Semantic Knowledge-Based-Engineering: The Codex Framework

J. Zamboni[a], A. Zamfir[b] and E. Moerland[c]
Institute of System Architectures in Aeronautics, German Aerospace Center, Hamburg, Germany

Keywords: Knowledge-Based Engineering, Object-Oriented Programming, Semantic Web Technologies, Collaboration, Complex-system Development, Collaborative Engineering.

Abstract: The development of complex systems within multi-domain environments requires an effective way of capturing, sharing and integrating knowledge of the involved experts. Modern Knowledge-Based Engineering (KBE) systems fulfill this function, making formalized knowledge executable by using highly specialized environments and languages. However, the dedication of these environments to their domain of application poses limitations on the cross-domain integration of KBE applications. The use of Semantic Web Technologies (SWT) delivers a domain-neutral way of knowledge formalization and data integration which promises to drastically reduce the effort required to integrate knowledge of multiple domains in a single representation. Especially within the complex field of aeronautical vehicle design the authors are working in, characterized by several individual disciplines having to be considered simultaneously, the combined usage of KBE and SWT technologies seems an attractive approach for the continued digitalization of the design process. In this paper, the COllaborative DEsign and eXploration (Codex) framework is presented which aims at merging these two technologies into a single framework that can be used to create domain-specific knowledge-bases and integrate these into a single model of the overall product. Formalizing and executing this model will lead to a more transparent and integrated view on complex product design.

1 INTRODUCTION

Designing the next generation of aeronautical vehicles requires a system of systems approach in which a large amount of disciplinary domains need to be integrated in an overarching product development process. Due to the current trend towards digitalization of processes, an increasing amount of data is generated during the design of these complex products. This data must be aggregated and processed into comprehensible information before decisions can be based upon it. These often manually executed tasks place an increasing burden on the low number of experts available, in particular when the processes are coupled and the tasks have to be repeated many times. The amount of data and ever-changing landscape of requirements and decisions can also lead to the emergence of errors and inconsistencies among the different models.

Knowledge-Based Systems (KBS) are developed with the aim of capturing and formalizing the experts' knowledge so that the data processing, requirements analysis, and decision making can be (partially) automated by using a computer for their execution (Milton, 2008). A KBS automatically propagates user decisions to the relevant requirements, components, and parameters, cascading these changes and re-validating the model. In this way, true concurrent engineering in which several disciplinary analysis can be performed at the same time becomes possible. These approaches promise to drastically reduce the time between each design iteration, ensure consistency with the formalized experts' knowledge and free up resources to assess the impact of specific decisions on the results.

In literature (La Rocca, 2012), Knowledge-Based Engineering (KBE) is defined as the merging of different disciplines such as Artificial Intelligence, Computer Aided Design and Object-Oriented Programming (OOP). It can be considered as an extension of KBS with a stronger focus on tasks that have a large impact on the engineering design process, namely solving computational problems, dealing with geometries and configurations. KBE application development is still a fairly new discipline and it has seen a limited use only in specific engineering sectors such

[a] https://orcid.org/0000-0003-3207-4138
[b] https://orcid.org/0000-0001-7579-9628
[c] https://orcid.org/0000-0003-4818-936X

Zamboni, J., Zamfir, A. and Moerland, E.
Semantic Knowledge-Based-Engineering: The Codex Framework.
DOI: 10.5220/0010143202420249
In *Proceedings of the 12th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2020) - Volume 2: KEOD*, pages 242-249
ISBN: 978-989-758-474-9

as aerospace and automotive. This can be traced to the fact that these specific sectors deal with the design of safety critical and complex systems that need to adopt new technologies to remain competitive. All this while under the pressure to use less resources, to shorten the time-to-market of the products, and to satisfy a large and continuously evolving set of requirements. KBE proves to be a good methodology for front-loading knowledge in the design process, thereby creating a better foundation for design-decisions that have to be made early in the product life-cycle, which often have significant influence on the product performance (Kulkarni et al., 2017).

Managing knowledge in multi-domain environments, such as product development, requires a collaborative framework that is well aligned with the knowledge engineering process. Creating such a framework for knowledge representation is one of the fundamental aspects in designing a KBE application (Sanya and Shehab, 2014). The KBE tools available on the market all adopt OOP as the core modeling approach. We argue that OOP has its limitations when it comes to fully capturing and integrating experts' knowledge as it limits possibility for meta-modeling to frame-based structures. Our research proposes a novel approach to knowledge formalization in KBE applications, in which a more generic knowledge structure using Semantic Web Technologies (SWT) (World Wide Web Consortium, 2015) is adopted in an effort to remove these limitations.

