

Artificial Intelligence for the Early Design Phases of Space Missions

Audrey Berquand
University of Strathclyde
James Weir Building,
75 Montrose Street
G1 1XJ Glasgow, UK
audrey.berquand@strath.ac.uk

Francesco Murdaca
University of Strathclyde
James Weir Building,
75 Montrose Street
G1 1XJ Glasgow, UK
francesco.murdaca@strath.ac.uk

Annalisa Riccardi
University of Strathclyde
James Weir Building,
75 Montrose Street
G1 1XJ Glasgow, UK
annalisa.riccardi@strath.ac.uk

Tiago Soares
European Space Agency,
Noordwijk, The Netherlands
tiago.soares@esa.int

Sam Generé
RHEA group,
Leiden, the Netherlands,
s.generé@rheagroup.com

Norbert Brauer
AIRBUS,
Bremen, Germany,
norbert.brauer@airbus.com

Kartik Kumar
Spacejunkies V.O.F (satsearch),
Noordwijk, the Netherlands,
kartik@satsearch.co

Abstract—Recent introduction of data mining methods has led to a paradigm shift in the way we can analyze space data. This paper demonstrates that Artificial Intelligence (AI), and especially the field of Knowledge Representation and Reasoning (KRR), could also be successfully employed at the start of the space mission life cycle via an Expert System (ES) used as a Design Engineering Assistant (DEA). An ES is an AI-based agent used to solve complex problems in particular fields. There are many examples of ES being successfully implemented in the aeronautical, agricultural, legal or medical fields. Applied to space mission design, and in particular, in the context of concurrent engineering sessions, an ES could serve as a knowledge engine and support the generation of the initial design inputs, provide easy and quick access to previous design decisions or push to explore new design options. Integrated to the User design environment, the DEA could become an active assistant following the design iterations and flagging model inconsistencies.

Today, for space missions design, experts apply methods of concurrent engineering and Model-Based System Engineering, relying both on their implicit knowledge (i.e., past experiences, network) and on available explicit knowledge (i.e., past reports, publications, data sheets). The former knowledge type represents still the most significant amount of data, mostly unstructured, non-digital or digital data of various legacy formats. Searching for information through this data is highly time-consuming. A solution is to convert this data into structured data to be stored into a Knowledge Graph (KG) that can be traversed by an inference engine to provide reasoning and deductions on its nodes. Knowledge is extracted from the KG via a User Interface (UI) and a query engine providing reliable and relevant knowledge summaries to the Human experts. The DEA project aims to enhance the productivity of experts by providing them with new insights into a large amount of data accumulated in the field of space mission design. Natural

Language Processing (NLP), Machine Learning (ML), Knowledge Management (KM) and Human-Machine Interaction (HMI) methods are leveraged to develop the DEA. Building the knowledge base manually is subjective, time-consuming, laborious and error bound. This is why the knowledge base generation and population rely on Ontology Learning (OL) methods. This OL approach follows a modified model of the Ontology Layer Cake. This paper describes the approach and the parameters used for the qualitative trade-off for the selection of the software to be adopted in the architecture of the ES. The study also displays the first results of the multi-word extraction and highlights the importance of Word Sense Disambiguation for the identification of synonyms in the context. This paper includes the detailed software architecture of both front and back-ends, as well as the tool requirements. Both architectures and requirements were refined after a set of interviews with experts from the European Space Agency. The paper finally presents the preliminary strategy to quantify and mitigate uncertainties within the ES.

TABLE OF CONTENTS

1. INTRODUCTION	2
2. THE DESIGN ENGINEERING ASSISTANT	2
3. TECHNOLOGY TRADE-OFF	6
4. SOFTWARE ARCHITECTURE.....	10
5. PRELIMINARY RESULTS	17
6. CONCLUSION.....	18
APPENDIX – VISUALISER OUTPUTS	18
ACKNOWLEDGMENTS.....	18
REFERENCES	19
BIOGRAPHY.....	20

1. INTRODUCTION

Knowledge of space mission design is a broad term encompassing a dozen subdomains involved in the spacecraft design and data accumulated since the 1950s, including many textbooks, reports or data in various formats. Most of the information is stored as unstructured data and has today become too time-consuming to search through. [13] introduces the concept of "corporate amnesia" by which an entity risks losing knowledge and eventually efficiency and money due to poor Knowledge Management (KM). The timeframe of a spacecraft feasibility study being limited, it is even more critical to ensure that experts have smooth, reliable and quick access to available knowledge. Expert Systems (ESs) can provide such support. An ES captures Human expertise in a computer program and mimics Human reasoning. An ES falls into the Artificial Intelligence (AI) field of Knowledge Representation and Reasoning (KRR). The general structure of an ES generally combines three components: the Knowledge Base or the Knowledge Graph (KG), the inference engine and the User Interface (UI) [14]. The KG includes the knowledge of the ES and is usually built manually, a time-consuming and arduous task. However, in the frame of the Design Engineering Assistant (DEA) study, the team is attempting to automatize the generation and population process of the KG. This feature led the team to explicitly separate the KG and the Database of Rules in the DEA architecture.

The project focuses on implementing an ES into the concurrent engineering process used for feasibility studies by many agencies, industries, and universities. Concurrent engineering methods were introduced at NASA and ESA in the 1990s, to accelerate the processes of mission definition and preliminary conceptions for new mission proposals with growing complexity [15]. The incentives to integrate an ES within the CE approach and the study process are presented in previous DEA publications [10] [12]. In these previous publications, the team also presented the goals of the project, the preliminary architecture, and incentives behind the implementation of an AI-based agent into the spacecraft design process. Preliminary results of automatic extraction based on space components datasheets are found in [11].

This paper will focus on the software trade-off and interactions with experts involved in concurrent engineering studies leading to the refinement of the tool requirements and the detailed software of both front and back-end. In a final chapter, initial results of the multi-word extraction and mention of the importance of Word Sense Disambiguation are presented.

2. THE DESIGN ENGINEERING ASSISTANT

This chapter provides the outlines of the DEA project, including the project mission definition, the high-level tool

architecture and identified challenges. In the second part, the outputs of the interviews led at ESA, ESTEC with experts from the Concurrent Design Facility (CDF) are summarized.

2.1. Mission Definition

The DEA is an ES for the early phases of space mission design. The team identified two main development steps towards the full implementation of an AI-agent into the process of concurrent engineering feasibility studies:

Stage 1, a Knowledge Engine (KE) - Developing and populating a KG that can be queried by the User is the first development stage of the DEA. The queries will be entered via a natural language interface, in English only. The UI will extract knowledge from the KG in order to provide knowledge summaries and data analytics including traceability and recommendations. It will also include an active User Feedback loop in order to acquire the tacit knowledge of the experts. The KG is primarily fed with data from the ESA CDF team (e.g., reports, presentations and engineering models).

Stage 2, a Space Mission Design Assistant - The integration in a modeling environment tool will transform the DEA into an active design assistant. Considered design environments for the DEA integration include the Open Concurrent Design Tool (OCDT) [16] used at ESA and the Concurrent Design and Engineering Platform CDP4 developed by RHEA [33] both based on the European standard ECSS-E-TM-10-25A, Annex A&C [17], and used in the frame of concurrent design studies. Being able to extract and assimilate the structured data from engineering models based on the ECSS-E-TM-10-25A model, the DEA will be able to monitor in the background the case study, actively provide design suggestions and flag model-inconsistencies in a non-invasive manner.

2.2. DEA High-Level Architecture

Figure 1 displays a preliminary architecture of the DEA. The architecture is divided between the smart-squid and smart-dog components, respectively the front and back-ends of the DEA. smart stands for Strathclyde Mechanical and Aerospace Research Toolboxes, "dog" for "domain ontology generation" and "squid" for "smart querying and insight for space mission design". The detailed software architectures of both tools are presented in chapter 4.

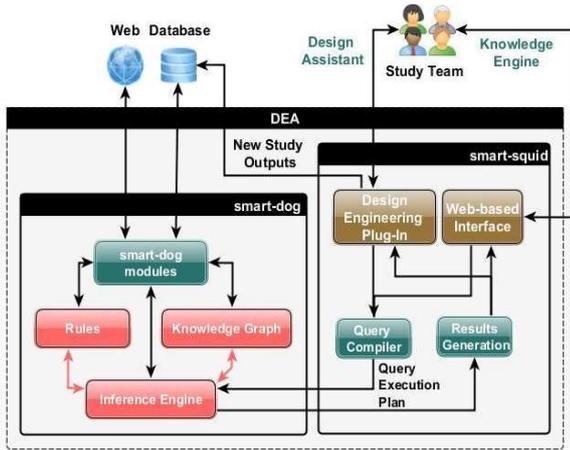


Figure 1. High-Level architecture of the DEA

2.3. Challenges

Challenges related to the development of the DEA have been detailed in previous publications [10], [12]. They included challenges concerning the KG generation and population, the expressivity of the language, time response and scalability, subdomains and continuous learning, the KG data modeling, access the data necessary to populate the KG, ensuring the data security within the KG, understanding the User intent and generating corresponding outputs. This paper will, however, focus more the challenges related to uncertainties quantification and mitigation.

Uncertainty is inherently attached to Human knowledge, and the available knowledge is generally imprecise, incomplete or fuzzy [3], [4]. As defined in [6] uncertainty refers to "any aspect of the model [of the real world] that cannot be asserted with complete confidence". Uncertainties originate from the data itself, are transmitted to the ES via the data modeling and the Information Retrieval (IR) process (e.g., related to the NLP methods employed) and are finally combined by the inference engine. Providing an answer to the End Users is not enough without indicating as well the output degree of certainty. This becomes a particularly essential metadata for medical diagnosis ES [1] [2].

Uncertainty quantification and management are fundamental to KRR within an ES, at all stages, from the back to the front-end. Four main questions arise with the issue of uncertainty within an ES:

- How to represent uncertain data from the data set or provided by User?
- How to combine uncertain data?
- How to infer from uncertain data?
- How to maintain the transparency on the uncertainty throughout the User interface?

Sections 4.1.2 and 4.2.3 will treat uncertainties respectively from a front and a back-end perspective. An additional layer of complexity is added in the case of the DEA as the ES is targeted to be as autonomous as possible: automatically

integrating new data into the KG and the feedback provided by the users via the UI.

2.4. Results from experts interviews

A User-centred approach holds the needs of the User at the center of each design decision throughout the project lifecycle. The DEA is a decision support system for Human experts. Therefore, it was essential for the team to include the Human component, the experts involved in feasibility studies, as early as possible into the project. Raising awareness on the topic of AI for space mission design, discussing with experts about how they could use the DEA and integrate it into their work process were the main goals of the interactions. The discussions contributed to lay the bases of the trust relationship between the DEA and its potential end-users. Trust is indeed a vital element for the success of the HMI and the adoption of the tool into the Users' work process. 47 experts from ESA, representing most of the fields involved in concurrent engineering studies, took part in round-table (40% of the experts) or face-to-face interviews over summer 2018. Figure 2 displays the background distribution of all experts involved.

