3-бөлім **Раздел 3** **Section 3**

Информатика **Информатика** **Computer Science**

IRSTI 28.23.39 DOI: https://doi.org/10.26577/JMMCS.2020.v108.i4.06

Zh.E. Sartabanova[1,*], V.T. Dimitrov[2], S.M. Sarsimbaeva[1]
[1]K.Zhubanov Aktobe Regional University, Aktobe, Kazakhstan
[2]Sofia University ST.Kliment Ohridski, Sofia, Bulgaria
*e-mail: sartabanova88@gmail.com

APPLYING THE KNOWLEDGE BASE OF CWE WEAKNESSES IN SOFTWARE DESIGN

The article deals with the issues of organizing software weaknesses by the software architect at the stage of its design using the developed ontological knowledge base of CWE weaknesses. The main goal of this research is to analyze the software defect system based on CWE and develop an ontology model (knowledge base) of this system for software architects. The use of artificial intelligence tools, in particular the development of knowledge bases based on weaknesses, will provide new opportunities for searching and researching software weaknesses. This model being developed will be useful for application by software developers, researchers in the field of software design and cybersecurity, as well as teachers of educational institutions that conduct courses in software development technology and information security. For developers, this model can serve as an assistant and reference when designing software, since weaknesses are organized by a well-known security tactic, helping the designer to embed security during the design process instead of detecting weaknesses after the software has been created. Researchers will be interested in studying and applying software weaknesses in their work. Teachers can use this model as a reference when studying and discussing security vulnerabilities in software design or architecture, as well as the types of errors that can be made during software development. The functions of the software architect are analyzed, and an example of the built ontological knowledge base of CWE weaknesses is given.

Key words: CWE, software weaknesses, ontology, knowledge bases, software architect, Protege, Semantic Web, SPARQL.

Ж.Е.Сартабанова[1,*], В.Т. Димитров[2], С.М.Сарсимбаева[1]
[1]Қ.Жұбанов атындағы Ақтөбе өңірлік университеті, Ақтөбе қ., Қазақстан
[2]Әулие Климент Охрид атындағы София университеті, София қ., Болгария
*e-mail: sartabanova88@gmail.com

Программалалық жасақтаманы жобалауда CWE әлсіздіктері туралы білім базасын қолдану

Берілген мақалада программалық жасақтама сәулетшісінің CWE әлсіз жақтарын білудің дамыған онтологиялық базасын қолдана отырып, оны жобалау кезеңінде программалық жасақтаманың әлсіздігін ұйымдастыру мәселелері қарастырылады. Бұл зерттеудің негізгі мақсаты - CWE негізіндегі программалық жасақтама ақауларының жүйесін талдау және программалық жасақтама сәулетшілері үшін осы жүйенің онтология моделін (білім базасын) жасау. Жасанды интеллект құралдарын қолдану, атап айтқанда, әлсіздіктерге негізделген білім базасын дамыту программалық жасақтаманың әлсіздіктерін іздеуге және зерттеуге жаңа мүмкіндіктер береді. Бұл әзірленген модель программалық жасақтаманы әзірлеушілердің, программалық жасақтама мен киберқауіпсіздік саласындағы зерттеушілердің, сондай-ақ программалық жасақтаманы әзірлеу технологиясы мен ақпараттық қауіпсіздік курстарын жүргізетін оқу орындарының оқытушыла

рына пайдалы болады. Бұл модель әзірлеушілерге программалық жасақтаманы жобалауда көмекші және анықтамалық бола алады, өйткені әлсіздіктер белгілі қауіпсіздік тактикасымен ұйымдастырылған, дизайнерге программалық жасақтама жасалғаннан кейін әлсіз жерлерді табудың орнына жобалау процесінде қауіпсіздікті енгізуге көмектеседі. Зерттеушілерге өз жұмыстарында программалық жасақтаманың әлсіздігін зерттеу және қолдану мәселелері қызықты болады. Оқытушылар бұл модельді программалық жасақтама дизайнының немесе архитектурасының әлсіз тұстарындағы қауіпсіздікті, сондай-ақ программалық жасақтаманы әзірлеу кезінде жасалуы мүмкін қателіктердің түрлерін зерттеу және талқылау кезінде анықтамалық материал ретінде қолдана алады. Программалық жасақтама сәулетшісінің функциялары талданады, CWE әлсіз жақтарын білудің онтологиялық базасының мысалы келтірілген.

