UNIVERSITY OF TARTU
Faculty of Science and Technology
Institute of Computer Science
Computer Science Curriculum

Mai Ristioja

OwlParser: Parsing OWL-based Blockchain Security Ontology

Bachelor's Thesis (9 ECTS)

Supervisor(s): Mubashar Iqbal, MSc
Prof. Raimundas Matulevičius, PhD

Tartu 2022

Parsing OWL-based Blockchain Security Ontology

Abstract:
Ontology representations are often used in areas when the knowledge is still incoherent and there exists a need to clarify and agree on definitions, concepts and relations. Ontology representations in their written form are however difficult to comprehend and thus tools for better exploring and understanding the information contained therein are in demand. Some tools providing such functionality exist, of which Protégé is the most common. This, however, is a professional-use application which can be difficult to use for less experienced user and which also needs installation. The aim for this thesis is to develop a web-based tool for exploring ontology representations based on the security and risk management (SRM) domain model and which also provides additional features to support ontology representations about blockchain security. The developed tool named OwlParser loads OWL files from the local computer or the web, allows browsing the content of the ontology representation and provides SRM-based navigation. We hope the tool will be useful for both students as well as professionals in helping to study and explore SRM-based ontology representations.

Keywords: ISSRM, exploring ontology, blockchain security concepts, web development

CERCS: P175 – Computer Science, System Theory

OWL formaadis oleva plokiahela turvalisuse ontoloogia parsimine

Lühikokkuvõte:
Ontoloogia esitusi kasutatakse sageli ebaselgetes või veel selgelt väljakujunemata valdkondades, kus vajatakse definitsioonide, kontseptsioonide ning nende vaheliste suhete kohta selgitusi ja kokkuleppeid. Kirjalikus vormis ontoloogia esitusi on aga keeruline hoomata ning seega eksisteerib vajadus tööriistade järele, mis aitaksid neis sisalduvat infot paremini uurida ja mõista. Leidub vastava funktsionaalsusega tööriistu, millest kõige laiemalt kasutatav on Protégé. Kuid Protégé on professionaalseks kasutuseks mõeldud rakendus, mis võib tavakasutajale osutuda keerukaks ning mis vajab ka paigaldamist. Selle lõputöö eesmärk on luua veebipõhine tööriist turvalisuse ja riski haldamise (SRM) valdkonna mudelil põhinevate ontoloogia esituste uurimiseks, mis pakub lisavõimalusi ka plokiahela turvalisuse ontoloogia esituste toetamiseks. Arendatud tööriist nimega OwlParser laeb ontoloogia esituse kohalikust arvutist või veebist, võimaldab ontoloogia sisu sirvimist ning pakub SRM-põhist navigeerimist. Me loodame, et sellest saab kasulik tööriist SRM-põhiste ontoloogia esituste uurimiseks nii tudengite kui professionaalide jaoks.

Võtmesõnad: ISSRM, ontoloogia, plokiahela turvalisuse mõisted, veebiarendus

CERCS: P175 - Informaatika, süsteemiteooria

Contents

1 Introduction

Ontology representations are often used in different domains when the domain is inconsistent, and there is a need to clarify definitions, concepts, and relations between them to avoid misunderstandings. Ontology representations are usually written down using languages created explicitly for this purpose, e.g., the Web Ontology Language (OWL) [1]. However, for the purpose of exploring the knowledge contained therein, the written form has several shortcomings. For instance, it does not offer a clear visual overview, and has rigid navigation and error-prone editing, especially when dealing with bigger ontological representations. To overcome the aforementioned deficiencies, several tools have been developed for working with ontologies, of which Protégé [2] is the most widely used. Protégé's features include editing and adding terms to ontology representations, and visualization of the contained hierarchies. However, Protégé is a rather detailed application suitable for professional use. This can make its use time-consuming and complicated for people with little previous experience with ontological representations. Protégé is a general tool for all OWL-based ontological representations and does not provide comfortable navigation for specific ontological representations.

The purpose of security risk management (SRM) is to provide means for evaluating information system's security. Ontology representations based on the SRM domain model are concrete examples that could benefit from the SRM domain model specific navigation in the user interfaces of ontology tools. The recently built healthcare security ontology (HealthOnt [3], [4]) is based on the SRM domain model and contains knowledge about blockchain and blockchain-based healthcare applications' security. The ontology is dynamic, extendable, and reusable since it can be integrated with other ontology representations in the information security domain. The need to comfortably explore this ontology and similar ones is the primary motivation of this work.

The goal of this thesis is to design and implement a web-based tool for easy browsing and exploring of the SRM-based ontologies, with built-in support for ontological representations about blockchain security. Based on this goal, the following research question is formulated: ***How to create a web-based tool for parsing and exploring SRM-based ontology representations?*** The web-based tool, named OwlParser, aims at ease of use for the practitioners to gain overview, find concepts, definitions and relations from the ontology representation. OwlParser allows for uploading an .owl file from the local computer, opening an ontology using an internationalized resource identifier (IRI) and provides options to open example ontology files. The tool supports browsing the content of the ontology and navigating it through an SRM-based navigation tree with filtering capabilities and contains SRM-based help information and tooltips. A recent version of OwlParser is hosted at `https://owlparser.cs.ut.ee`.

The development of OwlParser begins with the formulated problem statement. We design the two personas to represent the persons most likely to use the tool. Next, a set of initial requirements for the tool were formulated (and later refined). Use cases

and a use case diagram were created to assist the design process. Based on the initial requirements and use cases, an initial conception of architecture and a set of suitable technologies were selected for the tool. The components of the applications were then designed and developed. The introduction to the tool, tasks, and interview questions were created to evaluate the prototype of OwlParser in user research and usability testing. Test users were selected based on the personas, and user research was conducted. Finally, the results were analyzed, and OwlParser was enhanced based on user feedback.

The rest of this thesis is structured as follows. Chapter 2 presents background about the concepts of ontology, the SRM domain model, and blockchain. Chapter 3 describes the design of the tool, including the requirements, potential users, use cases, and architecture of OwlParser. Chapter 4 gives the implementation details as well as usage scenarios. Chapter 5 contains the descriptions of the evaluation of OwlParser. Chapter 6 presents an overview of the discussion, threats to validity, and future work.

2 Background

This chapter gives an overview of the security and SRM domain model and its concepts. We describe ontology, ontology representations, and the purpose of using it. We also give an overview of blockchain and its main security properties.

2.1 Security Risk Management Domain Model

SRM domain model (Fig. 1) [5], [6] helps to protect the assets of organization from all damages to information system security using risk management approach. Assets that are supported by information system must be protected against all corruption in terms of confidentiality, integrity, and availability, for they are susceptible to security risks.

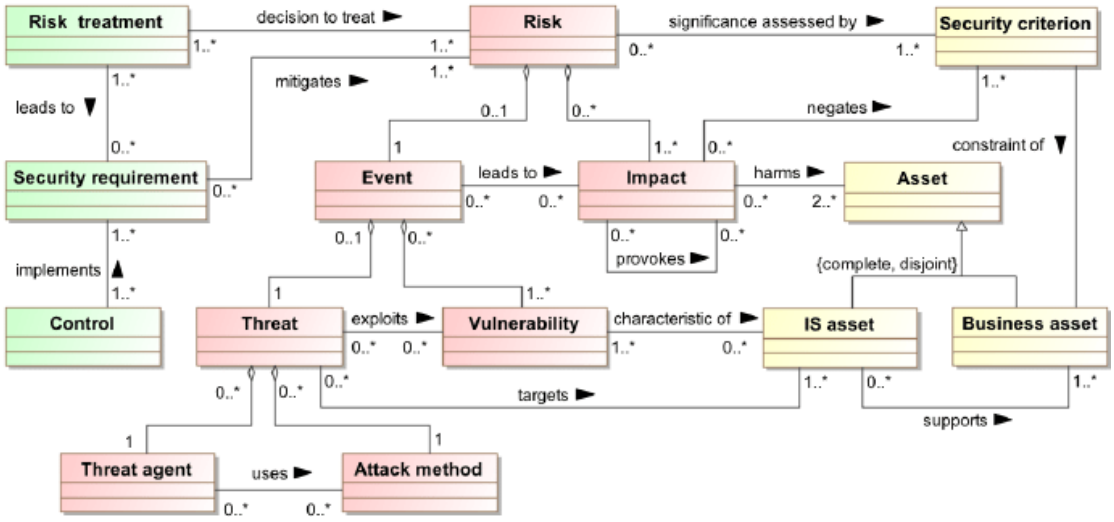

Figure 1. SRM domain model (adapted from [5], [6])

The concepts in the SRM domain model are distributed into three categories: asset-, risk- and risk treatment-related concepts.

Asset-related concepts define which assets need protection and by which criteria to ensure the safety of the assets. Assets can be anything that has importance to the organization in terms of attaining its objectives; assets are in turn divided into business assets and information system assets. The information system asset is a part of the information system that supports business assets. Security criterion describes the security needs of a business asset.

Risk-related concepts stand for the definitions of risk itself. A risk is the combination of one or more vulnerabilities and a threat that leads to an adverse impact on the assets. The impact is a consequence of a risk when the threat is carried out. The event in the

context of information system security means the combination of a threat and one or more vulnerabilities. A vulnerability is a weakness or a flaw of an information system asset. A threat describes a possible attack or security breach which might extend to damaging the assets. A threat agent may harm the information system assets. An attack method is a means of how a threat agent executes a threat.

Risk treatment-related concepts depict which decisions, requirements, and other means should be specified and implemented for the sake of mitigating potential risks. A security treatment is the decision to handle detected risks. A security requirement is the elaboration of a decision to mitigate the risk. Controls or countermeasures are instruments for enhancing security.