The Codex framework is currently being developed at the German Aerospace Center (DLR), aiming to overcome a large part of the aforementioned burdens. It targets to ease the creation of knowledge-based engineering tools, which can be easily integrated in a digitally coupled overarching product development process for aeronautical vehicle design. Codex is a continued development of the *model generator* (Zamfir et al., 2018) and it aims to improve the accessibility, extendability, and ease of cross-domain knowledge reuse.

The following section will highlight the challenges of collaboration in multi-domain environments and present possible solution to these. Section 3 focuses on human-machine interaction, the importance of the meta-modelling environment and its impact on knowledge formalization. Thereafter, section 4 will present the current state of the Codex framework and provide an example of multi-domain integration. This paper concluded with a summary and an outlook on future goals of this framework.

2 MULTI-DOMAIN COLLABORATION

One reason for the aforementioned low adoption of the KBE methodology can be be found in the large difference between the abstract, high-level language used in these applications compared to the highly specific domain languages and tools generally used within each discipline. Specific domain languages and environments allow for a high level of expressiveness, since the semantics are tailored to enable very precise modeling while using a set of vocabulary and concepts that the domain-expert is already familiar with. In contrast, to allow for integration and high-degree linking of knowledge from all stakeholders within a coherent knowledge base, a very abstract modeling language is required, which makes no assumptions on the domain it is used for and therefore features low expressiveness for specific domains. Therefore, the choice of the modeling language for a collaborative KBE application requires a trade-off between applicability to different domains (abstraction/generality of the language) and expressiveness within the domain itself.

Two examples of KBE tools used in the aerospace sector are *Pacelab APD* (TXT Group, 2020) and *ParaPy* (ParaPy B.V., 2019). The choice of modeling languages are *C#* for APD and *Python* with a specific flavor of annotations for Parapy, which makes knowledge-formalization straight-forward for programmers with experience in these languages. In both tools the meta-model is defined in a frame-like structure (Minsky, 1974), resulting from the necessity to create the model in the particular object-oriented programming language used by the application. Moreover, the model is defined in a hierarchical way as this structure is commonly used when modeling a product and its sub-parts within the engineering domain. Describing relationships among parameters from the same discipline or in the same system of components is fairly easy thanks to the inherent connections of this type of structure (e.g. parent and child relationships).

An issue that may arise from a frame-based and hierarchical structured knowledge capture approach is that higher complexity, especially via cross-branch connections, tends to increase rigidity of the knowledge base and makes it more complicated to use. This over-coupling diminishes the potential of knowledge re-usability since the coupled knowledge is not encapsulated anymore and its usage introduces many more (unwanted) dependencies. In fact, the hierarchical structure often does not even reflect the naturally emerging structure of collaboration, which resembles

a graph where many cross-branch connections exist (Alexander, 1965). This unconstrained structure yields a much higher degree of possible complexity in designing knowledge while decreasing the friction of cross-domain knowledge transfer.

Although this frame-like and hierarchical way of modeling has been proven to be effective in practice, within the Codex framework we choose a modeling approach that focuses on avoiding the emergence of conflicts that commonly arise in multi-domain environments. This approach is based on semantically expressing the precise meaning between models of different domains, which we assume to be key to create a highly collaborative KBE application.

SWT have the potential to tackle some of the challenges just mentioned. The models created using SWT are based on the formal semantics of RDF Schema (RDFS) and Web Ontology Language (OWL) (World Wide Web Consortium, 2015) which are unambiguous, precise, can be automatically validated for consistency and can be understood by experts from other domains. The increased meta-modeling capabilities allowed by these formal semantics enable the integration of common knowledge shared by multiple domains while still persisting all domain-specific interpretations.

Codex utilizes a hybrid modeling approach where the internal knowledge representation is as generic as possible (using SWT), while the modeling interface used by the engineer is as close as possible to his/her Domain-Specific Language (DSL) (i.e. the language commonly used within that specific domain). Examples of such DSLs within the field of aerospace are the Systems-Modeling-Language (SysML) (Object Management Group, 2020) and Common-Parametric-Aircraft-Configuration-Schema (CPACS) (Alder et al., 2020). Codex is able to translate these (and several other) DSLs to a generic knowledge representation, allowing different disciplines to unambiguously link parts of their knowledge – effectively creating an inter-domain knowledge representation of the overall product. The following section will elaborate on this aspect.

