Building Blocks in Standards: Improving Consistency in Standardization with Ontology and Reasoning

Marcello Balduccini[1] and Claire Vishik[2]

[1] Saint Joseph's University
Philadelphia, PA, USA
Email: marcello.balduccini@sju.edu

[2] Claire Vishik
Intel Corporation
Austin, TX, USA
Email: claire.vishik@intel.com

Abstract. International standardization in ICT has grown in importance due to the rapid technology development, the increasing need for interoperability, and the global nature of the digital infrastructure. However, technical resources available to international standards bodies have become more limited. With its focus on international collaboration and consensus, the standardization community has not invested significantly in the automation of the process of developing standards. In this paper, we describe potential gains in efficiency with an ontology-based approach and automated reasoning. As part of the exploratory phase of the project, we built a prototype ontology and evaluated the benefits of automated reasoning to improve the process of developing and harmonizing broadly understood ICT assessment standards. The exploratory phase confirmed feasibility and clarified the benefits of the ontology-based approach to standardization, but also highlighted difficulties and unsolved problems in this area.

– **Keywords:** Assessment standards, ontology, knowledge engineering, automated reasoning.

1 Introduction

In the past two decades, the importance of open international standards in ICT (Information and computer technologies) increased significantly, while the resources available to develop new standards became scarce (see, e.g., (Boje, 2015)). The dynamic cycle of technology development, the massive need for integration, where previously-independent technology domains have to be connected, and the global nature of the digital infrastructure combined to elevate the need for international standards. At the same time, the availability of industry experts decreased as companies shifted their focus to core technologies and products. While governments, universities, and consultants stepped in, the standardization community felt the strain as the need to develop and ratify new standards became more acute.

As a result, it often takes a long time to develop new standards and update the existing ones, and the pace of standardization has been out of step with the technology needs for some time. Additionally, as the body of available standards continued to grow and the diversification of the ICT space intensified, it has become more difficult to ensure consistency of approaches used in similar standards. Standards bodies, such as ISO, have taken a number of steps to improve the level of harmonization, enforcing unified formats, rules for references, and consistent terminology (see, e.g., ISO 704 "Terminology: Principles and Methods"). But creating a formal process to re-use building blocks used in standards beyond terminology and some other components has proved challenging.

In this paper, we propose an automated approach based on ontologies and reasoning that could pave the way for the re-use of standard building blocks in standards. In order to test our methodology, we have defined a set of building blocks in a sample of broadly understood assessment standards and created a prototype that demonstrates the feasibility of this approach.

While we don't advocate anything approaching automated standard generation, we consider mechanisms to improve consistency of the body of standards and speed up standards development necessary for the standardization community to keep pace with the needs of the technology and society. The open consensus driven process for developing international standards is one of the greatest achievements that made possible the development and deployment of the global digital infrastructure. The efficiency of this process will be enhanced by the ability to automate repetitive tasks and to achieve greater harmonization among diverse requirements described in standards.

The nature of standardization, a process based on collaboration and consensus, will always require a significant amount of time, we believe that the efficiency and consistency can be increased by employing knowledge engineering techniques. In addition to speeding up the development of consistent standards with non-conflicting requirements, the proposed approach can also lead to the development of more focused and context driven requirements, especially in the area of ICT assessment that is the purpose of the study described in this paper.

2 ICT Security and Privacy Assessment Standards

Over the past decades, a number of standards related to various types of assessments were developed in various standards bodies. For the purpose of this work, we describe assessment standards and those that can facilitate the evaluation of products, technologies and ecosystems. The assessment standards comprise a variety of approaches, including governance, secure development lifecycles, risk management, deterministic product testing, and many other. Examples of complementary approaches include the following:

1. ISO/IEC 15408. Evaluation criteria for IT security

2. ISO/IEC 17825Testing methods for the mitigation of non-invasive attack classes against cryptographic modules
3. ISO/IEC 18045. Methodology for IT security evaluation.
4. ISO/IEC TS 19249:2017 Catalogue of architectural and design principles for secure products, systems and application
5. ISO/IEC 19790:2012. Security requirements for cryptographic modules
6. ISO/IEC 19792:2009. Security evaluation of biometrics
7. Most standards from the 27000 series, e, g., ISO/IEC 27034
8. ISO/IEC 29134:2017. Guidelines for privacy impact assessment
9. ISO/IEC 29190:2015. Privacy capability assessment model

These standards are represented by highly structured documents that comprise similar building blocks, such as principles, requirements, or components of processes. Some of these building blocks are similar, but not identical in these standards, and others are very specific to the context that is the subject of a standard. In the course of the first stage of our project, we evaluated several standards documents, with the objective to:

1. Determine the extent of structural and semantic similarities in different assessment related standards.
2. Develop a prototype ontology to capture relationships among these building blocks.
3. Obtain the insights from this experiment.
4. Define the advantages of the approach with regards to the efficiency and flexibility of the standardization process that an ontology could provide.

For this project, we focused on structural building blocks, e.g., principles or processes. In other research projects focusing on defining ontologies for security related standards (see, e.g., de Franco Rosa 2018 1/2), building blocks were defined based on semantics, e.g., risk or vulnerability. Ideally, both structural and semantic building blocks need to be defined, but semantics could also be presented differently in the model, as demonstrated in this work.

3 Ontologies and their Use in Standardization and Related Areas

In order to evaluate the feasibility of structural and semantic analysis of ICT assessment standards based on ontology and other knowledge engineering methods, we need to start with the definitions.

