A Mapping Tool for Normative Requirements

JOÃO CARDOSO PINHO DA CRUZ, Instituto Superior Técnico, PT

Organizations tend to implement multiple compliance mechanisms in an effort to improve their security, from legislation to international standards. Blindly implementing these mechanisms, independently from one another, often leads to the duplication of work. The process of homogenizing, mapping and integrating compliance mechanisms solves this issue by allowing researchers to bridge the gap between them. However, these processes consume a lot of time and effort, which is the problem we seek to solve with this paper. We have designed, developed and evaluated a software tool that utilizes NLP and AI techniques to assist researchers in finding the intersections between compliance mechanisms, reducing time spent in the mapping process.

Keywords: Security, Compliance, DSRM, Standards, AI

1 INTRODUCTION

The impact of new security research on organizational security is limited due to organizations' focus on implementing compliance mechanisms, which may not always incorporate the latest research [17]. Compliance mechanisms come in different forms and originate from various sources, resulting in multiple constraints that may target different departments [34, 38]. Blindly implementing these different constraints can lead to duplicated work and wasted time [38]. To address this issue, researchers are developing mappings between different compliance mechanisms, utilizing normative requirements from documents such as legislation, policy or standards.

This paper addresses the practical, compliance, side of the coin and attempts to bridge the gaps between existing compliance mechanisms, bringing research to practical problems. In specific, we discuss how multiple different compliance mechanisms can be compared with the assistance of software, purposefully built to achieve this goal. Previous research has shown that comparing and mapping various compliance requirements is nothing trivial, especially when including multiple models into a single mapping [28]. These processes often require large amounts of effort and time.

To assist researchers, by automating parts of this process, a software tool was developed, tested and validated with data originating from the literature and a mapping created by the authors and validated with specialists. In specific, the researchers automated the comparison process through classic Natural Language Processing (NLP) techniques and through the use of pre-trained Artificial Intelligence (AI) models.

This paper follows the following structure after introduction: in the research methodology we present and reason our choice for the Design Science Research Method (DSRM). The Research Background is the following section, in which we present the required concepts that serve as the building blocks for this paper. We proceed to name the existing problems within this area of research and propose a solution for them. This solution is then demonstrated and evaluated with two practical examples. The conclusion summarizes the work done, identifies limitations, and suggests future research directions.

Author's address: João Cardoso Pinho da Cruz, joaopinhocruz@tecnico.ulisboa.pt, Instituto Superior Técnico, Av. Rovisco Pais, Nº 1, Lisbon, Lisbon, PT, 1049-001.

Term	Description
Normative requirement	Requirement pertaining to some compliance mechanism
Control	Normative requirement specific to the ISO or COBIT standards
Atomic normative requirement Atomic control	Requirement that can no longer be subdivided in multiple requirements [20]
ISO and COBIT	Standards Organizations

Table 1. Additional terms related to modelling and standards

2 METHODOLOGY

The present section introduces our chosen research method, DSRM, which we chose due to its rigorous nature and ease to verify the research resulting from the proper application of this methodology [15]. We applied DSRM to our research field, Information Systems (IS), based on the guidelines presented by Hevner in [15] and summarized in Figure 3.

The authors started off by identifying a problem within the field: the process of mapping between compliance mechanisms, such as international standards, legislation or internal policy, is extremely time consuming and cumbersome for researchers (currently under peer review).

To address the aforementioned challenge, we propose developing a software tool to automate certain repetitive aspects of the process. Our research involved the creation of such a tool and the generation of performance metrics based on literature artifacts. We evaluated the tool through specialist interviews and, finally, communicated our findings for peer review, completing the DSRM process.

3 RESEARCH BACKGROUND

In this section, we present the concepts required to understand our research.

3.1 Homogenization, Mapping and Integration

Homogenization consists in developing a strategy to compare two different models, from both a semantic and structural point of view [27] and is the first process in the pipeline. For models sharing dissimilar structures or semantics, this step is strictly necessary, but can otherwise be skipped. An applicable ontology can be used in this process, supplying a vocabulary and the necessary relationships to make the process easier to execute [8].

Mapping is the process that allows the comparison between two different models and is one that is commonly found across the literature [27]. The outputs (which can also be called artifacts) are often presented in the form of a mapping table, with one model presented across the table's columns and another on its rows. Integration takes the mapping one step further, combining the compared models into a single, unified, model [3].

In addition to the processes described in this subsection, we have included other relevant terms related to Homogenization, Mapping and Integration in Table 1.

3.2 Natural Language Processing (NLP)

Natural Language Processing is defined as an area of research and application that explores how computers can be used to understand and manipulate natural language text or speech to do useful things [9]. In this subsection we introduce NLP and AI. A summary of the most relevant terms for this subsection is presented in Table 4.

