On the Development of a Collaborative Knowledge Platform for Engineering Sciences

Jonas Jepsen[a], Arthur Zamfir[b], Brigitte Boden[c], Jacopo Zamboni[d] and Erwin Moerland[e]
Institute of System Architectures in Aeronautics, German Aerospace Center, Hein-Saß-Weg 22, Hamburg, Germany

Keywords: Knowledge-Based Engineering, Semantic Web Technologies, Web Application, Corporate Amnesia, Knowledge Management, Collaboration, Knowledge Platform, Codex.

Abstract: Knowledge is undoubtedly one of the most important assets for all organizations. Losing knowledge that once was considered part of an organization always poses a problem. When knowledge is concentrated on single individuals, as is often the case in research environments, this is particularly difficult to avoid. This is also true for software developed by individuals such as Knowledge-Based Engineering (KBE) applications. In this paper we show how the Codex Framework, which is based on Semantic Web Technologies (SWT), can be used for creating, maintaining and inspecting formalized knowledge in a way which also fosters its reuse and execution. We show use cases where Codex was applied and discuss the advantages and shortcomings compared to a conventional Object-Oriented Programming (OOP) approach. From these experiences we then draw conclusions on why the adoption rate of Codex is below expectation and present a newly developed collaborative knowledge platform which is designed to overcome the identified challenges.

1 INTRODUCTION

Organizational knowledge, the valuable pool of information and insights within a company, serves as the foundation for success in the fast-paced world of business (Nonaka and Takeuchi, 1995). However, an often overlooked and consequential challenge that companies face is corporate amnesia — the gradual erosion or forgetting of this vital institutional knowledge, experience, and lessons learned over time (Dalkir, 2005). As employee turnover rates rise and industries undergo technological transformations, organizations become increasingly fragile to losing their collective memory. This loss hinders their ability to make informed decisions, replicate past achievements, and avoid repeating previous failures, ultimately impacting their overall performance and long-term viability. In this section we discuss causes and effects of corporate amnesia in research environments as well as means to improve its prevention.

1.1 Corporate Amnesia in Engineering Research

In engineering research environments, avoiding corporate amnesia poses a unique challenge. Individuals typically enter these environments after completing their master's degrees and often continue their stay to conduct PhD studies. Moreover, researchers frequently dedicate their efforts to highly specialized topics, resulting in the accumulation and creation of a substantial amount of knowledge throughout their thesis work. As a result, the risk of corporate amnesia increases due to the transient nature of researchers and the significant personal expertise they develop within their specific knowledge domain. While a condensed portion of their knowledge is eventually documented in the form of a PhD thesis, it is crucial to recognize that a considerable amount of knowledge remains undocumented or is not reusable unless high effort is involved.

Nowadays, PhD theses in the field of engineering often include the development of some kind of software. Whether it is for demonstrating a new algorithm, method or methodology, performing simulations or analyses, or creating a design tool whose extent can vary from using simple handbook methods to fully fledged Knowledge-Based Engineering (KBE)

[a] https://orcid.org/0000-0002-3307-1978
[b] https://orcid.org/0000-0001-7579-9628
[c] https://orcid.org/0000-0002-2326-2762
[d] https://orcid.org/0000-0003-3207-4138
[e] https://orcid.org/0000-0003-4818-936X

Jepsen, J., Zamfir, A., Boden, B., Zamboni, J. and Moerland, E.
On the Development of a Collaborative Knowledge Platform for Engineering Sciences.
DOI: 10.5220/0012189600003598
In *Proceedings of the 15th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2023) - Volume 2: KEOD*, pages 208-215
ISBN: 978-989-758-671-2; ISSN: 2184-3228

applications (La Rocca, 2012). This software is often written, maintained and run by a single person. Although it contains a set of explicit instructions, operating the software requires a deep understanding of the underlying assumptions which are often only present as tacit knowledge gained while conducting the PhD.

The ease of extracting knowledge from software or adapting it to meet new requirements varies depending on how it was developed. In situations where the original creator is no longer accessible, these tasks can become highly challenging. As a consequence, a significant portion of the knowledge generated within the software is effectively lost when an individual departs, resulting in a notable case of corporate amnesia.