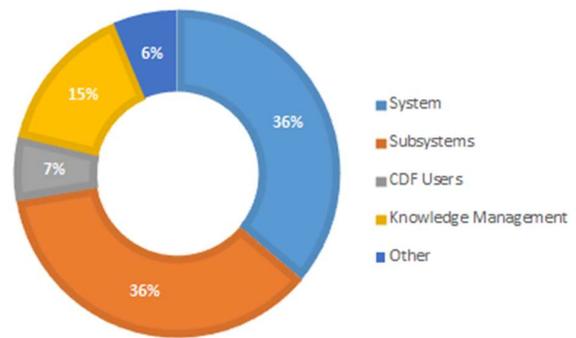


Figure 2. Background distribution of the experts

The elicitation process and results are thoroughly described in [12]. The main findings from the interviews are reminded in the following paragraphs. Figures 3, 4, 5 and 6 illustrate the main findings by displaying the experts' answers to different questions asked during the Round Table. However, the findings do reports comments collected from all experts interviewed (both during the Round Table and the face-to-face interviews).

The concept of the DEA KG raised a high interest among the experts. When kick-starting a study, the first activity is focused on knowledge reuse: experts look into previous similar missions, to get an idea of values range and architectures baseline validated by previous studies. The DEA could greatly simplify this task for them, quickly and easily making them aware of the relevant past missions. As shown in figure 3, a majority of experts spends between 25% and 50% of their time searching for information. Therefore, Knowledge Reuse is already part of the experts' work process. Figure 4 indicates the experts' familiar sources of information. Discussing with colleagues is a preferred option, it allows experts to add, on top of the information they find

in the reports, the tacit knowledge of their colleagues. However, Human experts might eventually forget some information or have a biased point of view.

Quantify the part of your work time spent researching through available information?

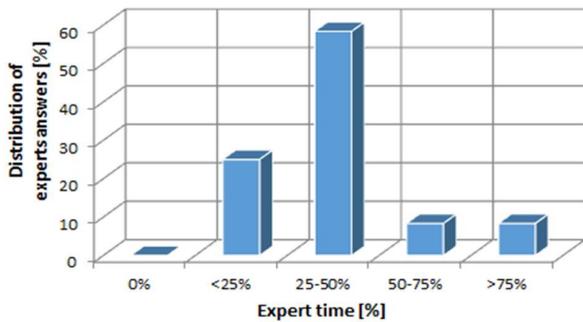


Figure 3. Round Table answer on their work habits

Where do you find the most useful information for your studies?

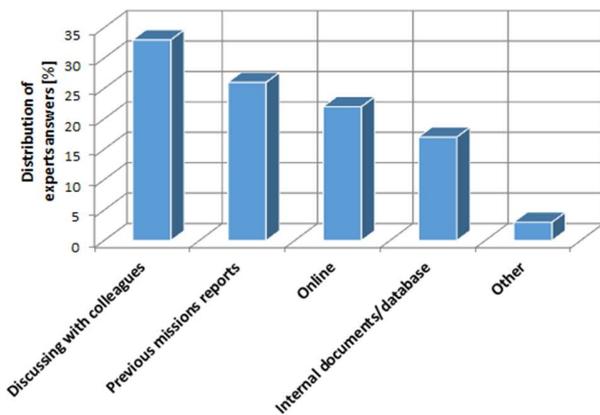


Figure 4. Round Table answer on their sources of information

The fact that the DEA could as well provide uncertainty measurements is a highlight for some experts who mentioned that the reliability of a source is often hard to judge. Experts argued that in addition to knowledge from early phases of space mission design, data from later phases, even flown missions would be very valuable to integrate to the KG. However, due to the limited timeframe of the project, the DEA team has to narrow down its focus on a first set of data, i.e., feasibility, phase 0 studies. Future work could naturally target the integration of other data sources to expand the KG.

One of the preferred outputs formats appears to be comparison tables as shown in Figure 5. However, experts frequently underlined their need to access the original source of information, i.e., the documents on which the KG is built, to look deeper into the summarized data provided by the DEA. Therefore a PostgreSQL database containing all the documents paths and parts extracted during the process of

creating the specific data has been added to the software architecture.

Vote for your preferred type of outputs

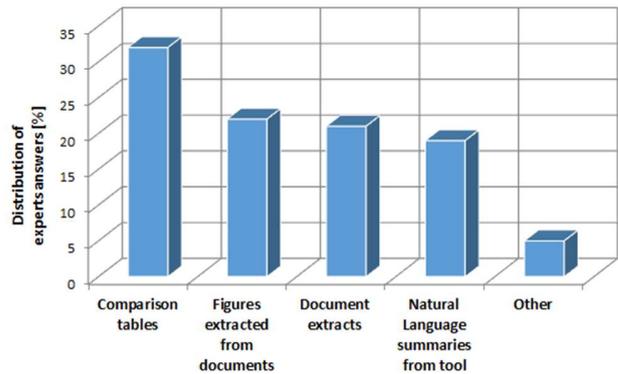


Figure 5. Round Table answer on the DEA Knowledge Engine preferred type of outputs

The experts' interviews also contributed to the refinement of the HMI, advising against the initial idea to adapt the outputs to the User background. Experts worried that the answer range would be narrowed down and prevent knowledge discovery. A set of various queries was defined with the experts allowing to understand better the kind of knowledge they were likely to request. These queries will also be used to train and test the DEA.

Finally, the concept of integrating the DEA as an active assistant within the design environment raised some interest among the experts. However, it was made clear that it would require more efforts, regarding testing and validation of the tool, for the experts to accept the DEA as an active assistant rather than as a passive knowledge engine. As shown in Figure 6, 70% of experts were unsure that they would trust the DEA. This percentage can be lowered by testing and validation, as well as demos involving the experts.

Would you rely on an AI-agent to mine information for you?

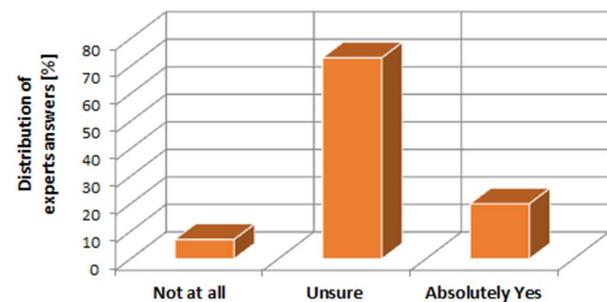


Figure 6. Round Table answer on the trust relationship with the DEA

Interaction with the ESA experts, as well as access to reports provided by the ESA CDF, led to the refinement of the project definition and architecture. The resulting requirements are presented in the following subchapter.

2.5. Requirements for smart-dog and smart-squid

The general requirements of the DEA and the more specific requirements of smart-dog and smart-squid were defined based on the findings of the literature review and discussions within the project team. The requirements were then refined after the interview sessions with space mission design experts, as presented in the previous subchapter, and after having accessed a sample of reports to be fed to the KG.

2.5.1. General requirements

DEA-GEN-01 The DEA project is composed of two complementary components: smart-dog (methods for the expert system back-end) and smart-squid (methods for the expert system front-end assimilation of structured design data).

DEA-GEN-02 The DEA shall target as End-users experts involved in space mission feasibility studies, across all domains.

DEA-GEN-03 The DEA primary source of data shall be feasibility reports of space missions from diverse types (e.g., Science, Earth Observation). Other sources of data should be integrated into the KG in further implementations.

DEA-GEN-04 The DEA shall remain a non-invasive assistant and support to the Human experts.

DEA-GEN-05 The DEA shall not increase the workload of the Human experts and therefore remain as autonomous as possible (e.g., automatic KG generation/population and User Feedback integration) and as easy to use as possible.

DEA-GEN-06 The DEA shall rely on open source software.

2.5.2. smart-dog

General requirements -

DEA-DOG-GEN-01 smart-dog corpus should allow automatic extraction of data from documents reducing the interaction with the human experts at the minimum.

DEA-DOG-GEN-02 smart-dog shall be able to convert, at least, text, unstructured data, in structured data.

DEA-DOG-GEN-03 smart-dog shall be able to be a neutral ontology learning system able to be adapted to different domains.

DEA-DOG-GEN-04 smart-dog shall discard sources not relevant to the domain in consideration.

DEA-DOG-GEN-05 smart-dog shall be able to parse, at least, .pdf, .docx, .pptx, .html documents.

DEA-DOG-GEN-06 smart-dog shall rely on the inference engine for the validation of the knowledge graph.

DEA-DOG-GEN-07 smart-dog shall rely on ontology learning techniques for the generation of the knowledge graph.

Knowledge Graph requirements-

DEA-DOG-KG-01 Each data that shall be inserted inside the KG shall be validated (human validation or ad hoc software solution).

DEA-DOG-KG-02 smart-dog corpus shall include sources from feasibility studies, for example, CDF reports.

DEA-DOG-KG-03 DEA KG shall contain all relevant terms for the space mission design domain extracted from the source data.

DEA-DOG-KG-04 smart-dog shall rely on ontology learning techniques for the generation of the knowledge graph.

DEA-DOG-KG-05 smart-dog shall provide an interface for validation and visualization.

Inference Engine requirements-

DEA-DOG-INF-01 The inference engine shall allow the following reasoning services: realization, classification, satisfiability, conjunctive query answering, entailment, consistency, explanation.

2.5.3. smart-squid

General requirements-

DEA-SQUID-GEN-01 smart-squid shall extract knowledge from the DEA Knowledge Graph.

DEA-SQUID-GEN-02 smart-squid shall allow the querying of the Knowledge Graph in natural language via a web-based User Interface.

DEA-SQUID-GEN-03 smart-squid shall extract the models from the design environments and allow the assimilation to the DEA Knowledge Graph.

DEA-SQUID-GEN-04 smart-squid shall quantify and mitigate the uncertainties related to the knowledge extraction.

DEA-SQUID-GEN-05 smart-squid should enable the collection of tacit knowledge from the experts via the User Interface and the Feedback loop.

DEA-SQUID-GEN-06 smart-squid shall autonomously analyze, detect conflicts and quantify the uncertainties linked to the User feedback before integration to the Knowledge Graph.

Interface of the Knowledge Engine -

DEA-SQUID-INT-01 The User queries are entered in natural language and in English via the User Interface.

DEA-SQUID-INT-02 smart-squid User Interface shall be User-friendly and operable with a minimum to no training.

DEA-SQUID-INT-03 smart-squid shall transcribe the User queries from natural language to internal representation adapted to the querying modules.

DEA-SQUID-INT-04 smart-squid should accept a wide range of queries w.r.t. the field of interest, the query format or formulation.

DEA-SQUID-INT-05 smart-squid User Interface should be error tolerant and flexible to the User queries, including robustness towards vague or fuzzy queries.

DEA-SQUID-INT-06 smart-squid shall facilitate the User search by allowing research refinement and suggesting basic queries.

DEA-SQUID-INT-07 smart-squid outputs shall be displayed in natural language in different format (i.e., images, text, graph) via the User Interface.

DEA-SQUID-INT-08 Traceability: smart-squid shall provide traceability and access to the source of information.

DEA-SQUID-INT-09 Uncertainty Measurement: the User Interface shall provide traceability of the uncertainty measure via a numerical representation of uncertainty factors or fuzziness quantifiers to the End User.

DEA-SQUID-INT-10 Transparency: smart-squid should provide and display the explanatory process of the output construction.

DEA-SQUID-INT-11 smart-squid shall transfer the User Feedback to smart-dog and smart-squid's query optimizer, only after an uncertainty quantification will have been performed on the information or comment submitted by the User. (ref. DEA-SQUID-GEN-06)

DEA-SQUID-INT-12 smart-squid User Interface should feature a recommender system to boost Knowledge Discovery.

DEA-SQUID-INT-13 smart-squid User Interface should generate basic analytics.