A typical NLP pipeline starts by splitting a document into sentences and the sentences into tokens. The data resulting from this process is used to build an inverted index. The keys for the index compose the vocabulary and the nodes they point to are documents from the corpus. Tokens can be further processed through Lemmatization or Stemming to remove derivational suffixes [5]. Both Lemmatization and Stemming have been shown to improve IR performance [5]. N-grams can be parsed from the text and added to the inverted index.

The inverted index provides some statistical data on which documents are more relevant for a given query. In specific, we can evaluate the TF and IDF metrics, both of which can be calculated through many different formulae. TF matches the terms present in both the query and the documents in the corpus, while IDF negatively ranks common terms across the corpus. When multiplied, these two metrics generate TF-IDF, which is one of the classic methods to rank documents for a given query [2, 32]. TF-IDF values for two different documents can then be put into a pair of vectors and the cosine similarity between them calculated, resulting in the final similarity scores.

3.2.1 Sentence Similarity through Artificial Intelligence (AI). Deep Learning has found big applications across many different fields, from reverse image search to speech recognition [35]. One particular application that is relevant to the researchers is Sentence Similarity and Text Transformers, through the use of text encoders.

Text encoders like BERT (Bidirectional Encoder Representations from Transformers) utilize a pre-trained deep learning model to generate a vector of text embeddings. These embeddings can be used in various functions, such as evaluating text similarity [10].

While BERT brought big improvements to the field, it is a very generalized Encoder and, as such, extremely slow, even on modern hardware, to finish pair regression tasks [31]. This problem was addressed by Reimeirs et al. in a 2019 study that specifically addressed sentence similarity, from which Sentence-BERT (SBERT) was created, achieving measured speedups of up to 780 times [31].

4 PROBLEM DESCRIPTION

Mapping between models is a complex, repetitive and time consuming process. When instantiated into real world examples, these mappings often take place between models that originate from compliance mechanisms, such as standards, legislation or policy. However, these mappings prove themselves extremely useful when implementing compliance mechanisms onto pre-existing systems that are already compliant with one of the mapping's parts [26].

The time consuming part of the mapping process is attributed due to its iterative, and repetitive, nature. Assume the example of two ISO standards with n and m atomic controls each: the mapping is be developed by comparing each control from standard 1 with each one from standard 2 [26]. This process necessarily results in a total number of (n * m) comparisons, both when using a supporting ontology [30] or otherwise [26].

Not only is the comparison part of the process extremely time consuming, but it can be compounded if the research is being carried out separately by multiple researchers and then reviewed and analysed as suggested by Mas et al [21], leading to further repetition of the already time consuming iterative process.

To summarize, we have found that the mapping of compliance mechanisms is a crucial but highly time-consuming research area. Furthermore, there has been limited research on improving the methods used in this process. This lack of attention to methodological improvements exacerbates the already significant time investment required for effective compliance mechanism mapping.

5 RESEARCH PROPOSAL

Our proposed solution to solve the named research problems is to utilize custom developed software to automate parts of the homogenization, mapping and integration processes. Our proposed solution is split into two parts, each presented in subsections.

The focal point for the first part of the solution is on the homogenization process. This part is based on converting the documents' requirements into a machine readable format. We refer to the outputs as "homogenized models".

The second part takes as input the homogenized models to attempt a semi-automated comparison. Through the parallel use of both classical NLP techniques and, more modern, neural-network processes. We assess that this automated comparison helps mitigate the research problem, by improving the efficiency of the mapping process, reducing time spent manually comparing every single pair of atomic controls.

5.1 Homogenization

We propose two methods for homogenizing compliance mechanisms (documents) into our desired structure. The first focuses entirely on homogenizing ISO standards, while the second, documented in subsubsection 5.1.2, is more generalized, accepting any kind of compliance mechanism. As required, the two methods produce interchangeable outputs.

The fundamental concept to highlight in the design of this system is the necessity of parsing the documents to extract their atomic controls. A single control can generate one or multiple atomic controls, by subdividing it until each control can no longer be divided.

5.1.1 Homogenizing ISO standards. The main benefit the authors identified from isolating the homogenization process of ISO standards derives from how they are structured. From a generalized point of view, nearly all of the ISO controls already are atomic controls (in a single statement) or can be easily extracted from a well defined enumeration.

We propose that the homogenizing system should take as input the ISO standard's PDF document and output a mapping of the clauses to a list of its atomic controls. Furthermore, every atomic control extracted should be easily reviewed and editable in an efficient manner. An example of how atomic controls are extracted can be found in Table 5.

