Informatics

Semantics-Based Exploratory Search on Manufacturing Knowledge

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Software Engineering & Internet Computing

by

Kevin Haller, BSc.
Registration Number 01325694

to the Faculty of Informatics

at the TU Wien

Advisor: Ao.Univ.Prof. Dipl.-Ing. Mag.rer.soc.oec. Dr.techn. Stefan Biffl
Assistance: Univ.Prof. Marta Sabou, PhD.

Vienna, 1st June, 2024

Kevin Haller | Stefan Biffl

Erklärung zur Verfassung der Arbeit

Kevin Haller, BSc.

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 1. Juni 2024

Kevin Haller

Acknowledgements

Firstly, I want to thank Prof. Stefan Biffl for his supervision of my master thesis and his recommendation of the book "Science Research Writing For Non-Native Speakers of English" by Hilary Glasman-Deal. It was a great help to me in terms of the improvement of my writing. My thanks go also to Prof. Marta Sabou. The finalization of this thesis took much longer than it should have, and their patience with me is highly appreciated.

I would also like to thank all those who took part in the informal interviews, and not forgetting the user study, for taking valuable time out of their busy lives. This goes to: Annisa, Felix, Florina, Laura, Peb, Tudor, Thomas and Sebastian.

Finally, I would like to thank my supportive family and the medical doctors at the Sisters of Mercy Hospital in Vienna, especially Dr. Maximilian Emich and Dr. Magdalena Neuhauser. I would like to also express my special appreciation for Dr. Martin Scharf, who managed to get me into well-needed hospital care in times when the COVID-19 pandemic was overwhelming our medical system. This thesis would still be unfinished without their help and support.

Abstract

Knowledge graphs have been identified within the manufacturing industry as a suitable technology for integrating multidisciplinary knowledge from heterogeneous data sources. Effectively reusing this knowledge can enhance decision-making processes, and thereby fostering a competitive edge. However, there has been limited attention given in production research to the active human involvement in exploring these manufacturing knowledge graphs. In this context, exploratory search systems present a promising solution to encourage such engagement.

Nevertheless, the majority of exploratory search systems are designed for general knowledge graphs where common knowledge suffices. Within the multifaceted and complex landscape of manufacturing, it is crucial to focus more closely on specific exploratory search features. Drawing from interviews with domain experts, this thesis highlights three distinct features that boost the efficacy of exploratory search for manufacturing environments. These include the system's ability to tailor itself to numerous engineering perspectives, the availability of transparent provenance information that notably assists engineers in their investigative work, and the need for comprehending as well as navigating complex and deep hierarchical structures.

An exploratory search system (ESS) has been designed, informed by an examination of common interface paradigms for generic exploratory search systems, coupled with an exploratory analysis of paradigms equipped to meet the identified requirements for the manufacturing domain. This system encompasses a rendering engine that supports an adaptive user interface, enabling a multifaceted configuration of visualizations and the underlying search algorithms. To address the need for provenance visibility, the system is structured to disseminate named graph information pertinent to each statement to the relevant UI components. A simple tree view was employed to overcome the challenge of navigating through hierarchical structures.

This ESS prototype was evaluated on a small scale with five participants for the pilot factory use case. The initial assessment of the ESS demonstrated its value for users executing exploratory search tasks in the domain of collaborative robotics. Potential areas for enhancement were identified, however. Users perceived the tree view design as not useful, and they frequently opted for managing multiple browser tabs over utilizing the memorization feature. Despite these potential areas for improvement, users were overall successful in achieving their search objectives.

Contents

CHAPTER 1

Introduction

The fusion of traditional manufacturing technology with modern information and communication technology is guiding a new paradigm shift in manufacturing, which is commonly referred to as the *fourth industrial revolution* [HA18]. This shift is investigated by several strategic initiatives such as "Industrie 4.0" in Germany [BHVH14], "Society 5.0" in Japan [Fuk18], or "MIC 2025" in China [WMZ+16]. Core to all of these initiatives is the digitization of multidisciplinary knowledge about production systems and processes.

Knowledge graphs (as defined in Section 2.3) are a promising solution to facilitate this digital transformation. As shown by the survey of Li et al. [LLW+21], knowledge graphs have received considerable attention in production research over the last years. In fact, it has been widely recognized as an important component of the next generation of information systems for manufacturing [LLW+21]. He et al. [HJ19] identify knowledge graphs as a technology to increase the reuse of manufacturing knowledge inside of a company. The effective reuse of manufacturing knowledge can better inform engineers in their decision making processes and consequently, establish a competitive advantage [ZZL+20]. Industrial enterprises such as Bosch or Siemens started to internally construct knowledge graphs related to manufacturing [HLHH18][STF+19].

Problem. While the manufacturing industry consists of a diverse range of sectors, all enterprises face similar issues with information management and sharing. Decision making is guided by information originating from many different sources, and different types of decision have to be carried out. Organizational groups inside manufacturing enterprises (e.g. design, production, maintenance, etc.) produce as well as select their own type of information and might have to gain knowledge from diverse sources to form a decision [PUJ+18]. The knowledge reuse for decision making is furthermore low, due to the complexity of production processes as well as the lack of clear standardization and formalization to knowledge in this field [HJ19].