DEA-SQUID-INT-14 New data: smart-squid should allow Users to propose new data to integrate to the Knowledge Graph via smart-dog via the User Interface.

Interface - Active Engineering Assistant -

DEA-SQUID-EA-1 smart-squid shall access and extract structured data from the active engineering model.

DEA-SQUID-EA-2 smart-squid shall include the design of a plug-in integrated to the design environment to interact with the User.

DEA-SQUID-EA-2 smart-squid should autonomously run queries over the DEA Knowledge Graph to support the design comparison.

DEA-SQUID-EA-2 smart-squid shall flag model inconsistencies to the User based on the Knowledge contained in the DEA Knowledge Graph.

DEA-SQUID-EA-3 Invasiveness: smart-squid plug-in shall not be invasive; the plug-in will only allow the DEA to provide suggestions to the User.

3. TECHNOLOGY TRADE-OFF

This chapter focuses on three main parts: the main software components that have been identified for the development of the expert system according to the requirements stated in section 2.5; the parameters selected for the qualitative trade-off to compare the different possible configurations for the software discovered and the last part highlights the software selected for the software configuration.

3.1. Software architecture

The DEA relies on an ES architecture taking into consideration four components: the KG, the inference engine, the Database of Rules and the UI. The following chapter will focus on the software components, interfaces and languages to be considered for the final architecture.

The different software components required according to the DEA requirements are:

The Ontology Learning System - A tool that includes all the algorithms and modules required to allow the (semi-) automatic generation and the population of the KG (i.e, Doodle II, Text2Onto, OntoLearn, HASTI, Ontogen, LexOnt, OntoGain) [7] [28].

The Rule Extraction - A module developed for the extraction of rules from the documents. These rules will be used by the inference engine to provide answers.

The Ontology Editor - A computer program enabling the User to perform actions on the ontology (i.e, Protegé, Apache Jena, ALTOVA, Fluent Editor, Grakn editor) [27].

The Inference Engine or Reasoner - The software that infers knowledge using the rules, and allows the reasoning on the information contained within the KG. (i.e, Hermit, Fact ++, Jfact, Pellet, Elk, Racer, Snorocket, FuzzyDL, Elephant, Grakn reasoner) [26].

The Rule Engine - The software that allows the statement of rules according to the language selected. This component could be merged with the inference engine in some tool (i.e, CLIPS, JESS, Drools).

The User Interface (UI) – The UI depends on the type of application and target users.

The compatibility of these components is not only related to a software point of view but also from a modeling point of view. In particular, the languages to be considered for the ES according to the software components identified above are:

Natural Language (NL) – NL is the most complex to be dealt with. NL is ambiguous and subjective, many concepts can be explained in different ways, and it is not easy for a machine to capture the different meanings in different context.

The Ontology Language (OL) – OL is the language selected for the data modeling. This language allows the introduction of concepts or entities, relationships, axioms and properties relative to the specific domain. The choice of the language introduces constraint in the level of expressivity. This selection is essential because it is one of the main factors, together with the choice of the reasoner that affects the query response time. (i.e., OWL, RDF(S), F-Logic, PR-OWL, Graql).

The Rule Language – The Rule Language is the language required to state the rules. It is usually not the same as the one used for the OL, and it has to be compatible with it (i.e, SWRL, Graql).

The Ontology Query Language - This is the language used to query the KG through the inference engine to provide the

correct answer. This language is different depending on the compatibility with ontology query languages, reasoners and ontology language (i.e., SPARQL, SPARQL-DL, Graql).

According to the languages analyzed, the language conversions required to satisfy compatibility and interoperability between the software components are:

- from Natural Language to Ontology Language and vice-versa;
- from Natural Language to Rule language and vice-versa;
- from Natural Language to Ontology Query Language and vice-versa.

The most general software architecture proposed for the DEA according to the analysis of the different factors is shown in Figure 7, while Table 1 summarizes the classes and the factors within each class.

If we focus on the selection of three components of the ES (the ontology editor, reasoner and rule engine), the possible combinations generate several solutions with different advantages and disadvantages. In particular, the adoption of one OL puts the constraint on the level of expressivity, which at the same time constrains the selection of the reasoner and the query language. Therefore, we first analyzed several ontology languages according to the following parameters:

- Logic
- Ontology Modelling Construct
- Compatible Reasoner
- Compatible Ontology Query Language

The OL considered are RDF(S), OWL 1 Full, OWL 1 DL, OWL 1 Lite, OWL 2 DL and Graql.

We analyzed several reasoners according to the following parameters:

- Expressivity
- Compatible Ontology Language
- Compatible Ontology Query Language
- Rule Language Support
- Reasoner methodology
- Supported Reasoning Services
- Licensing
- Availability
- Language implementation
- Last Release

The reasoners considered are Hermit, Fact ++, JFact, Pellet, ELK, Racer, Konclude, Snorocket, FuzzyDL, DeLorean, CEL, Mastro, TrOWL, ELephant, jcel, Chainsaw and Graql reasoner. Most of these reasoners are also rule engines. Therefore, we did not have to make a specific analysis.

We also analyzed the different Ontology Editors according to the following parameters:

- Availability
- Licensing
- Last version
- Portability
- Compatible Ontology Language
- Compatible Rule Language
- Compatible Reasoner
- Programming Language
- Extensibility
- Visualizer
- type of Application Programming Interface (API)

The Ontology Editors considered are Protege, Fluent Editor and Grakn.

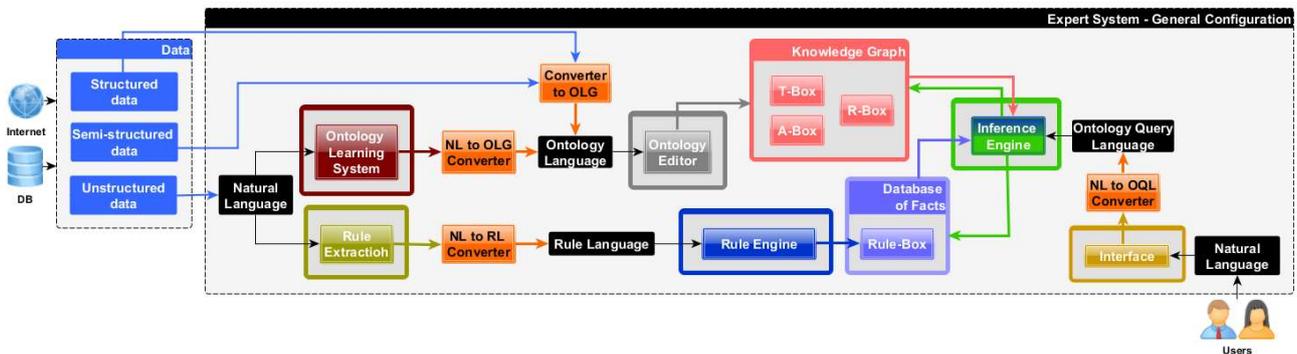


Figure 7 General software architecture for the DEA expert system

Table 1. Classes and factors for the software architecture

Software components	Type of languages	Language conversions
Ontology Learning System Rule Extraction Ontology Editor Inference Engine Rule Engine Interface	Natural Language (NL) Ontology Language (OLG) Rule Language (RL) Ontology Query Language (OQL)	NL to OLG, OLG to NL NL to RL, RL to NL NL to OQL, OQL to NL

3.2. Trade-off parameters

The parameters taken into account for the selection of the software for the general architecture of the expert system are listed and described below:

Compliance to Standards - There are several types of standards for data modeling, data exchange and interfaces (e.g., W3C, ECSS, SparQL, GraphQL, JSON-LD, OSLC or Open REST API). The compliance to standards is vital to create products universally understood and comparable. The adoption of new technology, if not developed through standards, should require a mappability with the primary standards.

Extensibility/Extendability - Extensibility is a design principle where the implementation considers future growth, so the possibility to extend a system through the addition of new functionality or modification of existing functionality.

Interoperability - Interoperability is a characteristic of a product or system, whose interfaces are completely understood, to work with other products or systems, at present or in the future, in either implementation or access, without any restrictions. The tool should be able to import and export from other relevant languages and data formats.

Learning Curve (Developer and End-User) – This represents the learning curve to implement or adapt the technology and the learning curve for users that will use the adopted technology.

Ontology constructs - The ontology constructs adopted depends on the ontology language selected (e.g., Classes, Properties, Datatype, Relationships, Roles). The adoption of one construct instead of another also depends on the type of data that has to be considered and on the level of expressivity. Moreover, these choices affect the query response time once a query is given to the inference engine.

Quality and Completeness of Documentation - The documentation is fundamental for User to use the tool and for Developers to understand the implementation and the possibility to add new features.

Portability - Portability between operative systems.

Project status - When a specific software, developed or under development is selected, it is essential to consider the factors that affect the status of the project (active users, active developers, releases, bug fixes, response time on questions or raised issues).

Reasoning services - These services guarantee consistency and reliability to the model and therefore also to the answers that will be provided to the users. It is essential to take into account the most critical reasoning services.

Query Response Time - The query response time depends on the size of the knowledge graph, on the ontology language and therefore on the level of expressiveness, and the type of queries.

Scalability - The tool can grow and manage increased demand, adapting to the changing needs or demands of its users.

Software Architecture Complexity - The software architecture complexity affects the timeline of a project. The interface between many software is not a trivial task, and it requires validation and quality check. For the expert system, the components listed in Table 1 needs to be interfaced.

Sources/Example projects - It is important to have sources to check the use of the software and applications for which it has been adopted.

Satisfaction of Users - The popularity of the software, together with the satisfaction of the users are also taken into account.

V&V Test Framework - The Validation and Verification Framework available for the selected technology.

3.3. Software selected

Two main configurations have been considered for the trade-off according to the requirements stated in 2.5.1:

- using Semantic Web technologies (figure 8.1 and table 2.1)
- using [Grakn](http://grakn.ai/) (<http://grakn.ai/>)

The selection of the software configuration and languages has been performed with a qualitative analysis using the parameters described in section 3.2., taking into consideration the impact of each of these parameters on the success and timeline of the research project. The results of the trade-off identified Grakn as the best solution.

Grakn is an intelligent database built for intelligent systems allowing the creation of a KG using the Graql data modeling. Grakn does not follow the standards for the semantic web but it is implemented in Java programming language, on top of Cassandra and it provides an API for Java, Python, and Node.js. It can be integrated into many applications and tools. The API allows interoperability with other tools, and it is possible to extend its functionalities thanks to the different interfaces. It has a unique and User-friendly visualizer allowing for validation of the knowledge graph. The data modeling is very intuitive and very fast to be learned especially for users with software background but not only. The ontology constructs in the Graql language can be mapped at a certain level with the standard for the semantic web and without creating limits in the use of the tool. Grakn is portable in different operating systems (Linux, MacOS, and Windows) and it has extensive documentation provided on the website that eases the learning curve for developers and new users. The project is very active and recently released the version 1.4.2 by the time this paper has been created. For Graql reasoner, all the reasoning services are provided for the Graql data modeling (Realisation, Classification,

Satisfiability, Conjunctive Query Answering, Entailment, Consistency, and Explanation) therefore matching the requirement for the development of the tool (**DEA-DOG-INF-01**). Query response time and scalability depend on the type of knowledge graph, data, and expressiveness adopted in the project.

