Auto-configurable Event-Driven Architecture for Smart Manufacturing

Hui Cao, Xing Yang

Auto-configurable Event-driven Architecture for Smart Manufacturing

Hui Cao*, Xing Yang

Department of Industrial Engineering, Tsinghua University, Beijing, China
caohui@tsinghua.edu.cn

Abstract. In order to meet the ever-changing customer demands, smart manufacturing needs to intelligently integrate and coordinate the entire manufacturing supply chain. Event-driven architecture has been considered as a promising enabler for smart manufacturing. However, in the primitive event-driven architecture all components work in parallel, and there are few frameworks or standards that deal with organizing and managing the components. This paper proposes an ontology based holonic event-driven architecture for smart manufacturing systems. Powered by the developed ontology model and Semantic Web technology, the ontology based architecture enables the distributed components in a smart manufacturing system to be configured autonomously and collaborate with each other even when they don't know much about their counterpart.

Keywords: Event-driven architecture, Smart manufacturing system, Ontology

1 Introduction

Shorter product lifecycle, higher dynamic market, and massive customized requirements drive today's manufacturing systems to be smart. In the United States, a coalition of companies, universities, manufacturing consortia and consultants called the Smart Manufacturing Leadership Coalition (SMLC) was built to develop a shared infrastructure that enables the implementation of smart manufacturing [1]. In the meantime, starting from 2011 the German government initiated a project to promote smart factory in Germany, so called Industry 4.0. The smart manufacturing systems (SMS) have the ability to adapt to new situations by manufacturing intelligence with real-time data [2], and they need to be autonomous, self-aware, and self-correcting [3].

Over the last three decades, multiple architectures and approaches have been developed to meet the evolving requirements of manufacturing systems. Holonic Manufacturing Systems were expected to be appropriate for the next generation manufacturing systems as they provide a high and predictable performance with a high robustness against disturbances and unforeseen changes [4], and agent-based approach was considered as the most popular technology for implementation Holonic Manufacturing Systems. However, in spite of all the enthusiastic academic researches on agent-based manufacturing systems, only few industrial applications were reported owing to the conceptual and technology difficulties. With the emerging of the service oriented architecture, Service Oriented Multi-Agent System (SoMAS) is proposed for creating more complex, flexible, and adaptive systems by integrating the highly interoperable Web services and autonomous software agents [5-7].

With the recent development of IOT technology, SMS are able to capture more and more real-time data. These data are usually in high volume, high variety while not exactly accurate, which bring big challenges to the current manufacturing systems. In order to meet the ever-changing customer demands, smart manufacturing also needs to intelligently integrate and coordinate the entire manufacturing supply chain [8]. Backboned by complex event processing technology, event-driven architecture (EDA) can reduce the amount of information communicated in the system [9], and is known to be strong at processing high-volume complex event streams [10]. Furthermore, owing to its extremely loose coupled and highly distributable nature, EDA has advantages of integrating and coordinating the various organizations and their manufacturing resource and information resource throughout the supply chain. These features make EDA a promising enabler for smart manufacturing. Several EDA based manufacturing systems have been proposed in literature, e.g. [9], [11] and [12].

However, in the primitive EDA all the event producers and consumers work in parallel, and we haven't found any frameworks or standards that deal with organizing and managing event service components in an EDA. In this case, when integrating a large variety of objects in different levels of different organizations using the primitive EDA, management, interoperability, security and privacy issues may arise. In this paper we propose an ontology based holonic EDA (Oh-EDA) to solve the management difficulties in the primitive EDA. With the ontology model, distributed event service components can be automatically configured and integrated to an Oh-EDA system like plug and play, and the service components can collaborate with each other even when they don't know much about their counterpart. The access to the event messages is distributed controlled based on the ontology based access control rules. The basic building blocks, ontology model and access control rules of Oh-EDA are introduced in the next section, which is followed by an illustrative case presented in section 3; while section 4 concludes the paper with the discussion of future works.

2 Auto-configurable EDA

An EDA is a software architecture that detects and responds to events [13]. An EDA usually consists of components that detect events, listen events, process reaction to events, and transmit events or message among its components [14]. In event-driven systems, actions are triggered by occurrences of events.

In Oh-EDA, the event service components are organized as holons. Each event service component is embedded with an ontology-based service description file which clarifies the feature of the service as well as its input and output events. An ontology reasoner is employed to infer the proper configuration of the components and thus enables the components to be autonomously configured and deployed into the system.