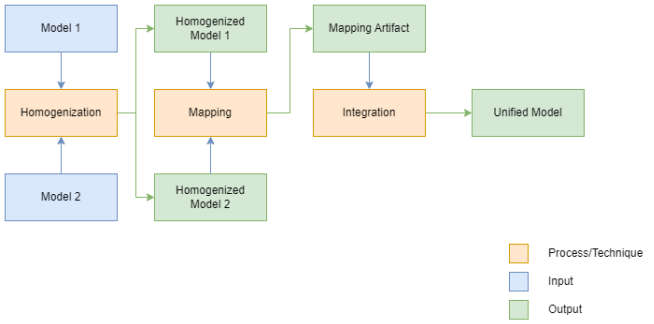

Fig. 1. Steps to create a unified model [26]

Automating the extraction of atomic controls makes the homogenization process more efficient as reviewers would spend considerably less time reading through the table outputted by the homogenizer than if they had to create such a table themselves.

5.1.2 Homogenizing other compliance mechanisms. We argue that creating a homogenization system that only applies to a single class of compliance mechanisms would be a design flaw. Therefore, we propose an alternative module that accepts any list of tuples matching section numbers to atomic controls. This system would require more user time but is capable of being applied to a broader range of compliance mechanisms.

5.2 Mapping

The authors propose that the research problems could be mitigated through the use of a software mapping tool. By automating the most repetitive parts of the mapping process, the comparisons between atomic controls, we can fast-track the whole process. In addition, such a tool could provide some error checking properties when used to compare against pre-existing mappings.

The proposed tool is split into two parts, using classical NLP methods and experimental pre-trained neural networks to compare sentences. The mapper can be easily parametrized with weights to value each method independently. The proposed tool iterates over each pair of atomic normative requirements and compares their similarity.

5.2.1 Mapping with classic NLP techniques. Classical NLP techniques, such as cosine similarity through the use of TF-IDF. These techniques should act as a baseline for the given tool, as they are now a "tried and true" method that was proposed in 1988 [32] and has been in use until the present day [1, 4].

When applied to our problem's, we propose a NLP pipeline to assist in the comparison process, which we summarized in Figure 2.

In order to perform the mapping process, we must determine which controls are the most similar. To achieve this goal, we propose to calculate the cosine similarity of the TF-IDF values between the two artifacts.

There are multiple alternatives to achieve the "raw" TF calculation presented. We opted to use the "log-TF" variation, given by the formula `log(1 + TF)`, where TF is the "raw" TF value. The basis

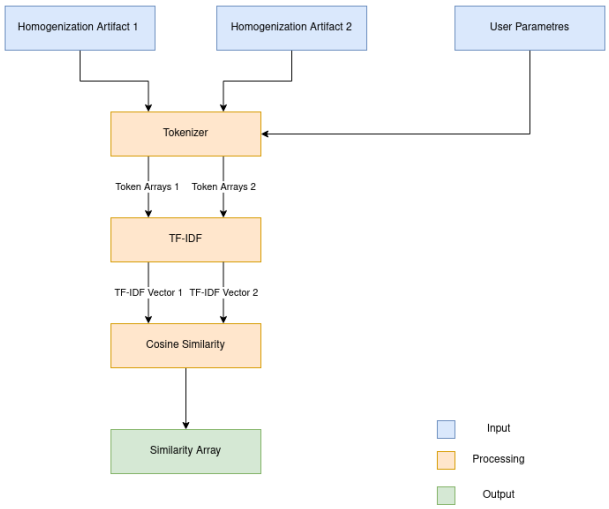

Fig. 2. Mapping with an NLP pipeline

for this choice, is to lower the risk of bias from controls containing a lot of term repetition.

To calculate the value for the IDF, we applied the smoothed out formula for the classic IDF calculation. In this scenario, N is the total number of documents within the corpus and #t is the number of documents in which the term t appears.

$$log(\left(\frac{N}{\#t+1}\right)+1$$

At last, TF-IDF is the simple multiplication of TF by IDF for a given term, in a given document from a given Artifact. This gives us the relevance of a token in relation to its context. By calculating these values to all of the controls in either of the two Preprocessor Artifacts, we gain a base we can use to compare them.

To compute the cosine similarity(csim), we first suggest utilizing a N x M zero matrix, with N equaling the number of controls for the Preprocessor Artifact 1 and M equaling the same for Preprocessor Artifact 2.

$$\begin{bmatrix} csim(n_1,m_1) & \dots & csim(n_1,m_M) \\ \vdots & \ddots & \vdots \\ csim(n_N,m_1) & \dots & csim(n_N,m_M) \end{bmatrix}$$

The result is a similarity matrix, with higher values for a given cell meaning to a higher likelihood of a mapping between the denoted controls.