As depicted in Fig. 1, this mechanism enables coupling multiple disciplines across the entire lifecycle of an aircraft configuration. Already in the early stages of development, this coupling is automatically managed by Codex and takes away the burden of having to manually communicate inter-domain dependencies. This mechanism ultimately results in the availability of more time for creative, value-adding design tasks to be performed by the engineers and largely reduces the amount of mistakes made in interdisciplinary communication.

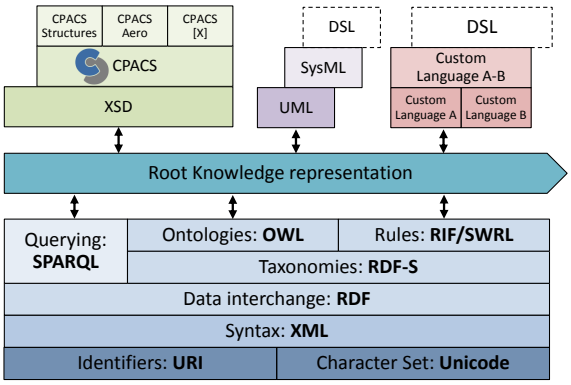


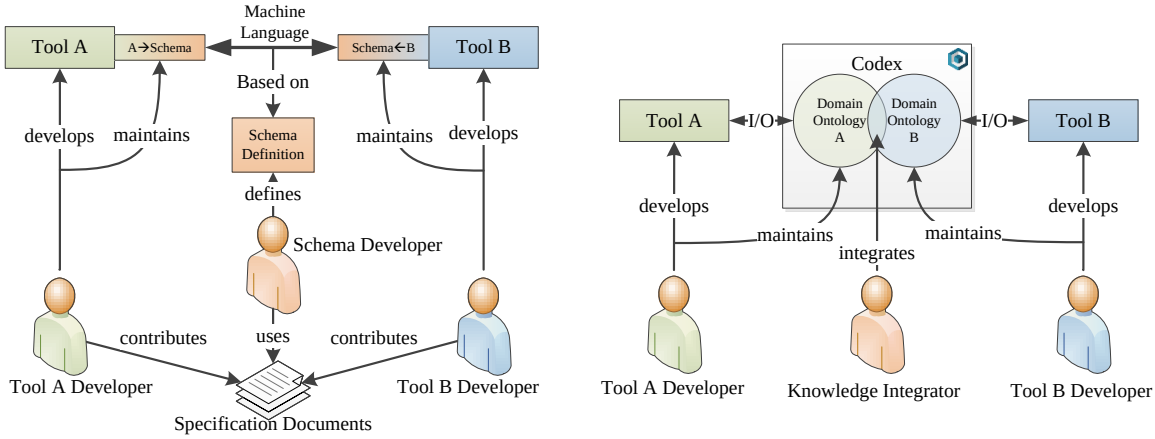
Figure 1: Coupling knowledge of different aerospace disciplines through language layers in Codex.

3 CHALLENGES IN MODELING FOR KBE APPLICATIONS

The following section highlights the challenges that arise when formalizing knowledge in KBE applications by first focusing on the interaction between humans and machines. Thereafter, the importance of the meta-modeling environment is discussed by comparing frame-based structures with a knowledge graph approach.

3.1 Human-machine Interaction

Effective collaboration among machines or software that should produce deterministic results demands a fixed communication protocol. Changes to that protocol often require making changes to the machines and software as well. When multiple tools/knowledge bases must be integrated, the problem of language barriers arises from the nature of having multiple domains, each with their own DSL. Each DSL is expressed through different software-systems, file-formats, and ontologies. When two or more domains have to share knowledge (collaborate), they often express the same (or at least similar) meaning of this shared subset of knowledge in their own DSL. To then integrate knowledge from a domain *A* and another domain *B* a Model-to-Model Transformation (M2M) needs to take place, which could manifest itself as a file-based export into a common format understood by both *A* and *B* or expressing the semantic meaning between *A* and *B* via ontological statements. The first method of integration is most commonly used but does not scale well with the number of domains involved (n), since for every additional domain the



(a) Human-machine protocol definition using a schema.