From our experience the quality of software written in research institutions covers the complete spectrum from legacy codebase with poor architectural cohesion, to a professionally crafted architecture and modular set of well written software applications. In between these extremes are many different levels of software quality, but the majority of scientists would rather focus on developing their domain knowledge than mastering the craft of software development (Kelly and Sanders, 2008).

Knowing that there are multiple angles from which corporate amnesia should be addressed by organizations, this paper focuses on technical solutions for solving two of the identified challenges — a suitable way for knowledge representation and means to reduce software development related tasks not directly contributing to doing science.

1.2 Preserving Organizational Memory

There are multiple approaches to knowledge management for keeping existing knowledge inside an organization and to support the creation of new knowledge. The authors of (Nonaka and Takeuchi, 2021) group knowledge in two major categories, explicit knowledge and tacit knowledge. While explicit knowledge is formal and systematic, like words, audio recordings and images, tacit knowledge is far more difficult to express and resides “within the heads of knowers” (Dalkir, 2005). Depending on the context, other types of knowledge are also addressed. The authors of (Li and Gao, 2003) argue that although often used synonymously, implicit knowledge should be distinguished from tacit knowledge since it refers to knowledge that can be articulated but for some reasons is not. It is crucial to acknowledge that all three mentioned types of knowledge, with their respective strengths and weaknesses, are essential for knowledge creation and fostering innovation. Although being a significant asset in driving innovations within companies, methods for retaining tacit knowledge is outside the scope of this paper. In this paper, our focus is on preserving organizational memory by encouraging scientists to externalize a significant portion of their explicit and implicit knowledge into a formalized and reusable model. In order to achieve that, each individual needs to be provided with an incentive to do so. The presented research attempts to free scientists from the burden of traditional software development.

In Section 2, we show how the Codex framework has been used for the development of KBE applications and discuss its advantages and disadvantages. Section 3 deals with the transition from Codex framework to the collaborative knowledge platform and presents the current state of the platform. In Section 4 we conclude with a summary and an outlook on the next steps.

2 SEMANTIC WEB TECHNOLOGIES TO THE AID

As arguments and dispute are integral to scientific progress, an effective knowledge structure for knowledge modelling should adopt a flexible and open-minded paradigm that allows for different opinions about the same subject to exist. By making use of Semantic Web Technologies (SWT) such as the Resource Description Framework (RDF) and the Web Ontology Language (OWL) as the base modelling language, it even supports explicitly interconnecting models from different domains or disciplines, which is one of the challenges for KBE applications identified by (Verhagen et al., 2012).

Schema-free languages such as RDF and OWL are an alternative to a central schema-based data exchange format such as CPACS (Alder et al., 2020). These languages provide a more flexible approach, without the need for building a consensus on a single data structure. This flexibility promotes interoperability between different domains since data can be represented, extended and shared without strict schema constraints. RDF and OWL enable the representation of data with rich semantics and relationships. This promotes better understanding and integration of data across domains. Moreover, collaboration can benefit from a shared understanding of the meaning and context of data, leading to more effective data integration and knowledge discovery.

The authors of (Zamboni et al., 2020) elaborate how a generic language such as RDF can be used to overcome the challenges of collaboration in multi-domain environments. The combination of a highly abstract base language and more expressive additions such as OWL allows easier cross-domain knowledge integration while simultaneously enabling independent and effective capturing of the precise meaning of each individual domain.

In the next section we give a brief introduction to the SWT-based Codex framework. We discuss specific characteristics of the different modules and give examples on how they are used. We then summarize the experiences made and discuss the benefits and shortcomings of the framework.

2.1 The Codex Framework

The **CO**llaborative **DE**sign and e**X**ploration (Codex) framework is a software framework for the development of KBE applications based on SWT. At its core, Codex makes use of SWT as a domain neutral knowledge representation in order to naturally support the integration of knowledge from multiple domains. SWT, especially RDF and OWL, are explicitly designed for linking knowledge and allow users to combine data from multiple domains. Codex enables users to model their knowledge in domain-specific knowledge bases which, if well designed, can be easily integrated, reused and build upon.