2.1 Building blocks

The main building blocks in Oh-EDA include event services, event service managers, event message brokers, and event service assemblies (ESAs), as illustrated in Fig. 1. ESAs are the basic units that compose an Oh-EDA system. An ESA assembles a set of event services working collaboratively to achieve some common objectives. In Oh-EDA, an ESA is a holon, which can work independently and also can be a part of another ESA. In a SMS, an ESA can represent any organization in the smart manufacturing value chain, e.g. an assembly line, a plant, a supplier, or the whole supply chain. The key components in an ESA are explained below.

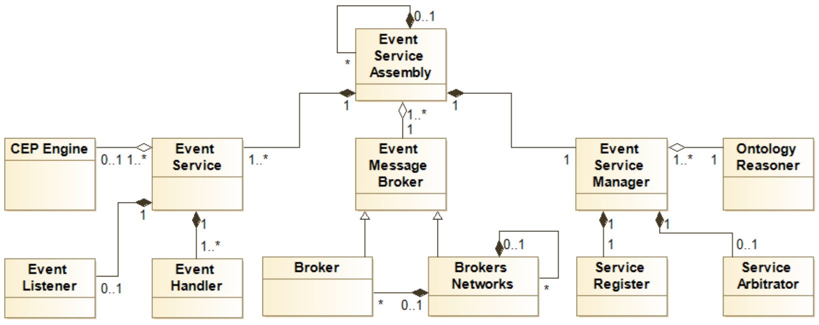


Fig. 1. Building blocks in Oh-EDA

Event service: An event service can be an event producer, an event processor, an event consumer, or their combination. The event service listens the events it subscribed via an *event listener*, and reacts to the events via *event handlers*. The events may need to be processed by a *CEP engine*. The CEP engine can be part of the service or be shared among various event services. The event handlers and the CEP engine may generate new events and publish them to the system via an *event message broker*.

Event message broker: The event message broker is the backbone of the event driven system. It enables event publishing, subscribing, and transmitting among the services. The message broker can be a single message *broker* or a distributed *network of brokers*.

Event service manager: The event service manager is responsible for the coordination, mediation, and management of the event services in an event service assembly. Event services have to be registered into the event service assembly via a *service register* before they can subscribe or publish events. In the registration, the service's access to different categories of events is granted or restricted according to the access rights that are deduced by an *ontology reasoner* based on predefined ontology-based access control rules. The *service arbitrators* are employed to solve possible conflicts among event services.

2.2 Ontology

An ontology is a formal, explicit specification of a shared conceptualization [15]. In this study, we propose an ontology model for event driven systems (EDO) that models the main entities, their relationships and related rules in an event-based SMS. We use OWL DL with Semantic Web Rules Language (SWRL) to represent EDO. OWL DL is a sublanguage of Web Ontology Language (OWL), and it permits efficient reasoning support [16]. SWRL is a rule extension of OWL DL, enabling Horn-like rules to be combined with an OWL knowledge base [17].

The top-level concepts in EDO include *Organization*, *Resource*, *Service*, and *Event*. Each of the concepts is a member of OWL class *owl*:*Thing*. The top-level concepts and their relationships are illustrated in Fig. 2. The *Organization* concept represents any organizations in the smart manufacturing value chain. The *Service* concept models the event-based services provided in the organizations, e.g. production scheduling, quality control, performance measurement, and cost accounting. The *Event* concept depicts any event message generated, published or subscribed by the services, while the *Resource* concepts represent all kinds of resources involved in the services.