(b) Human-machine protocol definition using Codex.

Figure 2: Comparison of legacy and Codex approaches.

number of potentially necessary M2M grows exponentially $(n * (n-1))$.

A solution to this problem is to extend the common format to all the involved domains. Such an approach has been taken within the aerospace community with CPACS, which was initiated by the German Aerospace Center and is nowadays continually developed as an open-source format within the aerospace community. This schema is required to include all knowledge that must be shared by at least two involved domains and forces these domains to agree on how this shared knowledge is expressed. This makes integration of tools within a multi-disciplinary design optimization workflow easier. Some successful implementations of this central data exchange format approach for multi-domain and cross-company collaboration have been shown in (Moerland, E., Pfeiffer, T., Böhnke, D. et al., 2017) and (Ciampa and Nagel, 2018). Utilizing a central schema results in a positive network-effect – the usefulness of the schema and the tools that utilize it increases with the number of domains and people participating.

This removes the M2M scaling issues, but often lacks reflection of the aforementioned graph structure of collaboration. This process demands the tool developer to provide a mapping between the common exchange language and the internal application model. In fact, effective collaboration among people demands a flexible communication protocol that can capture subtle semantic differences in meaning, allows for a variety of meanings about concepts to coexist, and supports the distributed nature of knowledge acquisition by enabling knowledge engineers to adjust their models at any time without having to know all or any of the dependencies on their model.

Moreover, the fixed structure of a schema, together with the high number of domain dependencies, makes modifications to it a labor intensive task, especially in a collaborative environment. This often results in a reluctance to introduce changes and therefore decreases development speed. These problems intensify with the number of domains and participants involved, which counteracts the positive network-effect – the more domains and people make use of the schema, the more rigid it becomes.

Figure 2a depicts the legacy approach, in which the communications protocol between people is based on natural language (or specification documents written in natural language) and the communication protocol for the machines (tools) needs to be manually derived from these specification documents by the schema developer.

With Codex we aim to provide the collaborators with a framework to add semantics to their exchange protocol and the tools with an application programming interface (API) to communicate their semantic model to the multi-domain environment, as shown in Figure 2b. These enhancements add formal meaning to the protocol and remove the necessity for a central schema while still allowing all tools to communicate. This also removes the previously mentioned negative network effect, since the labor intensive task of continuously keeping the schema-based inter-tool communication protocols updated, has been simplified. The task is now handled by the knowledge-integrator that can independently define the initial shared meaning between the different domain ontologies by using the formal semantics of RDFS and OWL. This shared meaning can be updated and modified without the need for the different domains to agree on a common representation, effectively creating less communication-overhead. Manual work is still re-

quired from the tool developer to maintain the semantic model of his/her tool whenever the tool internals are modified.

3.2 Meta-modeling Environment

In the OOP paradigm the class frame takes a central role as other constructs such as objects, methods and values cannot exist outside of their class. Information in one class can be re-used in others by inheritance and changes made to the classes are reflected in the objects instantiated from them. If a class is modified, all objects that depend on it become invalid. This strict separation of meta-model and model is resolved in OOP languages by ”freezing” the former while the latter is being built and used, thus ensuring that issues due to meta-model incongruities cannot arise during run-time. This is usually done by compiling the classes defined in the source code.

In KBE tools based on OOP languages the strict separation of meta-model and model is implicitly taken and becomes part of how the knowledge engineer interfaces with the knowledge base. A positive result of having the meta-model in compiled form is that performance during the knowledge execution phase can be improved. On the other hand, a direct drawback of this choice is that it takes time to compile the meta-model and switch to the modeling tasks. This can negatively impact the experience of the knowledge engineer as he/she would not be aware of the impact of the changes on the model until the time-intensive process needed to make such changes available is finished; in other words, rapid iteration of the KBE system is limited and the expert is encouraged to make as many changes as possible before compiling and assessing the changes. Furthermore, having to compile the meta-model before it can be used makes it less practical to have a KBE application deployed on the web, which is a major advantage when it comes to collaboration and adoption and one of the future goals of the Codex framework.