This knowledge base is made up of several domains, each of which describes a specific aspect or component of the aircraft model, such as the mission to be performed or its wings and tail. By using the formalized knowledge the tool computes aircraft masses, geometrical and performance properties as a function of external requirements in an automated way.

At the heart of codex is the *codex-semantic* module, an implementation of semantic modelling using RDF and OWL, as well as additional syntaxes such as Manchester Syntax and Functional Syntax. It allows users of the framework to efficiently model their knowledge using Domain Specific Languages (DSL), and to define and execute logical rules such as those defined by the OWL 2 Profiles specification (World Wide Web Consortium [W3C], 2012). Since Codex is a framework for developing KBE applications, in addition to the core module for semantic modelling, Codex includes multiple other modules, each providing its own functionality needed for KBE applications. The three most important of these modules are:

codex-rules provides an interface to a production-rule engine. Rule engines such as Drools (Red Hat, 2021) make it easy to formalize and execute rules that systematically “produce” connected entities within a model in a declarative way. Each rule has a Left-Hand-Side (LHS) which is basically a graph query similar to SPARQL SELECT. This LHS is compared against the dataset and, if not configured otherwise, rules are executed once on every match in the corresponding model. When triggered, these rules can add new statements to the model or execute user defined code.

codex-parametric adds the capability to define mathematical constraints which can then be evaluated by a solver. It serves as a means to describe, analyze, and solve a parametric system through a DSL implemented specifically for this purpose. For solving the system of equations *codex-parametric* implements a Solution Path Generator which analyses the system of equations and groups them into Strongly Connected Components (SCC) (Bölling, 2015). SCC have cyclic dependencies and can only be solved simultaneously. By looking at the amount and individual sizes of the SCC, we can get an idea of the problem complexity. In addition to problem complexity, this module analyses whether a set of constraints is either well defined, over determined, or underdetermined. When well defined it can solve the system of equations and return results for the missing parameter values.

codex-geometry provides the user with a large number of Computer Aided Design (CAD) functionalities which are essential for the development of KBE applications (La Rocca, 2012). It defines its own DSL to allow users to model their geometry efficiently.

Although each module implements its own DSL to provide a simple and intuitive way for modelling the corresponding knowledge, in the current version, it is still necessary to have a good understanding of programming.

2.2 Experiences with KBE Applications

During the past two years the Codex framework has been applied in different use cases. In this section we briefly present three of them and discuss the benefits of using the Codex framework over more conventional implementations.

Fighter Design

The authors of (Zamboni et al., 2022) give a great example on how a complex knowledge base that describes several aircraft systems, components and their relationships is effectively combined to provide a tool for the conceptual design of combat aircraft (Figure 1).

Compared to a previous Python-based imperative implementation the Codex framework offers many advantages when it comes to reusing the knowledge base. The higher degree of granularity provided by the splitting of component’s model and rule-sets allows users to precisely control which knowledge should be used.

Fuel System Verification

The authors of (Boden et al., 2022) present how the Codex framework and particularly *codex-geometry* is used in combination with *codex-rules* to generate CAD geometry for aircraft fuel systems (Figure 2). This geometry is then used for validating geometrical requirements using *codex-rules*.

The use of the Codex framework in this approach provides the user a great deal of flexibility when creating a design and makes it easy to extend the system with custom requirements and rules.

Liquid Hydrogen Fuel Tank

Another example for the use of the Codex framework is a KBE tool for the development of liquid hydrogen fuel tanks for aircraft. It was used for research on liquid hydrogen storage design trades for short-range aircraft (Burschyk et al., 2021).

The declarative approach of Codex framework allows the user to decide freely whether the design should be performed from the inside out, from outside in or even a combination of both.

