TermIt: Managing Domain Terminologies

Martin Ledvinka[1,*], Miroslav Blaško[1] and Michal Med[1]

[1]*Department of Computer Science, Faculty of Electrical Engineering, Czech Technical University in Prague, Technická 2, Prague 6, Czech Republic*

Abstract

Many domains use specific terminologies to describe concepts. Being able to explicitly manage such terminologies instead of relying on their common knowledge is beneficial both for newcomers and for people for whom the terminology is not their daily bread and butter. This is especially true for legislative terminologies. We present TermIt, a Semantic Web-based terminology manager that allows domain experts to create and manage high-quality terminologies, link them to normative documents as well as use them to annotate other related documents. We discuss the architecture of the system and the technologies used in its development.

Keywords

Terminology, Glossary, Thesaurus, SKOS

1. Introduction

People in different domains tend to use specific terminologies to describe concepts. The meaning of these terms can often be in a conflict with the meaning understood by lay folk (an *issue* in GitHub issue tracking does not necessarily mean a problem with the software), or there may exist subtle differences in the terms' semantics. In case of legislation or other normative documents, these subtle differences may have serious consequences. For example, according to the Czech Ordinance No. 268/2009 Coll (Ordinance on technical requirements for buildings), a *building* is an above-ground structure, including its underground portion, spatially concentrated and externally enclosed for the most part by perimeter walls and roof construction. On the other hand, according to the Czech Law No. 406/2000 Coll. (Energy Management Act), a *building* is an above-ground structure and its underground parts, spatially concentrated and externally largely enclosed by external walls and roof structure, in which energy is used to modify the indoor environment for heating or cooling purposes. The difference between these two meanings can be important in case of communication with a public office or in a legal dispute.

It is therefore advantageous to be able to manage a terminology of the domain consisting of *terms*, where each term has a *label*, i.e. the phrase it is called by and a *definition* describing the term's meaning. *TermIt* was created to support the creation, management and sharing of such terminologies.

2. Use Case Overview

TermIt is a terminology manager based on SKOS (Simple Knowledge Organization System) [1]. It is an information system with two main use cases:

1. Terminology management
2. Document annotation

Developers Workshop, co-located with SEMANTiCS'25: International Conference on Semantic Systems, September 3–5, 2025, Vienna, Austria

*Corresponding author.

martin.ledvinka@fel.cvut.cz (M. Ledvinka); miroslav.blasko@fel.cvut.cz (M. Blaško); michal.med@fel.cvut.cz (M. Med)

0000-0002-2451-2348 (M. Ledvinka); 0000-0002-9459-019X (M. Blaško); 0000-0002-3844-426X (M. Med)

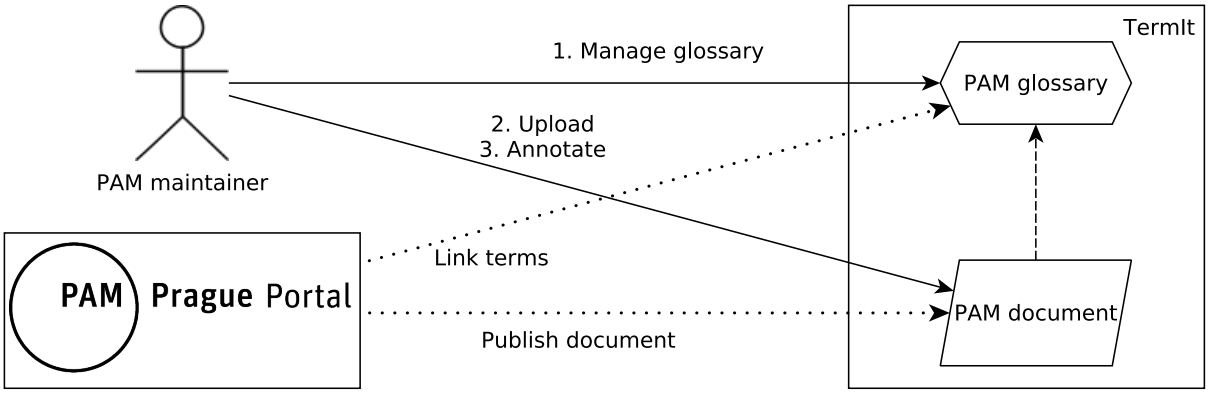

Figure 1: Interaction between the PAM Portal and TermIt. Solid lines represent user interaction, dashed line represents association between a glossary and a document and dotted lines represent system interaction.

