembly 1* to *Assembly 4*.

2. **Relations:** `DEPUTY OF(...)`
   The intermediate result set of the inner query `"..."` is the start for the traversal following specific relations. The detailed algorithm searching for concrete organization-specific and user-defined relations is described in [18]. In this example expression is the *deputy* relation integrated. The lookup starts in the resource level. There are no deputy relations, cf. figure 1.
   In the functional unit level is a self-reference that indicates a mutual deputyship of the resources. The organizational unit level will be considered if no suitable deputy is available.
   The deputy relation between the *Manufacturing Company* and the *Assembly Company* is an hyperedge ending in functional unit *Assembly*, cf. formula (7). Thus, the deputy relation is only valid for the functional unit *Assembly*. The intermediate result set then consists of the external resources *Assembly 1* to *Assembly 4*.
3. **Attributes:** `workpieceLength` $\geq$ `"30"`
   The final step of the lookup algorithm proves the fulfillment of all attribute requirements. The result set of the remaining resources is handed back to the "requester".

3 GBRM and Smart Factories

In this section, we demonstrate how the idea of smart factories and Graph-based Resource Management can be combined. Smart factory environments are usually decentrally organized. In order to tackle that fact, two crucial issues have to be addressed.

The first question is how the products that have to be produced know about the production facilities and their availability. This can be addressed by GBRM. Here a model of a production environment is set up, containing all facilities, their abilities, their availability and replacement/deputy relations (please see section 2.2). The product can formulate an organization query and receives the matching facilities.

Second, if the product knows the available resources, the next question is, which resource is chosen for the single working steps and when.

From the viewpoint of a production facility a lot of products are concurring for the dispatching of their working steps. So there has to be a strategy that leads to a facility working plan. Options for that like optimization on basis of the marginal return and auctions (e.g. when subcontractors can be used) are discussed deeply in [20]. In this article we will not address this point specifically.

In the following we will concentrate on the first question and will explain how the right production resources can be found using GBRM. Let us assume we would like to produce a chair with four legs, one seat and one back. According

to the working plan in table 2 we have to turn the four legs and to mill the seat and the back all with specific dimensions [7].

Task Number	Task	Specification	Amount
1	Turn chair leg	Turn, 40cm, octagonal	4
2	Mill seat	Mill, 30cm x 35cm x 5cm	1
3	Mill back rest	Mill, 20cm x 35cm x 5cm	1
4	Assemble chair	Assemble	1

Table 2. Working Plan of a Chair

After that we can assemble the single components of the chair. For the moment we disregard the fact that the leg, the seat and the chair can be produced in parallel. If an order for one chair arrives in our company a virtual chair object C_{Virt} will be created. This object knows the working plan and needs to figure out which machineries or production systems are able to carry out the single manufacturing operations. This can easily be done if there is a GBRM model of the underlying manufacturing system. If this model exists the only thing C_{Virt} has to do, is to specify a query and send it to the Organization-Server [8]. The server determines the resources that are able to carry out the specified production steps and sends them back to the C_{Virt} object. In our example the query `(Turner)(*).ATT.(workpieceLength` $\geq$ `"40" AND workpieceKind = "octagonal")` will lead to the result that only turner 2 will be appropriate. Via the use of deputy relations and propagation nodes alternative production scenarios can be modelled. If a production component has to be serviced, has a breakdown or in case of an employee beeing sick, it is easy to find a deputy. In figure 1 the hyperedge at the assembly node means that an assembly agent can take over the task of one of it's colleagues. If no resource is available propagation nodes denote that a task can be carried out by a subcontractor. Figure 1 shows the option to use a subcontractor for the assembly operation.

In a decentral smart factory environment the production facilities are represented by agents. On receiving the result set of the possible production resources C_{Virt} can start negotiating with the agents representing those facilities. The negotiations can follow different strategies with related patterns and optimization criteria. Examples are bilateral trading, auctioning or request for proposals. A deep insight on this subject is given in [20].

4 Conclusion and Outlook

In this paper we present an approach that brings the concept of smart factories and Graph-based Resource Management together. Because necessary production

[7] For clarification issues we leave out the bill of materials.
[8] There is a implementation of the GBRM theory called C-Org.

resources are described instead of enumerated, the maintenance effort will be significantly reduced and at the same time security in sense of finding valid identities (here human or automatic resources) is increased. For example, if a facility or a new manufacturing partner is added to the manufacturing system, the only thing is to update the graph model. From this moment on, all requests of products will consider the new situation. Via the usage of deputy relationships the model is robust against disturbances like machine breakdowns, illness or cyber attacks targeting the availability of resources.

