System for Semantic Technology-based Access Management in a Port Terminal

R. Szczekutek[1,b)], M. Ganzha[1,2,c)], M. Paprzycki[2,6,d)], S. Fidanova[3], I. Lirkov[3], C. Badica[4] and M. Ivanovic[5]

[1]*Faculty of Mathematics and Information Sciences, Warsaw University of Technology, ul. Koszykowa 75, 00-662 Warsaw, Poland*
[2]*Systems Research Institute, Polish Academy of Sciences, ul. Newelska 6, 01-447 Warsaw, Poland*
[3]*Institute of Information and Communication Technologies, Bulgarian Academy of Sciences, Acad. G. Bonchev, bl. 25A, 1113 Sofia, Bulgaria*
[4]*Software Engineering Department, Faculty of Automatics, Computers and Electronics, University of Craiova, 107 Decebal, Blvd., Craiova, RO-200440, Romania*
[5]*University of Novi Sad, Faculty of Sciences, Department of Mathematics and Informatics Trg Dositeja Obradovica 4, 21000 Novi Sad, Serbia*
[6]*Faculty of Informatics, Warsaw Management Academy, Kaweczynska 36, 03-772 Warsaw, Poland*

[a)]Corresponding author: szczekutekrafal@gmail.com
[b)]szczekutekrafal@gmail.com
[c)]M.Ganzha@mini.pw.edu.pl
[d)]Marcin.Paprzycki@ibspan.waw.pl

Abstract. Today's dynamic growth of the number of deployed solutions, based on the Internet of Things (IoT) concept, goes hand-in-hand with the need for new access control mechanisms. The Attribute Based Access Control (ABAC) paradigm seems to be a good alternative to the, commonly used, Role Based Access Control (RBAC) model, which does not necessarily work well in large-scale distributed systems. Specifically, the main drawback of the policy-based authorization solutions is the resulting high complexity of required policies, especially in the case of fine-grained access control (a clear case of lack of scalability). We believe that this problem can be appeased by incorporating semantic technologies into the decision-making process. In this paper we discuss a system for managing access control in a port terminal, based on application of semantic technologies. The proposed system is based on introduced and uses a modified version of the OntoPlay semantic application. The latter allows non-specialists to easily perform needed modifications of the ontology. The proposed system has been tested in a semi-realistic environment and in an emulator.

1 INTRODUCTION

Access control plays an important role in everyday life. Many resources have restricted access and are available only to a selected group of people. Recently, control of access becomes even more important, in the context of "smart devices", being capable of "talking to each other". Such machine to machine (M2M) communication enforces application of new authorization methods. Note that M2M access management can involve not only scenarios, in which M2M communication facilitates human access to some areas (*e.g.*, certain workers may have access to certain rooms within a building). It is also possible that M2M interactions may involve access to certain documents (*e.g.*, in a hospital, doctor may have access to different medical documents of a patient than the financial administrator). This being the case, let us note that while, in what follows, we focus on the first situation, presented results naturally apply also to the second.

Taking into account fast growing number of connected / interacting devices, it becomes obvious that, widely used, role-based access model is not good enough. Implementation of a fine-grained access using role-based approach (in large systems), very often leads to creation of an extremely large number of roles (so called, role explosion) [1]. This is partially related to the fact that role-based access control does not take into account any contextual data that could be assigned to a request for access. Such, contextual, data could reduce the number of role-based rules. Here, it has been noticed that Attribute Based Access Control seems to be better suited to handle such situations. Access policies may rely on attributes, values of which are being defined dynamically. This makes it possible to include context of the request into request evaluation. The problem is, that attribute-based policies also tend to grow and become complex and hard to maintain, especially with high granularity of access control [2, 3, 4]. Based on earlier work ([2, 5]), we believe that this problem of attribute based access control can be mitigated by inclusion of semantic technologies into the access-decision-making process. To be able to apply semantic technologies, the domain of a given access control system has to be *modeled in the form of an ontology*. Such ontology should reflect, in a formalized way, key components of the domain, and their relations. Obviously, such domain ontology does not have to be "created from scratch". As a matter of fact, if a domain model ontology exists, it should be re-used. However, care needs to be taken that it *captures all concepts necessary for access management*. Domain ontology allows application of a semantic reasoner, which will allow inferring values of attributes not explicitly defined in an access request. Using such "extra" (inferred) values, in an access policy, can decrease complexity of access management (see, also, Section 3). A solution that follows this approach has been described in [2, 5]. It introduces the SXACML system, which is based on the XACML standard for access control, which has been empowered with semantic reasoning.

The problem with the ontology-based systems is that it might be necessary to modify the ontology during its lifetime. One of basic examples is the need to adjust it to the dynamically changing domain. For instance, over time, new roles or permissions may need to be functionalized. Some modifications may be enforced by changes in the law. Finally, even simple changes situations where new contracts are signed and new partners (attribute values in policies) need to be added to the system (*e.g.*, Allied trucking company is allowed to access terminal of USM shipping within the port of New Orleans). In this context, currently available tools (that allow ontology management), have been developed to support semantic technology experts. This significantly decreases the chances of applying ontology-based systems in the real world. For instance, it is difficult to envision that each large port facility will hire an ontology engineer to support its access control systems. However, the OntoPlay project described in [3, 6] provides a potential solution to this problem. OntoPlay is an intuitive minimalist ontology management tool, dedicated to users inexperienced in semantic technologies. It has two main functionalities. First, it makes it possible to modify the underlaying ontology directly in a web browser. Here, modifying the ontology with OntoPlay should be relatively easy even for people with no familiarity with semantic technologies. This is because only a limited number of modifications is supported (see, section 3.4). Second, it allows building interfaces that are dynamically adjusting to the underlying ontology. Specifically, any ontology, defined in the RDF format, may be loaded and the interface will "represent it" allowing, for instance, formulating semantic queries. Moreover, if the underlying ontology is modified, *e.g.*, using OntoPlay itself (as above), the interface will immediately adapt itself to the change(s). Here, let us stress that, as will be seen (in section 3.4), OntoPlay is *not* a general ontology management tool (*e.g.*, with functionalities similar to Protege [7]). It is primarily an ontology-driven Web-interface, with selected, need-driven, functionalities added to support situations materializing in system like the one that is the topic of this work.

In this paper we focus on the problem of allowing subjects, with no explicitly assigned permissions, to access (a) specific resource(s). Access is to be based on the dynamically changing attributes and use of reasoner to establish if an entity with a given set of, explicitly stated and derived, attributes should be granted access to a given resource. It may often be the case that there are some explicitly defined privileges for a subject to access a particular resource. This case is simple for the system to handle. The only thing needed to be done is the verification of those privileges against the incoming access requests. The question is, how can the system automatically deduct access rights for an unknown, thus far, user. Simple answer could be to reject the access in all such cases. However, there might be a situation when such user should be granted access. That could depend on the relations between the subject and other entities from the system domain. One of those cases is when an unknown user inherits the rights from some other user that already has them. In such case, the system should grant access, even when no permissions were declared explicitly for a given user. Note that, this is precisely the situation when the relation between users could be stored in the ontology and necessary rights inferred with a semantic reasoner.

Without loosing generality, we have decided to study a particular use case, consisting of management of access control in a port terminal. Here, the access control system is expected to infer the access rights of truck drivers visiting the port. One of the possibilities is that the driver works for a company that signed a contract directly with the port authorities. More complex situation is when the driver's employer has an independent contract with a company that has access rights to the terminal (but does not have such explicit contract/access rights itself). It is desired that the driver of the subcontracting company will be granted access to the port terminal.

Upon consultations with representatives of an actual port (Valencia, Spain) we have established that the needed system should be equipped with an authorization element, dedicated to verifying incoming driver's permissions. Here, the actual "workflow" should be as follows. When the driver is visiting the port, (s)he is going to contact the system using a dedicated mobile application, in order to inform about the purpose of the visit. The application appends information about the company employing the driver to the message. The system uses received data in order to infer whether the driver's employer is a company with permissions to visit the port. The created port ontology is used for inferring the relation between the company and the port. With this approach, it should be possible to keep the access policy simple and concise. The basic rule is stated for the driver that is working for a company that has been contracted by the (Valencia) port. The transitivity of the relation, representing contracts between the companies, makes it possible to grant access to the subcontractors without the need to have them sign the contract directly with (Valencia) Port Authorities. Obviously, it is still necessary to add information about the company to the ontology, but with help of the OntoPlay software it is simple and intuitive.

The aim of this paper is to discuss, in detail, how the proposed system has been designed, implemented and tested. To this effect, in Section 2 we present analysis of requirements regarding the created system. In section 3, the structure of the system is explained, together with its functional aspects. Testing process is described in section 4.