An *ontology* is a formal, logic-based representation of knowledge typically aimed at supporting reasoning by means of logical inference. Broadly speaking, an ontology is a collection of statements in a logical language representing a given domain in terms of classes (i.e., sets) and subclasses of objects, individuals (i.e., objects of a specific class), and relationships between objects and/or classes. For instance, an ontology of standards produced by ISO/IEC JTC1 SC27 (IT Security Techniques)[1] might contain a $Document$ class capturing the set of all standard documents, and a subclass 27000_series. ISO_IEC_27032 would then be an individual of class 27000_series. To describe the title of a standard document, one might introduce a relationship $hasTitle(doc, str)$ where doc is a document and str is a string, e.g. $hasTitle(ISO\_IEC\_27032, \text{"}Guidelines\ for\ cybersecurity\text{"})$. Additionally, one could specify that $ISO\_IEC\_27034\_part\_1$ and $ISO\_IEC\_27034\_part\_2$ are part of ISO_IEC_27034 by means of a suitable relationship, e.g.:

$$partOf(ISO\_IEC\_27034\_part\_1, ISO\_IEC\_27034),$$
$$partOf(ISO\_IEC\_27034\_part\_2, ISO\_IEC\_27034).$$

Ontologies are utilized in a wide range of areas, from media to robotics, and from engineering to medicine. Researchers working in the areas of standardization have applied ontologies for a variety of purposes, but typically to create a semantic meta model of a complex space. Huang and Li (2010) used ontologies to link standards tags relating properties of the IoT space to the descriptions of the functions they denote. In the medical field, Cai et al (2014) used an ontology to standardize and classify adverse drug reactions based on Adverse Drug Reaction Classification System. Ramanauskaitė et al. (2013) described how ontologies could be used to map existing security standards, and Fenz & Ekelhart (2009) developed ontologies to formalize security knowledge and make it more amenable to various analyses. Gonzalez-Perez et al. (2016) developed an ontology for ISO software engineering standards, complete with a prototype demonstrating their approach.

Outside of the research literature, several standardization efforts have explored or used ontologies to make sense of complex subject areas, such as IoT (Internet of Things) or Smart Cities. Ontology-based standards exist, mostly to provide a model for complex interactions, such as ISO/TS 19150 ("Geographic Information—Ontology"[2]). However, the use of ontologies for harmonization and process efficiency that is common in, e.g., requirements engineering, has not been adapted to standardization, with the exception of some experimental studies. In the paper, we hope to demonstrate that some level of automation is feasible in the field of standardization as well and that it is compatible with the currently used standards development processes.

4 Ontology-Based Representation of Standard Documents

The analysis of various standard documents from the ISO/IEC 19000 series and 27000 series showed that documents can be viewed as consisting of a hierarchically-organized collection of *building blocks*, each describing a key component of the document. A relatively small number of structural building blocks is shared by all documents analyzed:

[1] https://www.iso.org/committee/45306.html, ISO/IEC JTC 1/SC 27 IT Security techniques.
[2] https://www.iso.org/standard/57465.html

- Concepts – concepts used throughout the document.
- Definitions – definitions of notions.
- Guidelines –guidelines pertaining to the standard.
- Principles – guiding principles used in the document.
- Process – a process (or task) being standardized.
- Purpose – purpose of the document or of a part of the document.
- Test – one or more tests being standardized, possibly used as part of a process.
- Misc – general-purpose block

The types of building blocks are formalized in the prototype ontology as subclasses of class “Building Block”, as shown in **Fig. 1**.

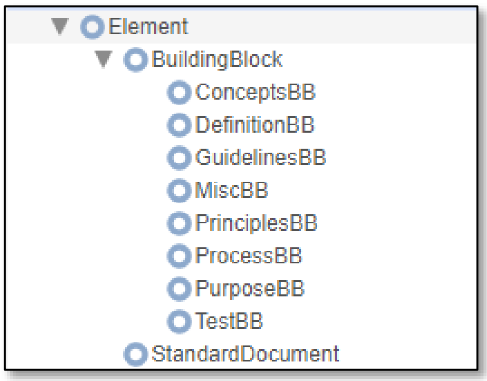


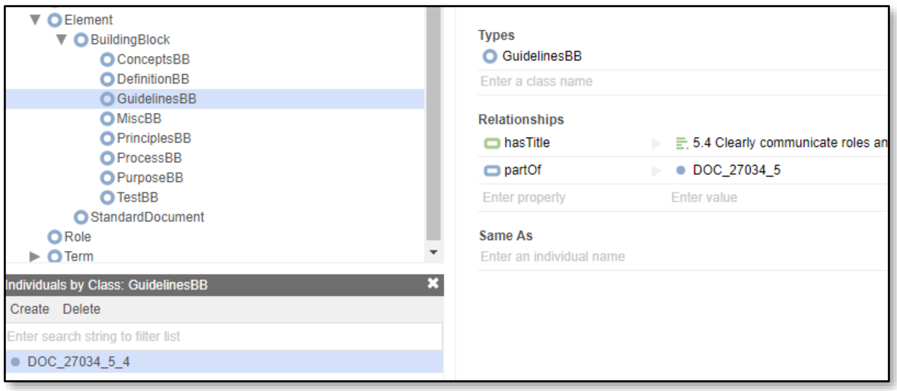
Fig. 1. Ontology fragment: classes for building blocks and standard document

Note that the specific selection of types of building blocks listed above is intended only for demonstration of the proposed formalization and not as a definitive list and that it is limited to structural building blocks. Analysis of further documents may well result in additional types of building blocks or in a reorganization of their hierarchy. Although the prototype ontology was built manually, the structured nature of standards provides an option to build it semi-automatically, by harvesting certain types of structures, clustering terms by frequency, and other methods developed by the text processing and knowledge engineering communities.

An occurrence of a building block in a document is captured by an individual of a suitable class. When available, a string representing the title is associated with a building block by means of relation $hasTitle$. Similarly, relation $hasBody$ specifies the main narrative of the building block. The concepts (see below) identified in the title are associated with the building block by relation $mainTopic$.

The hierarchy of building blocks that make up a document is captured by relation $partOf$, so that $x\ partOf\ y$ holds if building block y is a component of x, where x is an individual of class Element, i.e. a building block or a standard document (see **Fig. 1**).

For instance, a clause defining guidelines related to communicating roles and responsibilities might be captured in the ontology by an individual of class GuidelinesBB, which is linked to its parent clause by relation $partOf$. **Fig. 2** provides an illustration of such an individual.