Түйін сөздер: CWE, программалық жасақтаманың әлсіздігі, онтология, білім базасы, программалық жасақтама сәулетшісі, Protege, Semantic Web, SPARQL.

Ж.Е.Сартабанова[1,*], В.Т. Димитров[2], С.М.Сарсимбаева[1]
[1]Актюбинский региональный университет имени К.Жубанова, г. Актобе, Казахстан
[2]Софийский университет имени Святого Климента Охридского, г. София, Болгария
[*]e-mail: sartabanova88@gmail.com

Применение базы знаний слабостей CWE в проектировании программного обеспечения

В статье рассматриваются вопросы организации слабостей программного обеспечения архитектором программного обеспечения на этапе его проектирования с применением разработанной онтологической базы знаний слабостей CWE. Основной целью данного исследования является анализ системы дефектов программного обеспечения на основе CWE и разработка модели онтологии (базы знаний) этой системы для архитекторов программного обеспечения. Применение средств искусственного интеллекта, в частности разработку базу знании на основе слабостей даст новые возможности поиска и исследования слабостей программного обеспечения. Данная разрабатываемая модель будет полезна в применении разработчиками программного обеспечения, исследователям в области проектирования программного обеспечения и кибербезопасности, а также преподавателям учебных заведений, которые ведут курсы технологии разработки программного обеспечения и по информационной безопасности. Разработчикам данная модель может служить помощником и справочником при проектировании программного обеспечения, поскольку слабые места организованы известной тактикой безопасности, помогая проектировщику во встраивании безопасности в течение процесса проектирования вместо того, чтобы обнаружить слабые места после того, как программное обеспечение было создано. Исследователям будет интересны вопросы изучения и применения слабостей программного обеспечения в своих работах. Преподаватели могут использовать данную модель в качестве справочного материала при изучении и обсуждении безопасности по слабым местам дизайна или архитектуры программного обеспечения, а также типов ошибок, которые могут быть сделаны во время разработки программного обеспечения. Проанализированы функции архитектора программного обеспечения, приведен пример построенной онтологической базы знаний слабостей CWE.

Ключевые слова: CWE, слабости программного обеспечения, онтология, базы знаний, архитектор программного обеспечения, Protege, Semantic Web, SPARQL.

1 Introduction

Each organization makes sure that the software purchased and the software products developed are protected at all levels. To protect all their software products, companies adhere to the rules and standards of information security, as well as international recommendations, at all stages of development. To solve these problems, the developer community and MITRE Corporation created an international list of CWE software weaknesses. However, this information is not structured and there are contradictions in structuring these weaknesses by concepts, which leads to problems in the work of system users. These problems make it difficult to protect resources from cyber-attacks.

The analysis of publications [1–4] shows that one of the promising approaches to improving the efficiency of both search and analysis of information is an approach based on the construction of ontologies of the subject area. The authors propose an approach to implementing the organization of software weaknesses using ontology.

2 Architectural concept of CWE

A software architect is a person, whose main activity is to manage the process of creating, designing, developing and maintaining software. The main responsibility of the software architect according to international requirements is to develop or select the most appropriate architecture for the system (or systems), such that it meets the needs of the business, meets the requirements of stakeholders and achieves the desired results under specified restrictions. The functions of the architect are shown in Figure 1.