2.3 Towards Wider Adoption

Our experiences from using the Codex framework indicate a number of advantages for using a more declarative approach based on SWT for the development of KBE applications. In addition to advanced modelling capabilities, such as support for parameter metadata covering dimensions, units, numerical boundaries and computational tolerances, the Codex framework presented also simplifies extensibility. By separating components and rules it improves reusability of existing knowledge. For research software, where use cases might change quickly, the declarative approach is particularly useful as it makes the tool significantly more versatile. The flexibility to easily switch inputs and outputs allows KBE tools to be used in a variety of ways. As long as a user provides values for a sensible set of parameters, these parameters can be freely chosen.

Despite the listed significant advantages, we have observed some barriers in the adoption of the approach. In its current implementation, users still need to be familiar with the Kotlin programming language to use the Codex framework efficiently. Users still need to spend time on software related tasks, which however also holds true for the approach nowadays mainly taken by the engineering community which bases on the application of conventional OOP methods to create KBE tools. Users still need to invest time in software-related tasks, a requirement that also applies to the prevalent approach in the engineering community today, where conventional Object-Oriented Programming (OOP) methods are used to develop Knowledge-Based Engineering (KBE) tools. Finally, debugging a set of declarative rules can be more difficult than debugging imperative code. When the execution order of rules is determined by a rather complex algorithm there is no clear execution path to follow when debugging.

With this indication of the usefulness of SWT for the development of KBE applications and the main barrier of requiring to spend time on software development tasks, it became apparent that Codex has to be taken one step further. In the next section we present the concept and implementation of a collaborative knowledge platform.

3 A COLLABORATIVE KNOWLEDGE PLATFORM

Despite the demonstration of the value of Codex framework for the development of KBE applications, as indicated by the examples given in Section 2.1, the number of people adopting their developments to the framework was very limited. All adopters were

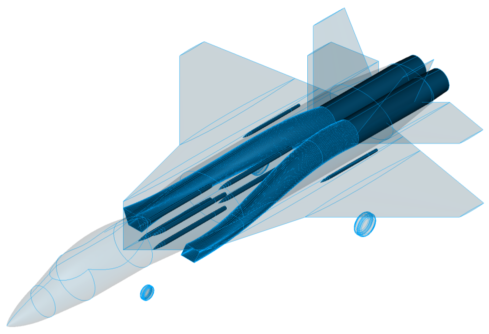


Figure 1: Combat Aircraft Design.

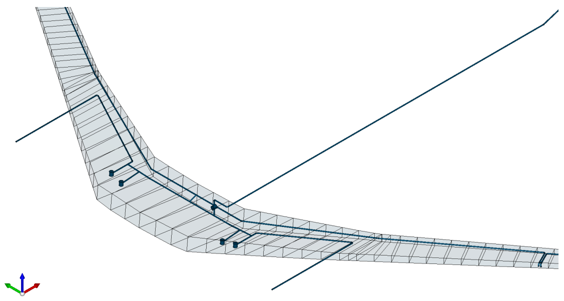


Figure 2: Fuel System Design.

people with an affinity towards software development and were either willing to learn, or already familiar with the Kotlin programming language. In addition, the developers also had at least a basic understanding of SWT. Although there are parallels between OOP and modelling in OWL, the learning curve can still be quite steep when switching from one to the other. Especially understanding the implications of the open world assumption and non-unique naming assumption can be difficult at first.

Recognizing that scientists prefer to engage in scientific work rather than software development (Kelly and Sanders, 2008), a web application was initiated to relieve users from the traditional software development demands. In order to achieve that, we decided to make even further use of SWT and also describe parametric and production rules in RDF. While this insight is significant, it is also crucial to enable the ”power users” who favour working with code over a GUI to do so. This led us to the implementation of the web-based Codex platform, described in the current section. Even though the original intent is creating a platform for developing KBE applications, we decided to build the core of the platform in a way that makes as few assumptions on its use as possible. Users should have full access to the core platform via a web API, enabling them to implement their own applications on top of the platform. The core of the collaborative knowledge platform serves as the backbone for client applications, such as graphical user interfaces, without prescribing their specific use-cases. It provides the community with a platform for effective knowledge collaboration as well as the ability to extend the capabilities of that platform without the need of maintaining their own server infrastructure.