2 RELATED WORK

One of interesting examples, regarding access control, is a case of large seaport facility. Such a port typically consists of multiple terminals that belong to different stakeholders. This results in complex structures and procedures of access control management. Ports have different methods for dealing with this problem. For instance, terminal in Portland introduced an authorization system based on magnetic cards that need to be scanned, while accessing the port area [8]. While far more advanced than access control managed based on "standard paperwork", this approach can still be improved. Further studies on the access control, in case of port terminals, led us to the seaport located in Valencia, Spain. It has been noticed, during the research, that the aforementioned facility is operated by an advanced IT system with a plan for extending it with even more complex functionalities. The interesting thing is that Port of Valencia already has some access control automation solution. It increased the capacity of their terminals significantly, making it far more productive [9]. Despite that, they still claim that there are areas in their access control mechanism that could be improved. Their system is not yet capable of dealing with some specific cases, in terms of making access decision for all "types" of arriving truck drivers. The system handles basic scenario, which assumes that the driver visiting the port is an employee of a company that signed proper contract with Valencia Port Authorities. For definiteness, let's call this company A. In such case, the driver is permitted to enter the port area. The problem is with more complex scenarios. There are situations when the port is visited by a driver that works for a company B; where B is a subcontractor of A. B signed a contract with A, but it has not signed any contract with Valencia Port Authority. It is expected that the system will grant access to driver representing B, because (s)he arrives on behalf of a company having permissions to enter the port. This scenario is not automatically managed, by the Valencia port IT system. However, it seems that it could be handled, with the usage of semantic technologies and attribute based access control mechanisms. Existence of relation between companies A and B could be reflected as a relation between two individuals in an ontology. Port personnel would need to add new company in the form of individual to the ontology, and reflect its contract with an existing company as a proper relation. After that, the resulting access rights would be inferred by applying semantic reasoning, on the access request, against the ontology. With this approach, the decision-making process could be automated even for the complex scenario. We considered this as an opportunity for making the Valencia port system even better and verify the possibility of applying ideas from [2] and [3] at the same time.

Interestingly, the proposed approach could also help solving one more issue that actually exists in the Port of Valencia – smart lighting management. Valencia Port wanted to have an intelligent lighting system that would dynamically adjust brightness, depending on the location of trucks in a terminal. This was supposed to work as an IoT solution. The trucks were to communicate their current location to the lights. Based on that information, lights should appropriately adjust their brightness. After all, system implemented in the port is able to communicate with the vehicles, but only those belonging to the terminal. This indicated the need to have port vehicles escorting every truck visiting the port. There was no interface, for the trucks / truck drivers to communicate with the system. This was not the most desired solution, so it has been decided to go for the movement detectors installed on every light and use them for the brightness adjustment. The problem with that is the precision of the detectors and the need to have a straight view on the moving truck. In the initial approach, no obstacles nor environment conditions would cause any problems. It seems that the system proposed in this paper could be used to implement the original idea. The problem with the lack of interface for the drivers to communicate with the system could be handled by the implementation of a dedicated mobile application. Based on popularity of modern mobile phones, this solution should be easily applicable and accessible. Drivers would install the application on their mobile phones and use it to communicate with the system. There would be no need for creating a dedicated hardware (*e.g.*, transponders) for that purpose. Truck's position could be established using phone's GPS (since phones would be connected to the power source in the truck, this would avoid problems with battery drainage when the GPS module is turned on). The system would then propagate the location information to the lights having them automatically adjusted. The drivers could be authenticated by the system using user accounts functionality that is to be build into the system in the first place.

Note that the same accounts mechanism could be used for identifying other system users, i.e. transportation company employees and port personnel. This would be used for limiting the access to some of the system functionality, mainly the ontology management. Here, for instance, three different privilege levels could be defined for that purpose: driver, company administrator, and administrator. The first two should operate only in the context of a given company being their employer. With this approach, the system would also protect stored data from unauthorized access.

The aforementioned mobile application could also be used by the drivers to inform the system about the purpose of their visit. The system could use that for locating the desired cargo and sending its position, together with protecting gates' positions to the driver. This would facilitate navigation through the port. With this background, let us move toward describing the developed solution and the main contributions of this work.

3 SUGGESTED SOLUTION

As outlined above, the proposed access management system has been designed on the basis of (1) analysis of solutions available in the area, and (2) requirements that materialize in the case of management of access to Valencia Port facility. Before proceeding, let us stress that this solution is, to a large extent, generic. In other words, it can be applied in multiple other domains. In each such case, domain ontology will have to be instantiated (created, re-used directly, or adapted). This ontology has to capture (and formally represent) all aspects of the domain that are crucial to access control. Here, it can be easily imagined that an existing domain ontology will have to be extended by adding a module consisting of concepts that are not core for the domain itself, but are important for access control. For example, in our earlier work ([5]), we had to represent family relations to be able to build a rule that if C is a father of Z and if Z is underage, then C has the right to access her medical records. Furthermore, application of mobile technologies is also a generic one, as today, in almost any context, availability of a mobile device, can be assumed. Furthermore, the access controlling module is created as an independent one and thus can be used in any other scenario. Finally, let us note that, as stated in [2], the access control module is a non-disruptive extension of a standard XACML architecture. This, further, shows the generic nature of the proposed approach. Let us now look into its details.

3.1 Architecture

Let us now consider how the proposed access control system can be implemented in practice. For this purpose we propose architecture presented in Figure 1. Here, the system consists of the Server – the core element, which binds all the remaining modules: Database, Authorization server, Mobile application and Port facility programs.

The core responsibility of the **Server** is to communicate with other parts of the system. First of all, it is supposed to exchange information with the drivers through a *Mobile application*. It also needs to coordinate the work with systems within *Port facilities* (*i.e.*, gates and lights) by letting them know about the incoming truck(s). *Server* provides also a web user interface for the users, as an entry point for accessing its functionalities. Those are, mainly: accounts management, ontology management/modification, and verification of the situation in the port. The *Server* module facilitates existence of three different roles in the system. The account assigned *Admin* role, has a full set of privileges to all system functionalities. It is dedicated to the system administrators as well as port authorities. It enables creation of (other) user accounts, including those with Admin role assigned to them. The *Company* role, is supposed to be used by the employees that will be managing the accounts assigned to their company. It allows to relieve the port staff from getting overloaded with account management tasks, making this management distributed. Lastly, the *Driver* role, is created for the drivers visiting the (Valencia) port terminal. This type of account is assigned the lowest level of privileges, narrowed down to the necessary minimum. The last major role of the server is to store all of the system's data.

Mobile application, represents an interface for the drivers visiting the port terminal, to communicate with the system. Its primary use is to authorize drivers. Moreover, it is used for sending the location of the truck as well as the

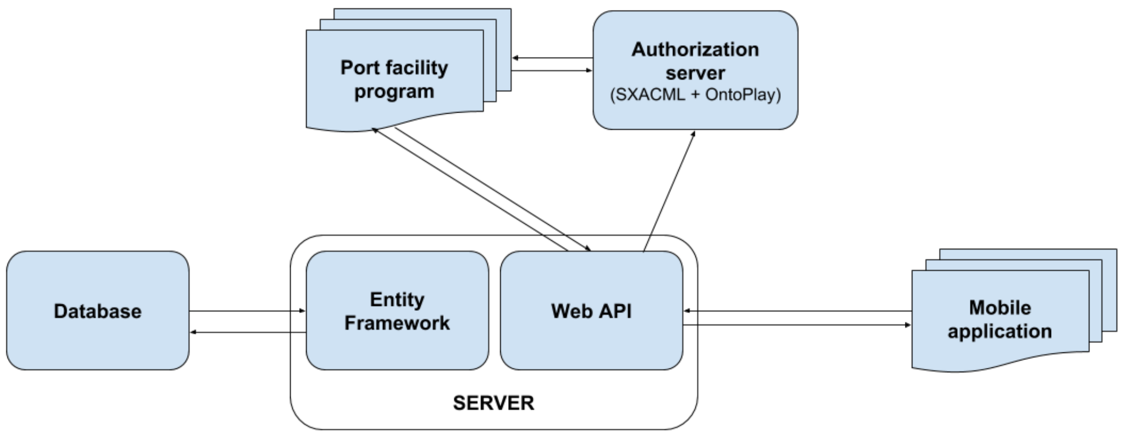


Figure 1. System architecture

visit purpose to the system. It is given the position of the cargo and intermediate gates in return, so the application is to provide some navigation functionality. This functionality is also used within the intelligent lights functionality. Note that the location functionality can be facilitated by any mapping software. Finally, for achieving greatest availability, mobile application should work on every commonly used mobile operating system, i.e. Android, iOS (and, possibly, Windows Phone/Mobile).

Port facility systems represent "local" systems that the access management will have to be interfaced with. In the context of our work, they involve functionality of gates and lights of the port. In our current considerations we do not take into account any other types of facilities present in the port. However, if need arises, they can be easily interfaced with our system. Note also, that this module would be replaced in case of management of access control to other facilities. Particular program, representing facilities to be accessed, should register related object in the system, listen for the messages sent by the *Server*, and send information about the facility state updates. Moreover, gate programs are supposed to verify the entrance permissions, for a particular driver, by connecting to the *Authorization server*. The importance of the proposed system is followed by a desire to achieve high availability and reliability, which makes it necessary to avoid single points of failures (SPOF). Following this requirement, it should be possible to dynamically attach (and detach) gates and lights to the system, without the need to restart it. This would make the system insusceptible to failures of particular gates or lights. Moreover, with this approach facilities rearrangements and maintenance breaks would not affect the whole system.