2.2 Ontology

Ontology is a standard specification of a shared theory that manages the structure of reality [7]. According [7], ontology means a specific kind of information object or computational artifact in the context of computer science. Ontologies are means of creating formal models of the system's structure, e.g., relevant entities and relations between them which all help achieve the goal. Entities are divided into concepts and relations by unary and binary predicates. The central part of ontology consists of a hierarchy of concepts with a direction from general to specific. Studer et al [8] defines ontology as a formal, explicit specification of shared concepts.

Ontology representations use languages created explicitly to represent the knowledge contained in the ontology in human-readable form and to enable the processing and sharing of knowledge [9]. One such language is Web Ontology Language (OWL), a semantic web language that is created for representing rich and elaborate knowledge about things, groups of things, and relations between things [1]. OWL is designed for use by applications and offers, in addition to the formal semantics, also additional vocabulary, and by this enables better interpretability of content compared to many other ontology languages [10]. OWL adds more opportunities to describe classes and properties, relations between classes, cardinality, equality, characteristics of properties, and enumerated classes. OWL is a part of the W3C (World Wide Web Consortium) recommended stack that is related to the semantic web (e.g., internet vision for the future where information is given explicit meaning and with that enabling better processing and integration of the available information from the web) [10].

2.3 Blockchain

Blockchain is a distributed ledger technology that is based on cryptography, makes it possible to move data through a peer-to-peer (P2P) network without the need for third-party control, and enables performing secure transactions [11]. According to Bodhke et al [12], the main goal of blockchain is to store and share data in a secure

manner. Blocks of data are connected with each other through links. When adding new data, a new link is created, and the chain of blocks lengthens. Effectively, it is impossible to change the blocks retrospectively, for change in the block results in a change in its cryptographical link, and the chain breaks. The authors argue that blockchain technology enables the treatment of different security attacks, including avoiding "single point of failure" attacks and web attacks that take advantage of distributed network nodes.

Iqbal and Matulevičius [13] describe different features of blockchain that allow its use for mitigating security risks in healthcare systems. The security features of blockchain, e.g., immutability and permissioning, are described in Table 1. The authors argue that these features enable properties of transparency and tamper-resistance, which are the backbone of the transactions' security, speed, and effectiveness.

Table 1. Blockchain security features (adapted from [13])

Feature	**Description**
Immutability	Impossible to delete or change added data retrospectively.
Decentralised	Network is managed by a group of distributed nodes.
Distributed	Computing power of peer-to-peer network is shared between the network nodes.
Consensus	Helps to maintain the immutability of the ledger.
Provenance	Every party can verify the authenticity of the action.
Tamper-evident	Any counterfeiting or tampering the content is detected because of the blockchain security characteristics.
Cryptography	Allows blocks of the blockchain to be securely connected.
Distributed access control	Blockchain-based access mechanisms to the resources are decentralised and distributed.
Permissioning	Only certain parties get permissions to perform some actions.
Pseudoanonymous	Masking the identity of the user.

3 OwlParser Design

In this chapter, we describe how the design of OwlParser evolved. We first depict two personas representing the likeliest potential users. We then formulate a number of use cases to clarify how a person uses the tool. Based on the use cases, we give the requirements and describe the architecture of the tool.

3.1 Personas

The application's users include IT specialties, security officers, security engineers, other cyber security professionals, and everyone willing to learn about blockchain security, specifically concepts concerning blockchain security and the SRM of traditional applications using blockchain [3], [14]. Here, we use two personas to analyze the potential users who most likely use the web-based ontology parsing tool.

3.1.1 Student

Name: Marian
Age: 24
Study domain: Computer Science
Level of study: 1st year Master's student

Technical skills	**Intrapersonal and business skills**	**Goals with the application**
• Coding (medium level Python and Java) • Software testing • Computer security basics • Operating system and network technology basics • Basics of mobile and front-end development	• Good communication and team working skills • Self-confidence • Critical thinking	• Get an overview of SRM concepts and see examples of entities belonging to SRM model • View a specific ontology and gain understanding of how different entities relate to each other • Gain enough information to pass the assignments about blockchain security

3.1.2 Security Analyst

Name: Andreas
Age: 38
Level of study: Master's in Cyber security

Technical skills	Intrapersonal and business skills	Goals with the application
• Security processes and solutions analysis and verification • Risk recognition and assessment • Cryptography basics • Network security design	• Good communication and team working skills • Problem solving • Self-responsibility • Decision competence • Design and innovation thinking • Willingness to improve • Legal and compliance skills • Product development skills • Product management skills • Human resources development skills • Business analysis skills	• Get an overview of specific area of blockchain security • See all possible countermeasures to mitigate certain threats

3.2 Use cases

To assist the design process and formulation of requirements for the user interface, and based on the personas and the problem statement, we formulated several use cases (Fig. 2) for the tool.

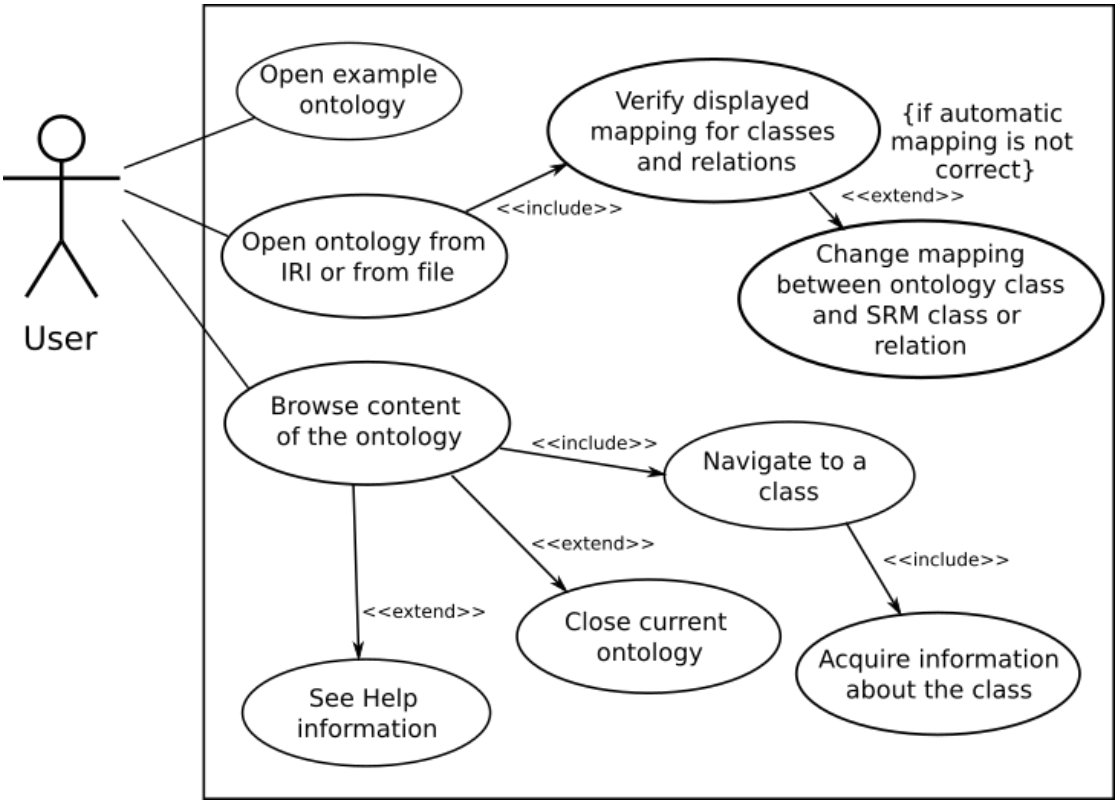

Figure 2. A use case diagram of the tool, high-level view.

3.2.1 Open example ontology

Actors:	User
Pre-conditions:	-
Normal flow:	1. The User clicks on a button to open an example ontology on the loading page, or clicks on an example ontology item in the menu on any page. 2. The tool loads the selected example ontology and navigates to the browsing page.
Exception flow:	If in step 2 of the normal flow the tool fails to download or load the example ontology it displays an error to the user.
Post-conditions:	The selected example ontology is loaded, and the browsing page is displayed.

3.2.2 Open ontology from a file

Actors:	User
Pre-conditions:	-
Normal flow:	1. The User is or navigates to the loading page or opens the ontology loading dialog from the menu. 2. The User activates the "Upload file" tab, if not active. 3. The User activates the file selection component to open a file selection dialog. 4. In the file selection dialog, the User selects an ontology file and confirms the selection. 5. The file selection dialog is closed. 6. The tool loads and parses the selected ontology file and navigates to the SRM classes mapping page.
Exception flow:	If in step 4 the User aborts file selection, the tool returns to the previous step state as if the User had not activated the file selection component. If in step 6 loading or parsing of the selected file fails, an error is displayed to the User, and the tool navigates to the loading page.
Post-conditions:	The selected ontology file is loaded, and the browsing page is displayed.

3.2.3 Open ontology from IRI

Actors:	User
Pre-conditions:	-
Normal flow:	1. The User is or navigates to the loading page or opens the ontology loading dialog from the menu. 2. The User activates the "Upload from IRI" tab if not active. 3. The User activates the "Load ontology from IRI" text field and types or copy-pastes the ontology IRI. 4. The User clicks the "Load" button. 5. The tool downloads and parses the ontology from the given IRI and navigates to the SRM classes mapping page.
Exception flow:	If in step 5 downloading or parsing of the selected file fails, an error is displayed to the User, and the tool navigates to the loading page.
Post-conditions:	The selected ontology file is loaded, and the browsing page is displayed.