3.1 Platform Concept

In (Zamboni et al., 2022) and (Boden et al., 2022) we established that SWT are an excellent medium for precise knowledge capture and cross-domain knowledge exchange. Despite these advantages, however, it is not widely used outside a few select domains that rely mostly on logical inference capabilities. By considering the analogy of RDF as a ”protocol for knowledge exchange,” to TCP/IP as a protocol for information exchange (Internet Engineering Task Force [IETF], 1981), we can deduce a hypothesis that might help explain its limited adoption despite its clear benefits. Protocols such as TCP/IP are extremely useful and at the core of what made the Internet successful and widely adopted. However, most people who actually use the Internet on a daily basis are completely unaware of its inner workings, and in particular have no idea how TCP/IP works or that it even exists, nor should they. Such infrastructure technology is essential for a system to work but it fades into the background as more abstraction layers are built on top of it (International Organization for Standardization [ISO], 1994) and what remains visible are domain-specific and easy to use interfaces such as web, desktop, or mobile applications where the individuals can focus on their task at hand instead of thinking about TCP/IP packets. Such layers of abstraction are mostly absent from the RDF “knowledge-protocol”. Of course, there are existing applications with graphical user interfaces, such as Protégé (Musen, 2015) but in order to effectively use these applications you still need to be familiar with the “knowledge-protocol” layer and understand RDF and OWL. Therefore, our hypothesis for the lack of wider adoption of SWT in engineering design is the result of missing high-level abstraction layers that will make the low-level protocol layer fade into the background and make effective knowledge collaboration based on SWT more accessible to non-experts.

With this hypothesis in mind, the aim was to create a collaborative knowledge platform utilizing open SWT standards. The expectation is that the platform will improve cross-disciplinary knowledge exchange upon its adoption by a dynamic community, especially in engineering sciences, such as the complex field of aerospace engineering. In order to achieve these goals and overcome the challenges described above, the platform must meet several essential requirements. The following is a condensed selection of the top-level requirements that comprise the key aspects of a collaborative knowledge platform.

Managing & Publishing RDF models Users MUST be able to create, read, update, delete and publish RDF models in a persistent data storage.

Access Control Capabilities User access to models MUST be controlled and manageable by users with corresponding permissions.

User Management There MUST be capabilities to manage (identify, add, update, and remove) users.

Authentication There MUST be capabilities to authenticate users.

Document Resource Storage There MUST be a document storage system for storing and sharing arbitrary data files (as resources, including published models).

Real-Time Synchronization Updates from users working on the same model SHALL be synchronized. This implies users inspecting a model are informed of updates from other users in real-time.

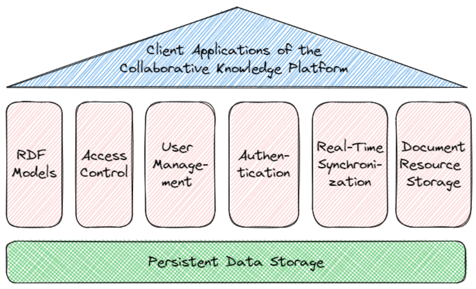

Client Applications of the Collaborative Knowledge Platform
RDF Models
Access Control
User Manage-ment
Authen-tication
Real-Time Synchroni-zation
Document Resource Storage
Persistent Data Storage

Figure 3: Collaborative Knowledge Platform.

Figure 3 illustrates that these six pillars, based on a persistent storage, are the foundation for the collaborative knowledge platform and can be used for a large variety of client applications. In Section 3.3, we provide an example of how these basic features can be utilized to implement a client for collaborative knowledge modelling, acting as the aforementioned higher-level abstraction layers on top of the RDF ”knowledge-protocol”.

This is largely in contrast to the Codex framework implementation where all the models are stored in memory only, unless explicitly exported to file. When deciding to directly store all the knowledge in RDF models, the exchangeability and reusability is significantly increased.

In the next section, we give a more detailed overview of the architecture and implementation of the core of our extensible collaborative knowledge platform.