In SWT standards every entity is a resource with equal importance. Classes, individuals, object- and data-properties can be linked together but their existence is not dependent on the presence of a specific entity, as is the case for objects and classes in OOP. Moreover, there is no strict separation between classes and individuals as, depending on the statements made, a resource can be both. For example, an expert could see an A320 as an individual of type Aircraft while a different (and not conflicting) view of another expert is that A320 is a class for a family of aircraft that shares a specific set of basic properties. Thus A320 can be simultaneously a class and an individual depending on the specific point of view.

We believe that the proposed approach within Codex can improve knowledge re-usability when compared to OOP based KBE tools. One reason for this is that only the resources that are actually needed can be re-used, leaving out unnecessary information that might actually overcomplicate the derived knowledge base. Another reason for this improvement can be found in the ability to freely mix classes and individuals; any model can become the starting point for a new meta-model without much overhead. Friction of implementing new knowledge is also decreased as the engineer does not need to worry about having to model his/her ideas in a strictly hierarchical language.

In Codex we adopt more flexible modeling capabilities allowed by SWT when building knowledge bases. Doing so makes the boundary between meta-model and model blurred, when compared to the OOP approach, leading us to decide not to compile any of the user-defined knowledge. In this way, the expert is allowed to modify and extend the meta-model and model simultaneously during run-time. This choice should allow for a faster iteration of knowledge formalization leading to a higher speed of innovation. Deployment of the complete knowledge base on the web becomes also possible allowing multiple users to collaborate concurrently on the same project.

4 THE CODEX FRAMEWORK

With the Codex framework we provide an implementation of a KBE application where semantic knowledge formalization plays a central role. The framework currently consists of several modules, that allow the different domain experts to solve many of their knowledge-engineering tasks in a holistic way. Codex is written in the *Kotlin programming language* (Kotlin Foundation, 2020), as it has the ability to leverage the syntax of the language to create so-called *type-safe-builders*, which are a way of creating custom DSLs within the language. This allows for the provision of an implementation of the aforementioned hybrid approach to knowledge formalization not just to users of our Graphical User Interface (GUI), but also to domain experts that like to express their knowledge in code, as well as developers of plugins for the Codex-framework.

Within Codex, functionalities are provided in the form of plugins, each possibly providing its own DSL and ontology. The core module of the framework is the *codex-semantic* module that makes use of existing and well-established standards for knowledge representation (Resource Description Frame-

work (RDF) (World Wide Web Consortium, 2015), OWL) and utilizes mature open-source libraries, such as those provided by the *Apache Jena Framework* (Apache Software Foundation, 2020). This module takes care of the creation and manipulation of OWL-models.

The modular plugin architecture of Codex also allows for existing tools, languages, and technologies to be connected to it. A plugin that has already been implemented is the *codex-parametric* module that allows users to semantically define and solve systems of parametric constraints with a DSL that closely resembles common mathematical notation. This plugin contains an execution engine that provides information about parameter dependency (e.g. over/under determination of the system of constraints) and solving capabilities.

Another plugin that is currently under development provides the ability to import and export models defined using the aforementioned CPACS schema. It also exposes this schema as an OWL ontology and additionally provides a custom DSL that can be used by engineers already familiar with CPACS to create and modify models. This allows Codex to seamlessly integrate with the already existing, vast eco-system of CPACS-enabled tools.

The knowledge asserted by the domain expert can easily be enhanced by simply adding the formal-semantics of RDFS and OWL as a set of rules that will infer additional knowledge. Moreover, the user can utilize this same mechanism (and the provided Rule-DSL) to add custom rules to the knowledge-base. While rules expressed through this mechanism are sufficient for semantic-inferences, they do not cover the definition of production-rules (Russell and Norvig, 2016), that demand additional expressions to be possible, such as the definition of a priority of rule-execution. These production-rules are an essential part of any KBE application, since they allow for the dynamic, rule-based creation of a large number of interconnected individuals. To satisfy this need, a dedicated module that allows such production-rules to be conveniently expressed via a custom DSL and executed using state-of-the-art algorithms and libraries is being developed. These user-defined rules will not be stored within the OWL model, but a more appropriate standard such as the *Rules-Interchange-Format* (World Wide Web Consortium, 2013).