2.1. Terminology Management

The terms in TermIt are grouped into *glossaries*, where each glossary corresponds to a `skos:ConceptScheme`.[1] Each term is a SKOS `skos:Concept` with basic attributes such as `skos:prefLabel`, `skos:definition`, etc. A hierarchy of terms can be built within a glossary (using `skos:broader`) as well as across glossaries (using `skos:broadMatch`). Additional non-hierarchical relationships between terms can be specified using `skos:related`(`Match`) and `skos:exactMatch`. Users are also able to create and use additional attributes that are not covered by the built-in data model.

To support use in multilingual use cases (for example, the Czech Standardization Agency uses TermIt to map building construction glossaries in cooperation with other European standardization agencies), all string-based term attributes may contain language-tagged values. The only requirement for a term is to have a label (`skos:prefLabel`) in the glossary's configured primary language.

Although domain experts, who are the primary intended users of TermIt, are not expected to do detailed modeling of the relationships between terms (this task requires an ontology engineer), TermIt does offer basic modeling capabilities to more advanced users. One already mentioned is the SKOS-based ability to create term hierarchies. Another is the ability to classify terms into categories. By default, a UFO-based [2] classification scheme is supported, but different options may be provided. Should a need arise to create a conceptual model of the domain, an ontology engineer could use a modeling tool such as OntoGrapher [3] (see Section 3) to further specify term data.

Data quality is a concern in many cases where people provide the input for an information system and terminology management is no exception. On the contrary, since the terminology is often based on normative documents or used in highly regulated conditions, the quality of the terminology is important. The solution in TermIt is twofold: 1) basic consistency rules, such as the requirement for a term to have a label that is unique within its glossary, are enforced; 2) a *validation service* that checks the quality of terms in a glossary using a (configurable) set of SHACL [4] rules is used. These rules check, for example, that a term has a definition, that the term's label is not used as another term's synonym (`skos:altLabel`), that a term's identifier is in line with the glossary namespace, etc.

Other terminology management features of TermIt include tracking changes made by users, the ability to create glossary snapshots, comment on terms, and a simple state-based workflow. The user may search for glossaries and terms using full-text search or use faceted search for more fine-grained term exploration. Glossaries may be exported/imported to/from RDF as well as MS Excel.

2.2. Document Annotation

Terminologies are often based on normative documents. TermIt therefore supports the attachment of documents (possibly consisting of multiple files) to glossaries. These documents can be used as a

[1] skos: is prefix for http://www.w3.org/2004/02/skos/core#

seed to populate the glossary based on suggestions from *Annotace*[2] [5] – a text analysis service, or as a reference for the definition of existing terms in a glossary. In fact, the definition of the term can be marked directly in the document, providing a clear link to its source.

Another use case for document annotation is having an existing glossary and finding the occurrences of its terms in the document. For example, the Prague Institute for Planning and Development maintains multiple glossaries that are then used to provide definitions of terms occurring in the documents it publishes. The Portal of Prague Planning Analytical Materials (PAM)[3] is one of the sites where such documents are published. Figure 1 illustrates how the PAM Portal and its maintainers use TermIt.

It is important to note that the document annotation is semi-automatic – the service suggests occurrences of terms, but an authorized user is ultimately required to approve or reject these occurrences. If a new version of a document is uploaded to TermIt, previous occurrences are retained, as long as the changes in the document are not too significant. Annotace expects the content to be HTML (or plain text that it transforms to HTML) and produces RDFa-annotated HTML.[4]

3. Technical Overview

TermIt as a system consists of the TermIt application and multiple optional services - Figure 2 illustrates its architecture. The TermIt application itself is designed as a monolith with back-end written in Java and front-end written in TypeScript. The back-end follows the layered architectural style [6] which is typical for web applications and promotes separating the business logic of the application from the infrastructural data access and web service layers. The additional services that can be used as part of TermIt are:

Annotace [5] is a text analysis service that can be used to annotate HTML documents with occurrences of terms from the provided glossaries and suggest new terms based on their significance in the text. The current implementation supports two lemmatizers: Spark[5] and MorphoDiTa [7] (more suitable for Czech).

Validation service (as already mentioned in Section 2.1) is used to determine the quality of the term metadata and provide the user feedback as to what they should improve. The service contains a predefined set of SHACL rules and the caller may choose which ones to use for individual validation calls.

Authentication service compatible with the OAuth2 standard can be used instead of the built-in internal user management and authentication mechanism. This configuration is suitable in case the target organization already has an authentication solution in place or when TermIt is to be used with OntoGrapher (see below).

OntoGrapher [3] is a Web-based conceptual modeling tool designed primarily for ontological engineers, facilitating a more detailed description of relationships between terms.