3.2.4 Verify classes and relations mapping

Actors:	User
Pre-conditions:	The tool has finished loading (or downloading) and parsing an ontology file and has navigated the User to the SRM class mapping page.
Normal flow:	1. The SRM class mapping page presents the User with the list of SRM classes and their (initially) automatically detected mappings, as well as editing controls. 2. The User may apply any of the following actions: • The User may add or change any mapping by clicking on the editing button next to the mapping. • The User may remove any mapping by clicking on the delete button next to the mapping. • The User may restore any mapping to their automatically detected initial value by clicking on the restore button next to the mapping. 3. The User confirms the SRM class mappings by clicking on the "Next" button. 4. The tool navigates to the SRM relations mapping page, which presents the user with the list of SRM relations and their (initially) automatically detected mappings, as well as editing controls, similarly to the SRM class mappings page. 5. The User may apply action to add, change, remove or restore the SRM relation mappings similarly as on the SRM class mapping page. 6. The User confirms the SRM relation mappings by clicking on the "Finish" button. 7. The tool navigates to the browsing page, which respects the provided SRM class and relation mappings.
Exception flow:	When in step 2 or 5 the User clicks on the "Cancel" button, the tool closes the loaded ontology and navigates to the loading page. When in step 5 the User clicks on the "Back" button, the tool navigates back to the SRM classes mapping page and continues with step 2.
Post-conditions:	The tool has navigated to the browsing page, which uses the provided SRM class and relation mappings.

3.2.5 Browse ontology content

Actors:	User
Pre-conditions:	The tool has navigated the User to the browsing page.
Normal flow:	1. The User may apply any of the following actions: • The User may navigate to view information about classes contained in the ontology by clicking on their links in the left-side navigation panel or the right-side main view. • The User may read information about the currently active class in the main view. • The User may view the SRM class hierarchy in the left-side navigation panel. – The User may filter the SRM class hierarchy. – The User may toggle the display of empty SRM categories in the SRM class hierarchy. • The User may view the SRM threats by the type of application these target in the left-side navigation panel. • The User may close the loaded ontology by clicking on the close button. • The User may load another ontology using the menu. • The User may view help information by clicking on the help button. 2. The User clicks on the close button to close the loaded ontology. 3. The tool navigates the User to the loading page.
Exception flow:	-
Post-conditions:	-

3.3 Requirements

The requirements were developed in response to the problem statement, research goals, and input from users. Several requirements, however, were added and amended repeatedly during the development process. The requirements define the expected behavior of OwlParser and its components and serve as development guides.

R1 Top-level application requirements

R1.1 The application is a web application.
R1.2 The application UI contains a header.
R1.3 The application UI contains the page content view.
R1.4 The application UI contains a footer.

R1.5 When first launched, the application displays the loading page in the page content view.

R2 Header and Footer

R2.1 The header always displays the name of the application.
R2.2 The header always displays a help button, which opens a modal help dialog when clicked.
R2.3 The header contains a drop-down menu that is visible on all pages except on the loading page.
  R2.3.1 The drop-down menu provides an option to load another ontology representation.
  R2.3.2 The drop-down menu provides options to load example ontology representations.
  R2.3.3 Loading an ontology representation from the drop-down menu works as on the loading page.
R2.4 The footer component contains links to the Git repositories for the application code and the example ontology files.

R3 Loading page

R3.1 The application contains a loading page that allows ontology representations to be loaded.
  R3.1.1 The loading page allows ontology representations to be loaded from local files on the user's computer.
  R3.1.2 The loading page allows ontology representations to be loaded from external IRIs.
  R3.1.3 The loading page displays the names of a number of example ontology representations and allows them to be loaded.
R3.2 When loading an ontology representation fails for any reason, the user is notified.
R3.3 Upon opening an ontology representation from a local file or an external IRI, the user is navigated to the SRM classes mapping page.
R3.4 Upon opening an ontology representation example, the user is navigated to the browsing page.

R4 SRM classes mapping page

R4.1 The SRM classes mapping page contains a list of SRM class mappings.
  R4.1.1 The mappings are sorted in alphabetically ascending order based on the name of the SRM class.
  R4.1.2 The default mappings are auto-detected from the loaded ontology.

R4.1.3 The user can trigger changing of the mapping via the mapping editing modal dialog.
R4.1.4 Each mapping can be unset.
R4.1.5 Each mapping can be reset to its initial (auto-detected) values.
R4.2 The SRM classes mapping page contains a button to cancel the operation, which returns to the loading page.
R4.3 The SRM classes mapping page contains a button to apply the SRM classes mapping and continue to the SRM relations mapping page,.
R4.4 The SRM classes mapping page does not allow the user to map multiple SRM classes to the same class in the ontology representation.

R5 SRM relations mapping page

R5.1 The SRM relations mapping page contains a list of SRM relation mappings.
R5.1.1 The mappings are sorted in alphabetically ascending order based on the name of the SRM relation.
R5.1.2 The default mappings are auto-detected from the loaded ontology.
R5.1.3 The user can trigger changing of the mapping via the mapping editing modal dialog.
R5.1.4 Each mapping can be unset.
R5.1.5 Each mapping can be reset to its initial (auto-detected) values.
R5.2 The SRM relations mapping page contains a button to cancel the operation, which returns to the loading page.
R5.3 The SRM relations mapping page contains a button to cancel the operation and return to the SRM class mapping page.
R5.4 The SRM relations mapping page contains a button to apply the SRM relations mapping and continue to the browsing page.
R5.5 The SRM relations mapping page does not allow the user to map multiple SRM relations to the same relation in the ontology representation.

R6 Mapping editing modal dialog

R6.1 The mapping editing modal dialog contains the dialog title.
R6.2 The mapping editing modal dialog contains a short explanation text about changing the mapping, which contains the name and type (either class or relation) of the mapping being edited.
R6.3 The mapping editing modal dialog contains a text input to filter the list of displayed options based on the text entered.
R6.4 The mapping editing modal dialog contains a list of applicable options of which only one may be selected.

R6.5 The mapping editing modal dialog contains a button to confirm the mapping. This button is disabled when no option is selected.
R6.6 The mapping editing modal dialog contains a button to cancel the editing and close the dialog.

R7 Browsing page

R7.1 The browsing page contains a left-side navigation panel.
R7.2 The browsing page contains a right-side main view.

R8 Navigation panel

R8.1 The navigation panel allows displaying the SRM-based navigation tree.
  R8.1.1 Next to the SRM-based navigation tree is a filter bar that allows filtering classes by the text inserted.
  R8.1.2 Next to the SRM-based navigation tree is a toggle control to hide or show empty categories in the tree.
  R8.1.3 The SRM classes in the SRM-based navigation tree are organized into three top-level categories for risk-related concepts, risk-treatment-related concepts, and asset-related concepts respectively.
  R8.1.4 The SRM-based navigation tree contains the class hierarchies of the respectively mapped classes in the loaded ontology representation.
  R8.1.5 All subitems in the SRM-based navigation tree are sorted in alphabetically ascending order.
  R8.1.6 SRM categories in the SRM-based navigation tree have tooltips containing definitions or descriptions of the respective categories.
R8.2 The navigation panel allows displaying the classes organized by category based on the type of application they target (i.e., traditional or blockchain).
R8.3 Clicking on a class link from the navigation panel switches the main view to display information about that class.

R9 Main view

R9.1 The main view contains a component that displays the title and creator of the loaded ontology representation as contained in the loaded ontology representation.
R9.2 The main view contains a button to close the loaded ontology representation and return to the loading page.
R9.3 The main view contains information about the currently active (selected) class in the ontology representation, including
  R9.3.1 a list of its derivations (superclasses) from the top-level classes (if any),
  R9.3.2 a list of its direct subclasses,

R9.3.3 information about the SRM relations it is part of,
R9.3.4 information about its use in other (non-SRM) triples contained in the loaded ontology representation.
R9.4 If no class is currently active (selected), this is communicated to the user.
R9.5 Clicking on a class link from the main view switches the main view to display information about that class.

R10 Help dialog

R10.1 The modal help dialog contains at least the following:
R10.1.1 general information about the tool,
R10.1.2 a short description and a diagram of the SRM domain model,
R10.1.3 descriptions on how to use the main components of the application.

3.4 Architecture

The design of the tool is based on requirements and also changed iteratively throughout the development process, as the requirements themselves evolved, and we tried out different technologies and discovered both new opportunities as well as technical needs. However, the conceptual workflow of OwlParser as shown in Figure 3 remained relatively static. It consists of 5 phases:

1. Loading the ontology representation by the user
2. Parsing the ontology representation into a set of triples of strings
3. Preprocessing the set of triples to create an inner model of the loaded ontology representation
4. Mapping of SRM classes and relations to those in the loaded ontology representation by the user
5. Browsing the ontology by the user

Phases 1, 4, and 5 are performed mainly by the OwlParser user, and phases 2 and 3 are internal to the tool itself. The separate phase 4 for mapping SRM classes and relations to their equivalents in the loaded ontology representation is needed because the respective identifiers (IRIs) are not standardized. In phase 3, OwlParser uses regular expressions to guess which identifiers in the loaded ontology representation correspond to their SRM equivalents and provides these as defaults. When example ontologies are loaded, these are always used, and phase 4 is skipped for the user.