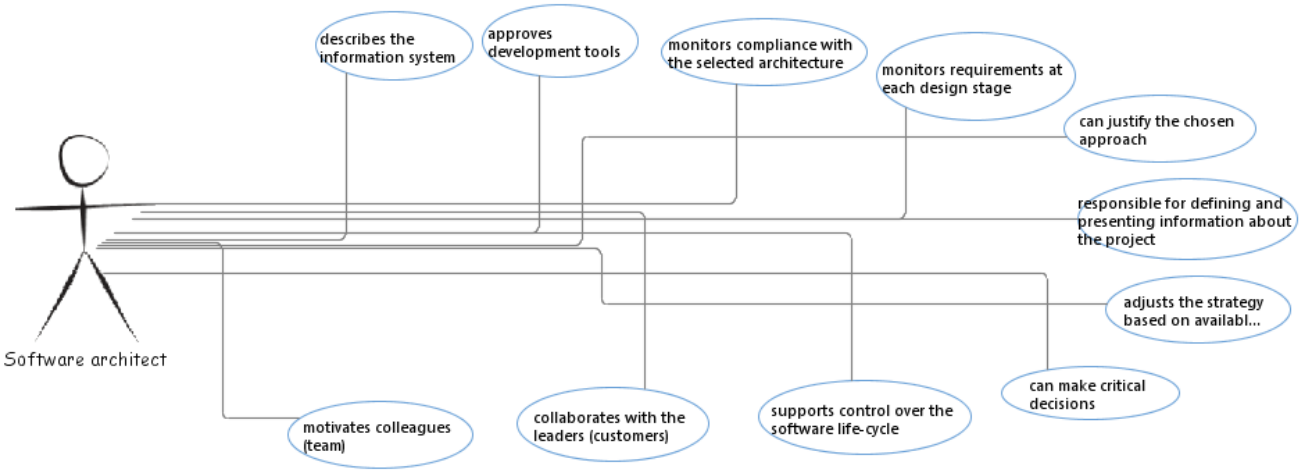

Figure 1 – Software Architect Functions [5]

In accordance with the Professional Standard in the Republic of Kazakhstan, the software architect has the following responsibilities: evaluating and analyzing systems, developing system solutions, developing IT strategies, concepts and architecture of information systems, introducing innovations in business processes, advising on the selection and implementation of optimal systems, from the point of view of the IT strategy of the enterprise, information technologies and using investments in information systems with maximum benefit. Its main goal is to develop a software architecture. The work functions of the software architect include [6]:

1. Creating Software Architecture Variants;
2. Assessment of software requirements and selection of software architecture option;
3. Documentation of software architecture and implementation of software tools;
4. Assessment of the possibility of creating a project architecture and identification of key scenarios;
5. Manage methods, interactions, and software upgrades;
6. Control over selection of software architecture option, implementation and maintenance of software tools.

3 The proposed ontology

An ontology is a system of objects, their properties, and relations between them for a certain area of knowledge. Special languages are being developed to work with ontologies, the most modern of which is the W3C consortium's OWL language. Information in the ontology is stored in OWL format, in the form of triplets. OWL is an information (knowledge) representation language that can be used to describe ontologies both in the semantic web and in various applied information systems. [7, 8]

A Query Language for RDF (RDQL) and Protocol and RDF Query Language (SPARQL) are currently available for executing queries to RDF stores. SPARQL is one of the recommended semantic web technologies for publishing data to the Internet [9].

The knowledge base development process was divided into the following stages. - Defining classes in the ontology;
- The organization of classes in a certain hierarchy;
- Property definition;
- The content of the instances of the class "CWEEntries»;
- The content of the instances of the class "CVEEntries»;
- The content of the instances of the class "CAPECEntries»;
- Checking the consistency of the knowledge base.

The architectural concept is represented as a graph. This concept includes 12 categories, each of which consists of classes, variants, bases, and composites. The category structure is shown in Figure 2.