The **Database** is supposed to store all of the needed system data. Proposed model is shown in Figure 2. All tables reflect corresponding system entities. Most of them are connected with each other, using a one-to-many relationship. However, there are two distinctions. (1) The *StreetLight* table has no links at all. The main reason for that is the idea of treating port lights as independent facilities. They are not assigned to the particular areas of the terminal as we don't want to limit their operational range. If one can light up the road for the truck driver in its range it should do so, no matter what area does it stand on. (2) The second exception is the many-to-many relationship between the *LoadingDock* and *Company* tables. Actually, this relation has been simplified in the presented figure. In reality, this relation is mapped with an additional table called *CompaniesDocksAccess*. It binds the areas of the port with the companies that are privileged to visit them. These are only static privileges, an optional, deficient mechanism for the proposed policy-based access control. There is one more link that looks different from the rest. It is an arrow leading from the *LoadingDock* to the *Area* table. It symbolizes the fact that loading dock is a subtype of a port area. Relation between the *Gate* and *Area* represents the fact that an area of the port could by accessible by more than one gate. The *Area* table is pointing to itself. This corresponds to the idea that port areas may be organized in a hierarchical structure. Some of them may be located inside another. Single loading dock may contain many different cargoes. One cargo may be the target of multiple different visit purposes. Every purpose can be assigned to any number of visits. Each truck will most probably be a subject of many different visits. A company is expected to be in possession of multiple trucks and cargoes.

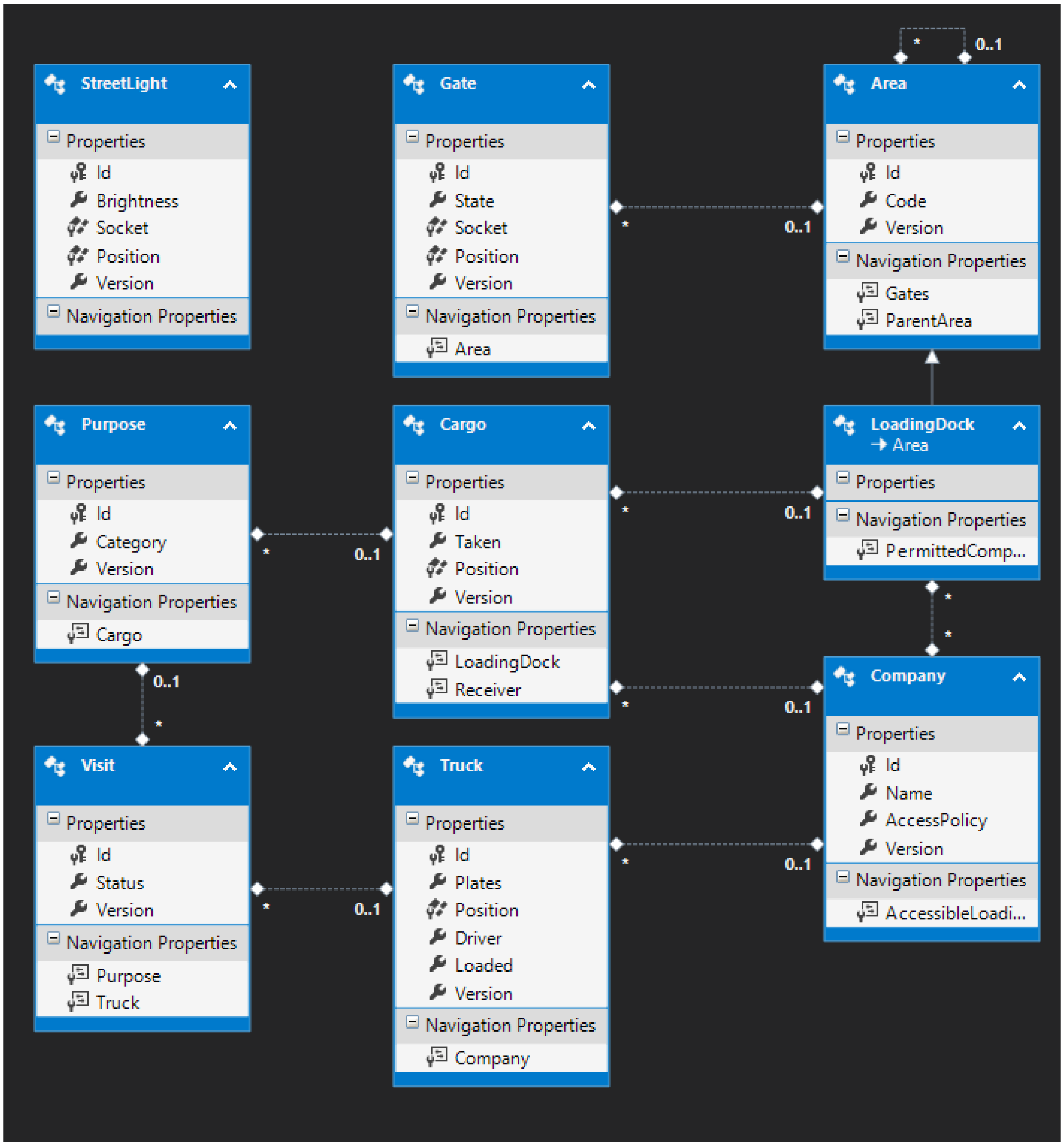
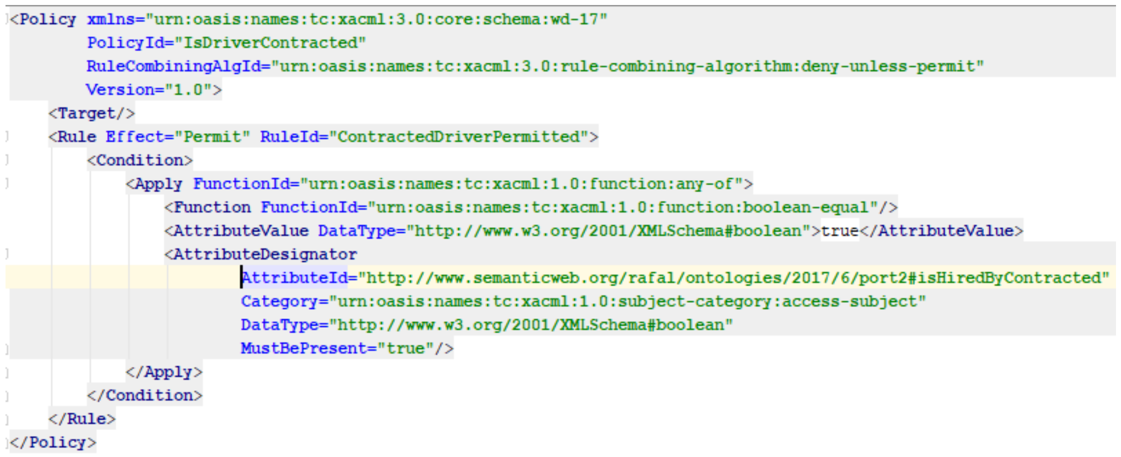


Figure 2. Database structure

3.2 Authorization Server

Since the *Authorization server* is the key component of the proposed system, we have deveoted a separate subsection to describe it in more detail. The **Authorization server** makes decisions about granting drivers access to the port terminal. It is queried by the gates, when drivers approach them. As indicated above, it is based on the SXACML architecture [2] and has been adapted to the port scenario. Overall, it is an implementation of the XACML standard, enriched with semantic technologies. Specifically, processing the access request is performed as in the XACML, with the significant difference occurring within the *Policy Information Point* (PIP). Here, the *Semantic PIP* uses semantic reasoner to dynamically provide derived information to supplement the request. In order for the component to work, it needs to be populated with an ontology defining the domain it is operating in. Dedicated port ontology has been created for that purpose (as one of our contributions). It has been described in section 3.3.

Access request attributes, together with the information derived from the ontology are tested against the predefined access policy. The one used in the proposed system was named *IsDriverContracted*. It has been presented in Figure 3. It states that the driver has the right to enter the port terminal and its internal areas only if (s)he works for a contracted company.



```
<Policy xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
        PolicyId="IsDriverContracted"
        RuleCombiningAlgId="urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:deny-unless-permit"
        Version="1.0">
    <Target/>
    <Rule Effect="Permit" RuleId="ContractedDriverPermitted">
        <Condition>
            <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:any-of">
                <Function FunctionId="urn:oasis:names:tc:xacml:1.0:function:boolean-equal"/>
                <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#boolean">true</AttributeValue>
                <AttributeDesignator
                        AttributeId="http://www.semanticweb.org/rafal/ontologies/2017/6/port2#isHiredByContracted"
                        Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"
                        DataType="http://www.w3.org/2001/XMLSchema#boolean"
                        MustBePresent="true"/>
            </Apply>
        </Condition>
    </Rule>
</Policy>
```