4.1 Multi-domain Integration Example

In this subsection, a simplified example of how multiple domains can be integrated with the help of Codex is discussed. The following provides some examples of parameters and rules present within multiple aerospace design domains, each represented by their individual namespace:

a: Aircraft Sizing Domain.

$$(a:Wing \quad rdf:type \quad owl:Class)$$

m: Mission Analysis Domain.

$$R \cdot \frac{g \cdot TSFC}{V} \cdot \frac{C_D}{C_L} = -ln(\frac{M_{end}}{M_{init}})$$
$$M_{end} = M_{init} - M_{fuel}$$
or
$$M_{fuel} = Mission\_Tool(R, C_L, C_D, TSFC, \dots)$$

t: Fuel-tank Sizing Domain.

$$Fuel\_Density = \frac{Fuel\_Mass}{Fuel\_Volume}$$
$$Tank\_Volume = Fuel\_Volume \cdot Factor$$

r: Requirements Domain.

$$Range \geq 6150km$$
"All fuel must be stored in the wing"

Through the use of the *codex-parametric* module's DSL and ontology, the above domains' knowledge can be captured; for example, an equation is modelled as an expression tree where each node is either of type operator, function (e.g. *sin(x)*), parameter, or constant. This expression tree is stored as a RDF graph. External parametric tools, such as "Mission_Tool", can be included in the knowledge graph as they behave like functions having a local name within the ontology, requiring input parameters and providing an output. During modeling, either of the definitions of $m:M_{fuel}$ (equation or external tool call) can be utilized to compute its value, as the constraints in *codex-parametric* are automatically reversed by use of numerical convergers.

During the knowledge integration phase, parameter equivalency between different domains can be quickly achieved with RDF statements such as:
$(m:M_{fuel} \quad owl:sameAs \quad t:Fuel\_Mass)$
$(m:R \quad owl:sameAs \quad r:Range)$.
This ensures that the two parameters are treated as the same entity by the execution engine, and consequently their calculated values will be equivalent.

Additionally, knowledge expressed through natural language, such as the requirement that "all fuel must be stored in the wing", can be integrated by RDF statements such as:
$(r:fuel \quad r:storedIn \quad r:wing)$
$(r:fuel \quad owl:sameAs \quad t:Fuel\_Volume)$
$(r:wing \quad rdf:type \quad a:Wing)$.

This example is visually represented in Figure 3 where the unconnected knowledge graphs are meaningfully integrated through the use of semantic statements. With this example we show that a schema is not necessary to bridge the gap between the different

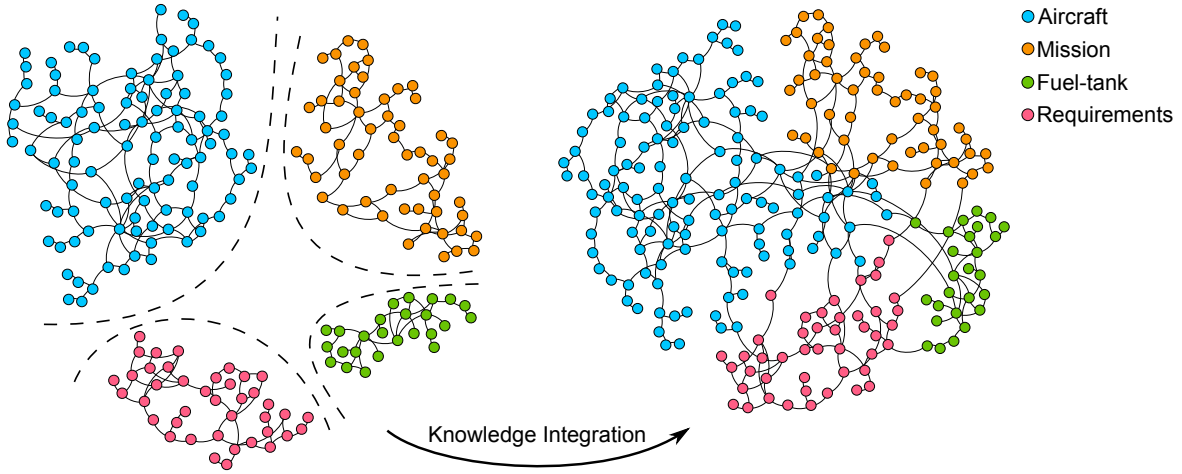


Figure 3: Multi-domain knowledge integration.

domains. Maintenance is only necessary if a change within a domain alters the meaning of the integrating statements.