DOC_27034_5_4

Types: GuidelinesBB

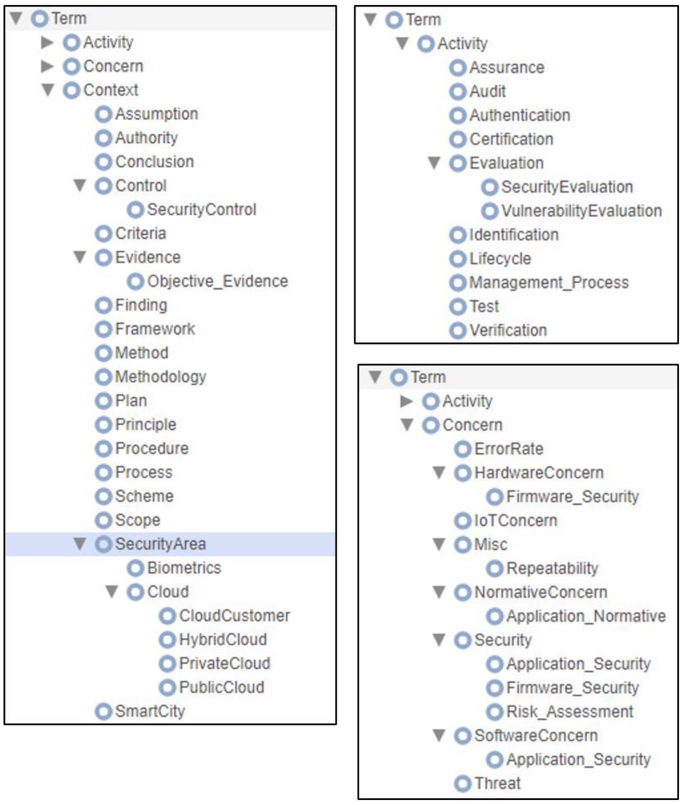
hasTitle	5.4 Clearly communicate roles an
partOf	DOC_27034_5

Fig. 2. Individual for clause 5.4 of ISO/IEC 27034

Next, we identified reoccurring relevant terms, such as “assurance,” “authority,” and “framework.” We noted that these terms are often used to form combinations, which appear to have importance in defining the content of a building block. The terms typically have specific functions within a combination, and we have identified three main categories of these: “Activity,” “Concern,” and “Context”. To formalize these combinations, we introduced the Concept class with associated relations $includesActivity$, $includesConcern$, and $includesContext$. Each relation informally states that a given Concept individual includes a term with the indicated function. The occurrence of multiple terms with the same function is formalized by multiple occurrences of a suitable relation.

The terms are currently organized hierarchically along three categories matching the categories of functions identified in the combinations.[3] For the sake of illustration, we created a notional set of frequent terms. The three branches of the term hierarchy are shown in **Fig. 3**.

[3] While this approach is convenient for illustration purposes, preliminary examination of a broader set of documents seems to suggest that the same term may be used for multiple functions. In that case, it may be more appropriate to organize the term hierarchy along other dimensions than the functions themselves.



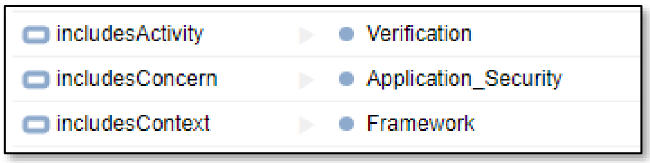
- Term
  - Activity
  - Concern
  - Context
    - Assumption
    - Authority
    - Conclusion
    - Control
      - SecurityControl
    - Criteria
    - Evidence
      - Objective_Evidence
    - Finding
    - Framework
    - Method
    - Methodology
    - Plan
    - Principle
    - Procedure
    - Process
    - Scheme
    - Scope
    - SecurityArea
      - Biometrics
      - Cloud
        - CloudCustomer
        - HybridCloud
        - PrivateCloud
        - PublicCloud
    - SmartCity

- Term
  - Activity
    - Assurance
    - Audit
    - Authentication
    - Certification
    - Evaluation
      - SecurityEvaluation
      - VulnerabilityEvaluation
    - Identification
    - Lifecycle
    - Management_Process
    - Test
    - Verification

- Term
  - Activity
  - Concern
    - ErrorRate
    - HardwareConcern
      - Firmware_Security
    - IoTConcern
    - Misc
      - Repeatability
    - NormativeConcern
      - Application_Normative
    - Security
      - Application_Security
      - Firmware_Security
      - Risk_Assessment
    - SoftwareConcern
      - Application_Security
    - Threat

Fig. 3. Ontology fragment: term hierarchy

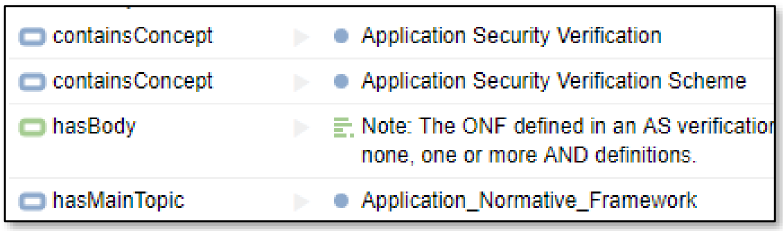
Fig. 4 illustrates the definition of a sample individual for the concept “application-security verification framework”.



includesActivity	Verification
includesConcern	Application_Security
includesContext	Framework

Fig. 4. Concept individual for “application-security verification framework”

In order to indicate that the narrative of a certain building block contains a concept, we introduce relation $containsConcept$, where $x\ containsConcept\ y$ means that building block x contains an occurrence of concept y. For instance, a clause might contain occurrences of concepts “Application-Security Verification” and “Application-Security Verification Scheme”. Hence, the definition of the corresponding building block would include the items shown in **Fig. 5**.



containsConcept	Application Security Verification
containsConcept	Application Security Verification Scheme
hasBody	Note: The ONF defined in an AS verificatio none, one or more AND definitions.
hasMainTopic	Application_Normative_Framework

Fig. 5. Concepts occurring in clause 10.3.19 of ISO/IEC 27034

In a similar way, concepts that occur in the title of a building block are captured by relation $hasMainTopic$.

The formalization of building blocks in assessment-related standards offers a foundation for several efficiency-related opportunities that will be the subjects of future work. Some of them are listed below:

1. Harmonize building blocks from different standards that are meant to denote the same context.
2. Pre-populate new or updated standards with typical building blocks that already exist.
3. Update building blocks across several standards when an error or an inconsistency are found or when the technology environment changes.
4. Build context-driven and focused assessments that combined components of several areas, such as privacy, SDL (Secure development lifecycle), safety, and governance.
5. Ensure greater harmonization across diverse standards and detect potential conflicts between assessment requirements.

Having built a prototype ontology in order to extract the recurring components in assessment standards, we can now evaluate the advantages provided by automated reasoning capabilities.