Any ontology contains a header and a body. Figure 3 shows the title of the ontology (knowledge base), added an annotation, you can add comments, and configure the import of the ontology. The ontology body contains descriptions of classes, properties, and individuals.

The "owl:Class" class is introduced in the OWL language. Classes are organized in a hierarchy using the "rdfs:subClassOf"property. Two complementary classes are important: "owl:Thing a superclass of any OWL class; "owl:Nothing a subclass of any OWL class. An instance of any OWL class includes an extension of the "owl:Thing" class. The extension of the "owl:Nothing" class is an empty set. Class descriptions are the building blocks for defining classes using axioms. To define classes and form various axioms, there are a number of constructs: "rdf:ID defining a named class; "rdfs:subClassOf" is extensional one class is entirely included in the other extensional; "owl:equivalentClass" is extensionally two classes coincide; "owl:disjointWith extensionally two classes do not intersect.

Classes are the main component of the OWL ontology. Classes can be named, disjoint, and hierarchical. When creating classes, the Protege editor initially already contains the "Thing"class - which is a class that represents a set containing all the objects in the subject area.

In the created knowledge base, the Thing class contains the following classes: CWEEntries - the main class under study, CAPECEntries, and CVEEntries. CWEEntries contains Software weaknesses, CAPECEntries-attacks related to the corresponding weaknesses, and cveentries-vulnerabilities.

An ontological graph is a bipartite graph whose vertices are concepts of the domain, and whose arcs are relations between them. An ontological graph is an information model of a domain that has the form of an oriented graph, whose vertices are classes, and whose arcs are relations or connections. Figure 4 shows the ontological graph obtained as a result of constructing the domain ontology in the Protege 5.5.0 editor [10].