5 CONCLUSIONS

Utilizing formal knowledge definitions in the engineering domain enables making the expert's knowledge reusable, modular, and machine executable. However, the reliance of KBE on OOP introduces limitations that might be a cause for the low adoption rate of this methodology. The choice of using a generic language (OWL) that makes no assumptions on the domain it is being applied to, enables a way to include other more specific approaches to system engineering such as the widely adopted SysML. Codex aims to augment these approaches by providing the ability to apply custom rules to these models, effectively making them executable. Moreover, it is one of our goals to leverage the inference mechanism combined with the model integration capability to infer new knowledge from the models described through these well established approaches.

The overall aim of the Codex framework is to provide a holistic knowledge-formalization and execution environment that reflects and supports the graph structure of an effective collaborative environment. Although Codex is at an early stage of development and the main expected improvements have yet to be validated, the framework can already be utilized to define, link, and use ontologies for the creation of executable KBE tools. A possible obstacle to its adoption is that semantic modeling might prove to be difficult for the experts involved in the product development process and the benefits do not outweigh the required learning curve. Another pitfall could arise when users do not follow semantic modeling best practices or specialize their ontologies too much, lowering their potential for re-usage within other domains.

The Codex framework is currently being developed to create a framework for knowledge digitalization and integration for usage within aeronautical vehicle design. When the framework is fully matured and has proved its usage within the design process, the opportunity arises to further open-up the design space by incorporating a larger amount of engineering and scientific domains, e.g. from other means of transport. With this, the basis is created for the overall design of more efficient, fully integrated mobility solutions.

Future developments include further integration with existing standards and tools, implementation of a rule-based production system, and the creation of a data-analytics and visualization environment that leverages the data-integration and exploration capabilities of SWT.

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, Bordeaux*.

Alexander, C. (1965). A city is not a tree. *Architectural Forum, 1965*, 122.

Apache Software Foundation (2020). Apache Jena. https://jena.apache.org/. Online; accessed Jul 27, 2020.

Ciampa, P. D. and Nagel, B. (2018). AGILE the Next Generation of Collaborative MDO: Achievements and

Open Challenges. In *2018 Multidisciplinary Analysis and Optimization Conference*. American Institute of Aeronautics and Astronautics.

Kotlin Foundation (2020). Kotlin Programming Language. https://kotlinlang.org/. Online; accessed Jul 27, 2020.

Kulkarni, A., van Dijk, R., van den Berg, T., and Rocca, G. (2017). A knowledge based engineering tool to support front-loading and multi-disciplinary design optimization of the fin-rudder interface. In *Aerospace Europe 6th CEAS Conference*, volume 680.

La Rocca, G. (2012). Knowledge based engineering: Between AI and CAD. Review of a language based technology to support engineering design. *Advanced Engineering Informatics*, 26(2):159–179.

Milton, N. (2008). *Knowledge Technologies*. Polimetrica, Monza IT.

Minsky, M. (1974). A framework for representing knowledge. *MIT AI Memo 306*.

Moerland, E., Pfeiffer, T., Böhnke, D. et al. (2017). On the design of a strut-braced wing configuration in a collaborative design environment. In *17th AIAA Aviation Technology, Integration, and Operations Conference*, page 4397.

Object Management Group (2020). OMG Systems Modeling Language (OMG SysML). https://www.omg.org/spec/SysML/. Online; accessed Jul 27, 2020.

ParaPy B.V. (2019). ParaPy - Knowledge Based Engineering Platform. Online; accessed Jul 27, 2020.

Russell, S. and Norvig, P. (2016). *Artificial Intelligence: A Modern Approach, Global Edition*. Pearson Education Limited.

Sanya, I. and Shehab, E. (2014). An ontology framework for developing platform-independent knowledge-based engineering systems in the aerospace industry. *International Journal of Production Research*, 52(20):6192–6215.

TXT Group (2020). Pacelab APD. https://www.txtgroup.com/markets/solutions/pacelab-apd/. Online; accessed Jul 27, 2020.

World Wide Web Consortium (2013). RIF Production Rule Dialect (Second Edition). https://www.w3.org/TR/rif-prd/. Online; accessed Jul 27, 2020.

World Wide Web Consortium (2015). Semantic Web. https://www.w3.org/standards/semanticweb/. Online; accessed Jul 27, 2020.

Zamfir, A., Jepsen, J., Moerland, E., and Nagel, B. (2018). Development of a modular knowledge-based model generator for the preliminary aircraft design process of the future. In *2018 Aviation Technology, Integration, and Operations Conference*. American Institute of Aeronautics and Astronautics.