We initially considered developing a server-side backend for parsing OWL files using OWL API [15], but this would have required the use of additional complex technologies

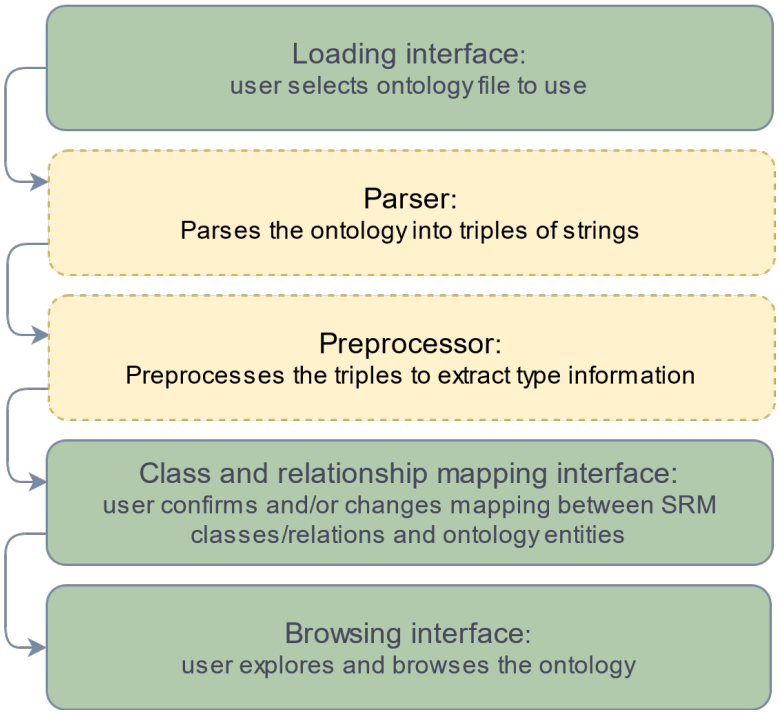

Figure 3. The diagram of the conceptual workflow of OwlParser.

as well as hosting the server backend. We instead created a client-side web application. The latter approach is possible thanks to many new web technologies, which altogether eliminate the strict need for a web server and allow the built application to also run from the filesystem of the user's computer.

3.5 Summary

In this chapter, we introduced two personas representing the target users. We elaborated on several use cases describing the user's interaction with the tool to assist with the design of the tool. Using the use cases and initial problem statement, we then defined the requirements for OwlParser. We also presented an overview of its architecture.

4 OwlParser Implementation

This chapter describes the implementation of the OwlParser. We briefly describe the technologies and development tools used in OwlParser, the data model, principal components, and usage scenarios.

4.1 Tool and Technology Stack

On the high level, OwlParser is a web application written in JavaScript [16] using the React [17] library for generating the user interface (UI). We also considered using TypeScript [18] for the benefit of type safety. However, since adding the required type annotations would considerably increase the developer's workload, we decided that using JavaScript is currently sufficient. The switch to TypeScript can also be made in the future. React was chosen because it allows for creating dynamic web applications with convenient component reuse and is relatively performant.

For the design of UI components Material UI [19] library was chosen. It supports React applications, has a wide choice of built-in components and implements the Material design [20] language, which is clean and minimalistic. OwlParser is built as a static site using npm [21] and can therefore be served as a static set of files from a web server or even from a filesystem directory on the user's computer. The build system is simplified by the use of create-react-app [22] scripts which were used to create the skeleton for the OwlParser code. OwlParser saves its working state in JavaScript variables and the web browser using Web Storage [23]. In Web Storage, the data is serialized as JSON [24].

Reading ontology representations from the user's computer is implemented using the File API [25], and downloading ontology representations from external IRIs is implemented using the Fetch API [26]. The ontology representation input to the application is assumed to have the syntax of RDF/XML [27]. The tool uses the rdfxml-streaming-parser [28] library to parse the input into a set of subject-predicate-object triples [29] which are then used to build an internal model about the ontology representation.

4.2 Data model

The internal data model in OwlParser is built from the subject-predicate-object retrieved from the loaded ontology representation. It contains necessary information about the ontology representation required by the user interfaces for mapping and browsing, including ontology metadata and relationships between classes. For the purpose of displaying information specific to blockchain security ontologies, we also extract from the triples whether a subject is part of a traditional or a blockchain application.

During generation, the internal model is also amended with preliminary mappings of SRM classes and relations with their counterparts contained in the input. These mappings are used by the UI to display a SRM based navigation panel and auxiliary information

in the main view. Due to the lack of a standard strictly specifying exact IRIs for the SRM classes and relations, the mapping is done by matching the identifiers of classes and relations to a set of SRM-specific regular expressions. We also provide a means for the user to verify and change these mappings when loading an ontology representation which is not directly an example provided by the application itself.

4.3 Components Design

Since most of the components of the tool are UI elements, the component architecture is centric on the respective React components. An outline of the React components is given in Figure 4. The App component is one of the top-level components containing all the other components, of which the Header and Footer components are visible on all application pages. These can be seen on the loading page screenshot in Figure 5 together with the LoadingContainer component, which handles ontology input.

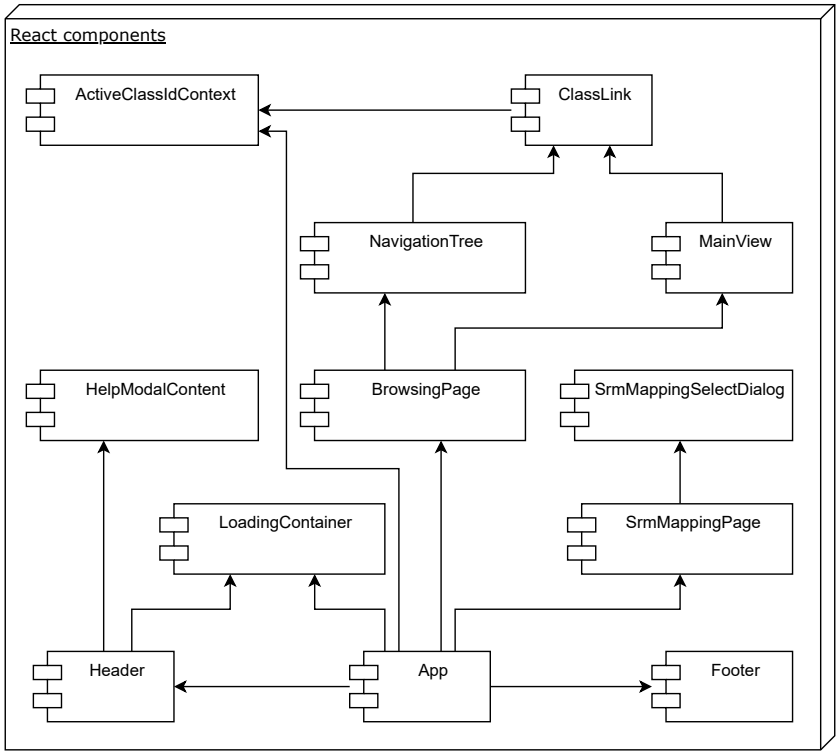

Figure 4. Main React components and their relations

The SrmMappingPage component (Fig. 6) handles the interactive mapping of SRM class and relation relations. It displays the list of mapping items, buttons to edit, restore and delete single mappings, and opens the SrmMappingSelectDialog component as a

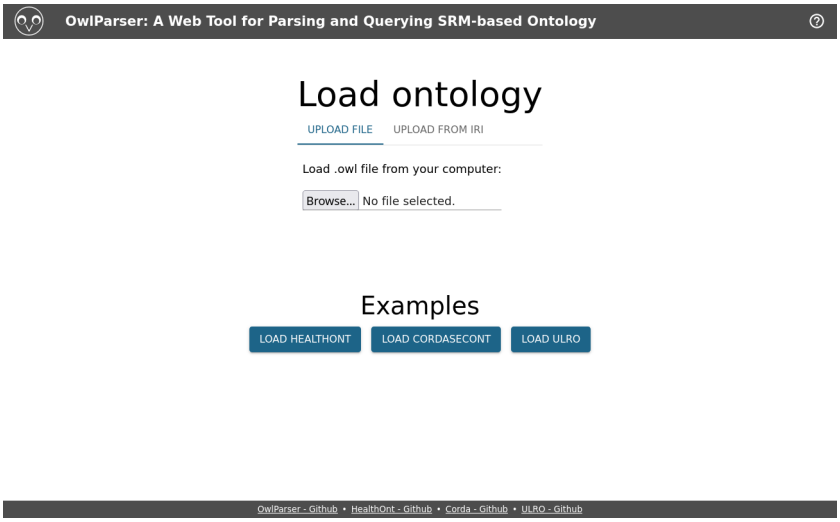

Figure 5. Loading page displaying the Header component in the top, the LoadingContainer component in the middle and the Footer component in the bottom.

modal dialog when the edit button is clicked. The BrowsingPage component (Fig. 7) constitutes the browsing inteface of OwlParser. It consists of the MainView component on the right and the sidebar on the left. The first tab of the sidebar displays the SRM class hierarchy in the NavigationTree component. The handle to the currently active class in the application is stored in the App component and made accessible to other components using the ActiveClassIdContext component. The ClassLink component (instances of which can be seen in Figure 7) renders links to classes in the loaded ontology and also uses ActiveClassIdContext to change the current active class in the App component.

The React JavaScript library strives to adhere to a programming style where components are written in a declarative fashion using JavaScript Syntax Extension (JSX), which is similar to (and may contain) HTML. React components can be nested using JSX. After each change in the components, React converts the resulting structure into the Virtual Document Object Model, compares it to the Document Object Model (DOM) of the web browser, and in a process called reconciliation, efficiently only updates the changed parts of the DOM. Components use functions called hooks to interact with the state and lifecycle of React. As an example of a React component, Figure 8 contains a code fragment of the ClassLink component of OwlParser.

Non-React components handle loading and parsing ontology representations, building the internal model for the loaded ontology, and managing storage. An overview of the most important non-React components and their relations with the rest of the system is given in Figure 9. The App component uses the useLocalStorage() custom React hook to serialize and store the application state in Web Storage as JSON. Since the

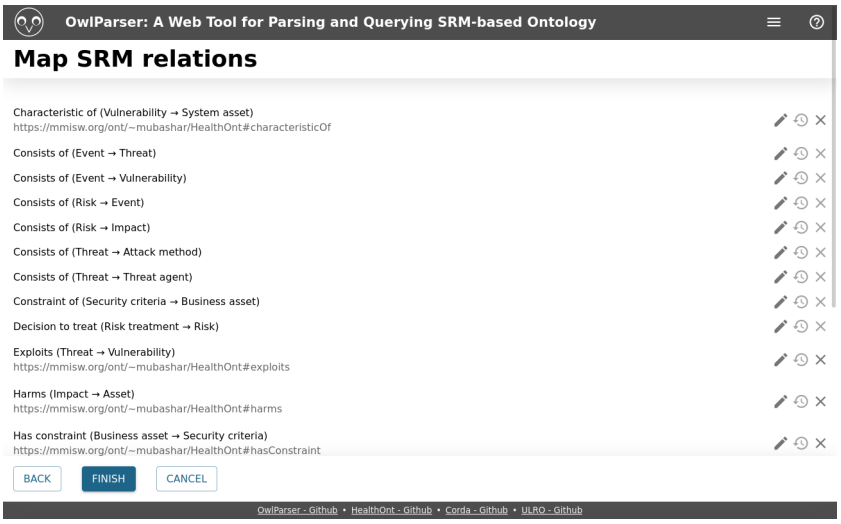

Figure 6. SRM relations mapping page displaying the Header component in the top, the SrmMappingPage component in the middle and the Footer component in the bottom.