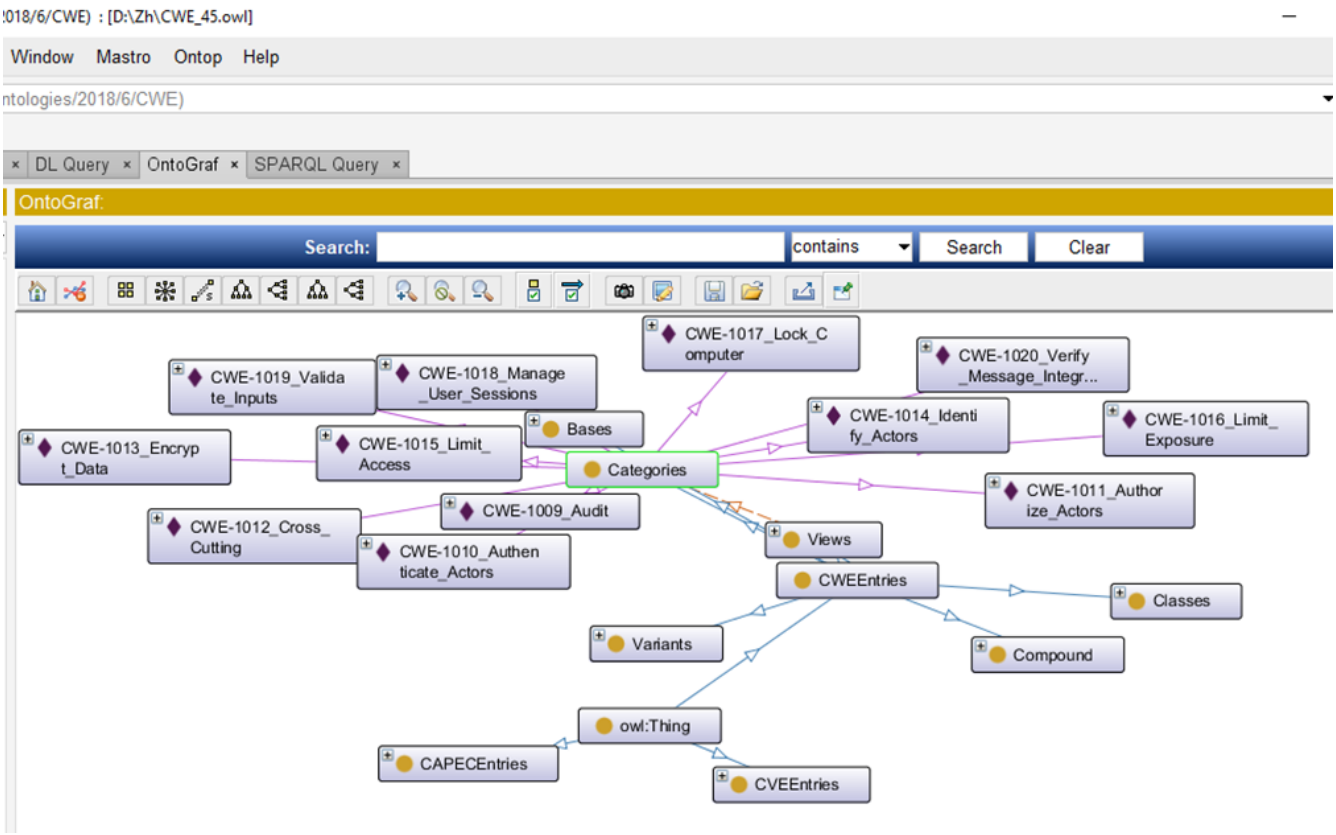

Figure 2 – The composition of the categories (the ontological graph)

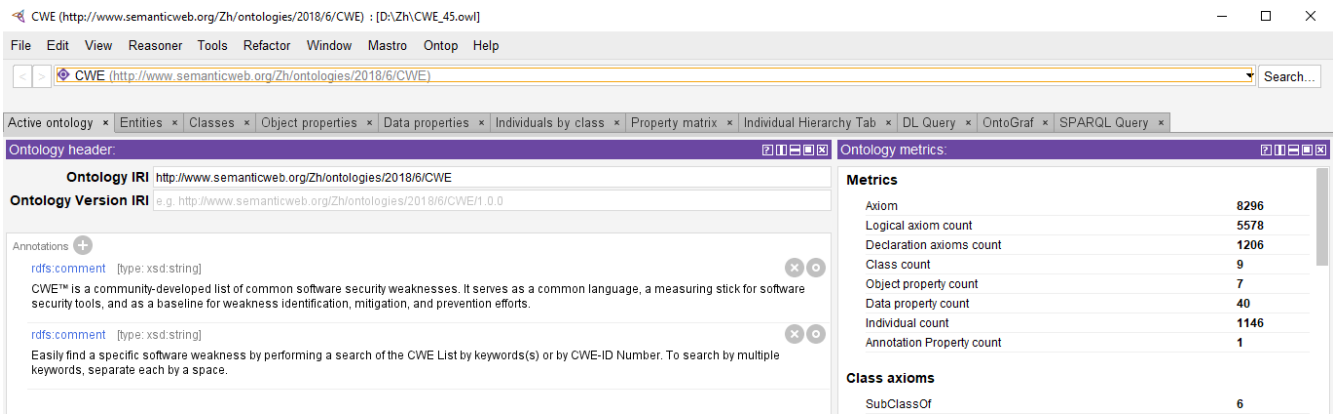

Figure 3 – Title of the ontology created by the knowledge base

The knowledge base you create consists of weaknesses, vulnerabilities, and attacks [11]. Since the knowledge base is implemented from the point of view of the architectural concept, the main class is CWEEntries. This class consists of the main view class, which includes 12 subclasses (category). The Categories class contains CWE records that contain a set of other records that share a common characteristic. [12]

Figure 5 shows the category content modeled in the Protege environment.