Regarding the software architecture complexity, figure 8.2 displays the advantages of the adoption of this tool, as it encases three of the main components for the expert system, table 2.1 (i.e., Knowledge Graph, Inference Engine and Database of Rules) enabling Knowledge Discovery and Analytics through the Grakn tool. This solution leaves us with the possibility to focus on the specific research for the PhD projects, avoiding the implementation of the interfaces between the main components of the expert system and considering one single language for the ontology, the rules, and the queries.

All Grakn projects can be found in the Grakn blog, <https://blog.grakn.ai/> (e.g., Biology applications, chatbots, movie recommenders). The popularity of the software and users satisfaction are witnessed by the recent introduction of Grakn in both Google and Amazon Cloud. Due to the different interfaces in different programming languages (Java, Python and JavaScript), Grakn allows many choices for the creation of the Validation and Verification Framework.

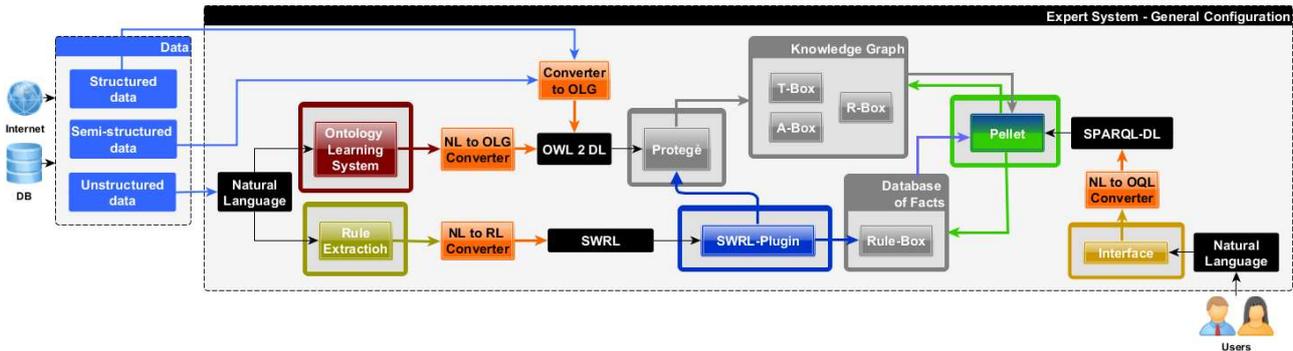


Figure 8.1. Software architecture for the expert system adopting Semantic Web Technologies

Table 2.1: Classes and factors for the software architecture with Semantic Web Technologies

Software components	Type of languages	Language conversions
Ontology Learning System Rule Extraction Protege (https://protege.stanford.edu/) + SWRL Plugin (https://github.com/protegeproject/swrltab-plugin) Pellet (https://github.com/stardog-union/pellet) Interface	Natural Language (NL) OWL 2 DL SWRL SPARQL DL	NL to OLG, OLG to NL NL to RL, RL to NL NL to OQL, OQL to NL

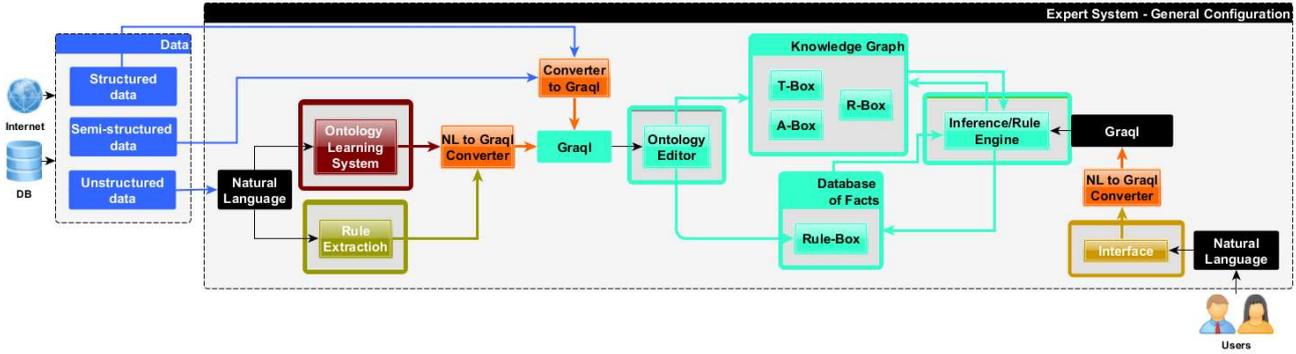


Figure 8.2. Software architecture for the expert system adopting Grakn technology

Table 2.2: Classes and factors for the software architecture with Grakn technology

Software components	Type of languages	Language conversions
Ontology Learning System Rule Extraction Grakn (Ontology Editor) Graql reasoner (Inference/Rule Engine) Interface	Natural Language (NL) Graql (Ontology Language, Rule Language, Ontology Query Language)	NL to Graql, Graql to NL

4. SOFTWARE ARCHITECTURE

As hinted in Figure 1, the DEA architecture is based on the alliance of two main set of methods: smart-squid and smart-dog respectively corresponding to the front and back-end of the DEA. This chapter will go more in details of each components software architecture.

4.1. smart-squid: DEA Front-End and structured data assimilation

smart-squid modules encompass all activities related to the front-end of the ES. In the frame of the DEA project, the front-end supports the Human Machine Interaction (HMI) via a web-based interface and a design environment plug-in.

Via the web-based interface, the User enters NL queries, which are decomposed and analyzed by smart-squid query engine, used to generate a query execution plan to extract and generate knowledge based on the KG and generated with inference engine. The web-based interface also enables the elicitation of the User tacit knowledge via a feedback loop.

Via the design environment plug-in, the DEA will access the structured data related to the working design. The detailed architecture of smart-squid is presented in Figure 9 and furthermore described in this subchapter. The next

subchapter will focus on uncertainties quantification and management from the front-end perspective.

4.1.1. Detailed Software Architecture

As shown in Figure 9, the software architecture of smart-squid is divided into two parts: one corresponding to the knowledge engine (continuous black lines and blue modules) and one to the design assistant integrated into the design environment (golden modules and dotted lines).

An example supports the description of the smart-squid architecture based on Earth Observation (EO) missions and L-Band Synthetic Aperture Radar (SAR), a type of remote sensing payload. A very basic KG was generated with grakn.ai (<https://grakn.ai/>) containing seven spacecraft from different agencies: ALOS-2 [18], NISAR [19], PAZ [20], SAOCOM-1A [21], Sentinel-1A [22], Tandem-L [23] and Tandem-X [24]. They all have SAR payloads but in different bands: L-band, C-band and X-band. For this example, the KG structure is on purpose very basic, only four entities were defined: "MissionType", "Spacecraft", "Payload Type" and "SAR Bandwidth". We imagine that the User is interested to search for L-band Earth Observation missions.

Web-based Interface - The User enters a NL query in English via a search bar on the web-based UI. The range of queries or requests, accepted by the tool has been refined after a set of experts interviews as described in 2.4. An example of a query could be: "Show L-band SAR satellites". To help the User formulate his/her research

intent into a written query, the UI will include refinement options presented as optional check-boxes. For instance, the User could select “Earth Observation” as the preferred mission type. Filtering or refinement options are used to better answer the User’s need and target more specific areas of the KG to decrease computational time.

The results of the User query are displayed on the same web-based UI. Outputs should be generated in different formats, for instance, comparison tables, extracted figures or images, or a summary of information. For the tool successful integration and acceptance by the experts, transparency, reliability and uncertainty measurements are essentials. The User should therefore be able to access the source of the information (cf. traceability requirement DEA-SQUID-INT-08). The original documents will be saved in a PostgreSQL and associated to the corresponding nodes of the KG. The DEA should be clear on the degree of certainty the User can have in the outputs (cf. uncertainty requirement DEA-SQUID-INT-09). The outputs will be associated to a numerical uncertainty quantification. Finally, the extraction and answer generation process should not be a "black box" (cf. transparency requirement DEA-SQUID-INT-10). The DEA should provide a simple overview of the analytical process to justify the outputs selection. Establishing the trust relationship between the DEA and the User is a crucial element for the integration of the tool into the experts' working habits.

The role of the UI and the knowledge extraction is also to boost knowledge discovery via the implementation of a recommender system encouraging the experts to explore alternative design options. The UI allows the User to submit new documents to the KG. The new data will go through the same filtering path as other documents submitted to smart-dog before being integrated or not to the KG. Finally, a significant part of the UI will be the integration of the User feedback loop, which is further developed in the last paragraph.

Query Compiler - The Query compiler receives a NL input and generates a query execution plan in the query language used to model the KG. Since the technology trade-off summarized in chapter 3 suggested the use of Grakn.ai, the query execution plan must be written in Graql, the querying and reasoning language of Grakn. The query compiler has three main modules: the NLP module, the Query analysis/optimizer module and the translation (i.e. "code generation to query language") module. The NLP module is similar to the one used by smart-dog. The role of the query compiler should also be to eventually grasp the real intent behind the User query (i.e., perform a semantic search). The query decomposition into a basic set of queries takes place in the Query analysis/ optimizer module. An optimized decomposition of the User query into basics ones allows extracting knowledge from the KG more efficiently. The queries are translated into the Graql query language as a Query Execution Plan.

Query Execution - The Query Execution relies on the Grakn API. Table 3 presents examples of Graql queries and their natural language equivalent. The construction of Grakn queries is furthermore explained in [25]. Grakn provides a visualizer on a local host exhibiting via a web-interface the outputs of a query. The visualization of the two first queries as well as the code used to generate the KG sample can be found in the Appendix while the visualization of the third query (i.e., corresponding to the example of L-band SAR mentioned previously) is displayed in Figure 10.

It has to be underlined that the Grakn.ai visualizer is not the DEA interface. The Grakn.ai visualizer inputs are written in Graql, and the outputs are a display of the relevant part of the KG corresponding to the User query, whereas the DEA interface targets NL inputs and will combine and generate knowledge based on the knowledge extracted and inferred from the KG.