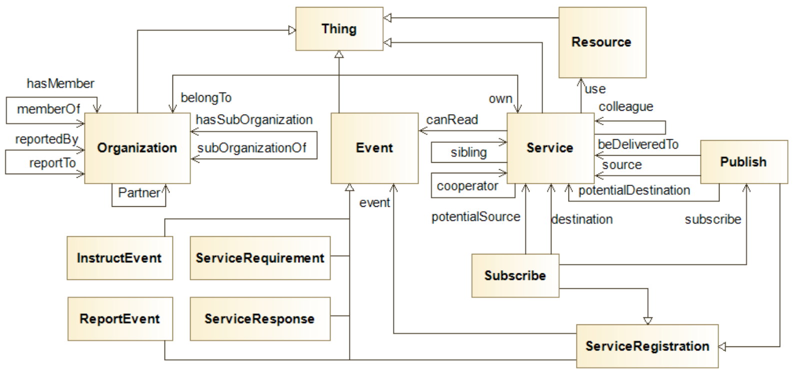


Fig. 2. Top-level concepts and their relations in EDO

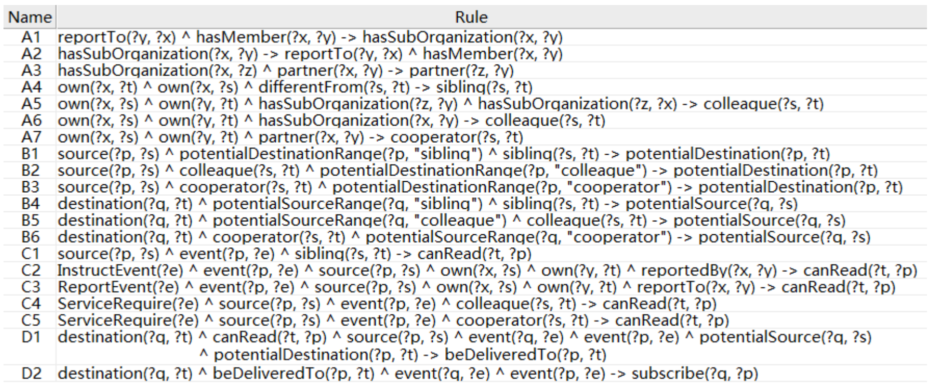
EDO supports a variety of organizational structures. Generally, an organization can be a *member* of another organizations, e.g. a manufacturer in a supply chain. The *reportTo*/*reportedBy* relationship depicts the chain of command either in a hierarchy, departmental, or matrix organization. Furthermore, an organization can be a *partner* of another organization, if they collaborate with each other but they are not in a same chain of command. The other relationships between the organization entities and the service entities are defined formally via SWRL rules A1 to A7 (represented with the SWRLAPI SWRL Syntax[1] used in the Protégé project) as displayed in Fig. 3. Among the object properties, *hasSubOrganization*, *memberOf* and *reportTo* are transitive properties.

2.3 Access control rules

In Oh-EDA, the access to the event messages is controlled via the distributed service registers based on the message subscribing and publishing mechanism. The event service components take the primary responsibility for the access control. When a service

[1] Please refer to https://github.com/protegeproject/swrlapi/wiki/SWRLAPISWRLSyntax

is registered into the system, it sends a set of *Publish* events and *Subscribe* events to the corresponding service register, stating the type of the event messages which the service is willing to publish and subscribe. The *Publish* events also suggest the accepted destinations of the event messages, while the *Subscribe* events suggest the potential event sources. In EDO, data properties *potentialDestinationRange* and *potentialSourceRange* have been defined for the *Publish* events and the *Subscribe* events respectively. The *potentialDestinationRange*/*potentialSourceRange* can be set as "sibling", "colleague" or "cooperator" to define the scope of the potential destination/source services instead of designating a specific service. Rules B1 to B6 shown in Fig. 3 are used to interfere the corresponding potential event destinations and sources for the publishing and subscribing.