References

1. N. Gronau, "Industrie 4.0," *Enzyklopedie der Wirtschaftsinformatik*, 2015.
2. J. Meißner, "Cyberphysische Produktionssysteme," *Productivity Management*, vol. 18, no. 1, pp. 21–24, 2013.
3. G. Weiss, *Multiagent Systems. A Modern Approach to Distributed Artifical Intelligence*. Cambridge: MIT Press, 2000.
4. W. Shen, Q. Hao, H. J. Yoon, and D. H. Norrie, "Applications of agent-based systems in intelligent manufacturing: An updated review," *Advanced Engineering Informatics*, vol. 20, no. 4, pp. 415–431, 2006.
5. E. D. Simmon, K.-S. Kim, E. Subrahmanian, R. Lee, F. J. de Vaulx, Y. Murakami, K. Zettsu, and R. D. Sriram, "A vision of cyber-physical cloud computing for smart networked systems," 2013.
6. S. Wang, J. Wan, D. Zhang, D. Li, and C. Zhang, "Towards smart factory for industry 4.0: a self-organized multi-agent system with big data based feedback and coordination," *Computer Networks*, vol. 101, pp. 158–168, 2016. Industrial Technologies and Applications for the Internet of Things.
7. E. Hozdić, "Smart factory for industry 4.0: A review," *International Journal of Modern Manufacturing Technologies*, vol. 7, no. 1, pp. 28–35, 2015.
8. L. Monostori, "Cyber-physical production systems: Roots, expectations and r&d challenges," *Procedia Cirp*, vol. 17, pp. 9–13, 2014.
9. K. D. Bettenhausen and S. Kowalewski, "Cyber-physical systems: Chancen und nutzen aus sicht der automation," *VDI/VDE-Gesellschaft Mess-und Automatisierungstechnik*, pp. 9–10, 2013.
10. T. P. Raptis, A. Passarella, and M. Conti, "Data management in industry 4.0: State of the art and open challenges," *IEEE Access*, vol. 7, pp. 97052–97093, 2019.
11. E. M. Martinez, P. Ponce, I. Macias, and A. Molina, "Automation pyramid as constructor for a complete digital twin, case study: A didactic manufacturing system," *Sensors*, vol. 21, no. 14, 2021.
12. M.-F. Körner, D. Bauer, R. Keller, M. Rösch, A. Schlereth, P. Simon, T. Bauernhansl, G. Fridgen, and G. Reinhart, "Extending the automation pyramid for industrial demand response," *Procedia CIRP*, vol. 81, pp. 998–1003, 2019. 52nd CIRP Conference on Manufacturing Systems (CMS), Ljubljana, Slovenia, June 12-14, 2019.
13. A. Lawall, D. Reichelt, and T. Schaller, "Propagation of agents to trusted organizations," in *Web Intelligence (WI) and Intelligent Agent Technologies (IAT), 2014 IEEE/WIC/ACM International Joint Conferences on*, vol. 3, pp. 74–77, August 2014.

14. A. Lawall, T. Schaller, and D. Reichelt, "Restricted relations between organizations for cross-organizational processes," in *Business Informatics (CBI), 2014 IEEE 16th Conference on*, vol. 2, pp. 74–80, July 2014.
15. T. Schaller, *Organisationsverwaltung in CSCW-Systemen*. PhD thesis, Bamberg University, 1998.
16. A. Lawall, T. Schaller, and D. Reichelt, "Cross-organizational and context-sensitive modeling of organizational dependencies in c-org," in *S-BPM ONE (Scientific Research)*, (Heidelberg), pp. 89–109, Springer-Verlag, 2014.
17. A. Lawall, "Hypergraph-based access control using formal language expressions - HGAC," in *DATA 2015 - Proceedings of 4th International Conference on Data Management Technologies and Applications, Colmar, Alsace, France, 20-22 July, 2015.*, pp. 267–278, 2015.
18. A. Lawall, T. Schaller, and D. Reichelt, "Local-global agent failover based on organizational models," in *Web Intelligence (WI) and Intelligent Agent Technologies (IAT), 2014 IEEE/WIC/ACM International Joint Conferences on*, vol. 3, pp. 74–77, Nov 2014.
19. M. Reed, J. F. Miller, and P. Popick, "Supply chain attack patterns: Framework and catalog," *Office of the Deputy Assistant Secretary of Defense for Systems Engineering: Washington, DC, USA*, 2014.
20. T. Mannmäusel, *Dezentrale Produktionslenkung*. PhD thesis, Universität Bamberg, 1997.