5.2.2 Mapping with neural networks. The authors have opted propose the use of the most recent developments in Machine Learning, as a method to improve the results deriving from the classical, NLP, methodology.

The chosen neural networks would have to be pre-trained and their goal would be to perform the "processing" parts of the mapping process, namely, the pair-wise comparisons between atomic controls.

The overall process would be similar to the one proposed with NLP, but instead of performing the cosine similarity comparisons with arrays, the output array would instead be filled out, cell by cell,

utilizing the output of the neural network with the two controls as inputs.

6 DEMONSTRATION

The Front-End is built through rendered HTML files, styled using CSS and made interactive through JS. The Front-End navigation is divided into two pages, one for interacting with the Extractor and Preprocessor and another for interacting with the Mapper.

6.1 Extractor & Preprocessor

Home Page - The Extractor & Preprocessor home page gives the user a dashboard that displays available documents, allows the upload of new ones and permits the creation and edit of each document's Preprocessor artifact. As an alternative to creating a new Preprocessor artifact from scratch, this step can be bypassed by uploading a CSV file.

Preprocessor Artifact Editing - The Extractor & Preprocessor Artifact Editing page allows the User to review, edit and delete the existing controls for the artifact selected from the Home Page. An example of the editing process can be seen in Figure 4

6.2 Main Server Block

The Main Server Block acts as an interface between the user's requests and the information stored in the MySQL database, and is composed of a web server, a templating engine, a database connector, and three data processing modules: the extractor, preprocessor, and mapper. A summary containing all of the required imports per module can be found in Table 6.

6.2.1 Web Server. The Web Server is the back-end for the Client, allowing him selected access to the Database. To build the Server, we choose to use the Flask Framework (Python 3.1) due to our familiarity with the software and its ease of use, efficient development process and expandability [18].

In our chosen implementation, the Web Server is the center block to the entire Main Server, it makes use of Python's `import` functionality to include the necessary core libraries and the developed modules: the Extractor, Preprocessor Mapper and Database Connector.

6.2.2 Templating Engine. Templating is a strategy that has gained a lot of popularity in the Web Development industry [12]. Templating makes use of a Templating Engine that takes as input a template file and, optionally, a set of variables. The output is a compiled version of the template with the variables replacing the placeholders pre-set into the template. We opted to use the Jinja2 Templating Engine since it is built with our Web Server framework, Flask, in mind and works natively right out of the box.

6.2.3 Database Connector. Connections with the MySQL server are managed through the Database Connector module. The connector provides not only the interface with which to connect to the database, but also an abstraction for all the queries required by the Webserver.

A .env (environment) file is read when the module is loaded, which provides the necessary environment variables to connect with the database, namely, the username, password, host name and database name. The .env file allows any user to configure which database to use, without requiring any knowledge of implementation or even programming expertise.

6.2.4 Extractor. The Extractor converts PDF contexts into a machine-readable format using the pdfplumber module [16]. The full text is merged into a string and passed to a function that utilizes Regular Expressions (RegEx) to extract Sections and Subsections.

Sections and Subsections are represented using an abstraction class called *StructuralElement*, with an identifier, title, and content. These elements are stored in a custom list type, *StructuralElementList*. The resulting list is written to a predefined output file and returned by the main function.

6.2.5 Preprocessor. The Preprocessor module takes the output of the Extractor as input, but can also be used with a StructuredElementsList from any source. Its main function is to generate a mapping of identifiers to a list of atomic controls by parsing the content of each StructuralElement. To identify controls within each section and subsection, we use a custom algorithm that utilizes NLTK [6] and regular expressions.

First, the algorithm loads each StructuralElement's content into NLTK's `sent_tokenize`, separating the sentences and non-atomic controls. These are stored in an intermediate mapping of identifier to list of non-atomic controls. The algorithm then parses each control to find enumerations, expanding them into atomic controls by applying their headers to each enumerated sentence. The final mapping from identifier to list of atomic controls is generated and returned as the Preprocessor Artifact.

6.2.6 Mapper. The Mapper's goal is to generate a Mapping Artifact out of the two inputted Preprocessor Artifacts. Mapping Artifacts are generated, through IR techniques, in a four step sequential process composed by Tokenization, Normalization, TF-IDF and Comparison. We developed this module with a focus on English, although it could be utilized in other languages with minor configuration changes. As such, we follow the standard English language models and stopwords conventions.

Tokenization - In order to generate a list of tokens for every control, we iterated over the list of controls and applied NLTK's `word_tokenize` algorithm [6].

Normalization - The first normalization step we took was to remove the English stopwords from every token list. This step has been shown to improve the accuracy of IR techniques, such as sentiment analysis [11], which uses TF-IDF, the same algorithm we use.