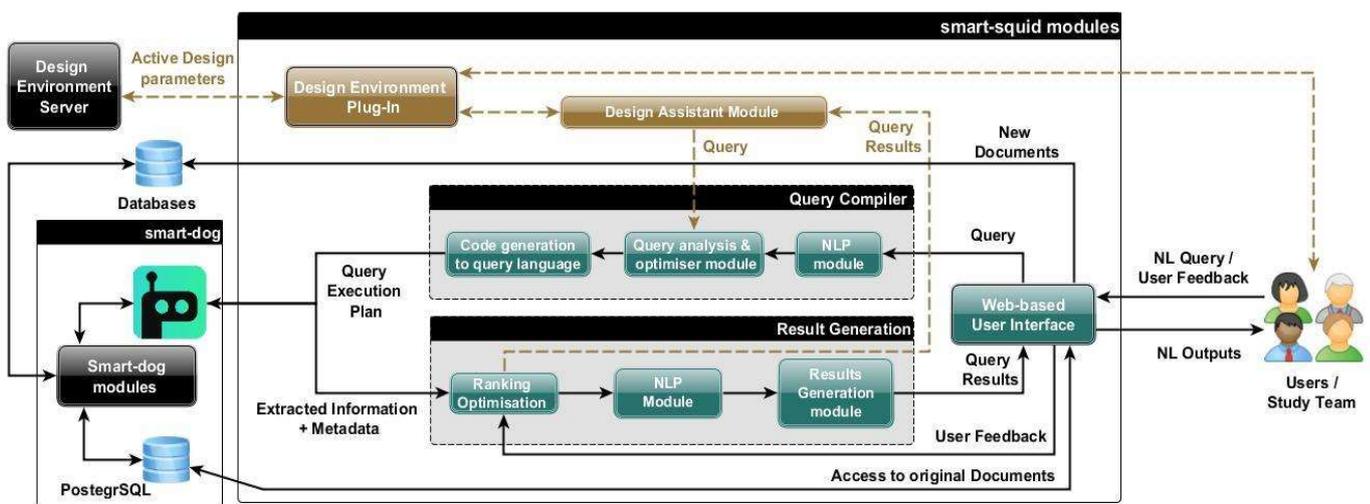


Figure 9. Detailed Architecture of smart-squid - Front-End of the DEA

Table 3. Equivalent User Queries examples in natural language and Graql language

NL query	Graql query
Show all Earth Observation missions	<i>match \$x isa missionType; \$y has name "EO"; offset 0; limit 20; get;</i>
Show all satellites with SAR payloads	<i>match \$x isa spacecraft; \$y isa payload; \$y has pname "SAR"; (\$x,\$y) isa HasPayload; \$z isa SARBand; (\$y,\$z) isa HasBandwidth; offset 0; limit 20; get;</i>
Show L-band SAR satellites	<i>match \$x isa spacecraft; \$y isa SARband; \$z has band "Lband"; (\$x,\$y) isa HasSARwithBandwidth; offset 0; limit 20; get;</i>

The query outputs shown in Figure 10 are correct as out of the 7 spacecraft entered into the test KG only 4: NISAR, Tandem-L, ALOS2, and SAOCOM-1A have L-band SAR payloads.

Results Generation - Once the candidate facts or knowledge have been extracted from the KG, the "Result Generation" module of smart-squid ranks the facts and transform the "raw" information into the useful format (e.g., comparison tables, text summaries) to be made available to the User via the UI.

Between the ranking and the organization of the outputs, the data has to be converted again to NL via a second NLP module. The ranking of the candidate facts is based on factors depending on certainty and relevance coefficients combinations associated with the data and a dynamic factor related to the User Feedback. In later phases, LearningToRank algorithm can be implemented to improve the facts ranking continuously. Introduction of Machine Learning into the ranking process has led to the emergence of LearningToRank algorithms also called Machine-Learned Ranking (MLR). [30] introduces different approaches to LearningToRank. Famous examples of MLR algorithms are RankNet and LambdaMART by Microsoft Research [31] [32].

Feedback Loop - There are two different kinds of knowledge: tacit (e.g., unspoken rules of know-how, "rule of thumb") and explicit (e.g., reports, presentations) [13]. Feedback can also be divided between implicit and explicit feedback. This paragraph focuses on explicit feedback. Integrating a User Feedback loop in the UI allows capturing part of the tacit knowledge from the users. For example, a User might want to update the launch date of a spacecraft because she/he has a more recent source of information. As defined by the requirement DEA-SQUID-GEN-06, the feedback loop should eventually be an automatic process to facilitate the tool maintenance.

It is both in the interest of the system and of the Users to allow continuous inputs of new information. The exact process of the feedback loop will be shared in later publications. As a preliminary option, it is considered that the User feedback could be collected by letting the User assign a grade to an output content or format (e.g., rating the usefulness from 1 to 10 of an output) and possibly add written comments as well (e.g., commenting on a value that should be updated or is not accurate). However, this automatized feedback process must be considered with care to avoid the injection of uncertainties and disequilibrium (i.e., unreliable or too subjective feedback) into the KG as discussed in the following section.

Integration to the design environment - As a design assistant, the DEA should be integrated into the study design environment (e.g., the OCDT [16] or the CDP4 [33] environment) to access the structured data of the active engineering models. The model, structured data, from the OCDT or CDP4 are based on the ECSS-E-TM-10-25A Annex A Unified Modeling Language (UML) model and rely on the JSON protocol for data export. To assimilate the engineering model data into the DEA KG, the ECSS-E-TM-10-25A model was mapped into a Graql schema. The last iteration of the engineering models is to be populated into this schema as data layers. The mapping process will be presented in a future publication.

The design assistant interface within the design environment has not been defined yet. It will be done at a later development phase of the project. An idea is, for instance, be to have a chatbot interacting with the User, that could be muted or unmuted depending on the User's need to use the DEA.

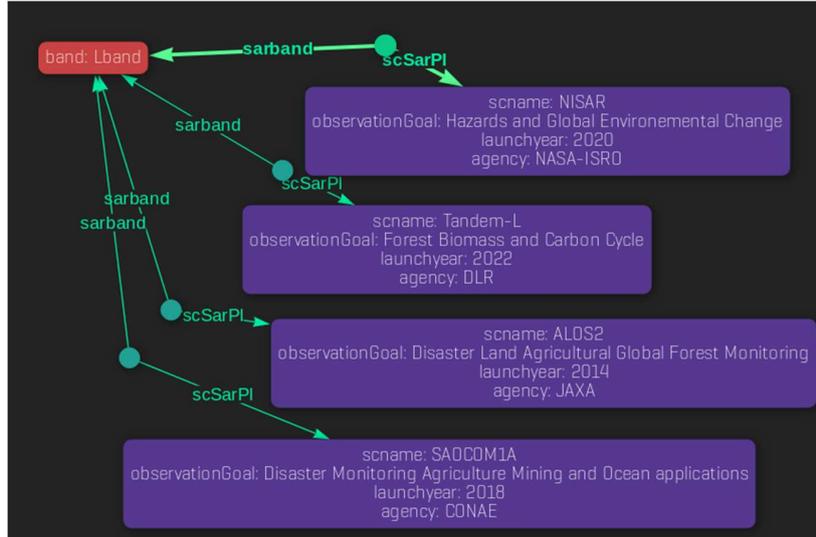


Figure 10. Grakn visualizer output for query "Show L-band SAR satellites".

Via the design environment plug-in, the DEA will be able to follow the design iteration as an observer, running in the background, and step in, to flag a model inconsistency or an outlier value. For instance, if a component parameter value is inconsistent with the datasheet information integrated into the DEA KG or if a value is outside the range of values typically found in similar missions according to the knowledge extracted from the DEA KG (i.e., outlier value). The DEA intends to be a non-invasive assistant and therefore will only provide suggestions of modifications to the experts (ref. DEA-SQUID-EA-3 on invasiveness requirement).

4.1.2. Uncertainty Management within the Front-End

Uncertainties within the front-end are partly inherited from the back-end. The uncertainties related to the back-end are discussed in 4.2.3. While smart-dog modules will deal with the uncertainty mitigation within the back-end, the traceability of the uncertainty quantification will likely be transmitted to smart-squid via Certainty Factors (CFs) associated to the data in the KG. CFs, represent the level of confidence in the associated data. They are an intuitive way to numerically illustrate the degree of uncertainty related to a piece of information. A CF can be expressed as fuzzy factors rather than crisp as described in [36]. CFs are often found in the literature. [1] describes, for instance, the uncertainty quantification and management for an epilepsy ES called HIPPOCRAT-EES. This ES simulates the reasoning of neuroscientists to diagnose epilepsy. It deals with uncertainty by ordering the potential diagnosis per increasing order based on CFs. The certainty factor presented with the output, the epilepsy diagnosis, is the sum of the weights assigned to each diagnosis parameters.

Additional uncertainties are injected via the User queries and the Feedback loop. The first source of uncertainty related to the User queries can arise from insufficient knowledge of the tool capacity and usage. To counteract this, the User-centred approach, already used during the requirement definition

phase, will be maintained during the next development phases. The experts contacted during the interviews will be regularly informed of the tool advancement and asked to provide feedback on the UI to ensure optimal user-friendliness of the DEA interface.

As it is not trivial to correctly render in words a Human thought, queries entered in NL via the UI might be vague or incomplete with fuzzy search goals. Mitigation methods found in the literature include the use of chain queries tracking [34], vague-query processor [35]. Semantic search contributes to better grasping the User intent [43]. Additional Machine Learning methods, including LearningToRank algorithms mentioned in 4.1.1.1 and query optimizers can furthermore contribute to improving the ranking and querying process. The UI also supports the narrowing of the research via the use of filters.

Typed queries often include some typos, misuse of words or concepts. To increase the flexibility of the UI, solutions found in the literature suggest the use of interfaces with error-tolerant and flexibility features [41] [42].

Finally, the uncertainty quantification related to the Feedback loop is probably the biggest uncertainty mitigation challenge of the DEA front-end. Allowing users to comment on data or submit new documents could expose the KG to imprecise, vague or wrong inputs. It is essential to integrate into the User feedback integration process a resilient uncertainty quantification and mitigation strategy to filter feedback and to avoid compromising the KG data. This filter will have to autonomously interpret the Feedback content and assign CFs to it, as the target is to let the DEA autonomously and continuously learn. In [2], an Expert Authentication Controller is used to support a User identification procedure. Restraining the list of Users allowed to provide feedback to a list of recognized experts could be the first filtering. Also mentioned in [2], the addition of new rules or data relies on a Conflict Detection module, including both syntax errors and content conflicts detection. One could imagine that a similar

module could be implemented into the DEA as a second layer of filtering. Eventually, the DEA interface could integrate some collaborative features that would allow Users to add comments directly to the data via the UI. Thread of comments would support the experts' interactions and could organically lead to the correct values recognized by the experts' majority.

Figure 11 summarises the uncertainty sources and potential mitigation methods from the front-end perspective.

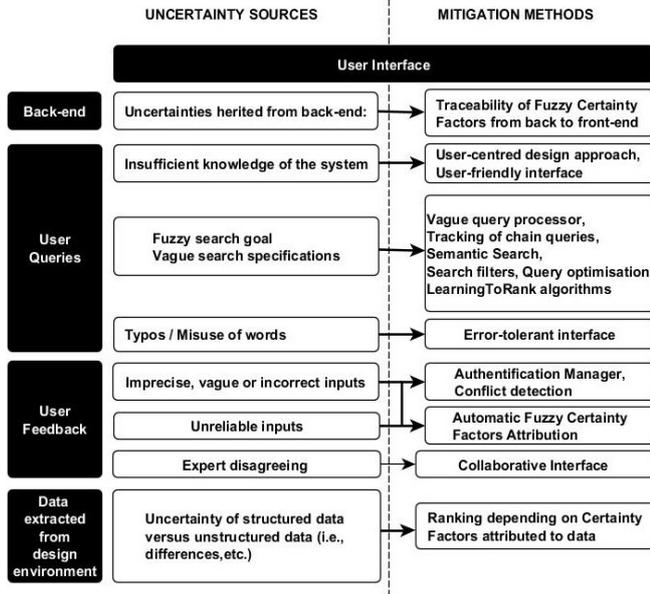


Figure 11. Summary of uncertainty sources and mitigation methods from the front-end perspective

Following the definition of the software architecture and the refinement of smart-squid requirements after experts' interviews, the next step will be to start the tool implementation. The implementation will first focus on the extraction of structured data from the design environments and on the NL inputs and querying of the KG.

4.2. smart-dog: DEA back-end

smart-dog represents the back-end part of an expert system. It encases the modules for the automatic creation of the KG coupled with the use of Grakn and its main components for inferring knowledge and validate and a relational database to store the data for fast processing, validation, and visualization. Figure 12 shows a general use of the smart-dog framework to perform the generation and the population of the KG [12]. The creation of the KG is iterative and requires human validation especially for the generation part. The interaction with the ontology expert will be allowed through a web-based interface, different from the DEA UI, that will give the possibility to validate the results obtained by the tool

before the insertion into the KG, in the frame of the DEA project for the space mission design domain.