Figure 3. Access policy

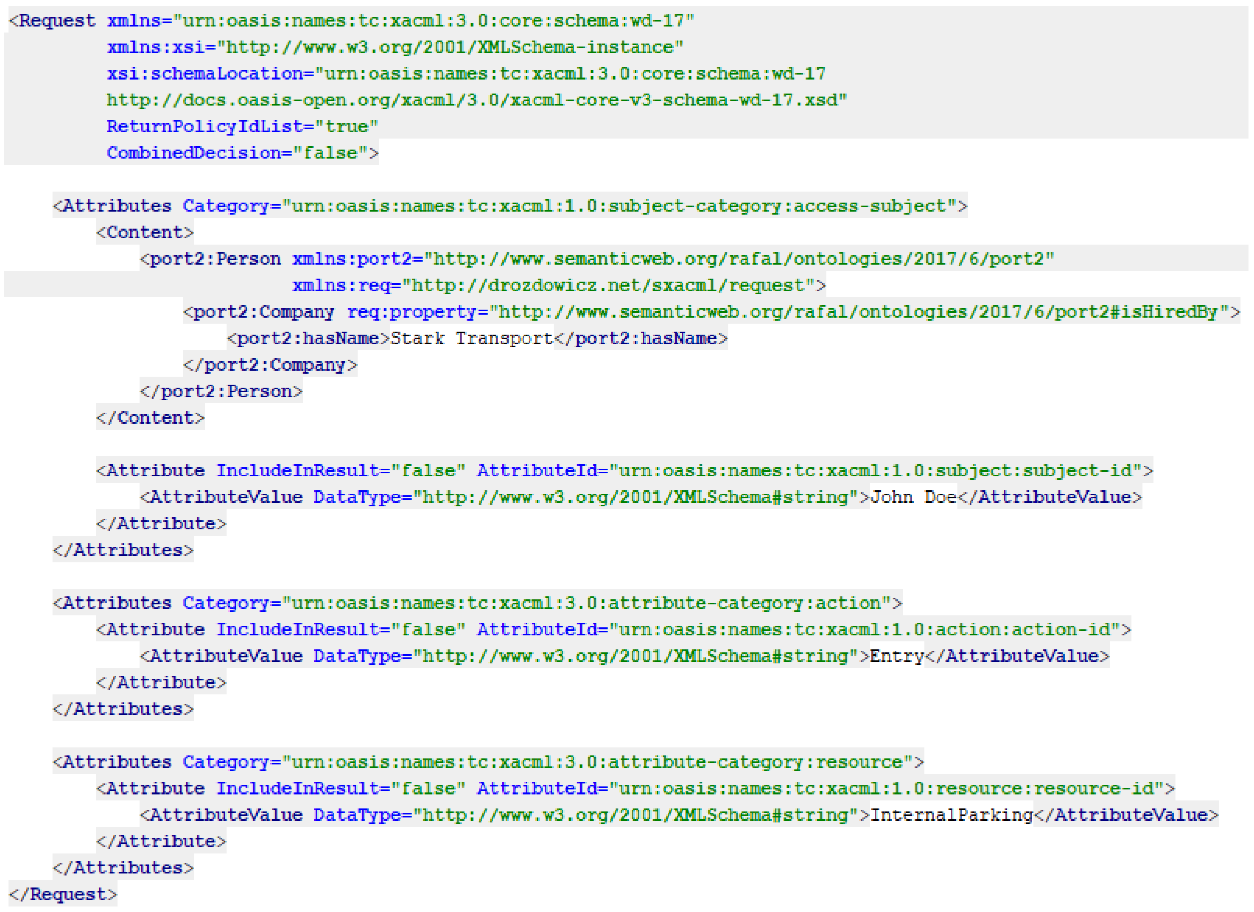
At first glance, this policy may look oversimplified, but this is one of the advantages of applying semantic reasoning. The key issue is how the *isHiredByContracted* attribute is declared in the ontology. Proper definition may result with handling all the desired cases, without the need to modify the policy, keeping it simple and easily maintainable.

Figure 4 presents the format of an **access request** processed by the authorization server. It needs to contain information about the driver and the company (s)he works for. In this case the terminal is visited by John Doe an employee of the Stark Transport company. Note that, in general, the request could be extended with attributes describing “any” action that is to be performed, and “any” targeted resource. In the current (initial) version of our system, we expect a single type of action, which is the entrance to one of the port areas (*e.g.*, internal parking, as the targeted resource).

An example of authorization server **response** has been presented in figure 5. It is concise and has to contain the decision of the system (*Permit* or *Deny*), followed by the status of processing the request (status:ok). It can also provide the id of the policy used for making the decision if the requesting subject sets the value of *ReturnPolicyIdList* attribute sent in the request to true. The response is parsed by the gate and corresponding action is taken based on the received decision.

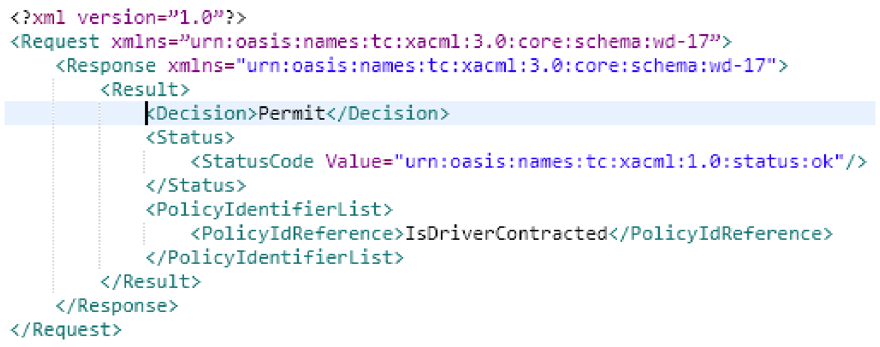
3.3 Ontology

Let us now provide some more details concerning ontologies, in the proposed system. An *ontology* is an explicit, precise and complete description of some part of the world, called the subject (or problem) domain. In general, the main purpose of creating an ontology is to unify concepts within a given field of knowledge. This is supposed to ensure that all entities, operating in this domain, understand its elements and properties in the same way. Clear definition of



```xml
<Request xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17
         http://docs.oasis-open.org/xacml/3.0/xacml-core-v3-schema-wd-17.xsd"
         ReturnPolicyIdList="true"
         CombinedDecision="false">

    <Attributes Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject">
        <Content>
            <port2:Person xmlns:port2="http://www.semanticweb.org/rafal/ontologies/2017/6/port2"
                          xmlns:req="http://drozdowicz.net/sxacml/request">
                <port2:Company req:property="http://www.semanticweb.org/rafal/ontologies/2017/6/port2#isHiredBy">
                    <port2:hasName>Stark Transport</port2:hasName>
                </port2:Company>
            </port2:Person>
        </Content>

        <Attribute IncludeInResult="false" AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">John Doe</AttributeValue>
        </Attribute>
    </Attributes>

    <Attributes Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action">
        <Attribute IncludeInResult="false" AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Entry</AttributeValue>
        </Attribute>
    </Attributes>

    <Attributes Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource">
        <Attribute IncludeInResult="false" AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">InternalParking</AttributeValue>
        </Attribute>
    </Attributes>
</Request>
```

Figure 4. Access request



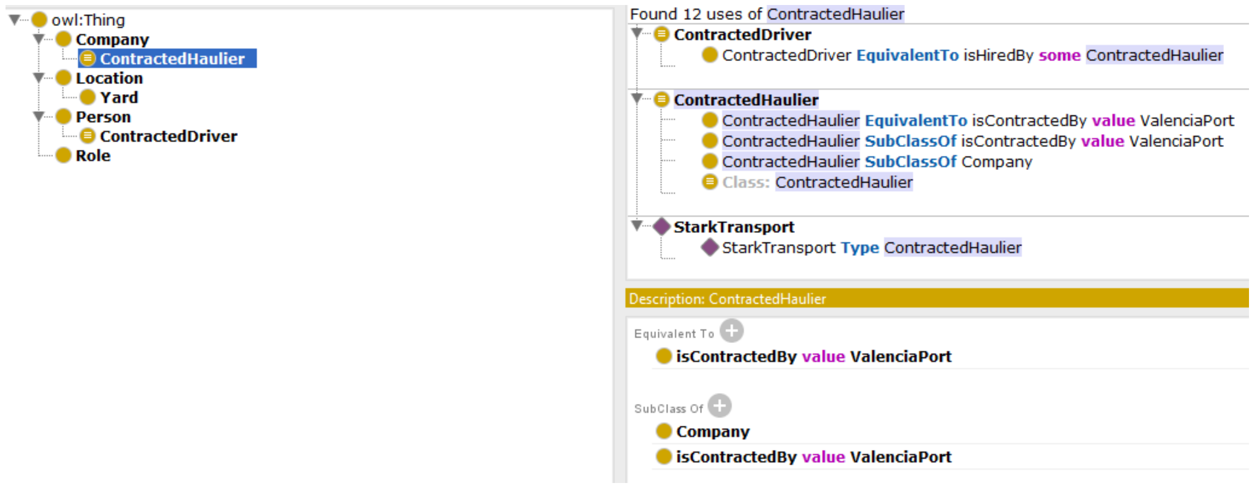
```xml
<?xml version=”1.0”?>
<Request xmlns=”urn:oasis:names:tc:xacml:3.0:core:schema:wd-17”>
    <Response xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17">
        <Result>
            <Decision>Permit</Decision>
            <Status>
                <StatusCode Value="urn:oasis:names:tc:xacml:1.0:status:ok"/>
            </Status>
            <PolicyIdentifierList>
                <PolicyIdReference>IsDriverContracted</PolicyIdReference>
            </PolicyIdentifierList>
        </Result>
    </Response>
</Request>
```

Figure 5. Authorization system response

the ontology makes it possible to process it automatically [10, 11]. Note that the completeness of an ontology is domain/application/context specific. For instance, concept of a truck will be approached differently by the driver (who uses it every day to transport goods, according to the rules of commercial vehicle movement on roads), and by the owner of a trucking company (for whom the truck is, among others, an investment that has certain value, has to be maintained and provide the income). In this context it has to be also stressed, that attempts to build an "ontology of everything" (one that would capture all meanings and contexts pertaining to a given domain and entities that this domain is comprised of) have failed. A typical example is the project CYC, which (after more than 20 years) is still to deliver an all agreed on ontology of the "world." These comments are important in the context of work reported on in this paper. as they show why it is possible to deliver narrowly-focused application-specific domain ontologies for various scenarios of access control management.