Name	Rule
A1	reportTo(?y, ?x) ^ hasMember(?x, ?y) -> hasSubOrganization(?x, ?y)
A2	hasSubOrganization(?x, ?y) -> reportTo(?y, ?x) ^ hasMember(?x, ?y)
A3	hasSubOrganization(?x, ?z) ^ partner(?x, ?y) -> partner(?z, ?y)
A4	own(?x, ?t) ^ own(?x, ?s) ^ differentFrom(?s, ?t) -> sibling(?s, ?t)
A5	own(?x, ?s) ^ own(?y, ?t) ^ hasSubOrganization(?z, ?y) ^ hasSubOrganization(?z, ?x) -> colleague(?s, ?t)
A6	own(?x, ?s) ^ own(?y, ?t) ^ hasSubOrganization(?x, ?y) -> colleague(?s, ?t)
A7	own(?x, ?s) ^ own(?y, ?t) ^ partner(?x, ?y) -> cooperator(?s, ?t)
B1	source(?p, ?s) ^ potentialDestinationRange(?p, "sibling") ^ sibling(?s, ?t) -> potentialDestination(?p, ?t)
B2	source(?p, ?s) ^ colleague(?s, ?t) ^ potentialDestinationRange(?p, "colleague") -> potentialDestination(?p, ?t)
B3	source(?p, ?s) ^ cooperator(?s, ?t) ^ potentialDestinationRange(?p, "cooperator") -> potentialDestination(?p, ?t)
B4	destination(?q, ?t) ^ potentialSourceRange(?q, "sibling") ^ sibling(?s, ?t) -> potentialSource(?q, ?s)
B5	destination(?q, ?t) ^ potentialSourceRange(?q, "colleague") ^ colleague(?s, ?t) -> potentialSource(?q, ?s)
B6	destination(?q, ?t) ^ cooperator(?s, ?t) ^ potentialSourceRange(?q, "cooperator") -> potentialSource(?q, ?s)
C1	source(?p, ?s) ^ event(?p, ?e) ^ sibling(?s, ?t) -> canRead(?t, ?p)
C2	InstructEvent(?e) ^ event(?p, ?e) ^ source(?p, ?s) ^ own(?x, ?s) ^ own(?y, ?t) ^ reportedBy(?x, ?y) -> canRead(?t, ?p)
C3	ReportEvent(?e) ^ event(?p, ?e) ^ source(?p, ?s) ^ own(?x, ?s) ^ own(?y, ?t) ^ reportTo(?x, ?y) -> canRead(?t, ?p)
C4	ServiceRequire(?e) ^ source(?p, ?s) ^ event(?p, ?e) ^ colleague(?s, ?t) -> canRead(?t, ?p)
C5	ServiceRequire(?e) ^ source(?p, ?s) ^ event(?p, ?e) ^ cooperator(?s, ?t) -> canRead(?t, ?p)
D1	destination(?q, ?t) ^ canRead(?t, ?p) ^ source(?p, ?s) ^ event(?q, ?e) ^ event(?p, ?e) ^ potentialSource(?q, ?s) ^ potentialDestination(?p, ?t) -> beDeliveredTo(?p, ?t)
D2	destination(?q, ?t) ^ beDeliveredTo(?p, ?t) ^ event(?q, ?e) ^ event(?p, ?e) -> subscribe(?q, ?p)

Fig. 3. SWRL rules in EDO

The publishing and subscribing are then reviewed based on the related event types and the organizations' policy. In EDO, the *Event* entities are subclassified as *InstructEvent*, *ReportEvent*, *ServiceRequirement*, *ServiceResponse*, and *ServiceRegistration*. The *ReportEvent* and *InstructEvent* depict the event messages disseminated up and down the chain of command respectively. The *ServiceRequirement*/*ServiceResponse* event represents the event messages that requires a service or response to a service requirement. The *ServiceRegistration* events are further specialized to the *Publish* events and the *Subscribe* events as stated before, modeling the publishing or subscribing of a type of events by a service. Rules C1 to C5 in Fig. 3 demonstrate some examples to define the organizational access control policy. Rule C1 states that an event message created by a service can be accessed by the siblings of the service. Rule C2 allows the *InstructEvent* entities to be delivered down the chain of commands, while rule C3 allows the *ReportEvent* entities to be delivered up the chain of commands. Rule C4 and C5 let the *ServiceRequirement* messages to be disseminated to all services in the service's organization and its partner organizations. Of course, these rules can be tailored by the organization based on its own security and privacy policies.

If a subscribing event and a publishing event are matched with each other and the organizational access control policies are followed, the subscribing and publishing will be registered to the system, as stated in rules D1 and D2 displayed in Fig. 3.

3 An illustrative case in smart manufacturing

Service components in Oh-EDA can be seen as agents in the agent-based architecture. Therefore Oh-EDA has all advantages that an agent-based architecture has. Besides, Oh-EDA has more desired capabilities and meets the requirements of SMS in many aspects, e.g. autonomous configuration, plug-and-play deployment, highly interoperable, easy to adapt, highly scalable and robust.