5 Inference and Reasoning Capabilities

An ontology can be used to achieve sophisticated inference and querying capabilities, and this is useful for the document analysis, such as the analysis of standards.[4] The SPARQL query language (Harris, Seaborne, & Prud'hommeaux, 2013) allows for efficient extraction from an ontology of data satisfying certain criteria. The basic syntax of a SPARQL query is:

[4] In fact, a whole hierarchy of ontology-based languages exist, with varying degrees of expressivity and computational complexity. A thorough discussion is beyond the scope of this paper.

```
$SELECT\ ?\,var_1\ ?\,var_2\ ?\,var_3\ \ldots$
$WHERE$
$\{\ ?\,var_x\ rel_A\ ?\,var_y\ .$
  $?\,var_y\ rel_B\ ?\,var_z\ .$
  $\ldots$
$\}$
```

where each variable $?\,var\_i$ is a placeholder for an ontology item of interest and every rel_i is the name of a relationship. Note the syntax used for relations in the $WHERE$ block, where $?\,var_x\ rel_A\ ?\,var_y$ stands for the more traditional mathematical logic atom $rel_A(?\,var_x\ ,?\,var_y)$. Typically, the variables from the $SELECT$ line also occur in the $WHERE$ block, but additional variables may appear in the block. When a tuple of items is found such that the $WHERE$ block is satisfied, the tuple of items corresponding to the variables from the $SELECT$ line is considered to be an answer to the query. For instance, consider the query:

```
$SELECT\ ?\,part\ ?\,doc\_class$
$WHERE$
$\{\ ?\,part\ partOf\ ?\,doc\ .$
  $?\,doc\ type\ ?\,doc\_class$
 $\}$
```

where $type$ is a built-in relationship stating that the left-hand side is an individual of the class mentioned on the right-hand side. Intuitively, the query identifies all documents that are part of another document ($?\,part\ partOf\ ?\,doc$), finds the class of the containing document ($?\,doc\ type\ ?\,doc\_class$) and yields all pairs consisting of a document part and of the class of the containing document. When evaluated against the sample ontology given earlier, the query will yield the answers:

$$(ISO\_IEC\_27034\_part\_1, 27000\_series)$$
$$(ISO\_IEC\_27034\_part\_2, 27000\_series)$$

For increased flexibility of representation, we make also use of an extension of propositional logic called *Answer Set Programming (ASP)* (Gelfond & Lifschitz, 1991) (Marek & Truszczynski, 1999). ASP is a rule-based language, where a rule is a logical formula of the form:

$$l_0 \leftarrow\ l_1, l_2, \ldots, l_m, not\ l_{m+1}, \ldots, not\ l_n.$$

Every l_i is a *literal*, i.e. an atom of the form $rel(c_1, c_2, \ldots, c_l)$ – where each c_i is a numerical or string constant – or an expression $\neg a$ for some atom a. Intuitively, the above rule states that, if all of $l_1, \ldots, l_m$ hold and *there is no reason to believe* (conveyed by the not keyword) that any of $l_{m+1}, \ldots, l_n$ hold, then l_0 must hold.

A *program* is a set of rules. When a literal l can be derived from a program Π, we write:

$$\Pi \vDash l$$

A formal description of the semantics of ASP can be found in (Gelfond & Lifschitz, 1991). For use in this paper, we will somewhat simplistically define an *answer set* of a program Π as one of the sets of literals that can be derived from Π.[5]

Following common representational practices, we allow for logical variables to be used as arguments of literals. A logical variable is a string denoted by an uppercase initial (unless enclosed by double-quotes). For instance, $partOf(Part, 27000\_series)$ is a literal whose first argument is the logical variable $Part$ and whose second argument is the string constant 27000_series. A rule containing variables is viewed as an abbreviation for all rules that can be obtained by replacing all variables by all possible constant symbols. For example, suppose one would like to identify all documents that are not subdivided further in document parts. This can be accomplished by the rules:

$$hasParts(Doc) \leftarrow partOf(Part, Doc).$$
$$monolithic(Doc) \leftarrow not\ hasParts(Doc).$$

Assuming that one is given information about document parts in the form of $partOf$ atoms, then the first rule will yield the atom $hasParts(d)$ for every document d that is subdivided. The second rule will yield an atom $monolithic(d)$ for every document d for which there is no reason to believe that $hasParts(d)$ holds. So, given information about ISO_IEC_27034 and ISO_IEC_27032 as in the prototype ontology, the rules will yield $hasParts($"ISO_IEC_27034"$)$ and $monolithic("ISO\_IEC\_27036")$.

5.1 Towards Automation of Information Extraction

A number of approaches have been developed to simplify the automated generation of ontologies or to support the semi-automatic generation of an ontology and a related knowledge base. They are typically based on the frequency of occurring terms, their positioning or proximity to other terms. Even with the automated generation, the maintenance of automated ontological models presents a problem in areas where ontologies are commonly used, such as Geographic Information Systems (GIS) or biomedical databases. Fortunately, advances have been made helping automate the creation or maintain the maintenance of single or linked ontologies. Innovative approaches are described in Alobaidi et al. (2018) for biomedical databases or in Fraga & Vegetti (2017) for manufacturing and product data. In pursuing the automation of the ontology creation process, we are helped by the fact that standards-related documents are highly structured, due to the requirements of standards bodies and traditions of standards development. As a result, the automation process is considerably simplified, although it requires a rigorous high-level model to make it meaningful.

At this stage of our work, we do not commit to a specific way of obtaining the information captured by the ontology. At some point, tools may be built for standard document creation that automatically create the ontology as a by-product. On the other hand,

[5] Certain programs have multiple answer sets. For example, $\{\ p \leftarrow not\ q.\ \ q \leftarrow not\ p.\}$ has two answer sets, $\{\ p\ \}$ and $\{\ q\ \}$, corresponding to two alternative, equally possible views of the world captured by Π..

one might develop dedicated information extraction techniques for processing existing documents. For our study, we have developed a basic but effective technique that enables extraction of information from existing documents. We describe it as it may lay the foundations of more sophisticated approaches.

The algorithm we devised takes in input an *input ontology* representing concepts and building block types (see **Sec. 4**), and a set of files containing clauses of a standards document. The algorithm produces an *output ontology* that can be used for the reasoning processes described in this paper. The output ontologies for multiple standards documents can be easily merged by applying traditional ontology-manipulation tools. To facilitate the task of the algorithm, each concept in the input ontology is augmented with a list of synonyms. For instance, concept "Application Security Verification Framework" has synonym "AS Verification Framework."