The ontology should include terminological knowledge of the field, in the form of specification of the terminology established for a given field, concepts existing in this field, attributes of these concepts, their properties and relations between them as well as existing bonds on these attributes, properties and relationships [10, 11]. In the case of access control, ontology is going to be used for inferring needed information that was not explicitly declared in the access request. This approach should decrease complexity of both the access policies and requests. As stated above, the SXACML approach provides mechanisms, based on the XACML standard, allowing use of ontologies in the decision-making process for inferring missing attribute values.

Going back to our application area, recall that world's seaports are often large facilities, with complex structure. Their area is split into a hierarchical set of terminals with varying (individual) access restrictions. Each terminal may belong to a different company. Furthermore, single terminal may store many containers belonging to different companies / stakeholders. Those companies may have signed contracts with different transportation companies for delivering and picking up cargo. To reflect such domain, we have created a dedicated port ontology. It is presented in Figure 6. Here, note that the proposed ontology is, on the one hand, realistic (based on interactions with representatives of Port of Valencia). On the other hand, if needed, it can be replaced with a different one, while principles of work of the proposed approach to access management will remain the same.



owl:Thing
- Company
  - ContractedHaulier
- Location
  - Yard
- Person
  - ContractedDriver
- Role

Found 12 uses of ContractedHaulier

ContractedDriver
- ContractedDriver EquivalentTo isHiredBy some ContractedHaulier

ContractedHaulier
- ContractedHaulier EquivalentTo isContractedBy value ValenciaPort
- ContractedHaulier SubClassOf isContractedBy value ValenciaPort
- ContractedHaulier SubClassOf Company
- Class: ContractedHaulier

StarkTransport
- StarkTransport Type ContractedHaulier

Description: ContractedHaulier

Equivalent To
- isContractedBy value ValenciaPort

SubClass Of
- Company
- isContractedBy value ValenciaPort

Figure 6. Ontology

Port area is captured by the class *Location*. Particular subareas and terminals may be defined as individuals of this class. During our research, this class has been treated as a simple representative of any part of the port territory, which is why its definition does not contain any location-specific attributes, like coordinates or category. Each area is identified in the ontology. based on the respective individual name. This approach was sufficient for our prototype focused on the access control and lights management. If need arises, the *Location* class could be extended, providing more complex solution. If one would need to have some kind of area categorization, then a proper subclass could be created. The *Yard* subclass is an example of such a category. Actually, it was created only as an example, and is not being used in the proposed solution.

In our guiding use case, access to port areas is requested by arriving truck drivers. Thus, the ontology contains *Person* and *Role* classes. The *Role* class indicates the role that a given person fulfills in the system. An instance of this class, named *Driver*, is used for determining that a person, communicating with the system, is a driver. While the most common case, in the port reality, is the visit of a truck driver, there still might be a different person interacting with the system. This is why more generalized classes were created (*Person* and *Role*), instead of a single *Driver* class. With such approach some user groups could be created (*e.g.*, drivers group, port personnel, or visitors).

Driver visiting the port is employed by a transportation company. This information is crucial for determining access rights. In reference, the *Company* class was defined in the ontology. It relates to the companies whose workers interact with the system. If a person is hired by some company, then there is the *isHiredBy* relation between *Person* and *Company* classes. Moreover, if a company signed a contract with another company, then this fact should be reflected in the ontology in a form of the *isContractedBy* relation. One instance of this class has already been created in the ontology (*ValenciaPort*). It represents Port Authorities. It is required for defining direct port contractors. Obviously, other companies can be added to the system dynamically, when needed, using the OntoPlay interface described in the next section.

To support access control automation, there was a need to create a subgroup (subclass) of people, working for the company that has access to the port terminal. It was called *ContractedDriver* class. It aggregates drivers that have the right to enter the port area. This fact is represented by the *isHiredByContracted* attribute, set to *true*. This is the place of binding with the rule defined in the access policy. If one has this flag set to true, then one is granted permission to enter the port area.

Group of companies, having access to the port, is represented by the *ContractedHaulier* class. So, in terms of the created ontology, the driver is a *ContractedDriver* if her/his employer is a *ContractedHaulier*. A single *ContractedHaulier* company was added to the ontology for test purposes. Its name is Stark Transport. As in all other cases, adding transportation companies can be easily achieved with help of the OntoPlay.

The initial condition of having the access to the port is to sign a contract directly with the Port Authorities. This has been explicitly stated in the ontology. In our case, a company is a *ContractedHaulier* if has a contract (*isContractedBy*) with the Valencia port (*ValenciaPort*). This condition is good enough only for the basic scenario, where the access is granted only to the direct contractors of the port. In order to handle the subcontractors case, the ontology had to be modified. The extension happens to be pretty simple. The assumption that a subcontractor of a company with access permissions should also be granted access led us to the conclusion that the *isContractedBy* relation is actually transitive. After marking it, accordingly, in the ontology, we ended up with a situation where the entire subcontractors chain inherits access rights from the direct port contractor, as expected. Obviously, more complex relations (beyond simple chain of permissions) can also be captured in the ontology. With the aforementioned modification of the *isContractedBy* relation, the authorization system properly recognizes if the truck working for the unknown (up till now) company should have permissions to enter the port area. The only action that needs to be taken by human is to populate the ontology with the information about the new company and its relation to the other company present in the system. There's no need for the subcontractor to sign a contract directly with Port Authorities.

The proposed ontology (created using Protege [7]) has been incorporated into the created access control system. It is used together with the access policy, during evaluation of the access requests, send by the gates in the name of the drivers. This is the key decision-making component in terms of access control in the created system.

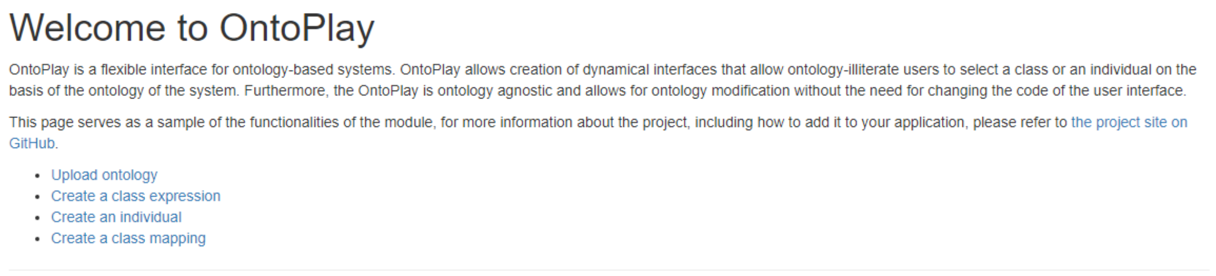
It has been mentioned that the information about new companies and their relation with others needs to be added to the ontology dynamically. Obviously, reverse operation – removal of a company that no longer has a contract with the given port – needs to be supported as well. This means that the initially prepared ontology will have to be adjusted / modified as the world, in which the system works, evolves. The responsibility of dynamical ontology management will, most probably, be delegated to the port personnel. To facilitate this functionality, and allow changes to be made by staff not versed in semantic technologies, OntoPlay software [3] has been integrated into the authorization server.

3.4 OntoPlay

As has been mentioned, there exist multiple tools for ontology management. The problem is that most of them cannot be easily used in the real-world as they require specialized knowledge about semantic technologies narrowing their users to experts in the field. This is where the OntoPlay project comes in handy.

The OntoPlay plug-in is an ontology management tool, originally designed to allow development of interfaces to ontology-driven systems. Later, capabilities related to, limited in scope, ontology maintenance for users inexperienced in semantic technologies have been added. It provides a flexible and intuitive web interface for managing the ontology. It is capable of be used with any ontology, specified in the RDF format, making the modification process as simple as possible [3, 6].

It has been mentioned that OntoPlay is not a general ontology management tool like Protege [7]). It provides a selected need-driven functionalities required by the system like the one discussed in this work. Figure 7 presents the main page of the OntoPlay web interface. First of all, any RDF-based ontology may be loaded into the system, and then it can be adjusted at runtime. This functionality could be useful for applying a different ontology when needed. In the proposed solution the initial ontology would be defined in the configuration, but it should be stressed that this feature makes it possibly to change the way the access control system works, without the need to restart it. The other three functionalities, currently available, are related to the modification of the particular elements of the given ontology. The interface makes it possible to create a new class or an instance (individual) as well as a new class mapping. Crucial functionality in terms of the described system is the possibility of adding / removing a new transportation company, being a new instance of the *Company* class. This functionality was added within the scope of this work.


Welcome to OntoPlay

OntoPlay is a flexible interface for ontology-based systems. OntoPlay allows creation of dynamical interfaces that allow ontology-illiterate users to select a class or an individual on the basis of the ontology of the system. Furthermore, the OntoPlay is ontology agnostic and allows for ontology modification without the need for changing the code of the user interface.

This page serves as a sample of the functionalities of the module, for more information about the project, including how to add it to your application, please refer to the project site on GitHub.

- Upload ontology
- Create a class expression
- Create an individual
- Create a class mapping

Figure 7. Main page of the Ontoplay web interface

3.5 Implementation

The proposed system has been implemented and tested (in a semi-realistic and in an emulated environments). Let us now provide key details concerning implementation. Major part of the software has been implemented using Microsoft technologies. Here, the main programming language used for the development was C#.