3.2 Platform Implementation

Based on the listed requirements, the groundwork for the collaborative knowledge platform has been established. The resulting system provides a simple but powerful foundation for all additional application features. It improves extensibility by focusing on a limited set of basic features and then implementing additional features as client applications of this basic service. This includes the user interface as well as the features connected to KBE as described in the use cases from Section 2.2. Every service that is build on top of the platform core could be replaced by a custom implementation, thus providing the community with an environment to build their own tools. Thanks to the open approach, the community can construct tools whilst making use of already existing tools, much like software libraries utilizing other software libraries. This implies that once domain knowledge is implemented and shared, others can easily leverage it to develop new domain-specific tools.

Figure 4 illustrates the current architecture of the collaborative knowledge platform. Looking at the requirements from the previous section one can map the responsibilities as follows.

The platform consists of a **Gateway** that is used to deliver the front-end application to the user’s browser. It also delegates requests to the Real-Time Service (RTS), Core Backend Service (CBS) and Document Resource Service (DRS) as needed.

The **Core Backend Service (CBS)** is at the center of the Codex platform. It implements user management, access control, and RDF model registration. Access control is implemented on a very fundamental level. There are four types of access which can be granted for each model, which are read, write, delete, and manage. More complex types of authentication can always be implemented by client applications which then act as a delegator for these fundamental permissions. Published models can either be stored in a separate dataset on the server or in the DRS. While storing published models in a separate dataset has the advantage of allowing SPARQL requests on those models directly, saving them in a content addressed DRS can provide several useful characteristics, such as verifying data integrity, guaranteeing immutability, and providing resilience to data loss. The latter aspect is especially useful since one of the challenges of SWT is to assure that a resource can be long-term locatable (dereferenceable).

The **Real-Time Service (RTS)** provides the ability to synchronize data across multiple users in (near) real time. It uses a modern approach to conflict resolution (Nicolaescu et al., 2016) that works in both centralized and decentralized (peer-to-peer) collaborative environments. Our implementation also supports the continuation of work when the network connection is lost, and the changes can be synchronized later in a conflict-free manner.

Authentication is outsourced to an **Authentication Server (AS)**.

In the next section the first client application which is implemented using this platform is presented.

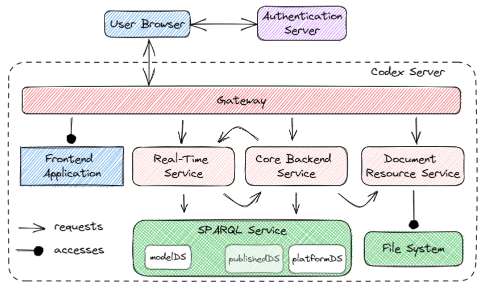
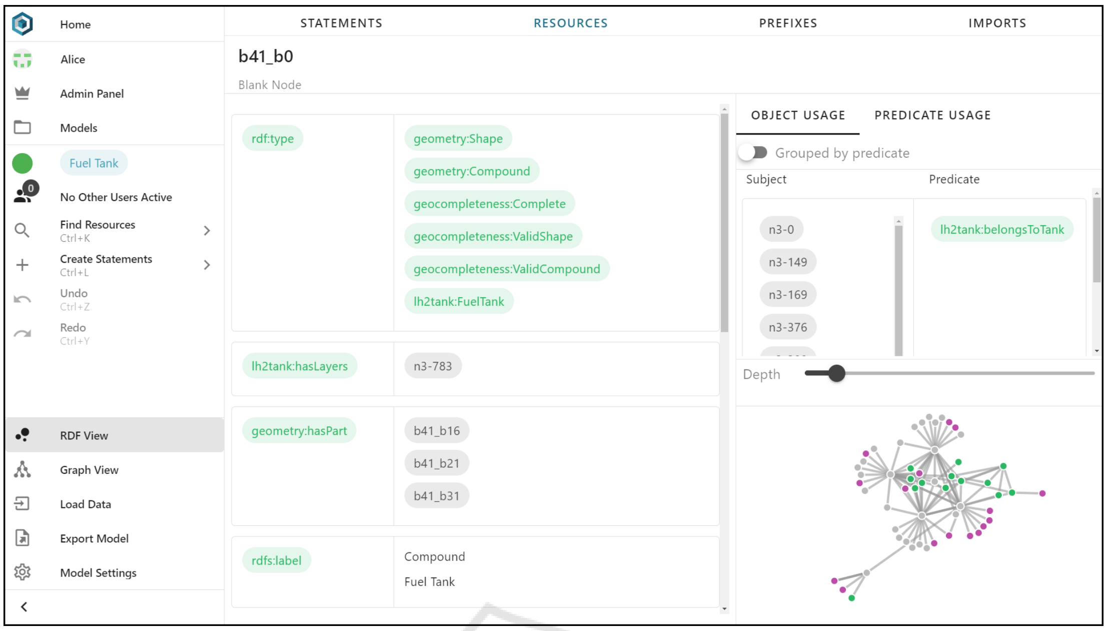