Each *clause file* lists the clause number (e.g., 10.3.19) followed by its title. The rest of the file contains the narrative of the clause. A special *document file* contains the number of the document (e.g., 27034) and its title.

The algorithm begins by creating an individual of class StandardDocument with the title and number provided by the document file. Next, each clause file is processed and a building block individual is created. The class of the individual is determined from the title of the clause, by identifying occurrences of the names of building blocks found in the ontology (augmented with plurals and other syntactic variations). If the building block cannot be identified, the algorithm defaults to type Misc. The clause number is used to identify the parent clause, if it exists. The association is made by means of relation $partOf$. The text of the title and of the text of the clause are linked with the individual by means of relations $hasTitle$ and $hasBody$.

Next, the clause title is inspected for occurrences of the concepts and their synonyms, which are then associated with the building block individual by means of relation $mainTopic$. Note that, in some cases, the title of a clause should be interpreted within the context set forth by higher-level clauses. For instance, in ISO/IEC 19792 ("Security evaluation of biometrics"), the title of clause 6, "Security evaluation", should be interpreted within the context of the document, i.e. "Security evaluation for biometrics." For this reason, the algorithm allows for the manual specification of additional text to be considered in the determination of the main topic of a clause.

Finally, the text of the clause is matched against the known concepts in a similar way. Matching concepts are associated with the building block individual by means of relation $containsConcept$.

For an example application of the algorithm, consider clause 3 of ISO/IEC 27034, with title "Terms and definitions." The input ontology from **Sec. 4** does not contain a building block matching the title, and thus the algorithm creates a building block individual of class Misc and suitable attributes (title, etc.). When the text of the clause is inspected for occurrences of known concepts, the algorithm finds various matches, including:

- Application Security Authority
- Application Security Verification
- Application Security Verification Scheme

- Application Normative Framework
- Application Security Audit

6 Automated Reasoning in Support of Standardization

We have identified two broad classes of reasoning tasks that appear to be relevant to the process of document creation and querying, the ***drill-down*** type of query and the ***harvest*** type of query.

For each type of the query, we formalized the type of reasoning involved by means a set of axioms that make it possible to carry it out through the general-purpose logical inference mechanisms introduced earlier. The reasoning process is broken down into two parts: ontology-based querying and non-monotonic reasoning. The former applies relatively light-weight inference mechanisms to efficiently extract a broad set of relevant pieces of information from the ontology. The latter applies more sophisticated inference mechanisms to the restricted set of pieces of information identified in order to obtained the desired answer.

The process of answering a query Q can thus be reduced to checking whether Q follows from ontology Ω and axioms Λ, i.e.:

$$\tau(\Omega) \cup \Lambda \vDash Q$$

where $\tau(\Omega)$ denotes the set of tuples extracted from Ω through ontology-based querying. At the current stage, ontology-based querying is implemented by SPARQL queries and checking for entailment ($\vDash$ in the above equation) is reduced to finding answer sets of suitable ASP programs. Next, we discuss in more details the types of reasoning tasks considered and the axioms that formalize them.

In the ***drill-down*** **task**, the user looks for clauses relevant to a certain concept and may progressively refine the query to focus on particular types of documents or sets of clauses. For example, a user might want to investigate all the building blocks, technical, procedural, or descriptive, that apply to deployment of IoT edge devices while developing a standard document describing a new IoT framework. An example query is:

"What are the typical components of the existing frameworks?"

In response to this query, the reasoning mechanism will identify all building blocks pertaining to the description of frameworks and return the list of building blocks contained in them and whose types reoccur in all frameworks. This is accomplished as follows:

- $\tau(\Omega)$ identifies (*) every building block whose main topic (given by relation $hasMainTopic$) is a concept whose context (relation $includesContext$) is term Framework, and (*) all building blocks that are part of it. The query yields a set of triples $(root, bblock, type)$ where $root$ is the identifier of the building block with suitable main topic, $bblock$ is one of its constituents, and $type$ is the type of $bblock$ (see **Fig. 1**).

- Λ identifies the building block types that occur in all triples of $\tau(\Omega)$. The two key axioms are:

$$
\begin{aligned}
&\neg common(T) \leftarrow \\
&\qquad root(R1), root(R2), \\
&\qquad contains\_\ bb\_\ type(R_1, T), \\
&\qquad not\ contains\_bb\_type(R_2, T).
\end{aligned}
$$

$$
\begin{aligned}
&common(T) \leftarrow \\
&\qquad contains\_bb\_type(R, T), \\
&\qquad not\ \neg common(T).
\end{aligned}
$$

  The first axiom finds every building block type that occurs in the description of one framework, but not in some other. Relations $root$ and $contains\_bb\_type$ are obtained from the triples of $\tau(\Omega)$ (axioms omitted). The second axiom states that a type T is common to all framework descriptions if it occurs in at least one of them and was not flagged by the first axiom.
- Q is the atom $common(T)$, i.e. one is interested in every building block type T for which $common(T)$ holds.

For illustration purposes, let us suppose that an ontology contains building blocks for two frameworks, an application security verification framework (clause 1), a hardware security verification framework (clause 3.5) and a process-related framework (clause 7.2). Specifically, for the former the ontology contains the following components:

- A purpose building block (clause 1.1)
- A concepts building block (clause 1.2)
- A test building block (clause 1.3)
- A guidelines building block (clause 1.4)

For the hardware security verification framework, the ontology contains:

- A concepts building block (clause 3.5.1)
- A purpose building block (clause 3.5.2)
- A misc building block (clause 3.5.3)

For the process-related framework, the ontology contains:

- A process building block (clauses 7.2.1)
- A concepts building block (clause 7.2.2)

One can check that, for this example, $\tau(\Omega)$ will contain the triples

$(1{,}1.1, PurposeBB), (1{,}1.2, ConceptsBB), (1{,}1.3, TestBB), (1{,}1.4, GuidelinesBB),$
$(3.5{,}3.5.1, ConceptsBB), (3.5{,}3.5.2, PurposeBB), (3.5{,}3.5.3, MiscBB),$
$(7.2{,}7.2.1, ProcessBB), (7.2{,}7.2.2, ConceptsBB)$

One can also check that the only atom of the form $common(T)$ entailed by $\tau(\Omega) \cup \Lambda$ is $common("ConceptsBB")$. The reasoning mechanism has thus determined that the only type of building block the (known) frameworks share is $ConceptsBB$. Note that, while this is a simple example, it can be easily extended by adding suitable axioms, e.g. to find building blocks that are most frequently occurring or occurring at least with a certain frequency.