Secondly, we piped the token lists to a function that modifies the corpus of the artifact. This function applies a Stemming or Lemmatization algorithm, or none, depending on the user supplied configuration. We have set lemmatization as the default behaviour for the system, since the impact on Precision and Recall is shown to be relatively small [37], but lemmatization reduces the corpus size considerably, leading to lowered computational times in future steps.

Neural Network Comparisons - The first implementation choice the authors have made, was the decision of which of the many sentence transformers to use. We have selected two, which

have been trained with two distinct sets of documents: bert-base-nli-mean-tokens [33] and legal-bert-base-uncased [7].

We have opted to experiment with a general case sentence transformer, BERT, since it should work for any given input with reasonable accuracy. This model should provide a baseline of what is possible to achieve.

Legal BERT, as the name would suggest, was trained under a large collection of legal documents. We assess that, due to the proximity of the compliance and legal fields, there is a fair possibility that comparisons made with a model specialized for law be as good, or better than those made with a generalized model.

7 EVALUATION

We have opted to evaluate this tool by comparing the mapping tables produced by the tool to those present in the literature. In such a model, we would be utilizing the mappings developed manually by humans as the "golden standard". For our approach to be viable, the mappings with which to compare should be the result of peer-reviewed studies, or at least, have some form of specialist input.

7.1 Evaluation methodology

We evaluated the tool utilizing two mappings: ISO 27001:2005 with ISO 20000:2005 and ISO 27001:2022 with ISO 22301:2019. Both mappings follow the same format: a mapping table, with each cell being a unique pair of sections relating to two distinct compliance mechanisms.

The mapping table generated for the standards ISO 27001 (2005) and ISO 20000 is compared against the one created manually by Pardo et al. [26]. This mapping table is a partial mapping in which each row or column has at least a non-zero value.

In regards to the mapping table generated for the standards ISO 27001 (2022) and ISO 22301, we have taken a different approach. We have manually mapped between these two standards. These results were then validated with a survey that we handed out to specialists. The survey gathered 10 responses.

We have ran both artifacts through ten variations of the testing suite, each one presents a unique combination of the weights given to the values generated by the AI model (AI %) and which model was used. A summary of the testing results is present in Tables 2 and 3. The best results were highlighted in **bold**.

Having presented the input artifacts for the testing, it is also important to present the metrics we utilize to evaluate how well the tool performs. We have opted to select the F1-score, given by the harmonic mean of Precision and Recall, as well as the average deviation from the test data, given by the formula $\frac{1}{I*J} \sum \sum_{i \in I, j \in J} |generated(i, j) - test(i, j)|$, assuming that I and J are the number of rows and columns, respectively, `generated` is the value generated by the tool and `test` is the value pulled from the artifact.

7.2 Evaluation results

The authors observed the testing data, and highlight that the accuracy varies greatly across the two different mapping artifacts. The reasoning is two-fold. Firstly, we must take into consideration that the tool is very accurate at predicting where no mapping is present, with an accuracy of 98.5% if we consider only the best model predicting a rank of 0 or not 0.

Secondly, ISO 27001_2005 - ISO 20000_2005, being only a partial mapping, containing a matrix where each cell's row or column has at least one mapping ranked at 1 or above, whereas the more recent mapping is a complete one. This leads to much less chances for the tool to rank a given cell at 0, leading to lower performance in general.

Model	AI %	F1	Delta
bert-base-nli-mean-tokens	0	**0,52**	**1,1**
bert-base-nli-mean-tokens	25	0,48	1,16
bert-base-nli-mean-tokens	50	0,41	1,32
bert-base-nli-mean-tokens	75	0,41	1,34
bert-base-nli-mean-tokens	100	0,37	1,4
nlpaueb/legal-bert-base-uncased	0	**0,52**	**1,1**
nlpaueb/legal-bert-base-uncased	25	0,48	1,16
nlpaueb/legal-bert-base-uncased	50	0,41	1,32
nlpaueb/legal-bert-base-uncased	75	0,41	1,34
nlpaueb/legal-bert-base-uncased	100	0,38	1,4

Table 2. Testing results for ISO 27001_2005 - ISO 20000_2005

Model	AI %	F1	Delta
bert-base-nli-mean-tokens	0	**0,96**	**0,05**
bert-base-nli-mean-tokens	25	0,95	0,05
bert-base-nli-mean-tokens	50	0,94	0,07
bert-base-nli-mean-tokens	75	0,94	0,09
bert-base-nli-mean-tokens	100	0,93	0,12
nlpaueb/legal-bert-base-uncased	0	**0,96**	**0,05**
nlpaueb/legal-bert-base-uncased	25	**0,96**	**0,05**
nlpaueb/legal-bert-base-uncased	50	0,95	0,07
nlpaueb/legal-bert-base-uncased	75	0,93	0,1
nlpaueb/legal-bert-base-uncased	100	0,93	0,12