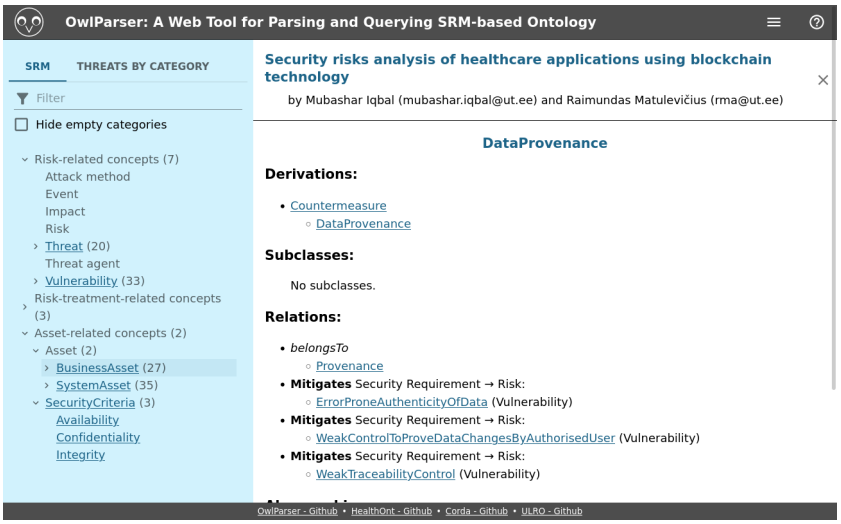

Figure 7. Browsing page displaying the Header component in the top, the Footer component in the bottom, and the BrowsingPage component in the middle, consisting of the left-side side panel showing the NavigationTree component and the right-side MainView component showing the description of the active class. The class links are rendered by the ClassLink component

```
export const ClassLink = ({classId, model, ...props}) => {
  return (
    <ActiveClassIdContext.Consumer>
      {([activeClassId, setActiveClassId]) => {
        let link = (
          <span className="classLink" onClick={() => setActiveClassId(classId)}>
            {minimizeOwlId(classId, model)}
          </span>
        );
        if (!("tooltip" in props)) {
          link = (<Tooltip title={classId} disableInteractive>{link}</Tooltip>);
        } else if (props.tooltip) {
          link = (<Tooltip title={props.tooltip} disableInteractive>{link}</Tooltip>);
        }
        return link;
      }}
    </ActiveClassIdContext.Consumer>
  );
};
```

Figure 8. Definition of the ClassLink component in OwlParser

application's state contains indirect self-referential loops that do not map to JSON, the decycle() and retrocycle() functions are used as intermediaries to convert between these formats. The handleUpload() and handleIriDownload() functions handle loading ontology representations from local files or IRIs in the App component. They stream the loaded ontology to parseStream(), which converts the data to subject-predicate-object triples. To build the internal model of the ontology representation, the triples are in turn passed to buildModel(), which also calls guessSrmClassOwlIds() and guessSrmRelationOwlIds() to automatically detect SRM classes and relations in the model for mapping purposes. These srmClasses and srmRelations objects provide metadata about the SRM model, including display strings, SRM relation domains and ranges, and regular expressions for SRM classes and relations auto-detection.

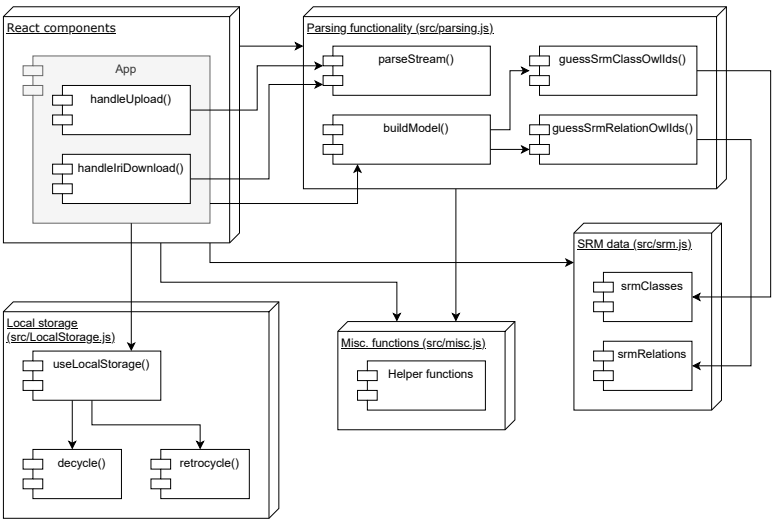

Figure 9. Important non-React components and their relations with the rest of the system

4.4 Usage Scenarios

The following example usage scenarios describe how potential users can interact with the OwlParser tool. The hypothetical users Marian and Andreas correspond to the personas given in Chapter 3.1.

Scenario 1: Finding vulnerabilities and countermeasures

Andreas wants to get an overview of all vulnerabilities the threat of data theft might exploit in the healthcare domain. He also wants to know what are the possible countermeasures to mitigate them. For that, he opens the OwlParser tool in a web browser, clicks on the "Load HealthOnt" button, and from the browsing page, he finds the "Threats" category from the navigation panel and expands it. Then he finds the "DataTheft" link under this category and clicks on it to open it in the main view. From the main view "Relations" section, he finds the keyword "exploits" and under it the keyword "union of" with the list of vulnerabilities (ImproperSecurityControlsForDatabase, IneffectiveCryptographicControl, WeakAccessControl). Under each of those vulnerabilities, he finds the countermeasures that the respective vulnerabilities are mitigated by.

Scenario 2: Finding all SRM model categories under risk-related concepts

Marian studies the SRM model and wants to recall what SRM model categories belong to risk-related concepts. For that, she opens the OwlParser tool in a web browser and clicks on any example file loading button. From the navigation panel, she can already see all subcategories under the Risk-related concepts category.

Scenario 3: Opening file and finding threats that target blockchain applications

Andreas has an SRM-based ontology file containing information about different threats that target information system assets and wants to find out which affect blockchain applications. He opens the OwlParser tool in a web browser, clicks on the "Browse" button under the "Upload File" tab, and loads the ontology file from his local computer. On the mapping pages, he ensures that the automatic mappings are correct and clicks "Next" and "Finish" to continue to the browsing page. He then opens the "Threats by Category" tab in the navigation panel and finds all the threats he was looking for under the "Blockchain application" section.

Scenario 4: Finding business assets that AccessControl supports

Marian needs to find out which business assets does the AccessControl system asset support in the healthcare domain. She opens the OwlParser tool in a web browser and clicks on the "Load HealthOnt" button. On the browsing page, she finds the "SystemAsset"

category in the navigation panel and expands it. She clicks on the "AccessControl" link from the expanded category to open its description in the main view. Under the "Relations" section in the main view, she finds the business assets supported by AccessControl under the keywords "supports" (MedicalRecord, PatientData).

4.5 Summary

In this chapter, we discussed the implementation of OwlParser[1]. We gave an overview of the tools and technologies used to build the tool. We introduced the internal data model with its components and explained the need to map SRM classes and relations. We also discussed the tool's components and described its usage scenarios.

[1] https://github.com/mairistioja/OwlParser-frontend

5 Evaluation

In this chapter, the validation process is described. We introduce the usability testing methodology, its main components, and its uses. The results of the user research and usability testing carried out for the research are then described and discussed. Also, threats to validity are mentioned and briefly discussed.

5.1 Study Design

User research and usability testing was carried out to validate the tool. Five experts were selected based on the description of personas to try out the application (Table 2). A separate interview was performed with each participant, consisting of a short introduction to the tool, asking the interviewee to perform a few simple tasks using the tool, and asking questions about the user's predisposition and the tool's usability. The user research and usability testing protocol that contains the purpose of the testing, short introduction, and task descriptions is given in Chapter 5.3. The consent to act as a participant in a research study is given in Appendix I. The interview questions are given in Appendix II.

Table 2. List of participants

	Expertise	**Mode**	**Time**
Expert 1	Blockchain, Ontology	Face-to-face	55 m
Expert 2	Security specialist	Face-to-face	45 m
Expert 3	Software development	Face-to-face	55 m
Expert 4	Software development, Security specialist	Face-to-face	40 m
Expert 5	Software development, Blockchain, Ontology	Zoom	50 m

5.2 Usability Testing

Usability testing [30] is a methodology to evaluate user experience. The testing is conducted in a way where the researcher interviews and asks the participant to perform tasks on a user interface. The researcher observes and listens to the participant to gather relevant feedback. This helps to identify and prioritize problems with the domain, as well as discover opportunities for improvement. Observing the actions of the participant also gives insight into the user's inclinations.

The central elements of usability testing are the researcher, the tasks, and the participant. The researcher's objective is to lead the participant through the usability testing process by giving directions, answering, and asking additional questions. The tasks are activities similar to those the participant might encounter in everyday life. The participant is either part of the target user group or has needs similar to those of the target user group.

Usability testing can be divided into qualitative and quantitative research. The focus of qualitative usability testing is on gathering insights and discovering problems in the user experience. Quantitative usability testing is focused on collecting metrics, such as task success and time spent on tasks. In the current thesis, we are using the qualitative usability testing approach.