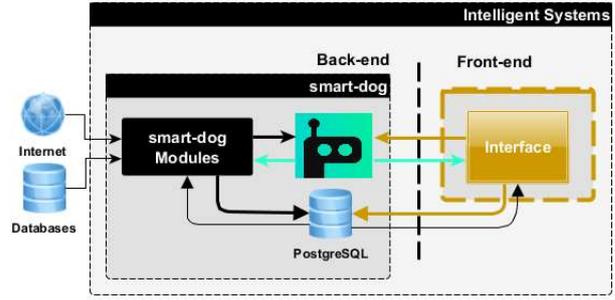


Figure 12. General use of the smart-dog framework

smart-dog has a modular architecture, allowing flexibility in the interfaces between the modules and guaranteeing static extensibility. Figure 13 shows the different modules, which are introduced in [12].

In particular, the NLP Module is not static, but it is dynamically improved iteratively with relevant domain terms. There are many libraries and dictionaries for NLP but developed for general language purposes. We target a particular domain. Therefore, the first step to improve the text processing is to identify the vocabulary for space mission design, to allow a better normalization and increase the performances of the modules receiving the NLP module outputs. The other modules rely on Machine Learning algorithms; therefore, their results will be improved when the input data will be better, following the quote "garbage in, garbage out." The Context Identification module is another critical module. It has to act as a filter to avoid the analysis of documents with topics not relevant to the domain considered, while, at the same time, it has to be able to distinguish between different topics relevant to the space mission domain to improve the outputs and overall automation of the process, especially for the ontology population part.

The coupling of the Context Identification Module with the Ontology Learning Module and with the NLP Module is fundamental for the performance of the tool. The NLP Module is coupled with the Ontology Learning Module to provide inputs for the extraction of the relevant information for the data modeling of the KG, but at the same time, the module itself is improved in the normalization phase after the validation of the outputs of the Ontology Learning Module.

The Grakn Module and the Validation Module are also coupled since the information validated can be automatically inserted. At the same time, Grakn itself has a visualizer that can be used for visual validation of the data modeling as well. The Ontology Population Module will be the last to be developed once all the KG will be generated.

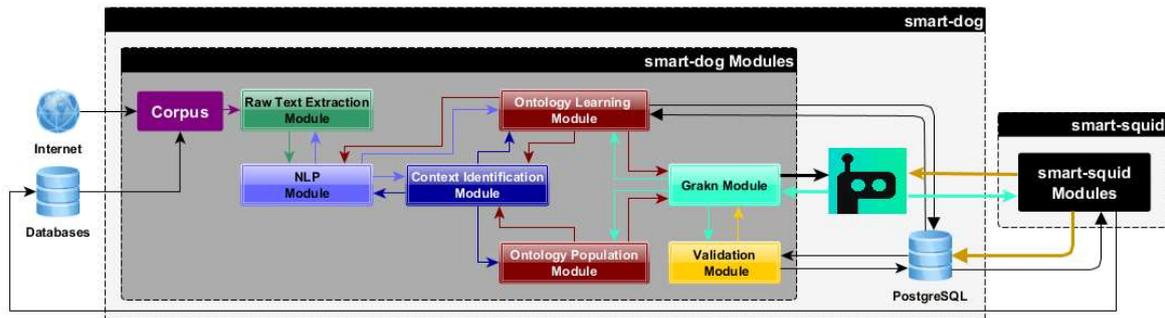


Figure 13. smart-dog modules

4.2.1. OLC approach and criticalities with the type of documents

Ontology learning (OL) is defined as the set of methods and techniques used for building an ontology from scratch, enriching, or adapting an existing ontology in a semi-automatic fashion using several sources. Ontology learning techniques rely on methods from various fields such as Machine Learning, knowledge acquisition, Natural Language Processing (NLP), statistics, and information retrieval. Such techniques facilitate and support the construction of ontologies by the ontology engineer. This is the reason why ontology learning frameworks have been developed in the last years and integrated with standard ontology engineering tools. Ontology learning can be applied to unstructured, semi-structured and fully structured data to support semi-automatic and cooperative ontology engineering. [7] The goal of OL is to guarantee automatic extraction of knowledge, possibly structured as composite or straightforward statements, from a given corpus of textual documents, to form an ontology. Many OL approaches follow a model named the Ontology Learning Layer Cake (OLC) [8], and share many features, such as statistical based information retrieval, machine learning, and data and text mining, resorting to linguistics based techniques for specific tasks. The Ontology Learning Layer Cake model aims at learning ontologies by using a multistep approach [9].

The goal of the smart-dog framework is to allow the generation and the population of the KG. [12] The multistep approach used for the OLC satisfies the needs for the generation part, but, for the population part in order to make the process the most automatic possible, it requires the extraction of lexico-syntactic patterns for the different data to be extracted. This is why we refer to a modified OLC. In other words, we want to consider also a metadata layer that will be used to extract patterns relevant for the ontology population phase and stored with the concept inserted into the knowledge graph.

As we mentioned in [12], the users of the DEA would like to have information coming from several types of sources that can be queried through the DEA UI. These sources are provided in different formats (.pdf, .pptx, .docx), forms (structured, semi-structured and unstructured) and they require specific solutions in order to be dealt.

In order to be able to deal with any source, we need a robust NLP Module and a robust Context Identification Module. Therefore, we decided to focus first on reports to be mined by the smart-dog, which are the output of phase 0 of the space mission design, final output of a CDF study at ESA. The reports contain information about the different disciplines in the space mission design, not only text but also figures, tables and formulas. In order to develop the Context Identification Module, the extraction of the different thesauri for the different disciplines shall be defined in order to allow the DEA to recognize the type of source and context and be able to deal with all other types of sources. Due to the choice of the source, it is necessary to split the reports into their subchapters and understand all the type of disciplines that can be identified and then define the thesauri. This pre-processing step is shown in section 5.2. The Raw Text Extraction Module rely on the TIKIA library which is not able to separate the chapters in the reports. The solution foresaw the implementation of pattern recognition in order to identify the single chapters inside the reports, but it is something that the Context Identification Module will be able to do once trained for the different disciplines. In the reports, it is possible to identify the main disciplines always included in the design and the one specific for some missions. However, for the training, it is necessary to give more information to generate the KG. The extraction of chapters is strictly related to the type of sources we selected because we focus first on one subsystem, but it is fundamental for the training of the Context Extraction module which will act not only as context identifier but as a filter.

Once we have the knowledge required for each discipline, we can test the extraction of information with other types of sources, such as requirements documents, presentations or data sheets. A first attempt in the extraction of data from datasheets have been made in [11], including issues to be solved. The approach we foresee will improve the extraction and reduce the number of issues.

4.2.2. Terms and Synonyms Extraction

Terms are the most basic building blocks in ontology learning. Terms can be simple (i.e., single word) or complex (i.e., multi-word), and are considered as lexical realizations of everything essential and relevant to a domain. The main tasks associated with terms are to *preprocess texts* and *extract*

terms. The preprocessing task ensures that the input texts are in a format supported by the ontology learning system. The extraction of terms, known as *term extraction* or *keyphrase extraction*, typically begins with tokenization or part-of-speech tagging to break texts into smaller constituents. Statistical or probabilistic measures are then used to determine the collocational stability of a noun sequence to form a term, also known as *unithood*, and the relevance or specificity of a term for a domain, also known as *termhood*. [8]. A Multi-Word Term (MWT) is a term that is composed of more than one word. The unambiguous semantics of a multi-word term depends on the knowledge area of the concept it describes and cannot be inferred directly from its parts [40].

Several algorithms can be adopted for the terms extraction and in particular for multi-word extraction: Counting Frequencies of terms, Tf.idf, Domain Consensus, NC Value, Domain Coherence, Syntactic patterns, probabilistic algorithms [39]. For multi-word extraction we decided to implement the C/NC Value method [40], the preliminary results are shown in section 5.3.

Because this process is iterative and it needs to be tuned carefully to guarantee reliable results, the next step is an improvement with the introduction of validated multi-words and synonyms.

The synonym level addresses the acquisition of semantic term variants in a language. An essential aspect of this work is the identification of the appropriate sense of the term in question, which determines the set of synonyms that are to be extracted. This involves standard word sense disambiguation algorithms. However, specifically in the ontology learning context, researchers have exploited the fact that ambiguous terms have precise meanings in particular domains allowing for an integrated approach to sense disambiguation and domain-specific synonym extraction. In contrast to using readily available synonym sets such as provided by WordNet and related lexical resources, researchers have also worked on algorithms for dynamic acquisition of synonyms by clustering and related techniques. On this basis, much work has been done on synonym acquisition from text corpora that is based on the distributional hypothesis that terms are similar in meaning to the extent to which they share syntactic contexts. [8]

Word sense disambiguation is the ability to computationally determine which sense of a word is activated by its use in a particular context [29]. This task is fundamental in the NLP Module to perform precise normalization in the text and the KG population phase because it requires a reliable KG able to boost the automatic extraction of data in a precise and meaningful way. Without knowledge, it would be impossible for both humans and machines to identify the meaning [29]. The extraction of synonyms is not trivial and an open field of research. Different approaches have been identified, but they require more in-depth analysis in order to be implemented in the smart-dog framework and adopted for the domain considered.

4.2.3. *Uncertainty in the back-end part*

In Expert systems, the word uncertainty can be related to inaccurate data, imprecise information or reliability of the results. An ES allows the User to assign probabilities, certainty factors, or confidence levels and many more techniques to any or all input data [37]. Regarding the back-end part, several sources have been identified. The main one is inherently related to the type of data used for building the knowledge graph, which are text data coming from different type of sources. The text is written in a natural language which is inherently ambiguous and imprecise as we mentioned in [12]. Moreover, the numerous ambiguities in natural language are rarely clarified during translation to a formal language. As a result, rules that are not expressed precisely in the formal language can be misinterpreted, and this creates inconsistencies that we have to take into account during the automation of the extraction process. This source of uncertainty needs to be addressed with the development of accurate conversion between the natural language and the data modeling language adopted for the DEA. In order to reduce the lack of precision in descriptive language, there has been some attempt in the use of Controlled Natural Language (CNL). CNL is a subset of natural language with restricted grammar and vocabulary in order to reduce the ambiguity and complexity inherent in full natural language. One example of a tool that relies on this method is Fluent Editor [38], but we discarded this option.

The complexity of understanding the natural language is a non-trivial task even for human interpretations. The DEA aims at developing a robust NLP and words sense disambiguation to deal with semantic variety and context identification. The documents, in general, are written by different experts. Therefore we have to deal with redundant information, inconsistencies or incompleteness of data. We instead assign null values for to incomplete or missing data for a specific entity or attribute, and we deal with this imperfections relying on an inconsistency resolution engine able to flag inconsistencies and eventually solved them if it has enough information. Every type of source will be provided with a level of fidelities that will be analyzed by the front-end part of the DEA. The front-end part will indeed inherit the uncertainty factor inserted into the KG in order to perform a complete analysis of the uncertainties in order to provide the most reliable and accurate answer to the users together with the level of uncertainty linked to the result.