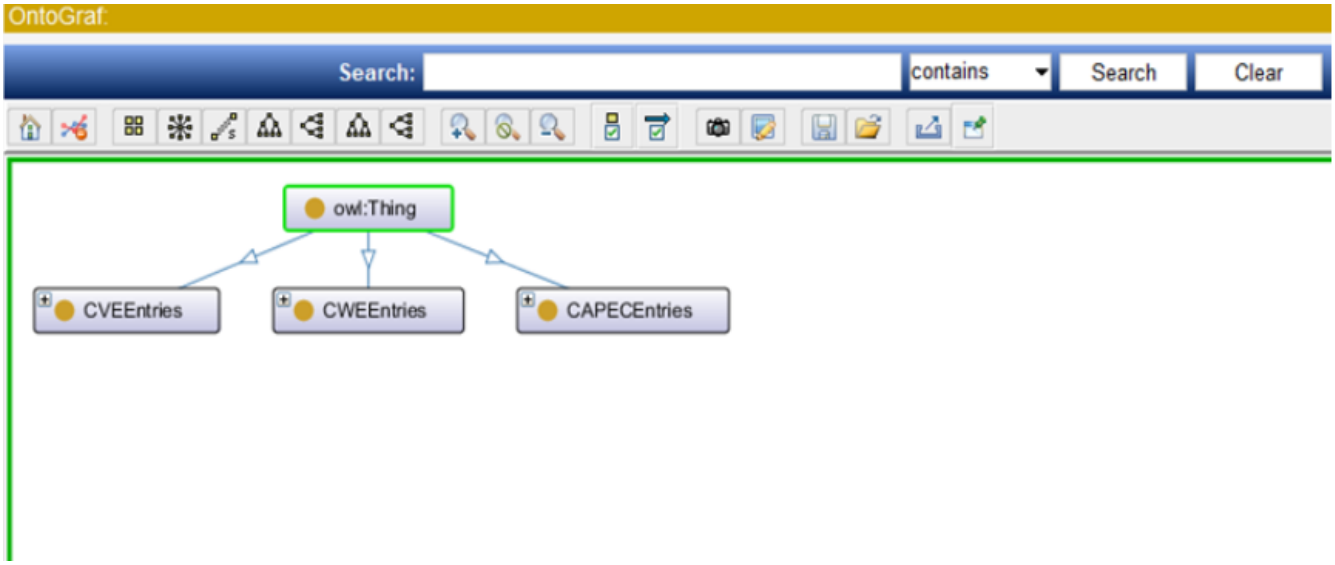

Figure 4 – The ontological graph

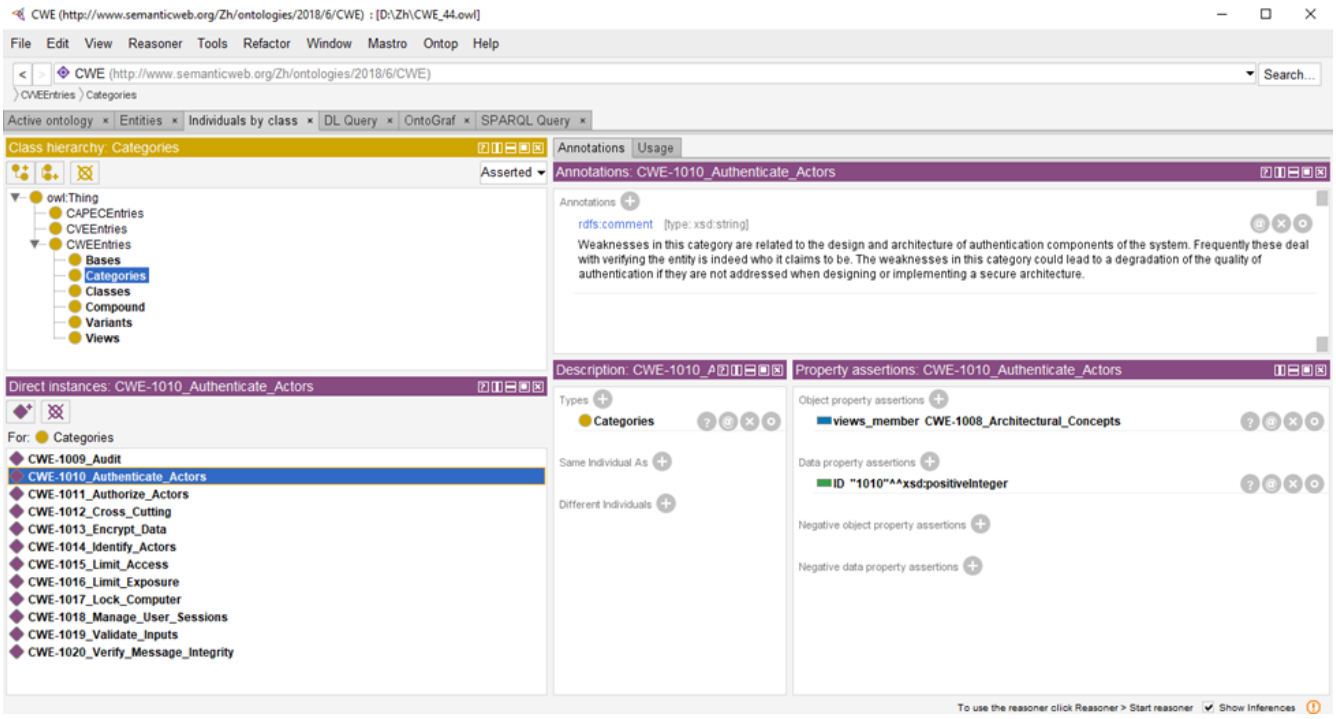

Figure 5 – The structure category

Each category includes classes, bases, variants, and, if there are composites.

The category has the following format: description - brief description-specified in the annotation), category ID - specified as a data property (data properties - ID), object property ? views_member. This property associates each category with the corresponding classes, bases, variants, and composites (that is, those that belong to a particular category).

Properties (rdf: Property) in OWL are relationships that bind classes or individuals. All

CWEEntries have two main objects:
1) object properties - properties of objects (owl:ObjectProperty) are relationships between two classes or individuals;
2) data properties - data type properties (owl:DataTypeProperty) - this property defines the relationship between the individual and the data value (numerical constraints).

The main properties of object properties include: references (designed to indicate the vulnerabilities available for a certain weakness), categories_member (this property is necessary for the relationship of each bases, variants, classes, compound with the corresponding categories), related_attack_patterns (this property indicates the relationship of weaknesses with possible attacks).

Properties of data properties include: id, modes_of_introduction, common_consequences, likelihood_of_exploit, demonstrative_examples, applicable platforms, weakness_ordinalities, detection_methods. [13, 14]

4 Application of the knowledge base

There are four types of data queries: default query, literal query, class instance query, class object property query. Below are various types of queries written in SPARQL.

Query 1 - a general standard query, It defines the structure of the knowledge base and shows the list of main objects.
Inquiry 1. Generic Standard Query:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
SELECT ?subject ?object
WHERE { ?subject rdfs:subClassOf ?object }
```

As a result of this query, a list of the main classes that are subordinate to the CWEEntries class in our ontology is displayed.

Query 2 - a query based on class object property data, It identifies the knowledge base elements associated with that property and shows the relationships between weaknesses through a specific object property.
Inquiry 2. Objects associated with a property of type categories_incudes:

```
PREFIX db: <http://www.semanticweb.org/Zh/ontologies/2018/6/CWE#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
SELECT ?x ?property ?y
WHERE { ?property rdfs:subPropertyOf* db:views_includes.
?property rdfs:domain ?x .
?property rdfs:range ?y .
}
```

The result of the second query is to display the relationship between categories and objects that are subject to categories using the categories_includes property.

Query 3 - Demonstrates the query based on the data type property. Defines the likelihood_of_exploit property values for all categories and displays the list of categories and the corresponding value for these categories of the likelihood_of_exploit property. Inquiry 3. likelihood_of_exploit property values for categories:

```
PREFIX db: <http://www.semanticweb.org/Zh/ontologies/2018/6/CWE#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
SELECT ?t ?y WHERE
{// ?x db:categories_member ?t.
?x db:likelihood_of_exploit ?y
}
```

As a result of the third query, a list of values of the likelihood_of_exploit property for categories objects is displayed.

5 Conclusion

CWE projects are hierarchical classifications and are used primarily in this capacity. They provide a way to formally describe information security phenomena for use as a common language. The main focus is on common use, so MITRE attracts specialists from both the scientific community and industry to develop it. These projects serve as an important tool for constructive interaction in the field of information security.

The authors have designed an ontological model of these weaknesses from the point of view of the architectural concept. The article considers the possibilities of using the ontology of software weaknesses.

The developed software is integrated with the Protege ontology editor, which is used for ontology design and maintenance. As a result, a system has been created that provides an effective filling of the knowledge base with knowledge, has an ergonomic interface for expanding the existing model, combines the ability to convert new changes in the SID standards to the OWL language and the ability to visualize the schemes of structural models of paths, knowledge of which is extracted from the used database.

Queries for selection by condition, in the case of an ontological database, are made in SPARQL. It is a language for querying data represented in the RDF model, and it is also a Protocol for transmitting requests and responses.

References

[1] Bhandari P., Singh M., *Formal Specification of the Framework for NSSA* (2nd International Conference on Intelligent Computing, Communication & Convergence. Procedia Computer Science **92**, 2016), 23-29. DOI: 10.1016/j.procs.2016.07.318.

[2] Sanjay Kumar Malik, Rizvi Sam., *An ontology framework for semantic web illustrating ontology merging* (7th International Conference on Next Generation Web Services Practices, 2011), 227-232. DOI: 10.1109/NWeSP.2011.6088182.

[3] Khoroshevsky V.F. "Ontology Driven Software Engineering: Models, Methods", *Implementations Ontology of designing* **9**:4(2019), 429-448 [in Russian]. DOI:10.18287/2223-9537-2019-9-4-429-448.

[4] H. Arman, A. Hodgson, N. Gindy, "An ontology-based knowledge management system to support technology Intelligence", *International Journal of Industrial and Systems Engineering* **5**:3(2010), 377-389.

[5] "Software Architecture" , https://softwarearchitectures.com

[6] "The national chamber of entrepreneurs of the Republic of Kazakhstan Atameken" , https://atameken.kz.

[7] "OWL 2 Web Ontology Language" , https://www.w3.org/TR/owl2-overview/

[8] Hitzler P, Krцtzsch V, Rudolph S. *Foundations of Semantic Web Technologies* (Chapman & Hal l. CRC, 2009).

[9] "SPARQL Query Language for RDF" , https://www.w3.org/TR/rdf-sparql-query/

[10] "A free, open-source ontology editor and framework for building intelligent systems" , http://protege.stanford.edu/

[11] Sartabanova Zh., Dimitrov V. *Overview of the CWE software weaknesses system* (Proceedings of the international scientific conference "Problems of applied mathematics and computer science" , 2017), 309-311.

[12] "CWE (2020). Common Weakness Enumeration - A Community-Developed List of Common Software & Hardware Weakness Types" , http://cwe.mitre.org/

[13] Sartabanova Zh., Dimitrov V. *Modelling of CWEs on the CWE-287 example* (CEUR Workshop Proceedings, **2464**, 2019), 90-94.

[14] "Recommendation ITU-T X.1525" , https://www.itu.int/rec/T-REC-X.1525-201504-I