Table 3. Testing results for ISO 27001_2022-ISO 22301_2019

8 CONCLUSION

Implementing compliance mechanisms, such as standards, policy, or legislation, can offer organizations many benefits. However, implementing multiple mechanisms can be problematic, as blindly implementing them one after another often leads to duplication of work, increased costs, confusion and auditing problems. One solution to these issues is through a mapping study, where two compliance mechanisms are compared to find intersections between them. This process may or may not lead to their integration, generating a single model out of the two (or more) mapped mechanisms. However, mapping can also prove to be time consuming, which can be problematic if it is used to reduce wasted time in the implementation of multiple compliance mechanisms. In this paper, we propose a software tool that automates parts of this process, splitting it into two parts: an extractor and a mapper.

The tool is designed to extract atomic controls from a compliance mechanism's PDF document or CSV file and generate a dictionary indexed by their section, subsection and subsubsection. The tool then uses classical NLP and semantic comparison provided by AI models for sentence similarity to compare two dictionaries and generate a weighted value

We have evaluated our tool by comparing it against two manually made mappings. The evaluation has shown that the best results were achieved using either no AI support or a low setting of the legal model. In both cases, the tool has shown high accuracy (over 95%) when attempting to guess whether or not a given pair is ranked 0 or not 0, which is a great starting point for a review process of the tool's outputs.

We note that the work presented in this paper is not without its limitations. We have tested our tool only with two instances of mappings between international standards. Moreover, we have tested our tool using only embeddings generated by two AI models, both of which derive from BERT [7, 10, 31].

For Future Work, we propose that the testing could be expanded to other types of compliance mechanisms, so that the tool's efficacy can be studied when applied to them. Interesting examples could include organizational policy, other standards like the COBIT Framework or legislation (such as the GDPR). We also suggest that other AI models could be studied, preferably utilizing a different basis for their training, possibly utilizing novel techniques such as Word Mover's Distances (WMDs) [22].

A APPENDIX A - ADDITIONAL TABLES AND FIGURES

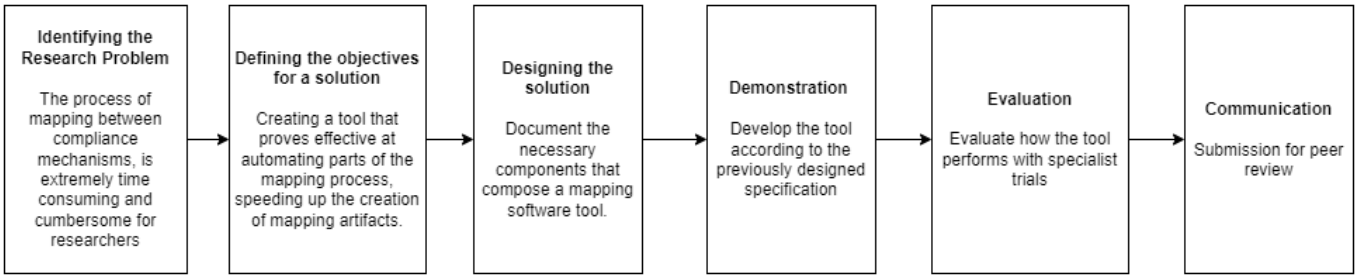

Fig. 3. Summary of the DSRM guidelines applied to our study

Term	Description
Information Retrieval IR	The process of analyzing text and identifying mentions of semantically defined entities and relationships within it [13]
Document	Unstructured text input
Lemmatization	Lemmatization removes inflectional and attempts to return the dictionary form of a given token [5].
Stemming	Stemming is a procedure to reduce all words with the same stem to a common form [5]
Inverted index	Mapping of each token to which documents contain it [19]
Token	Word within a vocabulary
Vocabulary	Set of unique tokens
Corpus	Set of documents
n-gram	Expression containing n tokens
TF	Term Frequency
IDF	Inverse Document Frequency

Table 4. Common NLP terms and their definitions

Original control	Atomic controls
The organization shall: a) identify the risks of disruption to the organization's prioritized activities and to their required resources; b) analyse and evaluate the identified risks; c) determine which risks require treatment	The organization shall identify the risks of disruption to the organization's prioritized activities and to their required resources;
	The organization shall evaluate the identified risks
	The organization shall analyse the identified risks
	The organization shall determine which risks require treatment

Table 5. Example of the conversion of a control into its composing atomic controls