Using the natural language facts are described with such terms as often and sometimes, frequently and hardly ever. As a result, it can be challenging to express knowledge in the precise IF-THEN form of production rules. Therefore it is necessary to adopt fuzzy rules and fuzzy functions able to convert natural language into information that can be interpreted by the machine. Moreover, inferring knowledge from incomplete data can be taken into account adopting fuzzy rules and performing fuzzy inference through the adoption of fidelity factors.

Figure 14 summarizes the sources and possible mitigations described above.

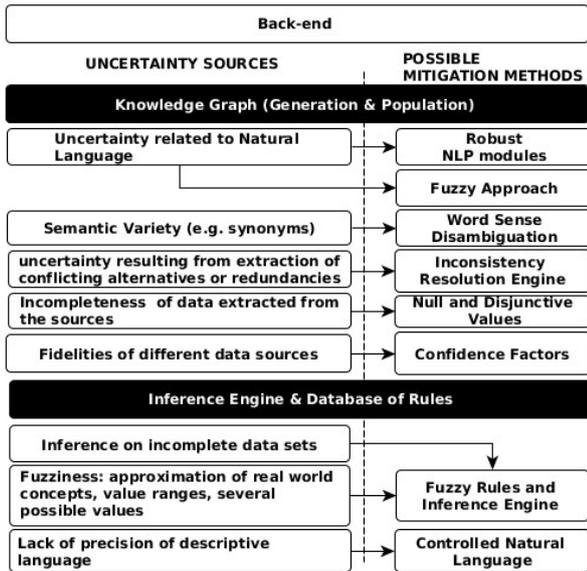


Figure 14. back-end uncertainties sources and possible mitigations methods.

5. PRELIMINARY RESULTS

This section focuses on the preliminary results for the pre-processing part required for the creation of the Context Identification Module, therefore on the preparation of the training set to be used for context identification. This approach is strictly dependent on the type of sources we decided to adopt, which contains information from many disciplines which are difficult to be parsed with a generalist approach focused on general English vocabulary. The second part of the section instead shows the preliminary results in the multi-word extraction task in order to improve the NLP Module and the overall recognition of entities inside the text.

5.1. Chapters Extraction

Using pattern recognition for the titles of the chapters in the reports, we separated them and the corresponding raw text to be given as input to the NLP Module in smart-dog. This solution is necessary for training the ML algorithms to recognize the different disciplines but also to create a test case for the Context Identification Module to identify the part of the documents and extract automatically specific parts depending on the context or the discipline considered. Figure 15 shows the chapters extracted from 9 reports available online of different type of missions from 2004 to 2015 [44]. It is possible to notice that there is no standard name for the disciplines. Therefore, it is a good test base to apply clustering. The final goal foresees the development of an algorithm able to automatically extract relevant parts from different sources for any specific space mission discipline using as input for training the words and synonyms extracted for each discipline. However, we will first focus on a single discipline.

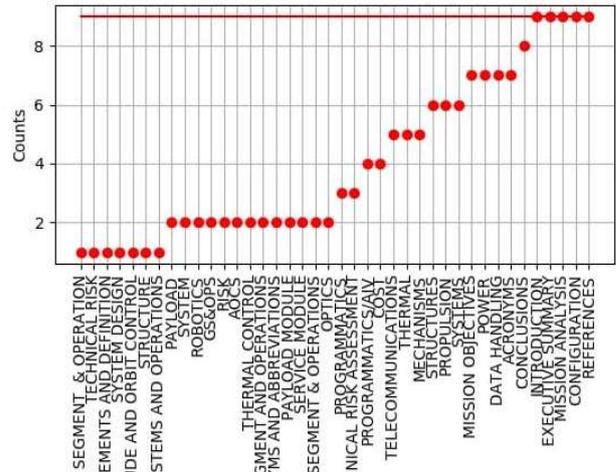


Figure 15. chapters extracted from mission reports

5.2. Multi words extraction

Figure 16 shows the results for 2-grams of the multi-words identified implementing the C Value Method. The linguistic filters selected according to the space mission domain are: ['NN', 'JJ'], ['JJ', 'NN'], ['NN', 'NN'], ['NN', 'NN', 'NN'] and ['JJ', 'NN', 'NN'], where NN and JJ are respectively Nouns and Adjectives according to the Penn Treebank list of part-of-speech tags. These results are preliminary and shall be run on a bigger corpus and validated. The same will be done for 3-grams. Moreover, the NC Value method shall be evaluated.

Multi words will improve the normalization for the NLP module and also give the power to the DEA to understand the compound inside a text, not just simple words uncorrelated. In this way also the space mission design vocabulary can be identified.

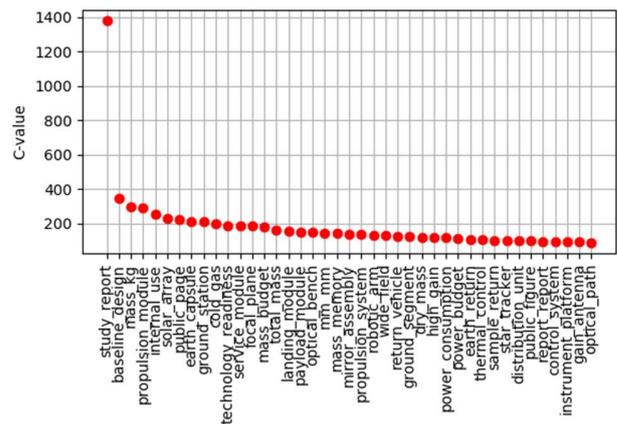


Figure 16. 2-grams extraction for multi-words in the corpus using the C value method

6. CONCLUSION

After reminding the outline of the DEA project, the paper presented the detailed software architectures of both front and back-ends as well as the tool requirements. Architectures and requirements were defined based on the literature reviews and a set of experts interviews led at ESA, ESTEC in July-August 2018. The experts involved in the interviews had experience in concurrent engineering for space mission feasibility studies. They all acknowledged the potential of the DEA, first as a knowledge engine then as an integrated design assistant, to facilitate KM and knowledge reuse in their field. The experts' attitude w.r.t the DEA project was encouraging concerning the success of the tool integration into their work process.

The preliminary results detailed in section 5.2 display the importance of the multi-word extraction in order to guarantee the recognition of compounds in the text, otherwise not recognizable by the tool. The C Value Method shall be improved with the NC Value Method. The next step foresees its implementation and the test with a more significant corpus related to one discipline, also considering other types of material, like books or datasheets.

For the back-end part, the next steps foresee the validation of the multi-word terms with the selection of more linguistic filters and with a more significant corpus; the implementation

of solutions for the synonyms layer; implementation and test of the context identification module.

For the front-end, the next steps will focus on the extraction of structured data from the design environments of OCDT and CDP4 and the assimilation to the DEA KG based on the DEA ontology. The next steps will also focus on implementing the first modules of the query manager: generating Graql queries from complex User NL queries.

APPENDIX – VISUALISER OUTPUTS

This Appendix includes more details on the Grakn.ai example generated with Grakn 1.3.0 to illustrate the architecture of smart-squid in 4.1. Figures 17 and 18 corresponds respectively to the visualization of the User queries "Show all Earth Observation missions" and "Show all satellites with SAR payloads". The code used to generate the KG of the example can be provided upon request to the authors.

ACKNOWLEDGMENTS

The DEA project started in January 2018 and will last three years. The team is based at the Intelligent Computational Engineering (ICE) lab of the University of Strathclyde, Glasgow (UK) and is supported by partners: ESA, Airbus, Rhea, and satsearch. The DEA team would like to warmly thank their partners for their valuable support.

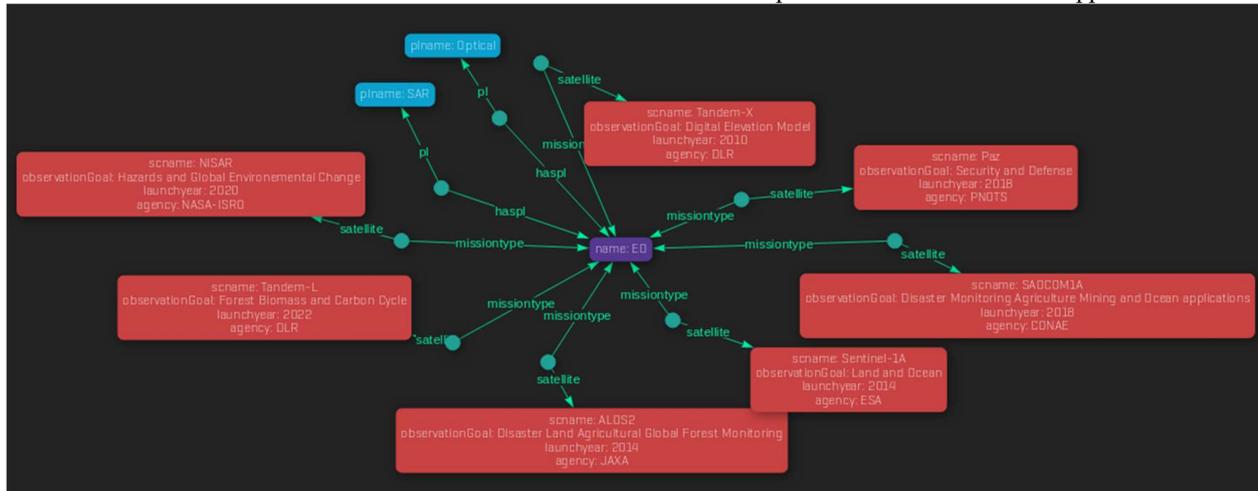


Figure 17. Grakn visualizer output for query "Show all Earth Observation missions"

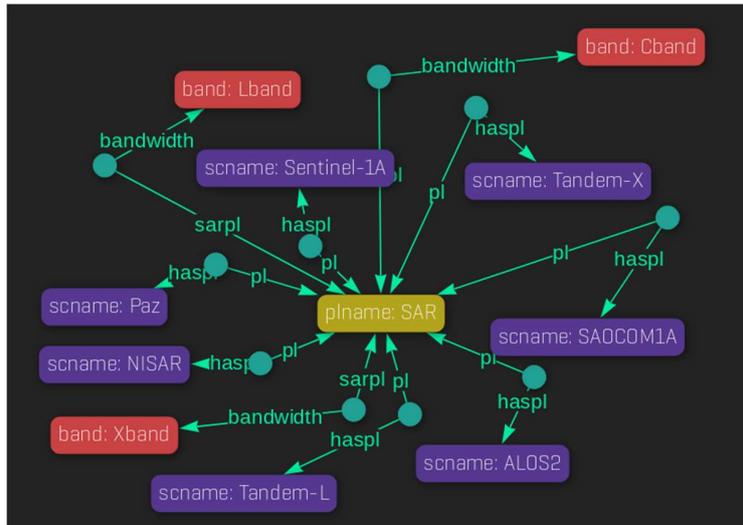


Figure 18. Grakn visualizer output for query "Show all Earth Observation missions"