The **Server** has been implemented suing ASP.NET technologies. The machine to machine connections is based on the ASP.NET Web API. Here, Web API controllers are dedicated for handling authorization, gates, street lights and trucks. The user interface of the server has been created using ASP.NET MVC technology. It is accessible from most common web browsers. The related controllers support account management, port-related functionalities and login session handling. The interface has also a reference to the user interface of the authorization server, which enables port personnel to modify the ontology in the contexts needed for day-to-day activities. Moreover, the server ensures the existence of the necessary roles, i.e. Admin, Company and Driver, by creating them during initialization if they're missing. System's data is stored by the server in a dedicated database.

The system is using the MSSQL **Database** created using Entity Framework, an Object Relational Mapping tool, developed by Microsoft. The tool enables to generate the structure of the database directly from the implementation of classes. This approach is called Code First, and it has been applied in the created system.

The **Mobile application** was implemented using the Xamarin framework, which makes the implementation of a cross-platform application much faster and easier. It enables the developer to write the application logic once, and use it for every platform. Moreover, with the usage of the Xamarin.Forms extension, the user interface implementation could also be shared between the platforms. This was a convenient way for achieving greater availability by having the application working on every commonly used mobile operating system, i.e. Android, iOS, and Windows Phone. User may log in to the application to be authorized by the system. Then a visit purpose is defined. It is supposed to contain the information about the type of visit (pick up or drop off), id of the "container(s)," name of the company ordering the visit, and truck license plate number. Finally, the application informs the user about the final destination and intermediate gates positions.

There is also a location tracking module, implemented in the application. It is used for obtaining the current position of the device. Asking for the device information like this usually requires the usage of a native API of the platform the application runs on. This would force one to have a separate implementation for every mobile operating system. Thankfully, there is an open source project called Geolocator Plugin, for Xamarin and Windows, created by James Montemagno and published on GitHub as well as in a form of a NuGet package. It provides the possibility of writing a single code for handling the location related logic and share it between all the platforms. The location tracking module notifies the server about the current position of the devices after every change [12].

The application has been integrated with the external maps providers (Google Maps, HERE Maps and OSRM). They are used for calculating routes, displaying maps in the application, and launching external mobile applications of map providers. It is important to note that usage of the API of those providers is limited. They both require the user to create a developer account. Moreover, free usage of HERE Maps is only possible for a predefined period. Google Maps does not put any time limits, but it restricts number of free request that may be processed during the day. Exceeding that amount results with charging fees. OSRM seems to be a good alternative as its open source and completely free. Unfortunately, because of that, the service may sometimes be overused by its clients and stop responding properly, which may be critical in some situations.

Gates and lights are operated by the console applications written in C#. State of a gate is represented by a Boolean flag indicating whether the gate is opened or not. Street light state is in a form of a decimal number determining the brightness.

The only part of the system not based on the Microsoft technologies is the authorization server. This specific element was implemented in Java and Scala with the usage of an open source framework called Play.

4 EXPERIMENTS

4.1 Access Control Experiments

The process of system verification (via experimentation) has been divided into two parts. The first one was focused on testing the entire workflow of all the components in a semi-realistic environment. This was crucial to ensure that the created system actually works and can complete the needed functionality. Because of the distance and lack of possibility to connect the system to the actual Valencia port facilities (we would not be allowed to “play with” real-life system within a million euro enterprise), the tests were executed locally. However, we have tried to make them as realistic as possible. A dedicated scenario has been prepared within local real estate. We believe that this introduced a certain level of credibility to the solution, above typical fully-virtual emulations. As a matter of fact, as it can be seen, using this approach allowed us to spot a potential problem that would have not been captured if the tests were fully emulated.

In our scenario, the user was an inhabitant trying to get access to her car located at the internal parking of a gated community. Although the scenario differs from the port reality, all its elements are strictly analogous. In other words, the user represented the driver. Internal parking was a reference to the terminal area. The car indicated a cargo located somewhere inside the terminal. Person with a laptop running the gate program was standing next to the community wicked, to imitate a terminal’s gate. The results are presented in Figure 8.

Here, it is important to note that the initial attempts to perform the scenario were unsuccessful. The precision of the established user geolocation was too low. The value was so inaccurate that the user standing next to the gate was recognized as being far from it. Hence, the gate did not even initialize the user authorization procedure. It should be stressed that gate is configured to start the authorization when the incoming truck is in a predefined range. This way we were limiting the number of requests processed (potentially simultaneously) by the system and better synchronize gate opening with truck arrival. One way of dealing with the problem was to slightly increase the range of the gate. This was a simple, but only temporary, solution. We still wanted to have the initial configuration working. The problem was unexpected, because, at the same time, the Google Maps application on the phone was showing the current position very precisely. It could be the case that the cross-platform location plugin was not able to take full advantage of the phone’s GPS module accuracy. As we have found, the plugin is under active development a new version was released, with some breaking changes significantly increasing the accuracy on the Android-based devices [12]. Unfortunately, testing it was impossible as it required a device with Android 8.0 while there was no such version available for the devices we had (other than emulators). After all, together with increasing the gate range, the implementation of the application was improved increasing the stability of the location module. Those fixes made it possible to test the aforementioned scenario. The test has been successful. Person using mobile application was granted access to the internal parking where the car, being the target of the visit, was located.

In order to verify applicability of the system in the actual location of the Valencia port terminal, in the second stage of testing, full emulation environment had to be developed. To achieve this goal, the mobile application was

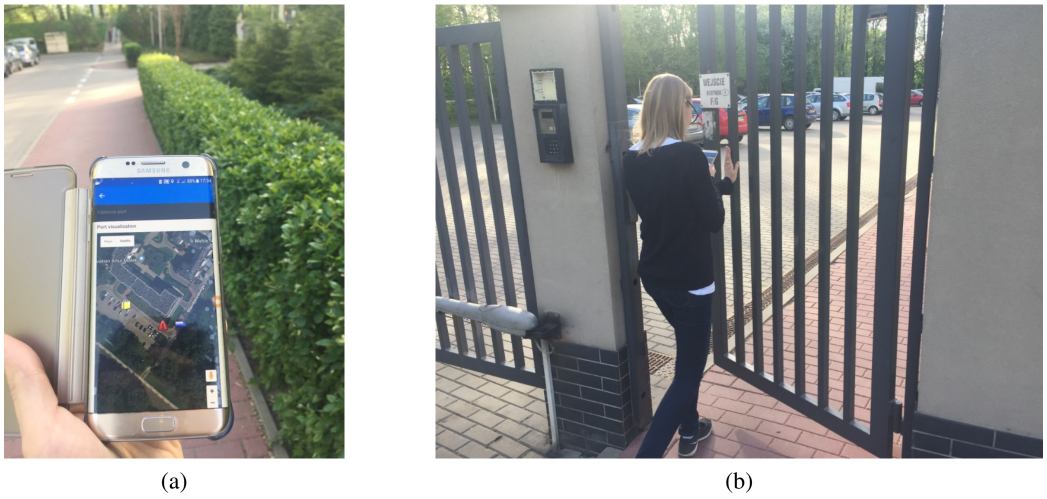
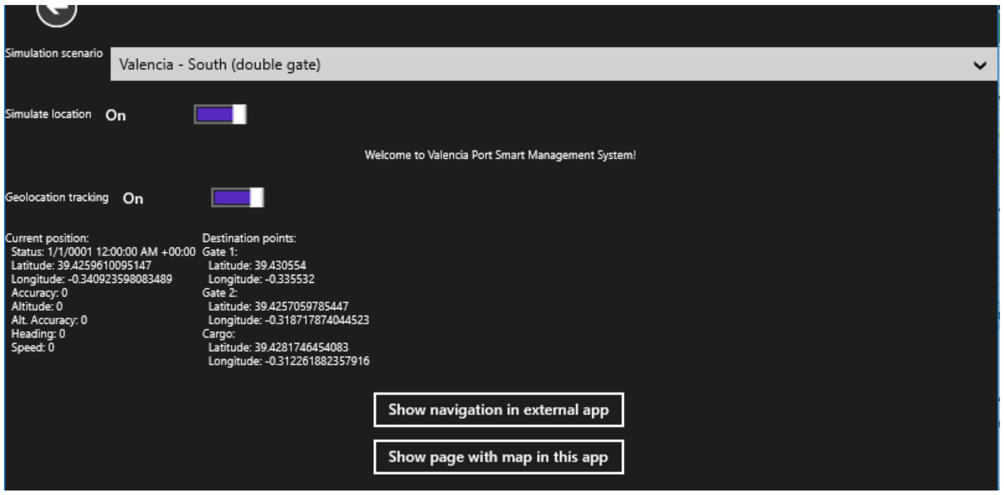


Figure 8. Mobile application usage (a) for traversing the gate (b) in a real-life scenario

extended with a location simulator. It is accessible from the page of the application presented in Figure 9.



```
Simulation scenario    Valencia - South (double gate)

Simulate location    On

Welcome to Valencia Port Smart Management System!

Geolocation tracking    On

Current position:                           Destination points:
  Status: 1/1/0001 12:00:00 AM +00:00       Gate 1:
  Latitude: 39.4259610095147                  Latitude: 39.430554
  Longitude: -0.340923598083489               Longitude: -0.335532
  Accuracy: 0                               Gate 2:
  Altitude: 0                                 Latitude: 39.4257059785447
  Alt. Accuracy: 0                            Longitude: -0.318717874044523
  Heading: 0                                Cargo:
  Speed: 0                                    Latitude: 39.4281746454083
                                              Longitude: -0.312261882357916
```