Figure 4: Codex Server Architecture.



b41_b0

Blank Node

rdf:type	geometry:Shape geometry:Compound geocompleteness:Complete geocompleteness:ValidShape geocompleteness:ValidCompound lh2tank:FuelTank
lh2tank:hasLayers	n3-783
geometry:hasPart	b41_b16 b41_b21 b41_b31
rdfs:label	Compound Fuel Tank

Subject	Predicate
n3-0 n3-149 n3-169 n3-376	lh2tank:belongsToTank

Figure 5: Resource view of the web application.

3.3 A User Interface for Knowledge Modelling

As a first client implementation a Knowledge Modelling Interface for RDF and OWL models is currently being developed. It serves as a proof of concept for the platform and its extensibility, and also as the base for the migration of the additional KBE features described in Section 2. The client's scope is reduced in comparison to the Codex framework and so far, covers the modelling aspect and some client specific features (user settings, user feedback and admin panel).

The web interface shown in Figure 5 allows users to create and manage their models and the corresponding access control. To store additional information about the user's model, the client actually requests two models to be created on the platform, one for the user's model data and the other one containing meta data on the model. This is one of the advantages of having a simple open platform, it does not force clients into a predefined structure beyond the few simple implications of RDF, but allows them to define their own custom data structure in a semantic way. If desired a client specific meta-model ontology can be defined and also stored on the platform either privately or openly to share it with others.

One of the most important features for the client is the capability for users to collaborate with each other on a model in real-time. Therefore, updates to the model are exchanged between all users currently "connected" to the model in real-time. To raise awareness on which users are currently connected to a model, this information is indicated by the client on the left side of the view as shown in Figure 5. Users connected to the same model are shown by their user icons. OWL imports play a special role for models since they provide a mechanism to link different models to each other. It allows for simple reuse of models and is also important for integrating between different models. Due to this special role the client provides a specialized interface for managing model imports. In the model settings, users with manage permissions can manage access control for all users.

4 CONCLUSIONS & OUTLOOK

"Scientists are primarily interested in doing science, not software." (Kelly and Sanders, 2008). In engineering research environments, this inherent fact can be linked to cases of corporate amnesia, implying a loss of organizational knowledge. In this paper, the discussion centers around why engineering research organizations are particularly affected by corporate amnesia, and it is demonstrated how technical solutions such as SWT can be leveraged to preserve organizational knowledge While the field of knowledge management is wide, this research focuses on the domain of KBE. As illustrated in Section 2, previous efforts have shown the value of SWT for the development of KBE applications. It enhances the reusability of knowledge, simplifies extensibility, and provides the flexibility to choose inputs and outputs, resulting in more versatile applications. In this respect, the Codex framework has generally been proven valuable. However, experience has shown that the barrier to using the Codex framework is currently too high for widespread adoption by scientists. To broaden the accessibility of technological solutions to a larger user group, a decision was made to develop a web-based application using the current framework. This application offers a collaborative user interface that enables non-experts in SWT to engage in effective knowledge collaboration.