	Extractor	Preprocessor	Mapper	Database Connector	Web Server	Templating Engine	Libraries
collections			x		x		
dotenv [36]				x			
functools					x		
flask [23]					x		
ginja2 [24]						x	
math			x				
mysql [39]				x			
nltk [6]		x	x				
numpy [14]			x		x		
os	x	x			x		
pandas [25]			x		x		
pathlib	x						
pdfplumber [16]	x						
pickle	x	x					
re	x	x			x		x
sklearn [29]			x				
statistics					x		
string			x				
traceback					x		
sys	x	x					
uuid				x			
werkzeug					x		

Table 6. Module usage per Main Server Block module

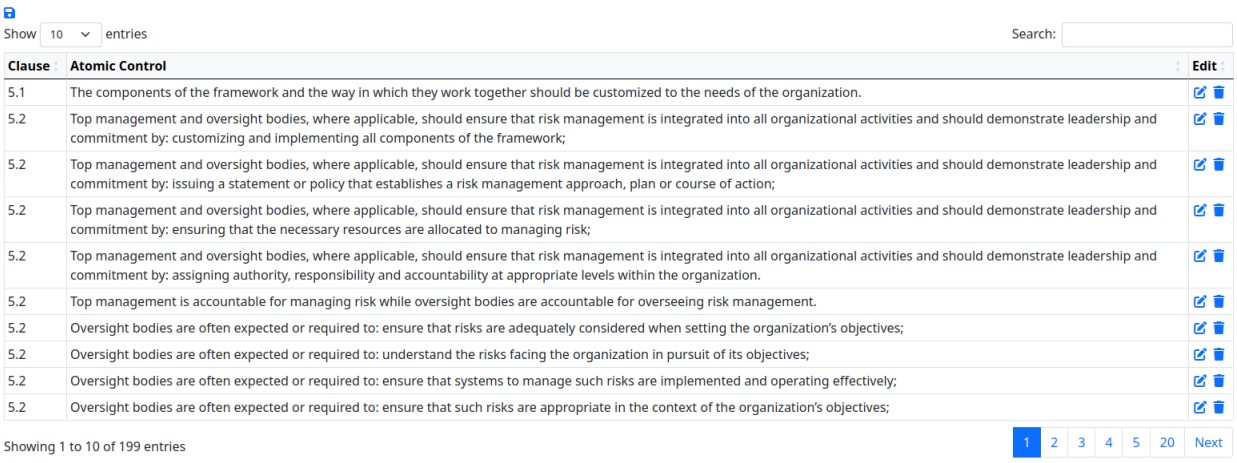

Fig. 4. Editing an existing Preprocessor Artifact

REFERENCES

[1] Palakorn Achananuparp, Xiaohua Hu, and Xiajiong Shen. 2008. The evaluation of sentence similarity measures. In *Data Warehousing and Knowledge Discovery: 10th International Conference, DaWaK 2008, Proceedings 10*. Springer, 305–316.

[2] Akiko Aizawa. 2003. An information-theoretic perspective of tf–idf measures. *Information Processing & Management* 39, 1 (2003), 45–65.

[3] Rafael Almeida, Renato Lourinho, Miguel Mira da Silva, and Rúben Pereira. 2018. A model for assessing COBIT 5 and ISO 27001 simultaneously. In *2018 IEEE 20th Conference on Business Informatics (CBI)*, Vol. 1. IEEE, 60–69.

[4] Issa Atoum, Ahmed Otoom, and Narayanan Kulathuramaiyer. 2016. A comprehensive comparative study of word and sentence similarity measures. *arXiv* (2016).

[5] Vimala Balakrishnan and Ethel Lloyd-Yemoh. 2014. Stemming and lemmatization: A comparison of retrieval performances. (2014).

[6] Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.".

[7] Ilias Chalkidis, Manos Fergadiotis, Prodromos Malakasiotis, Nikolaos Aletras, and Ion Androutsopoulos. 2020. LEGAL-BERT: The Muppets straight out of Law School. In *Findings of the Association for Computational Linguistics: EMNLP 2020*. Association for Computational Linguistics, Online, 2898–2904.

[8] Balakrishnan Chandrasekaran, John R Josephson, and V Richard Benjamins. 1999. What are ontologies, and why do we need them? *IEEE Intelligent Systems and their applications* 14, 1 (1999), 20–26.

[9] KR1442 Chowdhary and KR Chowdhary. 2020. Natural language processing. *Fundamentals of artificial intelligence* (2020), 603–649.

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv* (2018).

[11] Kranti Vithal Ghag and Ketan Shah. 2015. Comparative analysis of effect of stopwords removal on sentiment classification. In *2015 international conference on computer, communication and control (IC4)*. IEEE, 1–6.

[12] David Gibson, Kunal Punera, and Andrew Tomkins. 2005. The volume and evolution of web page templates. In *Special interest tracks and posters of the 14th international conference on World Wide Web*. 830–839.