3.1. Technologies

TermIt uses Semantic Web technologies at all levels of its implementation, yet the software libraries it uses ensure that it does not differ from regular Web applications in terms of software engineering and development. The data are stored in a triple store. In particular, GraphDB[6] is used for its superior performance and support for custom inference rules. TermIt uses a number of rules, for example, to infer inverse relationships between terms or a term and the glossary to which it belongs. RDF named

[2] https://github.com/kbss-cvut/annotace, accessed 2025-07-30
[3] https://uap.iprpraha.cz/en, accessed 2025-07-30
[4] https://www.w3.org/TR/rdfa-core/, accessed 2025-07-30
[5] https://sparknlp.org/, accessed 2025-07-30
[6] https://graphdb.ontotext.com/, accessed 2025-07-30

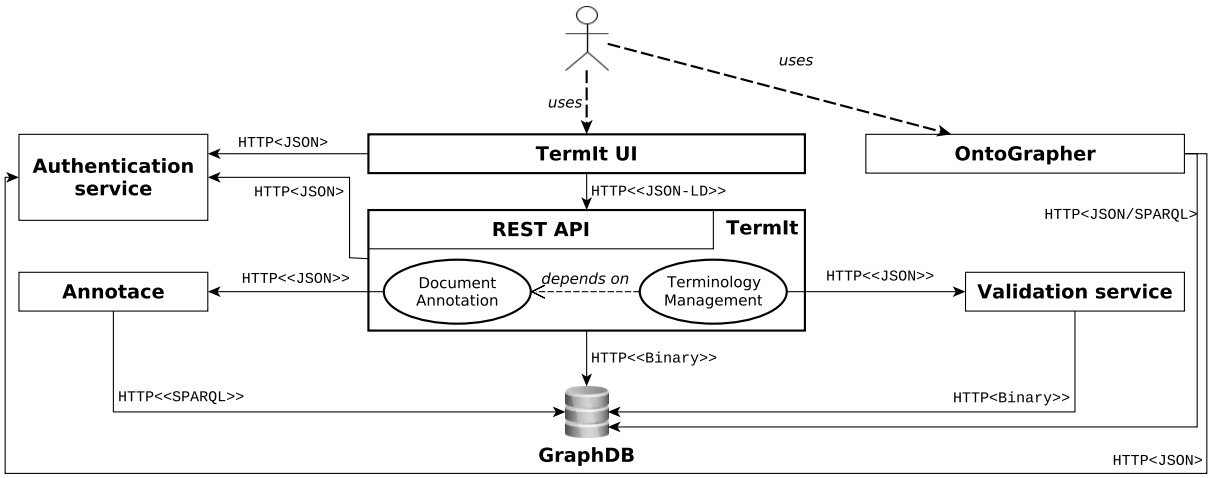

Figure 2: TermIt architecture. Solid lines represent invocation, dashed lines represent dependency, bold dashed line represents user interaction. The architecture depicts the core TermIt application (bold border) and the services it may use.

graphs are used to structure the data – each glossary is stored in a separate named graph (its IRI[7] coincides with the glossary's identifier) and additional named graphs are used for related technical data (change records, comments, etc.).

The repository is accessed using the Java OWL Persistence API (JOPA) [8] – a semantic data persistence library. The back-end uses Spring Boot[8] and provides a REST API for the front-end and other clients to use. The API is documented by an OpenAPI[9] specification. The API supports JSON and JSON-LD as data-exchange formats. The front-end communicates with the back-end using JSON-LD, but other clients (such as the aforementioned Portal of Prague Planning Analytical Materials) use JSON. JSON-LD (de)serialization is handled by the JB4JSON-LD library [9], which is also developed by TermIt's authors.

The front-end is a client-side application written in TypeScript using React. It follows common React application development practices – it uses a state management library, routing, authentication, and authorization workflow – and the user interface (UI) is localized to Czech and English. Since all UI texts are extracted into localization files, adding a translation for another language would be trivial. As mentioned earlier, the front-end communicates with the back-end using JSON-LD with *context*. From experience, this increases the amount of data transferred for a single object retrieval, but in situations where objects repeat in the content, the amount of data actually decreases significantly, because JB4JSON-LD uses an object's identifier for its repeated occurrences. jsonld.js,[10] is then used to ensure that all references are replaced with the full object for the rest of the front-end code.

The whole system is published and deployed using the Docker[11] containerization platform. This ensures the runtime is the same for all instances and nothing (besides Docker itself) needs to be installed on the host system.