The platform concept, as discussed in Section 3.1, and its implementation (covered in Section 3.2), serve as a foundation for a wide range of applications. While the usability of the platform is demonstrated through the implementation of the web application for knowledge modeling in Section 3.3, it is yet to be determined whether the goal of achieving widespread adoption can be realized. This depends on the effort required for modeling domain knowledge on the platform, and whether the Return on Investment (ROI) justifies this effort. The answer to this question can only be provided once additional KBE features and domain-specific interfaces for engineering are developed and tested. Therefore, after finalizing the publication of models, the KBE related services will be addressed and additional user interfaces are going to be developed. Meanwhile, even in the absence of those specific KBE features, the platform will be utilized for knowledge capturing and integration tasks. This way, it aims to prevent corporate amnesia and instead foster a robust corporate memory.

REFERENCES

Alder, M., Moerland, E., Jepsen, J., and Nagel, B. (2020). Recent Advances in Establishing a Common Language for Aircraft Design with CPACS. In *Aerospace Europe Conference 2020*.

Boden, B., Cabac, Y., Burschyk, T., and Nagel, B. (2022). Rule-based Verification of a Geometric Design using the Codex Framework. In *33rd ICAS*.

Burschyk, T., Cabac, Y., Silberhorn, D., Boden, B., and Nagel, B. (2021). Liquid hydrogen storage design trades for a short-range aircraft concept. In *DLRK 2021*.

Bölling, M. (2015). *Lösungspfadbasierte Analysen im Entwurf komplexer Systeme*. PhD thesis, University of Stuttgart.

Dalkir, K. (2005). *Knowledge Management in Theory and Practice*. Elsevier.

International Organization for Standardization [ISO] (1994). Information technology — Open Systems Interconnection — Basic Reference Model: The Basic Model. ISO/IEC 7498-1, ISO.

Internet Engineering Task Force [IETF] (1981). Transmission Control Protocol. Technical Report RFC 793, Internet Engineering Task Force.

Kelly, D. and Sanders, R. (2008). Assessing the Quality of Scientific Software. First International Workshop on Software Engineering for Computational Science and Engineering, Leipzig Germany, May 2008.

La Rocca, G. (2012). Knowledge based engineering: Between AI and CAD. Review of a language based technology to support engineering design. *Advanced Engineering Informatics: the science of supporting knowledge-intensive activities*, 2012(26):159–179.

Li, M. and Gao, F. (2003). Why Nonaka highlights tacit knowledge: a critical review. *Journal of Knowledge Management*, 7(4):6–14.

Musen, M. A. (2015). The protégé project: a look back and a look forward. *AI Matters*, 1(4):4–12.

Nicolaescu, P., Jahns, K., Derntl, M., and Klamma, R. (2016). Near Real-Time Peer-to-Peer Shared Editing on Extensible Data Types. In *Proceedings of the 2016 ACM International Conference on Supporting Group Work*, page 39–49, New York, NY, USA.

Nonaka, I. and Takeuchi, H. (1995). *The knowledge-creating company: How Japanese companies create the dynamics of innovation*. Oxford University Press, New York.

Nonaka, I. and Takeuchi, H. (2021). Humanizing strategy. *Long Range Planning*, 54(4):102070.

Red Hat (2021). Drools Documentation 7.49.0.Final. Available at: https://docs.drools.org/7.49.0.Final/drools-docs/html_single/index.html. Accessed: 20 June 2023.

Verhagen, W. J., Bermell-Garcia, P., van Dijk, R. E., and Curran, R. (2012). A critical review of Knowledge-Based Engineering: An identification of research challenges. *Advanced Engineering Informatics*, 26(1):5–15.

World Wide Web Consortium [W3C] (2012). OWL 2 Web Ontology Language Profiles (Second Edition). W3C Recommendation, World Wide Web Consortium.

Zamboni, J., Zamfir, A., and Moerland, E. (2020). Semantic Knowledge-Based-Engineering: The Codex Framework. In *12th IC3K*, volume 2, pages 242–249.

Zamboni, J., Zamfir, A., Moerland, E., and Nagel, B. (2022). A semantic knowledge based engineering framework for the rapid generation of novel air vehicle configurations. In *33rd ICAS*.