5.3 Usability Testing Protocol

Purpose: You are being invited to participate in the user research because your profile is related to the field of computer science, information security. The purpose of this research is to perform usability testing of the web-based tool that has been developed for exploring and analyzing security risk management (SRM) based ontologies. The process consists of a short introduction of the tool, performing a few tasks using that tool, and answering some questions about using the tool. The purpose of the questions is to discover your prior experience and knowledge about ontologies and the field of computer security. Furthermore, there are questions about the tool's usability according to the tasks performed. Altogether, the process will not take more than 30-40 minutes.

Short Introduction to the Tool: The tool is a web-based application called OwlParser, and its main function is to make the information in SRM-based ontology files readily available without installing any special ontology tools. The following is a short introduction to the user interface of the tool.

On the loading page, there is an option to upload an ontology file from the computer or the web using an IRI (Internationalized Resource Identifier). Also, there is the option to load an example ontology. When an ontology is loaded, the application then provides an interface to map the classes in the loaded ontology to SRM classes, which are prefilled with guessed mappings. The mappings can be changed and must be confirmed to continue. After clicking "Continue," the application similarly displays an interface to map the ontology relations to SRM relations. When clicking "Continue" again, the browsing page of the tool opens.

The browsing page consists of two panels. The left-side panel contains the SRM-based navigation tree and provides an option to hide or show the categories of SRM classes not present in the loaded ontology. Under the SRM-based navigation tree, an extra details section contains ontology classes that do not belong to the SRM model. The navigation panel also has a "Threats" tab which displays the threat classes of the ontology - based on the type of application they target. Clicking on a class link in the navigation panel displays information about that class in the right-side main view. Also, clicking on any class link in the main view opens the information about that class.

Tasks: The purpose of these tasks is to evaluate the user experience of the application on first use, and more specifically with respect to the ease of learning, the efficiency of use,

ease of remembering, understandability, and overall satisfaction.

- Load the example file and find out what vulnerabilities the threat of data theft might exploit and what could be the possible countermeasures to mitigate them.
- Find all SRM model categories that belong to risk-related concepts (including empty categories).
- Find the threats that affect blockchain applications.
- Find out which business assets does AccessControl system asset support.

After each task, the following questions are asked to evaluate the ease of use and ease of remembering:

- How complicated was the task, and why so?
- Did you remember immediately how to begin the task? How easy or difficult was it to start the task?

Interview Questions: The interview questions are given in Appendix II.

5.4 Results

In this chapter, we describe and summarize the user research and usability testing results based on the usability criteria of applicability, ease of use, ease of learning, task efficiency, subjective satisfaction, and understandability. We also present and discuss the numerical ratings received from the interviews.

5.4.1 Applicability

Three of the respondents were familiar with the concept of ontology, and two of them have used ontology representations in their professional life. The respondents who had used ontology representations before unanimously answered affirmatively to the questions about using the ontology representations for getting an overview of some concepts, constructing or gaining information, avoiding misunderstandings, and clarifying some definitions (questions 3-6). For the question about what tools they used, Protégé was the most common. Other than that, only some drawing tools and UML diagrams were mentioned. About Protégé, one respondent mentioned that it is a good tool for creating ontology representations, but it does not provide a good visual overview if there is a need to highlight some aspects.

The respondents were familiar with computer security. They were either working in the field of computer security or knew the area through their research and studies. Of the

respondents, two claimed to have broad knowledge about blockchain security; one said that he is currently writing a paper on blockchain security and the other two had at least general knowledge about it.

On the question about rating the applicability of the tool, some respondents only gave a numerical rating. Among the respondents who also gave reasons for this, one mentioned that the tool contains much information, and it is nice to have this tree view on the left to see the hierarchy of different elements as well as many links in the webpage where the different concepts can be accessed quickly. Another respondent added: "It can be a nice alternative to Protégé because with Protégé, you have to download a .exe file, open it, and so on, but this is much simpler and can be used especially if you want to show the ontology to someone." The respondent who gave the lowest rating noted that "if I would have such file then, of course, it would be useful to me to understand what are the different risks, what are the connections and kind of like another version of database where I can look some information... At the moment, it is not very applicable." One respondent suggested adding some visual diagrams to the tool (See Table 3). This led us to add the diagram of the SRM model to help modal.

Table 3. Suggestions for applicability

Suggestion	Number of respondents suggesting	Implemented in the final version
Add visual diagrams to the tool	1	Partly

5.4.2 Ease of use

The respondents commended the general ease of use. For example, one of them pointed out that "it is very clear and it has very clear interface. Everything is in the correct position." Additional suggestions were made to highlight the tabs on the navigation panel more and also to separate the "Load example" button more from the main form on the loading page (See Table 4). These simple suggestions were implemented.

Using the loading page was straightforward to all respondents. Using the mapping interface brought questions from some respondents about the page's purpose and suggestions for improving it. These included showing explanatory text on the page, showing more information on the input while editing, having the mappings sorted alphabetically, and also returning to the previous page without canceling the process (See Table 4). In response to these suggestions, we added explanatory texts showing more information to the mapping page and to the mapping editing modal dialog, and implemented alphabetical sorting of mappings. Returning to the previous page without canceling the process

was partly implemented. However, the effect of going back is that the changed mappings are not saved and must be re-entered when continuing.

Table 4. Suggestions for ease of use

Suggestion	Number of respondents suggesting	Implemented in the final version
Separate the “Load example” button more from the main form on the loading page	1	Yes
Highlight the tabs on the navigation panel more	1	Yes
Showing more information on the input while editing the mapping	1	Yes
Sort the mappings alphabetically	1	Yes
Return from the mapping interface to previous page without cancelling the process	1	Partly
Sort the names in navigation tab in alphabetical order	1	Yes
Rename the “Threats” tab to “Threats by Category” for clarity	1	Yes
Add tooltips that show more detailed information about specific categories	1	Yes
Change the checkbox “Show empty categories” to "Hide empty categories"	2	Yes
Add additional ontologies as examples	1	Yes
Add some export options (including a means to export the mapping so that it can be used with the same file again from the beginning of the workflow)	1	No
Add collaborative editing features	1	No

Using the navigation panel was very easy, according to three of the respondents. Suggestions for improvement included organizing the names in alphabetical order, renaming the “Threats” tab to “Threats by Category” for clarity, and adding means to get more detailed information either as the form of simple question mark buttons or something similar to get more detailed information about specific categories (See Table 4). To answer these suggestions, alphabetical ordering was added, the tab name was renamed according to the suggestion, and getting more information about categories was implemented as

tooltips showing definitions for those categories.

According to three of the respondents, using the main view was also very easy. An example statement: "It is clear. Everything is in the headings and subheadings, and bullet points. I like that there is navigation control as well. When I click on here, it moves to the detailed information page. It's clear, and it should be like this." Another respondent remarked it was well-structured but a little harder to use because he was not yet accustomed to interacting with this interface. One of the respondents said it was a bit complicated because of the foreign terminology, and it would be good to read the documentation first about what the different relationships mean. Also that it is not clear what the name of the active page is.

On expectations of some parts of the tool to work differently, three respondents said everything was working as expected. One respondent stated that he expected the checkbox "Show empty categories" to work the opposite way because usually everything is shown by default, and there is an option to hide the empty categories (See Table 4). It was later implemented according to the suggestion.

In response to the questions about the improvement of the application and missing functionality, several suggestions were given (See Table 4). Suggestions included to add of additional ontologies as examples, to add a search bar for the navigation panel, to show the relationships between different concepts using diagrams, to add some export options (including a means to export the mapping so that it can be used with the same file again from the beginning of the workflow), to provide additional help information, to slightly improve the design of the tabs on the navigation panel to improve their visibility, and perhaps even adding collaborative editing features. In response, the suggestions were implemented except for adding exporting options which would have probably added too much of a workload and adding collaborative editing features, which would have gone too much out of scope for the current thesis.

5.4.3 Ease of learning

The respondents stated that the introduction to the tool was clear, and the tool was easy to learn. On the question of which parts of the tool were difficult to understand, two respondents said that the mapping interface needed more explanation (See Table 5).

Table 5. Suggestions for ease of learning

Suggestion	Number of respondents suggesting	Implemented in the final version
Show explanatory text on the mapping interface	2	Yes

5.4.4 Task efficiency

In response to the question about which activities were cumbersome to perform, two respondents suggested adding a search bar for quicker searching by name. One suggestion was to optionally skip the mapping interface or add navigational buttons to the top so that it would not be required to go through all the rows of individual mappings (See Table 6). In response to the suggestions, a search bar was implemented, navigational buttons in the mapping interface were re-designed so that these are always visible and the need for scrolling is eliminated. Skipping the mapping interface was only implemented for the example files because in the general case, the application might show incorrect results and behave in unexpected ways.

Table 6. Suggestions for task efficiency

Suggestion	Number of respondents suggesting	Implemented in the final version
Add a search bar for the navigation panel	3	Yes
Give an option to do the mapping in the background	2	No
Organise navigational buttons in a more accessible way	1	Yes

5.4.5 Subjective satisfaction

Generally, the respondents were satisfied with the tool. One of them worded the reason for it: “I would say this tool could generalize our knowledge. It is time-consuming to get knowledge from ontology. If an ontology is very big and I explain it to someone, it can take a lot of time to explain everything. Using this tool, the knowledge is categorized in a very good way, so if I want to look at some class then I can easily go there and all information I can find in the summary in the right panel.“

On the question about how likely they would use the tool in the future, two respondents answered that they would use it. One of them said he would recommend it to his students and researchers. One respondent stated that if he had the file, he would use the application. The other two respondents said that it is not very likely; the reasons for it included OwlParser not yet having enough functionality implemented and the respondent not planning to work in a field where OwlParser would be needed.