4. Deployments

TermIt has been used or is planned to be used by several organizations:

- Prague Institute for Planning and Development
- Czech Standardization Agency

[7] Internationalized Resource Identifier
[8] https://spring.io/projects/spring-boot, accessed 2025-07-30
[9] https://spec.openapis.org/oas/latest.html, accessed 2025-07-30
[10] https://github.com/digitalbazaar/jsonld.js, accessed 2025-07-30
[11] https://www.docker.com/, accessed 2025-07-30

- Czech Digital and Information Agency – used in a project of an *assembly line of semantic vocabularies* for the Czech legislation [10]
- ČEPS – Czech Transmission System Operator – currently in evaluation phase
- Ministry of Health of the Czech Republic – currently in negotiations

5. Conclusions

We have introduced TermIt, a terminology manager based on Semantic Web technologies. We argued that a clear definition of terminology is important, especially in domains where misunderstanding the meaning of a word or a phrase can have an impact on a conversation, a dispute, or a legal act.

Building TermIt with Semantic Web technologies has had a number of benefits. A well established model (SKOS) was used to describe the terms, including their relationships, while retaining the possibility to easily define additional properties. The terms have stable and globally valid identifiers in the form of IRIs, and the terminologies can be published as machine-readable data. On the other hand, some drawbacks have also been discovered – mainly issues with performance for larger amounts of data (glossaries with hundreds or thousands of terms or deep hierarchies).

TermIt can be used in cooperation with additional tools, such as a validation service, a text analysis service, and a tool for modeling detailed relationships between terms.

A more detailed description of TermIt use cases as well as a comparison with other relevant tools can be found in [11].

Declaration on Generative AI

The author(s) have not employed any Generative AI tools.

References

[1] A. Miles, S. Bechhofer, SKOS Simple Knowledge Organization System Reference, W3C Recommendation, W3C, 2009. https://www.w3.org/TR/skos-reference, accessed 2025-07-30.

[2] Guizzardi, Giancarlo, Ontological Foundations for Structural Conceptual Models, Ph.D. thesis, University of Twente, 2005.

[3] A. Binder, P. Křemen, OntoGrapher: a Web-based Tool for Ontological Conceptual Modeling, in: Proceedings of the 21st CIAO! Doctoral Consortium, and Enterprise Engineering Working Conference Forum 2021 co-located with 11th Enterprise Engineering Working Conference (EEWC 2021), CEUR-WS, 2021. URL: https://ceur-ws.org/Vol-3115/paper2.pdf, accessed 2025-07-30.

[4] H. Knublauch, D. Kontokostas, Shapes Constraint Language (SHACL), W3C Recommendation, W3C, 2017. URL: https://www.w3.org/TR/2017/REC-shacl-20170720/.

[5] L. Saeeda, M. Med, M. Ledvinka, M. Blaško, P. Křemen, Entity linking and lexico-semantic patterns for ontology learning, in: The Semantic Web, Springer, Cham, 2020, pp. 138–153. doi:10.1007/978-3-030-49461-2_9.

[6] F. Buschman, R. Meunier, H. Rohnert, P. Sommerlad, M. Stal, Pattern-Oriented Software Architecture Volume 1: A System of Patterns, volume 1, Wiley, Hoboken, New Jersey, 1996.

[7] J. Straková, M. Straka, J. Hajič, Open-Source Tools for Morphology, Lemmatization, POS Tagging and Named Entity Recognition, in: Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations, Association for Computational Linguistics, Baltimore, Maryland, 2014, pp. 13–18. URL: http://www.aclweb.org/anthology/P/P14/P14-5003.pdf, accessed 2025-07-30.

[8] M. Ledvinka, P. Křemen, JOPA: Accessing Ontologies in an Object-oriented Way, in: Proceedings of the 17th International Conference on Enterprise Information Systems - Volume 1: ICEIS, INSTICC, SciTePress, 2015, pp. 212–221. doi:10.5220/0005400302120221.

[9] M. Ledvinka, Java Binding for JSON-LD, in: Proceedings of the 19th International Conference on Web Information Systems and Technologies - Volume 1: WEBIST, INSTICC, SciTePress, 2023, pp. 207–214. doi:10.5220/0012168500003584.
[10] K. Klíma, M. Blaško, P. Křemen, M. Nečaský, M. Ledvinka, A. Binderová, M. Švagr, F. Kopecký, Assembly line: a tool for collaborative modeling of ontologies in public administration, in: 2023 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C), IEEE, 2023, pp. 24–29.
[11] P. Křemen, M. Med, M. Blaško, L. Saeeda, M. Ledvinka, A. Buzek, TermIt: Managing normative thesauri, Semantic Web 16 (2025) SW–243547. doi:10.3233/SW-243547.

A. Online Resources

TermIt source code is available on GitHub under the GPL license:

- TermIt back-end at https://github.com/kbss-cvut/termit,
- TermIt front-end at https://github.com/kbss-cvut/termit-ui,
- TermIt Docker Compose configuration (including installation instructions) at https://github.com/kbss-cvut/termit-docker.

A demo instance of TermIt is available at https://kbss.felk.cvut.cz/termit-demo.