Show navigation in external app
Show page with map in this app

Figure 9. Application page with the simulation functionality

Having the starting position, intermediate points and the desired destination, the developed simulator calculates the route. Detailed geometry of the track is obtained by asking an external map service provider (HERE Maps, or OSRM). If there are some long fragments received (which often happens for the straight parts of the route), simulator splits them into smaller fragments of predefined maximal length. After the final geometry is established, the simulator starts to update the location point by point in a predefined interval. This ensures that the truck will not suddenly "jump" by an enormous distance within a single step of the simulation, and makes the simulated movement smooth, and more realistic. The simulator treats gates as intermediate points that need to be visited. If one is right in front of the truck, a request is sent to the server to ask for the gate's state. Normally, there would not be the need for this, as the driver sees whether the gate is opened or not. Simulator however, needs to get this information from the server. Furthermore, assuming that, in the near future, self-driving trucks will be introduced, this step will be necessary. If the gate is opened, the movement simulation continues, and the truck traverses the gate. Otherwise, truck's movement is stopped. Gate state is being checked repeatedly in a predefined interval. The truck stops until the gate opens. In this way it was possible to test the system behavior in several different scenarios, directly bounded to the actual Valencia port area. Gates and street lights locations were set to the real positions of the related Valencia port facilities like shown in Figure 10.

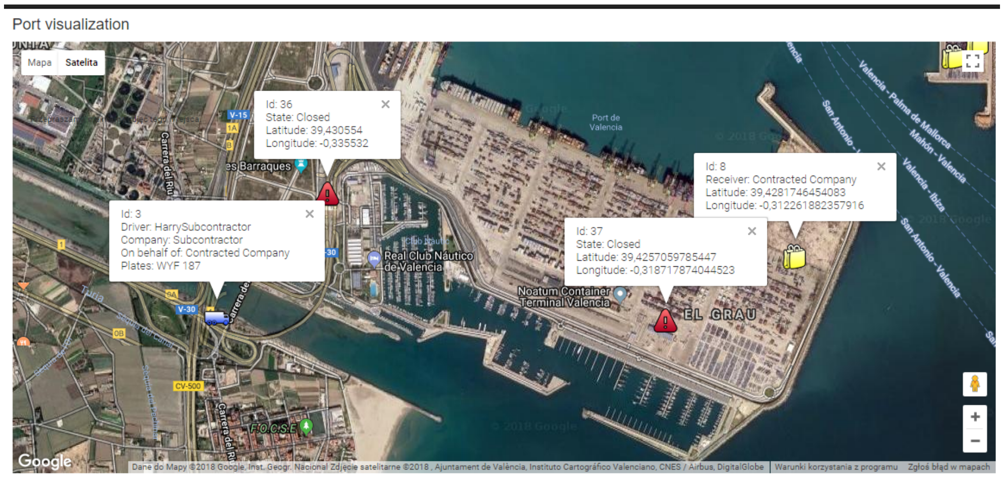
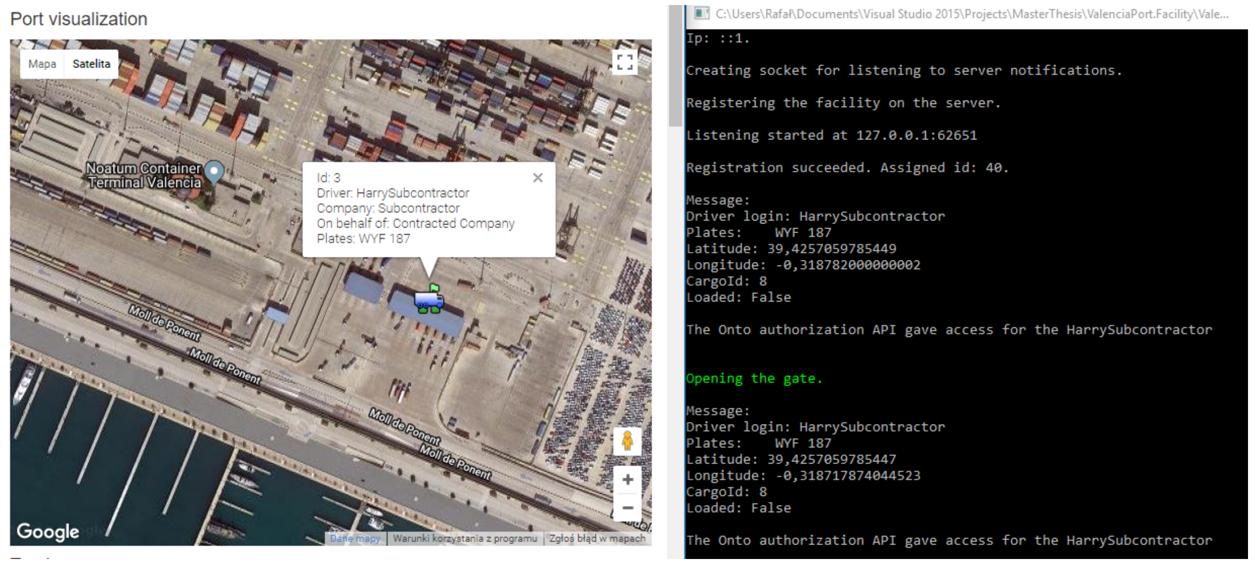

Port visualization
Mapa
Satelita
Id: 36
State: Closed
Latitude: 39,430554
Longitude: -0,335532
Id: 3
Driver: HarrySubcontractor
Company: Subcontractor
On behalf of: Contracted Company
Plates: WYF 187
Id: 8
Receiver: Contracted Company
Latitude: 39,4281746454083
Longitude: -0,312261882357916
Id: 37
State: Closed
Latitude: 39,4257059785447
Longitude: -0,318717874044523
Port de Valencia
Real Club Náutico de Valencia
Noatum Container Terminal Valencia
EL GRAU
Google

Figure 10. Valencia port map with the facilities, trucks and cargoes markers

We have tested the system in multiple scenarios, here we will report only two scenarios of most interest. One of them was a basic situation when the port was visited by the truck driver working for the company that has a legal agreement signed directly with the port authorities. The other, more complex one, was related to the visit of a truck driver working for a company related to the port indirectly. There is no contract between the company and the port. However, it has been subcontracted by a company that has the rights to enter the port area. Both scenarios where simulated ending up with a successful visit. Figure 11 presents the situation of traversing the gate by the truck. Each gate was opened automatically after asking the authorization server for the access rights of the driver.


Port visualization
Mapa
Satelita
Noatum Container Terminal Valencia
Id: 3
Driver: HarrySubcontractor
Company: Subcontractor
On behalf of: Contracted Company
Plates: WYF 187
Moll de Ponent
Google
C:\Users\Rafał\Documents\Visual Studio 2015\Projects\MasterThesis\ValenciaPort.Facility\Vale...

```
Ip: ::1.

Creating socket for listening to server notifications.

Registering the facility on the server.

Listening started at 127.0.0.1:62651

Registration succeeded. Assigned id: 40.

Message:
Driver login: HarrySubcontractor
Plates:    WYF 187
Latitude: 39,4257059785449
Longitude: -0,318782000000002
CargoId: 8
Loaded: False

The Onto authorization API gave access for the HarrySubcontractor


Opening the gate.

Message:
Driver login: HarrySubcontractor
Plates:    WYF 187
Latitude: 39,4257059785447
Longitude: -0,318717874044523
CargoId: 8
Loaded: False

The Onto authorization API gave access for the HarrySubcontractor
```

Figure 11. Gate opened for the truck after automatic rights verification

Final step of the simulations is presented in Figure 12. The drivers reached their destinations as expected.