The design of the user interface was mentioned to be clear, intuitive, user-friendly, and minimalistic. One suggestion was to add a little more distance between the text rows

(See Table 7), which was taken into account and implemented. Also, the respondents did not find any issues with the performance and responsiveness of the tool.

Table 7. Suggestions for satisfaction

Suggestion	Number of respondents suggesting	Implemented in the final version
Add a little more distance between the text rows	1	Yes

5.4.6 Understandability

The respondents stated that the tool was easy to understand, one said it was highly understandable, and two added that the short introduction to OwlParser was helpful. One of them stated: "Very easy to understand the tasks and the tool as well, where to go, what to look, easy to find." The purpose and functionality of all elements of the application were mostly clear. For two respondents, an explanation of the mapping interface was necessary. Another added that some user interface elements needed some explaining.

5.4.7 Ratings for the Tool

The respondents gave numerical ratings for every usability aspect of the tool on a scale from 1 to 5, 1 being least desirable and 5 being most desirable. We computed the mean values to see the central tendency of the ratings, standard deviations to see the variation within the ratings, modes to see the most common ratings, and medians to see which value falls in the middle separating the higher half from the lower half of ordered ratings. The means, standard deviations, modes, and medians of the ratings are shown in the Table 8.

The most varied ratings were given to the applicability of the tool, reasons for that were mentioned earlier. Most respondents still rated the applicability as 5 (most desirable). Ratings for ease of use were high with a mean value of 4.8, mode and median of 5. Ratings for ease of learning were the highest possible. The task efficiency was rated slightly lower than most others, with a mean value of 4.4, a standard deviation of 0.49, and both mode and median as 4. The subjective satisfaction with the tool has a mean value of 4.6, a standard deviation of 0.49, and both mode and median of 5. The understandability of the tool has a mean value of 4.6, and both mode and median as 5. The standard deviation of 0.8 for understandability is larger, which refers to more variety in the answers. Though some aspects of the tool can be improved in the future or have been improved after the usability testing, from the given ratings can be concluded that OwlParser is a highly usable tool.

Table 8. Means, standard deviations, modes and medians for ratings of different usability categories (on the scale from 1 - least desirable to 5 - most desirable)

Question	$\bar{x}$	σ	Mo	Median
How would you rate the applicability of the tool for you?	4.1	1.11	5	4.5
How would you rate the ease of use?	4.8	0.4	5	5
How would you rate the ease of learning?	5	0	5	5
How would you rate the task efficiency?	4.4	0.49	4	4
How would you rate your satisfaction with the tool?	4.6	0.49	5	5
How would you rate the understandability of the tool?	4.6	0.8	5	5

5.5 Summary

In this chapter, we described the evaluation of OwlParser. We gave an outline of the study design and explained the usability testing methodology, its purpose, and its main elements. We included a usability testing protocol which we developed and used while conducting the user testing interviews. It consists of the purpose of the testing, an introduction of the tool, tasks for the participants to perform, and interview questions. We also described and discussed the results of the testing.

6 Discussion and Future Work

This thesis provides a web-based ontology parsing tool to leverage the SRM domain model-based ontology representations without acquiring knowledge of ontology, an editing tool, or SPARQL queries. The OwlParser is built using React, and it is freely available online. To produce the desired outcomes of this thesis, we formulated the following research question: ***How to create a web-based tool for parsing and exploring SRM-based ontology representations?*** We have indeed developed such a tool, and the usability testing results have shown that it is well received. The test users rated the tool created as part of this thesis highly and found it to be generally satisfactory, easy to learn, easy to use, understandable, and efficient. Most respondents also affirmed that the tool is applicable and valuable for them personally. Our present work has a few limitations, which we discuss below. Furthermore, our work paves the way for a variety of future work directions.

6.1 Threats to Validity

The validity of results is a subject of several threats. The selection of test users might have influenced the results, especially due to their small number. To mitigate this, we tried to select test users from different relevant backgrounds. The validity of results is prone to social threats such as hypothesis guessing and evaluation apprehension. Hypothesis guessing might have been a threat because respondents knew the purpose of the research and might unintentionally have based their answers on the intended results and given more agreeable answers. Evaluation apprehension might have been a threat because it is a human tendency to give less correct answers while trying to look better. This might especially be of concern regarding questions the respondents could have interpreted to rate their personal performance. We tried to mitigate these threats by explaining to the test users the testing goals, stressing our need for honest feedback, and assuring them that the interviews were anonymous.

OwlParser is a simple yet usable tool. However, it has some *technical limitations* as well. For example, OwlParser currently supports a small subset of OWL features and might not work with more complex OWL files, e.g., those containing OWL property restrictions in class descriptions. SRM classes and relations can only be mapped to their exact counterparts, which have names. More complex mappings are currently not supported. Persistence of state in OwlParser is presently subject to the limitations of Web Storage, meaning that state may be cleared with web browser caches, and persistence may be undefined in case the application is started from the local filesystem. In the future, adding a back end with storage capabilities could improve user experience, especially if custom mappings are often needed.

6.2 Future work

In line with the goals of this thesis, OwlParser can be extended to enhance usability and ease of use for parsing and exploring ontology representations. In addition to overcoming the limitations stated above, many further developments are possible.

For example, given current functionality, the user interface of OwlParser can be polished to provide a better user experience. Controls to customize the user interface can be added, e.g., to resize or hide the sidebar or toggle full-screen mode. Users might also prefer to customize or re-order the details shown in the main view to show more relevant information to the user, for example, by using collapsible sections. Making the application more mobile-friendly should also be considered.

A significant concern is the lack of a standard SRM ontology representation, which could serve as a common base for all SRM ontology representations. This lack is why OwlParser uses regular expressions to map or fit the loaded ontology representation into the SRM model and why the SRM classes and relations mapping interfaces exist. If such a common base were standardized, such complex mapping logic and the complex mapping user interfaces could perhaps be avoided altogether. Furthermore, support for other models besides SRM could be added more easily.

On the technical level, the OwlParser code needs to be refactored to enable testing, and tests need to be implemented. To increase code quality enforcing stricter rules and switching from JavaScript to TypeScript are also recommended.

7 Conclusion

This thesis aims to develop a web-based tool for easy exploring of SRM-based ontology representations. To achieve this goal, we researched the security risk management (SRM) domain model, ontology, and blockchain security features. We designed and implemented a web-based tool, OwlParser, to make understanding SRM-based ontologies easier and user-friendlier compared to the available tools. To assist the design process, we created two personas representing target users and defined some use cases. We formulated the requirements based on the use cases and problem statement. To implement OwlParser, many different development tools and technologies were used.

We evaluated an initial prototype of OwlParser through user research and usability testing. For that, five different persons were interviewed separately, asked to perform some tasks, and provide feedback to improve the tool's usability. The feedback obtained from the test users was later used to enhance the tool further.

To achieve the goal as mentioned above, we formulated the research question: ***How to create a web-based tool for parsing and exploring SRM-based ontology representations?*** Based on the experience we gained from the development process and the usability testing results, we can affirm that the developed tool fulfills its purpose to parse and query SRM-based ontology representations. The usability categories of applicability, ease of use, ease of learning, task efficiency, subjective satisfaction, and understandability acquired high ratings. Thus we can conclude that the tool is highly usable.

References

[1] OWL Working Group. (Dec. 11, 2012). "Web ontology language (OWL)", [Online]. Available: https://www.w3.org/OWL/ (visited on 01/10/2022).

[2] Stanford University. (2020). "protégé", [Online]. Available: https://protege.stanford.edu/ (visited on 07/25/2022).

[3] R. Matulevicius, M. Iqbal, E. A. Elhadjamor, S. A. Ghannouchi, M. Bakhtina, and S. Ghannouchi, "Ontological representation of healthcare application security using blockchain technology", *Informatica*, vol. 33, no. 2, pp. 365–397, 2022. DOI: 10.15388/22-INFOR486. [Online]. Available: https://doi.org/10.15388/22-INFOR486.

[4] Mubashar Iqbal and Raimundas Matulevičius. (2022). "Security risk analysis of healthcare applications using blockchain", [Online]. Available: https://mmisw.org/ont/~mubashar/HealthOnt (visited on 08/07/2022).

[5] É. Dubois, N. Mayer, P. Heymans, and R. Matulevičius, "A systematic approach to define the domain of information system security risk management", in *Intentional Perspectives on Information Systems Engineering*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 289–306. DOI: 10.1007/978-3-642-12544-7_16.

[6] R. Matulevičius, *Fundamentals of Secure System Modelling*, 1st ed. Springer International Publishing, 2017, p. 225, ISBN: 978-3-319-61717-6. DOI: 10.1007/978-3-319-61717-6.

[7] N. Guarino, D. Oberle, and S. Staab, "What is an ontology?", in *Handbook on Ontologies*, S. Staab and R. Studer, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 1–17, ISBN: 978-3-540-92673-3. DOI: 10.1007/978-3-540-92673-3_0. [Online]. Available: https://doi.org/10.1007/978-3-540-92673-3_0.

[8] R. Studer, V. Benjamins, and D. Fensel, "Knowledge engineering: Principles and methods", *Data & Knowledge Engineering*, vol. 25, no. 1, pp. 161–197, 1998, ISSN: 0169-023X. DOI: https://doi.org/10.1016/S0169-023X(97)00056-6. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0169023X97000566.

[9] O. Corcho and A. Gómez-Pérez, "A roadmap to ontology specification languages", in *Knowledge Engineering and Knowledge Management Methods, Models, and Tools*, R. Dieng and O. Corby, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 80–96, ISBN: 978-3-540-39967-4.

[10] OWL Working Group. (Feb. 10, 2004). "OWL Web Ontology Language Overview", [Online]. Available: http://www.w3.org/TR/owl-features/ (visited on 01/10/2022).