Let us now turn our attention to ***harvest*** queries that can assist in, e.g., the creation of new standards by identifying and harvesting relevant components of existing documents. A sample harvest query might ask:

"What are the relevant building blocks for a Security development lifecycle (SDL) framework for IoT?"

Let us assume that background knowledge related to SDL is available to the reasoning mechanism through the ontology or encoded by axioms such as:

$decomposed\_in(sdl, "Audit").$
$decomposed\_in(sdl, "Verification").$
$decomposed\_in(sdl, "Assurance").$

Additional background knowledge might state that, in the case of frameworks for auditing, verification, and assurance, one should consider IoT-related concerns, hardware-related concerns and software-related concerns. For illustration purposes, we show the axioms that determine if two concepts are similar based on the above statements about concerns and activities:

$is\_audit\_verification\_assurance("Audit").$
$is\_audit\_verification\_assurance("Verification").$
$is\_audit\_verification\_assurance("Assurance").$

$is\_hw\_sw\_iot\_concern(CRN) \leftarrow tc\_subclassof(CRN, "HardwareConcern").$
$is\_hw\_sw\_iot\_concern(CRN) \leftarrow tc\_subclassof(CRN, "SoftwareConcern").$
$is\_hw\_sw\_iot\_concern(CRN) \leftarrow tc\_subclassof(CRN, "IoTConcern").$

$tc\_subclassof(X, X) \leftarrow class(X).$
$tc\_subclassof(X, Y) \leftarrow subclassof(X, Z), tc\_subclassof(Z, Y).$

$similar\_to(C_1, C_2) \leftarrow$
$includesActivity(C_1, ACT),$
$includesActivity(C_2, ACT),$
$is\_audit\_verification\_assurance(ACT),$
$includesContext(C_1, "Framework"),$
$includesContext(C_2, "Framework"),$
$includesConcern(C_1, CRN_1),$

$$\begin{aligned}&includesConcern(C_2, CRN_2),\\ &is\_hw\_sw\_iot\_concern(CRN_1),\\ &is\_hw\_sw\_iot\_concern(CRN_2).\end{aligned}$$

The first three axioms describe the set of activities of interest. The next three axioms capture the set of (possibly indirect) subclasses of $HardwareConcern$, $SoftwareConcern$ and $IoTConcern$ (see **Fig. 3**). To identify possibly indirect subclasses, the axioms leverage relation $tc\_subclassof$ (abbreviation of transitive closure of $subclassof$), which is defined in a standard way by the following two axioms. The last axiom states that two concepts C_1 and C_2 are similar if they include the same audit, verification or assurance activity, have context $Framework$, and include concerns that are (possibly indirect) subclasses of $HardwareConcern$, $SoftwareConcern$ or $IoTConcern$.

The query can thus be reduced to finding building blocks for frameworks whose main topic is an SDL framework for IoT or a framework similar to it, as defined by the above axioms. That is, given an axiom:[6]

$$\begin{aligned}relevant(ROOT\_BB) \leftarrow\ &\\ &concept(C_2),\\ &includesContext(C_2, "Framework"),\\ &includesConcern(C_2, "IoTConcern"),\\ &decomposed\_in(sdl, ACT),\\ &includesActivity(C_2, ACT),\\ &hasMainTopic(ROOT\_BB, C_1),\\ &similar\_to(C_1, C_2).\end{aligned}$$

query Q is captured by atom $relevant(ROOT\_BB)$.

Given the prototype ontology, since SDL includes verification activities, the application-security verification framework discussed earlier is potentially relevant to the query – were it not for the fact that it is related to application security, rather than IoT. A similar argument holds for the hardware verification framework. This gap is bridged by the background knowledge.

Thus, one can check that, while no building block exists for an SDL framework for IoT, $\tau(\Omega) \cup \Lambda$ entails two atoms of the form $relevant(ROOT\_BB)$: $relevant(1)$, indicating the application-security verification framework (clause 1) is a match for the query, and $relevant(3.5)$, indicating that the hardware security verification framework (clause 3.5) is another match for the query. In fact, both have context $Framework$, describe an activity of interest ($Verification$), and have a concern that is a subclass of $HardwareConcern$, $SoftwareConcern$ or $IoTConcern$. Specifically for clause 1, while the application-security verification framework does not mention concern $SoftwareConcern$, it is related to application security, which, based on the knowledge from **Fig. 3**, is a software-related concern.

[6] We omit the axiom for matching of SDL framework for IoT, which is straightforward.

7 Benefits and obstacles

The area of standardization, and specifically a sample of security assessment related standards that we have evaluated, demonstrate excellent premises for knowledge engineering and automated reasoning. The creation of standards is a structured collaborative process that results in highly structured documents. Thus, extracting building blocks from standards, organizing them within an ontology, developing reasoning tools, and incorporating the use of ontology and reasoning at certain stages of the development of standards is feasible, as was confirmed by our experiment.

Moreover, some level of automation and harmonization in the development of standards would speed up the development process. With limited resources, increasing specialization of experts, diverse ICT infrastructures, and shortening technology lifecycles, expanded use of technology will be vastly beneficial.

Although the knowledge engineering community developed a wide range of tools to support the ontology-based management of some standard building blocks, the nature of the standardization process makes the initial investment in the technology and process difficult for standards bodies. Most standards experts are volunteers, spending a fraction of their working time on the development of standards and specifications. Even with the highly structure nature of standards, the planning and creation of the initial ontology is a daunting task, but it may become necessary as the need for standards continues to expand.