The only action needed to be taken was the addition of the companies (Contractor, Subcontractor), together with the relations information, to the ontology. The OntoPlay plug-in was used for that purpose. The Administrator, representing person from the port personnel, navigated to the Authorization server interface. Then, the option to add a new individual was chosen. Figure 13 presents the action of adding the Subcontractor company. All options, based

Port visualization

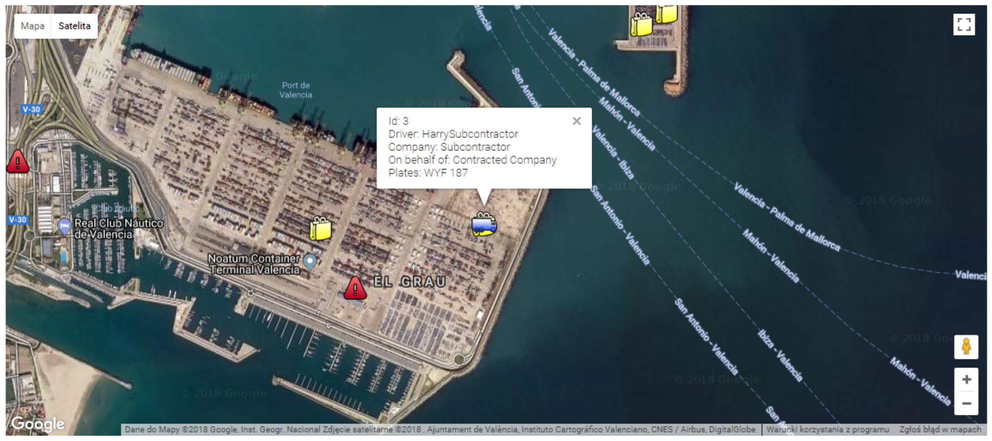
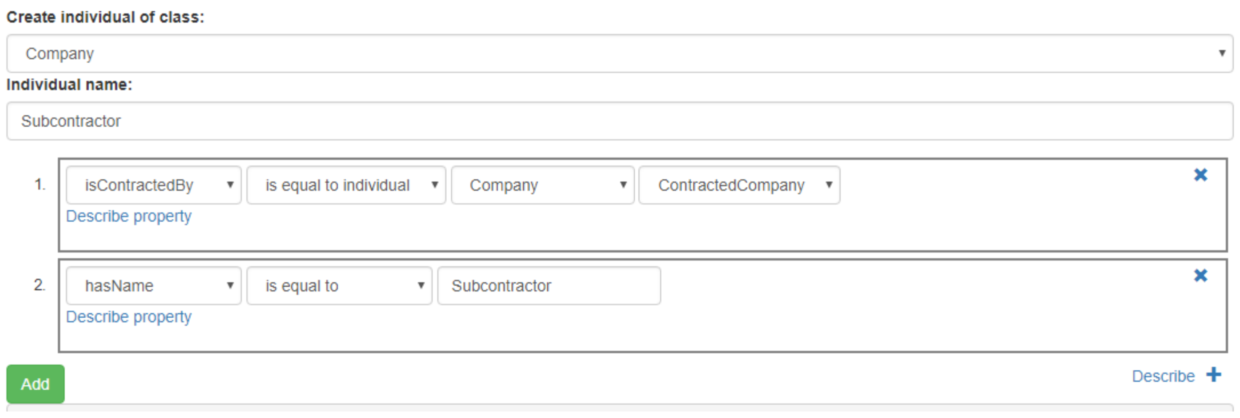

Id: 3
Driver: HarrySubcontractor
Company: Subcontractor
On behalf of: Contracted Company
Plates: WYF 187

Figure 12. Truck successfully reached its destination

on the ontology definition, were preloaded and selectable within dedicated drop-down menus. The Administrator simply needed to choose the *Company* class and put the name of the new instance (Subcontractor). Furthermore, it was possible to define Company-specific attributes. The first one, crucial from the authorization point of view, was indicating that the Subcontractor is contracted by the company, represented in the ontology with the *ContractedCompany* instance. As you can see in the figure, all the inputs for the first attribute were drop-down lists (indicated by the down arrow). Each of those boxes was displayed dynamically, based on the values chosen within the ones before. The other attribute is the definition of the company name. In this case it is equivalent to the individual name, but this does not necessarily has to be the case. During the entire process of adding a new company, the user is supported by the interface, being suggested what values for what attributes might be chosen.


Create individual of class:
Company
Individual name:
Subcontractor

1. isContractedBy | is equal to individual | Company | ContractedCompany
Describe property

2. hasName | is equal to | Subcontractor
Describe property

Figure 13. Adding information about new company to the ontology

4.2 Intelligent Lights Experiments

The final part of the experiments was related to the street lights control. The assumption was, that the lights will dynamically react to the incoming trucks, by adjusting their brightness, accordingly. This functionality was tested only in the emulated scenarios as there was no smart lighting infrastructure available in the local environment. During the tests, number of street lights were added to the virtual port area. Afterwards, trucks movement was simulated and the reaction of the lights was observed both in the street light console program and through the server interface presented in Figure 14. The lights are represented as markers with a surrounding circular shapes representing the areas currently illuminated by them. The radius of the circles changes dynamically, depending on the lights brightness. As a proof of concept, this functionality was tested with some simple formulas, based on the distance of the trucks. It should be noted that the same solution could be used for implementing much more complex lighting system.

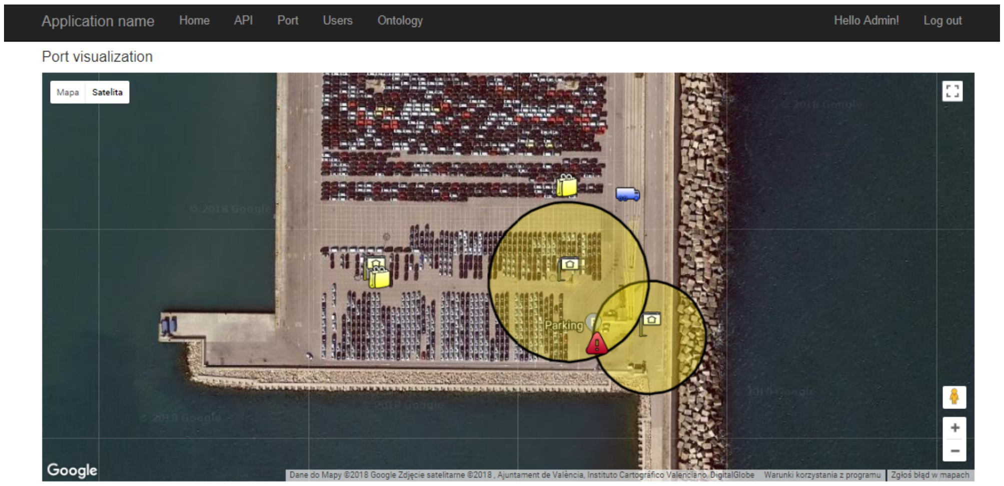


Figure 14. Testing lighting system

Tests ended up successfully. The lights were adjusting dynamically as expected.

CONCLUDING REMARKS

We have discussed an implementation of the access and lights management system for a port terminal that takes advantage of the ontology-based reasoning in the decision-making process. It is an example of how semantic technologies could be used to enrich the experience of working with the access control. Usage of the SXACML solution combined with the OntoPlay project helped building an efficient and convenient access management system for a port terminal. Pure main contributions were: (a) formulation of an access management oriented mini-port ontology, (b) modification of OntoPlay to add functionality needed in our scenarios, development of a "Port of Valencia Emulator" for testing the system, and the key contribution (d) implementation and testing of a complete access management solution for a port environment. Based on our work, we can state that it is reasonable to assume that combining ABAC with semantic technologies is a promising approach that can be applied in many different fields, especially IoT-based solutions.

ACKNOWLEDGEMENTS

Work presented in this paper was supported in part by: European Union's "Horizon 2020" research and innovation programme as part of the "Interoperability of Heterogeneous IoT Platforms" (INTER-IoT) project under Grant Agreement No. 687283, PAS-RAS bilateral project "Semantic foundation of the Internet of Things", PAS-BAS bilateral project "Practical aspects of scientific computing", as well as a collaboration agreement between University of Novi Sad, University of Craiova, SRI PAS and Warsaw University of Technology.

References

[1] A. Elliott and S. Knight, "Role explosion: Acknowledging the problem," in *Software Engineering Research and Practice* (2010), pp. 349–355.

[2] M. Drozdowicz, M. Ganzha, and M. Paprzycki, "Semantic policy information point - preliminary considerations," in *ICT Innovations 2015, Advances in Intelligent Systems and Computing*, Vol. 399, edited by S. Loshkovska and S. Koceski (Springer International Publishing, 2016), pp. 11–19.

[3] M. Drozdowicz, M. Alwazir, M. Ganzha, and M. Paprzycki, "Graphical interface for ontology mapping with application to access control," in *ACIIDS 2017* (Springer International Publishing, 2017), pp. 46–55.

[4] M. Ed-Daibouni, A. Lebbat, S. Tallal, and H. Medromi, "Toward a new extension of the access control model ABAC for cloud computing," in *Advances in Ubiquitous Networking* (Springer Singapore, 2016), pp. 79–89.

[5] M. Drozdowicz, M. Ganzha, and M. Paprzycki (2016) *Journal of Medical Systems* **40**, 238.

[6] M. Drozdowicz, M. Ganzha, M. Paprzycki, P. Szmeja, and K. Wasielewska, "Ontoplay – a flexible userinterface for ontology-based systems," in *CEUR Workshop Proceedings*, Vol. 918, edited by S. Ossowski, F. Toni *et al* (2012), pp. 86–100.

[7] A free, open-source ontology editor and framework for building intelligent systems, `https://protege.stanford.edu/`, 2018.

[8] Marine terminal security guidelines, `https://popcdn.azureedge.net/pdfs/Mar_Scrty_Trmnl_Gdlns.pdf`, 2009.

[9] J. G. de la Guia and M. Llop, Valencia port community system developement strategy and practice, `http://www.skematransport.eu/uploadfiles/Valencia\%20Port\%20Community\%20System\%20developement\%20strategy\%20and\%20practice.pdf`, 2010.

[10] W. Paluszyński, Reprezentacja semantyczna: sieci semantyczne i ontologie, `http://zpcir.ict.pwr.wroc.pl/~witold/ai/ai_onto_s.pdf`, 2018.

[11] Ontology development 101: A guide to creating your first ontology, `https://protege.stanford.edu/publications/ontology_development/ontology101-noy-mcguinness.html`, 2018.

[12] Geolocator plugin for Xamarin and Windows, `https://github.com/jamesmontemagno/GeolocatorPlugin`, 2018.