REFERENCES

- [1] MARAKAKIS, E., VASSILAKIS, K., KALIVIANAKIS, E., & MICHELOYIANNIS, S. (2005). EXPERT SYSTEM FOR EPILEPSY WITH UNCERTAINTY 1, (DECEMBER), 19–21.
- [2] GIANNOULIS, M., KONDYLAkis, H., & MARAKAKIS, E. (2018). COSMOS: A WEB-BASED, COLLABORATIVE KNOWLEDGE SYSTEM USING ONTOLOGIES AND MANAGING UNCERTAINTY. PROCEEDINGS OF THE 11TH PERVERSIVE TECHNOLOGIES RELATED TO ASSISTIVE ENVIRONMENTS CONFERENCE ON - PETRA '18, (JUNE), 441–448.
- [3] MOTRO, A., & SMETS, P. (1997). *UNCERTAINTY MANAGEMENT IN INFORMATION SYSTEMS -- FROM NEEDS TO SOLUTIONS*.
- [4] SONAL DUBEY, R. K. PANDEY, & S. S. GAUTAM. (2014). DEALING WITH UNCERTAINTY IN EXPERT SYSTEMS. *INTERNATIONAL JOURNAL OF SOFT COMPUTING AND ENGINEERING (IJSCE)*, 4(3), 105–111.
- [5] WALLEY, P. (1996). *MEASURES OF UNCERTAINTY IN EXPERT SYSTEMS. ARTIFICIAL INTELLIGENCE (VOL. 83)*.
- [6] MOTRO, A. (1992). SOURCES OF UNCERTAINTY IN INFORMATION SYSTEMS. *PROCEEDINGS OF THE WORKSHOP ON UNCERTAINTY*, 1–18.
- [7] ABEER, A., ABDULMALIK, A.. (2015). ONTOLOGY CONSTRUCTION FROM TEXT: CHALLENGES AND TRENDS.
- [8] STAAB, S.; STUDER (2007). *HANDBOOKS ON ONTOLOGIES*
- [9] BROWARNIK A., MAIMON O. (2015). ONTOLOGY LEARNING FROM TEXT DEPARTING THE ONTOLOGY LAYER CAKE
- [10]. F. MURDACA, A. BERQUAND, A. RICCARDI, T. SOARES, N. BRAUER, AND K. KUMAR, "ARTIFICIAL INTELLIGENCE FOR EARLY DESIGN OF SPACE MISSIONS IN SUPPORT OF CONCURRENT ENGINEERING SESSIONS," IN *SECESA*, 2018, NO.
- [11] F. MURDACA, A. BERQUAND, A. RICCARDI, T. SOARES, N. BRAUER, AND K. KUMAR, "KNOWLEDGE-BASED INFORMATION EXTRACTION FROM DATASHEETS OF SPACE PARTS," IN *SECESA*, 2018, NO. 1.
- [12] A. BERQUAND, F. MURDACA, A. RICCARDI, T. SOARES, N. BRAUER, AND K. KUMAR, "TOWARDS AN ARTIFICIAL INTELLIGENCE-BASED DESIGN ENGINEERING ASSISTANT FOR THE EARLY DESIGN OF SPACE MISSIONS AUDREY BERQUAND," IN *69TH INTERNATIONAL ASTRONAUTICAL CONGRESS (IAC)*, 2018, NO. OCTOBER, PP. 1–5
- [13] DALKIR, K. (2017). *KNOWLEDGE MANAGEMENT IN THEORY AND PRACTICE*. (THE MIT PRESS, ED.) (3RD ED.). THE MIT PRESS.
- [14] PETER J.F. LUCAS & LINDA C. VAN DER GAAG. *PRINCIPLE OF EXPERT SYSTEMS*, 1991
- [15] J.L. SMITH, *CONCURRENT ENGINEERING IN THE JET PROPULSION LABORATORY PROJECT DESIGN CENTER* (1997)
- [16] ESA OCDT COMMUNITY PORTAL LOGIN [HTTPS://OCDT.ESA.INT/](https://ocdt.esa.int/) (ACCESSED 17.09.18).
- [17] ECSS, "ECSS-E-TM-10-25A – ENGINEERING DESIGN MODEL DATA EXCHANGE (CDF) – (20 OCTOBER 2010) EUROPEAN COOPERATION FOR SPACE STANDARDIZATION."
- [18] ALOS-2 WEBSITE: <https://directory.eoportal.org/web/eoportal/satellite-missions/a/alos-2>
- [19] NISAR WEBSITE: [HTTPS://NISAR.JPL.NASA.GOV/](https://nisar.jpl.nasa.gov/)
- [20] PAZ WEBSITE: [HTTPS://DIRECTORY.EOPORTAL.ORG/WEB/EOPORTAL/SATELLITE-MISSIONS/P/PAZ](https://directory.eoportal.org/web/eoportal/satellite-missions/p/paz)
- [21] SAOCOM-1A WEBSITE: [HTTPS://DIRECTORY.EOPORTAL.ORG/WEB/EOPORTAL/SATELLITE-MISSIONS/S/SAOCOM#MISSION-STATUS](https://directory.eoportal.org/web/eoportal/satellite-missions/s/saocom#mission-status)
- [22] SENTINEL 1A WEBSITE: [HTTPS://DIRECTORY.EOPORTAL.ORG/WEB/EOPORTAL/SATELLITE-MISSIONS/C-MISSIONS/COPERNICUS-SENTINEL-1](https://directory.eoportal.org/web/eoportal/satellite-missions/c-missions/copernicus-sentinel-1)

- [23] TANDEM-L WEBSITE: [HTTPS://WWW.TANDEM-L.DE/](https://www.tandem-l.de/)
- [24] TANDEM-X WEBSITE:
[HTTPS://DIRECTORY.EOPORTAL.ORG/WEB/EOPORTAL/SATELLITE-MISSIONS/T/TANDEM-X](https://directory.eoportal.org/web/eoportal/atellite-missions/t/tandem-x)
- [25] GRAKN BLOG ON QUERY CONSTRUCTION:
[HTTPS://DEV.GRAKN.AI/ACADEMY/GET-QUERIES.HTML](https://dev.grakn.ai/academy/get-queries.html)
- [26] ABBURU S. (2012) A SURVEY ON ONTOLOGY REASONERS AND COMPARISON. INTERNATIONAL JOURNAL OF COMPUTER APPLICATIONS VOL 57 No 17.
- [27] ALATRISH E. (2013). COMPARISON SOME OF ONTOLOGY EDITORS. MANAGEMENT INFORMATION SYSTEMS. VOL 8 No.2, pp 018-024.
- [28] SHAMSFARD M. BARFOROUSH A. A.(2003) THE STATE OF THE ART IN ONTOLOGY LEARNING: A FRAMEWORK FOR COMPARISON. THE KNOWLEDGE ENGINEERING REVIEW, VOL. 18:4, 293–316, CAMBRIDGE UNIVERSITY PRESS
- [29] NAVIGLI R.(2009) WORD SENSE DISAMBIGUATION: A SURVEY. ACM COMPUTING SURVEYS, VOL. 41, No. 2, ARTICLE 10.
- [30] LIU, T.-Y. (2011). *LEARNING TO RANK FOR INFORMATION RETRIEVAL*. SPRINGER.
- [31] RICHARDSON, M., PRAKASH, A., & BRILL, E. (N.D.). BEYOND PAGERANK: MACHINE LEARNING FOR STATIC RANKING.
- [32] BURGES, C. J. C. (2010). *FROM RANKNET TO LAMBDA RANK TO LAMBDA MART: AN OVERVIEW*.
- [30] LIU, T.-Y. (2011). *LEARNING TO RANK FOR INFORMATION RETRIEVAL*. SPRINGER.
- [31] RICHARDSON, M., PRAKASH, A., & BRILL, E. (N.D.). BEYOND PAGERANK: MACHINE LEARNING FOR STATIC RANKING.
- [32] BURGES, C. J. C. (2010). *FROM RANKNET TO LAMBDA RANK TO LAMBDA MART: AN OVERVIEW*.
- [33] CDP4: [HTTPS://WWW.RHEAGROUP.COM/NEWS/CDP4-OPEN-SOURCE-COMMUNITY-EDITION](https://www.rheagroup.com/news/cdp4-open-source-community-edition)
- [34] RADLINSKI, F., & JOACHIMS, T. (2005). QUERY CHAINS: LEARNING TO RANK FROM IMPLICIT FEEDBACK.
- [35] MOTRO, A. (1992). SOURCES OF UNCERTAINTY IN INFORMATION SYSTEMS. *PROCEEDINGS OF THE WORKSHOP ON UNCERTAINTY*, 1–18.
- [36] ZADEH, L. A. (1983). THE ROLE OF FUZZY LOGIC IN THE MANAGEMENT OF UNCERTAINTY IN EXPERT
- [37] SONAL D.; PANDEY, R. K.; GAUTAM, S. S.(2014) DEALING WITH UNCERTAINTY IN EXPERT SYSTEMS
- [38] A. SEGANTI, P. KAPLANSKI, P. ZARZYCKI (2015) COLLABORATIVE EDITING OF ONTOLOGIES USING FLUENT EDITOR AND ONTORION
- [39] ASTRAKHANTSEV, N. A.; FEDORENKO, D. G.; TURDAKOV, AND D. YU.METHODS FOR AUTOMATIC TERM RECOGNITION IN DOMAIN-SPECIFIC TEXT COLLECTIONS A SURVEY (2015)
- [40] FRANTZI, KATERINA; ANANIADOU, SOPHIA; MIMA, HIDEKI (2000) AUTOMATIC RECOGNITION OF MULTI-WORD TERMS: THE C-VALUE/NC-VALUE METHOD
- [41] MOTRO, A. (1990). FLEX: A TOLERANT AND COOPERATIVE USER INTERFACE TO DATABASES.

IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, 2(2), 231–246.

[42] ICHIKAWA, T., & HIRAKAWA, M. (1986). ARES: A RELATIONAL DATABASE WITH THE CAPABILITY OF PERFORMING FLEXIBLE INTERPRETATION OF QUERIES. *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING*, SE-12(5), 624–634.

[43] CHITRE, N. (2016). SEMANTIC WEB SEARCH ENGINE. *INTERNATIONAL JOURNAL OF ADVANCE RESEARCH IN COMPUTER SCIENCE AND MANAGEMENT STUDIES*, 4(7).

[44]

[HTTP://WWW.ESA.INT/OUR_ACTIVITIES/SPACE ENGINEERING TECHNOLOGY/CDf/CDf_REPORTS](http://www.esa.int/Our_Activities/Space_Engineering_Technology/CDf/CDf_Reports)

BIOGRAPHY



Audrey Berquand is a PhD student from the Intelligent Computational Engineering (ICE) lab at the University of Strathclyde. Audrey received her diplôme d'ingénieur from EPF, France and her MSc in Aerospace Engineering from KTH, Sweden. She graduated from the Space Studies Program of the International Space University in Summer 2017. Before joining the ICE lab, she worked three years for Earth Observation at the European Space Agency (ESA). Her work focuses on the front-end of the DEA project and on the assimilation of structured design environment data. Her PhD is half-funded by ESA in the frame of a Networking Partnership Initiative (NPI).



Francesco Murdaca is a PhD student at the Department of Mechanical and Aerospace Engineering of the University of Strathclyde. He received both his Bachelor and Master from Politecnico di Milano in Aerospace Engineering. Before moving to Glasgow, he worked for a space company based in Madrid, Deimos Space, to develop a tool for one subsystem (C&DH) of the satellite to be used in a Concurrent Design Facility (CDF). His PhD focuses on the back-end of the DEA project.



Dr. Annalisa Riccardi is a Lecturer in Computational Intelligence at the Mechanical and Aerospace Department of the University of Strathclyde. She has more than eight years of experience in optimization techniques and applications. She is currently involved in projects on data-driven modeling and decision making for engineering design.