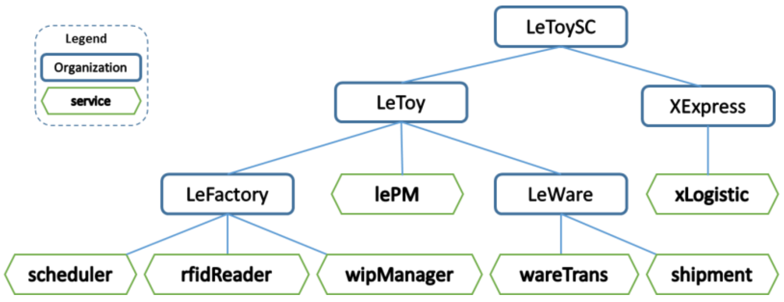
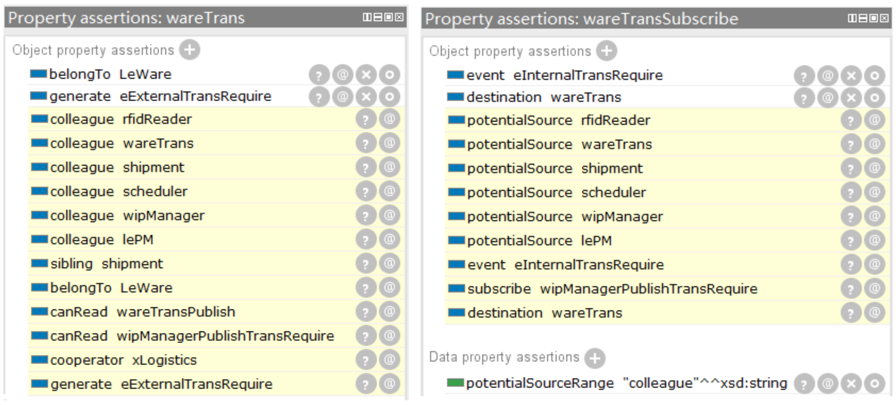


Fig. 4. The demonstration case

In this section we present a tiny show case to demonstrate the use of Oh-EDA in a SMS as shown in Fig. 4. The case involves a toy manufacturer named *LeToy* and a third party logistics provider *XExpress*. LeToy has a manufacturing plant *LeFactory* and a warehouse *LeWare*. RFID technology is employed in the factory to monitor the movement of WIPs and the *rfidReader* service is responsible for translating the RFID data into the WIPs' movement events. LeFactory owns a scheduling service *scheduler* which receives customized orders and determines the production schedule. The production schedule is then disseminated to the *wipManager* service, which dispatches the real-time production tasks to each workstations and controls the production processes based on the WIPs' movement events. For an instance, if a WIP leaves its final workstation and enters the dispatch zone, the *wipManager* would conclude that the production has been done. Therefore it would publish an order-status-change event and require a transportation of the final product to the warehouse. The *wareTrans* service owned by LeWare provides such transportation service. Once the final product is delivered to the shipment area, the *shipment* service will publish a transportation requirement to its third party logistics partner. Besides, LeToy measures its performance via the *lePM* service, which uses the information provided by the other services, e.g. order-status-change, to calculate the performance measures. Table 1 shows a part of OWL code in Turtle syntax for representing the knowledge about the SMS. In Table 1, namespace *edo* refers to the general EDO concepts and relations, and namespace *edsms* presents the concepts in the manufacturing domain. Fig. 5 shows the inferred knowledge (the properties with the yellow background) for the wareTrans service and the wareTransSubscribe event using Protégé reasoner tool. It shows that the transportation requirements from the wipManager service will be delivered to the wareTrans service.

Table 1. Sample OWL code in Oh-EDA system

```
:LeToy rdf:type owl:NamedIndividual , edo:Organization ;
        edo:memberOf letoysc:LeToySupplyChain ;
        edo:partner xexpress:XExpress .
:LeWare rdf:type owl:NamedIndividual , edo:Organization ;
        edo:subOrganizationOf :LeToy .
:wareTrans rdf:type owl:NamedIndividual , edsms:TransportService ;
        edo:belongTo :LeWare ;
        edo:generate edsms:eExternalTransRequire .
:wareTransPublish rdf:type owl:NamedIndividual , edo:Publish ;
        edo:event edsms:eExternalTransRequire ;
        edo:source :wareTrans ;
        edo:potentialDestinationRange "cooperator"^^xsd:string .
:wareTransSubscribe rdf:type owl:NamedIndividual , edo:Subscribe ;
        edo:destination :wareTrans ;
        edo:event edsms:eInternalTransRequire ;
        edo:potentialSourceRange "colleague"^^xsd:string .
```