[13] Ralph Grishman. 2015. Information extraction. *IEEE Intelligent Systems* 30, 5 (2015), 8–15.

[14] Charles R. Harris, K. Jarrod Millman, Ralf Gommers Stéfan J. van der Walt, and Pauli Virtanen et. al. 2020. Array programming with NumPy. *Nature* 585, 7825 (Sept. 2020), 357–362.

[15] Alan Hevner, Samir Chatterjee, Alan Hevner, and Samir Chatterjee. 2010. Design science research in information systems. *Design research in information systems: theory and practice* (2010), 9–22.

[16] jsvine. 2022. pdfplumber. https://github.com/jsvine/pdfplumber.

[17] Klaus Julisch. 2008. Security compliance: the next frontier in security research. In *Proceedings of the 2008 New Security Paradigms Workshop*. 71–74.

[18] PS Lokhande, Fankar Aslam, Nabeel Hawa, Jumal Munir, and Murade Gulamgaus. 2015. Efficient way of web development using python and flask. (2015).

[19] Ajit Kumar Mahapatra and Sitanath Biswas. 2011. Inverted indexes: Types and techniques. *International Journal of Computer Science Issues (IJCSI)* 8, 4 (2011), 384.

[20] Olivier Mangin, Béatrix Barafort, Patrick Heymans, and Eric Dubois. 2012. Designing a process reference model for information security management systems. In *Software Process Improvement and Capability Determination: 12th International Conference, SPICE 2012. Proceedings 12*. Springer, 129–140.

[21] Antonia Mas, Antoni Lluís Mesquida, Esperança Amengual, and Bartomeu Fluxà. 2010. ISO/IEC 15504 best practices to facilitate ISO/IEC 27000 implementation. In *International Conference on Evaluation of Novel Approaches to Software Engineering*, Vol. 2. SciTePress, 192–198.

[22] Vít Novotnỳ, Eniafe Festus Ayetiran, Michal Štefánik, and Petr Sojka. 2020. Text classification with word embedding regularization and soft similarity measure. *arXiv* (2020).

[23] pallets. 2022. flask. https://github.com/pallets/flask.

[24] pallets. 2022. jinja. https://github.com/pallets/jinja.

[25] The pandas development team. 2020. *pandas-dev/pandas: Pandas.*

[26] César Pardo, Francisco J Pino, and Félix Garcia. 2016. Towards an integrated management system (IMS), harmonizing the ISO/IEC 27001 and ISO/IEC 20000-2 standards. *International Journal of Software Engineering and Its Applications* 10, 9 (2016), 217–230.

[27] César Pardo, Francisco J Pino, Félix García, Mario Piattini, and Maria Teresa Baldassarre. 2012. An ontology for the harmonization of multiple standards and models. *Computer Standards & Interfaces* 34, 1 (2012), 48–59.

[28] César Pardo, Francisco J Pino, Félix García, Mario Piattini Velthius, and Maria Teresa Baldassarre. 2011. Trends in harmonization of multiple reference models. In *Evaluation of Novel Approaches to Software Engineering: 5th International Conference, ENASE 2010, Revised Selected Papers 5*. Springer, 61–73.

[29] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[30] Simona Ramanauskaitė, Dmitrij Olifer, Nikolaj Goranin, and Antanas Čenys. 2013. Security ontology for adaptive mapping of security standards. *International Journal of Computers, Communications & Control (IJCCC)* 8, 6 (2013), 813–825.

[31] Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv* (2019).

[32] Gerard Salton and Christopher Buckley. 1988. Term-weighting approaches in automatic text retrieval. *Information processing & management* 24, 5 (1988), 513–523.

[33] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *ArXiv* abs/1910.01108 (2019).

[34] Razieh Sheikhpour and Nasser Modiri. 2012. An approach to map COBIT processes to ISO/IEC 27001 information security management controls. *International Journal of Security and Its Applications* 6, 2 (2012), 13–28.

[35] Pramila P Shinde and Seema Shah. 2018. A review of machine learning and deep learning applications. In *2018 Fourth international conference on computing communication control and automation (ICCUBEA)*. IEEE, 1–6.

[36] theskumar. 2022. python-dotenv. https://github.com/theskumar/python-dotenv.

[37] Michal Toman, Roman Tesar, and Karel Jezek. 2006. Influence of word normalization on text classification. *Proceedings of InSciT* 4 (2006), 354–358.

[38] Basie Von Solms. 2005. Information Security governance: COBIT or ISO 17799 or both? *Computers & Security* 24, 2 (2005), 99–104.

[39] Michael Widenius and David Axmark. 2002. *MySQL reference manual: documentation from the source.* " O'Reilly Media, Inc.".