[11] T.-F. Lee, H.-Z. Li, and Y.-P. Hsieh, "A blockchain-based medical data preservation scheme for telecare medical information systems", *International Journal of Information Security*, vol. 20, no. 4, pp. 589–601, Aug. 2021, ISSN: 1615-5270. DOI: 10.1007/s10207-020-00521-8. [Online]. Available: https://doi.org/10.1007/s10207-020-00521-8.

[12] U. Bodkhe, S. Tanwar, K. Parekh, P. Khanpara, S. Tyagi, N. Kumar, and M. Alazab, "Blockchain for industry 4.0: A comprehensive review", *IEEE Access*, vol. 8, pp. 79 764–79 800, 2020. DOI: 10.1109/ACCESS.2020.2988579.

[13] M. Iqbal and R. Matulevičius, "Blockchain as a countermeasure solution for security threats of healthcare applications", in *Business Process Management: Blockchain and Robotic Process Automation Forum*, J. González Enríquez, S. Debois, P. Fettke, P. Plebani, I. van de Weerd, and I. Weber, Eds., Cham: Springer International Publishing, 2021, pp. 67–84, ISBN: 978-3-030-85867-4.

[14] M. Iqbal and R. Matulevičius, "Corda security ontology: Example of post-trade matching and confirmation", *Baltic Journal of Modern Computing*, vol. 8, no. 4, pp. 638–674, 2020, ISSN: 22558950. DOI: 10.22364/bjmc.2020.8.4.11.

[15] University of Manchester. (Nov. 7, 2020). "OWL API by owlcs", [Online]. Available: https://owlcs.github.io/owlapi/ (visited on 07/26/2022).

[16] ECMA International. (Jun. 2021). "ECMA-262, ECMAScript® 2021 Language Specification". version 12th edition, [Online]. Available: https://262.ecma-international.org/12.0/ (visited on 08/05/2022).

[17] Meta Platforms, Inc. (Jul. 14, 2022). "React - A JavaScript library for building user interfaces", [Online]. Available: https://reactjs.org/ (visited on 07/20/2022).

[18] Microsoft Corporation. (2022). "TypeScript: JavaScript with syntax for types", [Online]. Available: https://www.typescriptlang.org/ (visited on 07/26/2022).

[19] Material UI SAS. (2022). "MUI: The React component library you always wanted", [Online]. Available: https://mui.com/ (visited on 07/26/2022).

[20] Google LLC. (2022). "Material Design", [Online]. Available: https://material.io/ (visited on 07/26/2022).

[21] npm, Inc. (Jul. 20, 2022). "npm", [Online]. Available: https://www.npmjs.com/ (visited on 07/20/2022).

[22] Facebook, Inc. (2022). "Create React App", [Online]. Available: https://create-react-app.dev/ (visited on 07/26/2022).

[23] Web Hypertext Application Technology Working Group. (Jul. 20, 2022). "HTML Standard, 12 Web storage", [Online]. Available: https://html.spec.whatwg.org/multipage/webstorage.html (visited on 07/20/2022).

[24] ECMA International. (Dec. 2017). "ECMA-404, The JSON data interchange syntax". version 2nd edition, [Online]. Available: https://www.ecma-international.org/publications-and-standards/standards/ecma-404/ (visited on 08/05/2022).

[25] World Wide Web Consortium. (Jun. 4, 2021). "File API, W3C Working Draft", [Online]. Available: https://www.w3.org/TR/FileAPI/ (visited on 07/20/2022).

[26] Web Hypertext Application Technology Working Group. (Jul. 8, 2022). "Fetch, Living Standard", [Online]. Available: https://fetch.spec.whatwg.org/ (visited on 07/20/2022).

[27] World Wide Web Consortium. (Feb. 25, 2014). "RDF 1.1 XML Syntax, W3C Recommendation", [Online]. Available: https://www.w3.org/TR/rdf-syntax-grammar/ (visited on 07/20/2022).

[28] R. Taelman. (Aug. 11, 2021). "GitHub - rdfjs/rdfxml-streaming-parser.js at v1.5.0", [Online]. Available: https://github.com/rdfjs/rdfxml-streaming-parser.js/tree/v1.5.0 (visited on 07/20/2022).

[29] World Wide Web Consortium. (Feb. 10, 2004). "Resource Description Framework (RDF): Concepts and Abstract Syntax, W3C Recommendation 10 February 2004", [Online]. Available: https://www.w3.org/TR/2004/REC-rdf-concepts-20040210/ (visited on 07/26/2022).

[30] Kate Moran. (Dec. 1, 2019). "Usability Testing 101", [Online]. Available: https://www.nngroup.com/articles/usability-testing-101/ (visited on 08/05/2022).

Appendix

I Consent to Act as a Participant in a Research Study

Study title: User research and usability testing of the web-based ontology parser

Principal investigator: Mai Ristioja

Introduction: Because of your profile as a student of computer science (alternatively, as a security officer), you are being invited to participate in the user research and usability testing of the web-based tool that has been developed for exploring and analyzing security risk management (SRM) based ontologies. The purpose of this interview is to validate the tool by completing some simple tasks and asking questions about the different aspects of usability.

Content of the study: This study is conducted by a research group from the Computer Science Unit of the University of Tartu (Estonia). Firstly, a short introduction to the tool is given. Then you are asked to perform a few simple tasks and after each of them, there are a few questions about each of them. Following that, there are questions that cover basic information about your prior experience with ontologies and computer security, in addition to questions that cover the different usability aspects of the tool.

Participation requirements: A person who matches the created profiles of potential users of the application and that is a fluent English speaker is eligible to participate.

Duration of the study: The interview will take about 30-40 minutes of your time.

Risks and benefits: The risks that are associated with this research are no greater than those ordinarily encountered in daily life. There are no direct benefits to participants but the researchers anticipate potential societal benefits being derived from their research.

Privacy and confidentiality: The researchers will follow the following procedure to protect participants' identities during this study: The original audio files will remain on the original recording device which is only accessible to the Principal Investigator. The audio files will be transcribed, potential identifiers will be removed or aggregated and the original audio files will be deleted afterward.

Your data and consent form will be kept separate. Your consent form will be stored securely and will not be disclosed to third parties.

By participating, you understand and agree that the data and information gathered during this study may be used by the University of Tartu for publication purposes. However,

any identifiable information will not be mentioned in any such publication or dissemination of the research data and/or results. The University of Tartu requires all research records to be maintained for at least 5 years following the final reporting or publication of a project. Aggregated data will thus be archived by the Principal investigator for that timespan.

Questions about the Study: If you have any questions, comments, or concerns about the study either before, during, or after participation, please contact Mai Ristioja (mai@ristioja.ee).

Voluntary participation: Your participation in this research is voluntary. You may discontinue participation at any time during the research activity. Your decision regarding whether to participate in this study will not result in any loss of benefits to which you are otherwise entitled.

I am age 18 or older. I have read and understood the information above. I want to participate in this research and continue with enrollment in the study _ Yes _ No

Participant: The above information has been explained to me and all of my current questions have been answered. I understand that I am encouraged to ask questions, voice concerns or complaints about any aspect of this research study during the course of this study and that such future questions, concerns, or complaints will be answered by a qualified individual or by the investigator listed on the first page of this consent document.

Investigator: I certify that I have explained the nature and purpose of this research study to the above-named individual(s), and I have discussed the potential benefits and possible risks of study participation. Any questions the individual(s) have about this study have been answered, and we will always be available to address future questions, concerns, or complaints as they arise. I further certify that no research component of this protocol was begun until after this consent form was signed.

Participant

Investigator

II Interview Questions

Applicability

- How well are you familiar with ontologies? If applicable, please specify the main goals you have had previously with ontologies.
- What purposes have you had with ontologies?
- Have you ever used ontologies to get an overview of some concepts?
- Did you use ontologies before to construct or gain information?
- Have you ever used ontologies to avoid misunderstandings?
- Have you ever used ontologies to clarify some definitions?
- What tools have you used previously to explore ontologies?
- What do you know about blockchain security?
- What do you know about computer security in general?
- On the scale 1 to 5, how would you rate the applicability of the tool for you (1 - not applicable at all, 5 - easily applicable)?

Ease of use:

- How easy it was to use the tool in general?
- How easy it was to use the loading page?
- How easy it was to use the mapping interface?
- How easy it was to use the left-side panel?
- How easy it was to find information from the right-side panel?
- On first use, did you expect some part(s) of the tool to work differently?
- Which functionalities do you think are missing?
- Which aspects of the application would you like to see improved and how?
- On the scale 1 to 5, how would you rate the ease of use (1 - very difficult to use, 5 - very easy to use)?

Ease of learning:

- Did the short introduction to the tool make sense?
- Were there any aspects of the tool which were difficult to understand on first use? Which ones?
- On the scale 1 to 5, how would you rate the ease of learning of the tool (1 - very difficult to learn, 5 - very easy to learn)?

Task efficiency:

- Which activities did you find cumbersome to perform?
- On the scale 1 to 5, how would you rate the task efficiency (1 - not efficient at all, 5 - very efficient)?

Subjective satisfaction:

- How satisfied are you with the tool?
- How likely would you use the tool in the future?
- What do you think of the design of the user interface?
- How satisfied are you with the performance and responsiveness of the application?
- On the scale 1 to 5, how would you rate your satisfaction with the tool (1 - not satisfied at all, 5 - very satisfied)?

Understandability:

- How easy or difficult it was to understand the tool?
- Did you understand the purpose and functionality of all elements of the application?
- On the scale 1 to 5, how would you rate the understandability of the tool (1 - not understandable at all, 5 - very understandable)?

Conclusion:

- Do you have anything else to add or suggest?

III Licence

Non-exclusive licence to reproduce thesis and make thesis public

I, **Mai Ristioja**,
*(*author's name)

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to

   reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,

   Parsing OWL-based Blockchain Security Ontology,
   *(*title of thesis)

   supervised by Mubashar Iqbal and Raimondas Matulevičius.
   *(*supervisor's name)

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.

4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Mai Ristioja
08/08/2022