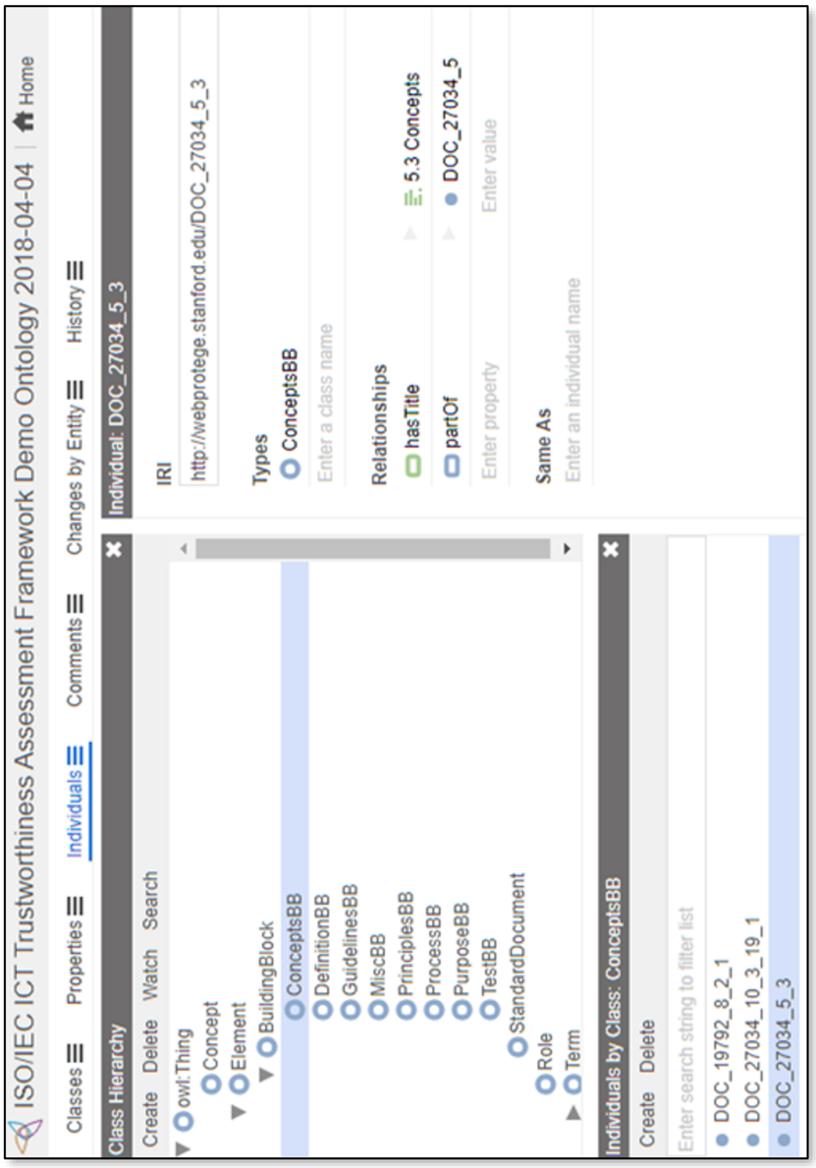
Another obstacle is connected to the positioning of the ontology within a standardization field. In order to be effective, the ontology itself and/or the building blocks it organizes need to be standardized. Although many areas of technology development have become automated and now rely on tools and processes these tools enforce, this change has not yet happened in standardization. The development of assessment-related standards has decade long traditions that served the field well. Creating an ontology to formalize building blocks and relationships among them may require reassessment of some practices.

However, we believe that the circumstances have aligned to make automation of some parts of the standardization process more important. The lack of cross-domain expertise among experts, the sheer number of existing standards, and the needs of the global digital infrastructure make some levels of formalization of building blocks in standards inevitable. And greater formalization is likely to lead to increase in automation.

8 Conclusions and Future Work

The first stage of our work was exploratory, but it confirmed the feasibility of the ontology-based management of standards, while also highlighting many difficulties along the way. We have built a prototype ontology based on a sample of ISO/IEC standards and determined that there are easily detectable building blocks that can help technologists working in the areas of security or “trustworthiness” assessment to build and update standards faster, to detect conflicts, and to promote harmonization within the field. Screenshots of the prototype system can be found in **Fig. 6** and **Fig. 7**.

We have also taken stock of the obstacles to introducing ontologies into standardization. Overcoming these obstacles will be an important area of the future work. Defining the structure of the most common building blocks as well as methodologies to promote harmonization in standards will be another important direction. The improvement of the techniques for automated reasoning in the context of ICT assessment-related standards will continue to be important. Finally, applications of this work to adjacent areas, such as automated classification of standards, will be explored.



owl:Thing
- Concept
- Element
  - BuildingBlock
    - ConceptsBB
    - DefinitionBB
    - GuidelinesBB
    - MiscBB
    - PrinciplesBB
    - ProcessBB
    - PurposeBB
    - TestBB
  - StandardDocument
- Role
- Term