Property assertions: wareTrans

Object property assertions

- belongTo LeWare
- generate eExternalTransRequire
- colleague rfidReader
- colleague wareTrans
- colleague shipment
- colleague scheduler
- colleague wipManager
- colleague lePM
- sibling shipment
- belongTo LeWare
- canRead wareTransPublish
- canRead wipManagerPublishTransRequire
- cooperator xLogistics
- generate eExternalTransRequire

Property assertions: wareTransSubscribe

Object property assertions

- event eInternalTransRequire
- destination wareTrans
- potentialSource rfidReader
- potentialSource wareTrans
- potentialSource shipment
- potentialSource scheduler
- potentialSource wipManager
- potentialSource lePM
- event eInternalTransRequire
- subscribe wipManagerPublishTransRequire
- destination wareTrans

Data property assertions

- potentialSourceRange "colleague"^^xsd:string

Fig. 5. Inferred properties for *wareTrans* and *wareTransSubscribe*

4 Conclusion and further work

This paper reports an ongoing work towards the auto-configurable SMS based on Semantic Web technology and event-driven architecture. The basic building blocks of the architecture and the top level ontology concepts, relations and access control rules have been developed. While the manufacturing domain specific ontologies and access control rules need to be further elaborated, and an industrial scale manufacturing system is under development for verifying and evaluating our proposed architecture.

Acknowledgement

The work was supported by the National Natural Science Foundation, China (71301084 and 71332005).

References

1. Davis, J., Edgar, T., Porter, J., Bernaden, J., Sarli, M.: Smart manufacturing, manufacturing intelligence and demand-dynamic performance. Comput Chem Eng 47:145-156 (2012).
2. Kumaraguru, S., Morris, K.C.: Integrating real-time analytics and continuous performance management in smart manufacturing systems. IFIP International Conference on Advances in Production Management Systems. Springer (2014).
3. Feeney, A.B., Frechette, S.P., Srinivasan, V. A portrait of an ISO STEP tolerancing standard as an enabler of smart manufacturing systems. J Comput Inf Sci Eng 15: 021001 (2015).
4. Babiceanu, R.F., Chen, F.F.: Development and applications of holonic manufacturing systems: a survey. J Intell Manuf 17:111-131 (2006).
5. Leitão, P.: Agent-based distributed manufacturing control: A state-of-the-art survey. Eng Appl Artif Intell 22:979-991 (2009).
6. Leitão, P.: Towards self-organized service-oriented multi-agent systems. in Service Orientation in Holonic and Multi Agent Manufacturing and Robotics 41-56. Springer (2013).
7. Del, V.E., Rebollo, M., Botti, V.: Enhancing decentralized service discovery in open service-oriented multi-agent systems. Autonomous agents and multi-agent systems 28:1-30 (2014).
8. SMLC: Implementing 21st century smart manufacturing. Workshop Summary Report (2011).
9. Kasakow, G., Menck, N., Aurich, J.C.: Event-driven production planning and control based on individual customer orders. Procedia CIRP 57:434-438 (2016).
10. Dunkel, J., Fernández, A., Ortiz, R., Ossowski, S.: Event-driven architecture for decision support in traffic management systems. Expert Syst Appl 38:6530-6539 (2011).
11. Kong, J., Jung, J.-Y., Park, J.: Event-driven service coordination for business process integration in ubiquitous enterprises. Comput Ind Eng 57:14-26 (2009).
12. Theorin, A., Bengtsson, K., Provost, J., Lieder, M., Johnsson, C., Lundholm, T., Lennartson, B.: An event-driven manufacturing information system architecture for Industry 4.0. Int J Prod Res 55:1297-1311 (2017).
13. Chandy, K.M.: Event driven architecture. In: Liu, L., Özsu, M.T. (eds.) Encyclopedia of Database Systems 1040-1044. Springer (2009).
14. Taylor, H., Yochem, A., Phillips, L., Martinez, F.: Book Event-driven architecture: How SOA enables the real-time enterprise. Pearson Education (2009).
15. Studer, R., Benjamins, V.R., Fensel, D.: Knowledge engineering: principles and methods. Data Knowl Eng 25:161-197 (1998).
16. Antoniou, G., Van Harmelen, F.: Web ontology language: Owl. In: Staab, S., Studer, R. (eds.) Handbook on Ontologies 67-92. Springer (2004).
17. Masoumzadeh, A., Joshi, J.: Osnac: An ontology-based access control model for social networking systems. In: IEEE Second International Conference on Social Computing (SocialCom) 751-759. IEEE (2010).