- DOC_19792_8_2_1
- DOC_27034_10_3_19_1
- DOC_27034_5_3

Individual: DOC_27034_5_3

IRI
http://webprotege.stanford.edu/DOC_27034_5_3

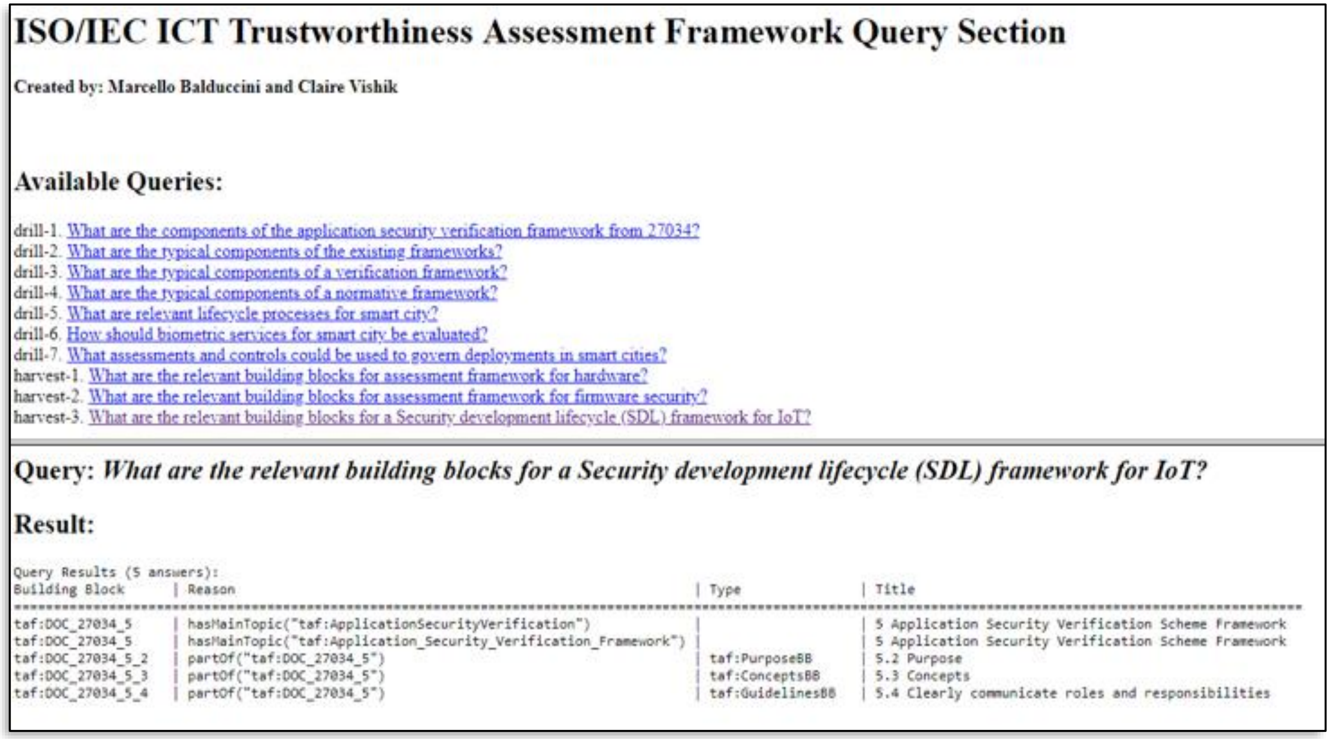
Types
ConceptsBB

Relationships

hasTitle	5.3 Concepts
partOf	DOC_27034_5

Same As

Fig. 6. Prototype system: ontology creation page



ISO/IEC ICT Trustworthiness Assessment Framework Query Section

Created by: Marcello Balduccini and Claire Vishik

Available Queries:

drill-1. What are the components of the application security verification framework from 27034?
drill-2. What are the typical components of the existing frameworks?
drill-3. What are the typical components of a verification framework?
drill-4. What are the typical components of a normative framework?
drill-5. What are relevant lifecycle processes for smart city?
drill-6. How should biometric services for smart city be evaluated?
drill-7. What assessments and controls could be used to govern deployments in smart cities?
harvest-1. What are the relevant building blocks for assessment framework for hardware?
harvest-2. What are the relevant building blocks for assessment framework for firmware security?
harvest-3. What are the relevant building blocks for a Security development lifecycle (SDL) framework for IoT?

Query: ***What are the relevant building blocks for a Security development lifecycle (SDL) framework for IoT?***

Result:

```
Query Results (5 answers):
Building Block      | Reason                                                         | Type              | Title
=====================================================================================================================================================
taf:DOC_27034_5     | hasMainTopic("taf:ApplicationSecurityVerification")           |                   | 5 Application Security Verification Scheme Framework
taf:DOC_27034_5     | hasMainTopic("taf:Application_Security_Verification_Framework") |                 | 5 Application Security Verification Scheme Framework
taf:DOC_27034_5_2   | partOf("taf:DOC_27034_5")                                      | taf:PurposeBB     | 5.2 Purpose
taf:DOC_27034_5_3   | partOf("taf:DOC_27034_5")                                      | taf:ConceptsBB    | 5.3 Concepts
taf:DOC_27034_5_4   | partOf("taf:DOC_27034_5")                                      | taf:GuidelinesBB  | 5.4 Clearly communicate roles and responsibilities
```

Fig. 7. Prototype system: query demonstration interface

References

Alobaidi, M., Malik, K. M., & Hussain, M. (2018). Automated Ontology Generation Framework Powered by Linked Biomedical Ontologies for Disease-Drug Domain. *Computer Methods and Programs in Biomedicine*.

Boje, D. M. (Ed.). (2015). *Organizational change and global standardization: Solutions to standards and norms overwhelming organizations*. Routledge.

Cai, M. C., Xu, Q., Pan, Y. J., Pan, W., Ji, N., Li, Y. B., & Ji, Z. L. (2014). ADReCS: an ontology database for aiding standardization and hierarchical classification of adverse drug reaction terms. *Nucleic acids research*, *43*(D1), D907-D913.

de Franco Rosa, F., Jino, M., & Bonacin, R. (2018). Towards an Ontology of Security Assessment: A Core Model Proposal. In *Information Technology-New Generations* (pp. 75-80). Springer, Cham.

de Franco Rosa, F., Jino, M., Bueno, P. M. S., & Bonacin, R. (2018, April). Coverage-Based Heuristics for Selecting Assessment Items from Security Standards: A Core Set Proposal. In *2018 Workshop on Metrology for Industry 4.0 and IoT* (pp. 192-197). IEEE.

Fenz, S., & Ekelhart, A. (2009). Formalizing information security knowledge. In *Proceedings of the 4th international Symposium on information, Computer, and Communications Security* (pp. 183-194). ACM.

Fraga, A. L., & Vegetti, M. (2017). Semi-Automated Ontology Generation Process from Industrial Product Data Standards. In *III Simposio Argentino de Ontologías y sus Aplicaciones (SAOA)-JAIIO 46 (Córdoba, 2017).*

Gelfond, M., & Lifschitz, V. (1991). Classical Negation in Logic Programs and Disjunctive Databases. *New Generation Computing*, *9*, 365–385.

Gonzalez-Perez, C., Henderson-Sellers, B., McBride, T., Low, G. C., & Larrucea, X. (2016). An Ontology for ISO software engineering standards: 2) Proof of concept and application. *Computer Standards & Interfaces*, *48*, 112-123.

Harris, S., Seaborne, A., & Prud'hommeaux, E. (2013). SPARQL 1.1 query language. *W3C recommendation, 21*(10).

Huang, Y., & Li, G. (2010). A semantic analysis for internet of things. In Intelligent computation technology and automation (ICICTA), 2010 international conference on (Vol. 1, pp. 336-339). IEEE.

Marek, V. W., & Truszczynski, M. (1999). Stable Models and an Alternative Logic Programming. In *The Logic Programming Paradigm: a 25-Year Perspective* (pp. 375–398).

Ramanauskaitė, S., Olifer, D., Goranin, N., & Čenys, A. (2013). Security ontology for adaptive mapping of security standards. *International Journal of Computers, Communications & Control (IJCCC)*, *8*(6), 813-825.