A suitable search system over manufacturing knowledge graphs could increase the reuse. However, a lot of production research is dedicated to the utilization of knowledge graphs for autonomous decision making systems, e.g., Rožanec et al. [RLR+22]. In contrast, less attention has been given to the creative participation of engineers in the exploration of manufacturing knowledge graphs. In particular, we consider two concrete use cases in two different organizations in which the need for such a search system arises.

(UC1) The first organization is a production plant manufacturer[1] from the metallurgy domain. In this company, several engineering disciplines work together to build a production system. An engineering activity is initiated across several disciplines such as mechanical and electrical engineering. In order to make well-informed design decisions for a planned production plant, engineers with potentially different perspectives using distinct tools and vocabularies within the company have to necessarily exchange engineering information, which constitutes a challenge. Simulation experts are tasked with the simulation of designed production plants, and subsequently, the optimization with respect to several key indicators. Yet, obtaining a coherent and complete system design to be used as input for simulation currently requires tedious manual work by a simulation experts to identify, reconcile and merge different versions and viewpoints on the same system component provided by engineers from diverse disciplines.

(UC2) Secondly, researchers from the Aspern pilot factory collaborate as part of various research projects. The Aspern factory is one of several Austrian "learning and experimentation factories" initiated and co-funded by the Austrian Federal Ministry of Transportation, Innovation, and Technology [HSSW16]. It currently hosts a series of valuable collaborative and industrial robotic arms as well as a wide range of supporting tools (grippers, 3D cameras, projectors, etc.), which can be used by students, researchers, and companies for their own purposes. The detailed capabilities of these machines are, however, unknown to a wider, potentially interested audience because (1) they are not publicly available, (2) they depend on the application-context (e.g., so-called robotic skills like assembling, drilling, screwing, etc. which are implemented in software), and (3) they continuously evolve and change. In this context, interested students, researchers, and companies are unaware of the availability and capabilities of the manufacturing technology in the Aspern factory, which contributes to a relatively low usage degree of these expensive, state of the art production machines.

Solution. Systems that enable exploratory search over manufacturing knowledge graphs can support engineers in their decision and sense making. Exploratory search (see Section 2.4) is an open-ended and multi-faceted information-seeking activity. It is commonly used in the context of scientific discovery, learning and decision making [WR09]. However, most publicized exploratory search systems focus on general knowledge graphs for which common knowledge is sufficient. We argue that within the complex environment of manufacturing, closer attention has to be paid to particular exploratory search features.

[1] Use case partner cannot be named, due to confidentiality agreements.

Contribution. To that end, this thesis investigates the three research questions outlined in the subsequent Section 1.1. The methodology of this thesis is discussed in Section 1.2, while the structure of the remaining thesis is outlined in Section 1.3.

1.1 Research Questions & Aims

This thesis endeavors to investigate ways of leveraging exploratory search mechanisms to assist engineers in the domain of manufacturing. The thesis will probe into the subsequent three research questions:

RQ I. ***What are special requirements for exploratory search in the manufacturing domain?***

In the multifaceted landscape of manufacturing, engineers may have particular needs pertaining to an exploratory search system. These needs might not be exclusively unique to manufacturing professionals, but are often overlooked in the design of exploratory search systems for general domains. The majority of publicized exploratory search systems are intended for use on general knowledge graphs, where common knowledge suffices, and the domain of engineering and manufacturing has not received substantial attention to date.

Aim. Informal interviews will be held with stakeholders from the partnering organizations, which are from the field of manufacturing. This approach aims to collect insights into potential requirements that, while not exclusive to manufacturing professionals, may often be overlooked in the design of exploratory search systems for more general domains.

RQ II. ***Which of the common exploratory mechanisms are most suitable to address the needs of the two presented use cases? What interface paradigms are of particular interest for engineers in manufacturing?***

Over time, a variety of exploratory search systems have been publicized, experimenting with diverse search and interface paradigms. While certain semantic-based exploratory search systems such as Aemoo [NPG$^+$17], for example, visualize entities and their relationships as a graph, others try to mask the graph structure of the underlying data model. This thesis, in particular, is focused on facilitating exploratory search within manufacturing knowledge. As such, it seeks to pinpoint those interface paradigms best equipped to aid engineers in a manufacturing environment.

Aim. Several recurring interface paradigms are found among the exploratory search systems publicized in the past two decades. These common paradigms will be thoroughly examined bearing in mind that this thesis concentrates on the manufacturing domain. Drawing on insights gained from investigating research question RQI with the partnering organizations, interface paradigms will be proposed and highlighted that could be particularly beneficial for the manufacturing domain.

RQ III. ***What effect do the selected interface paradigms have on the exploration utility of the search system, when applied on manufacturing knowledge?***

An assessment of the interface paradigms selected from research question RQII is required to determine their capability to aid engineers in navigating manufacturing knowledge. This thesis will refer to this capability as exploration utility. As previously stated, most publicized exploratory search systems concentrate on general knowledge graphs, leading to the question of whether the commonly used performance indicators in the evaluation of these systems are also applicable for exploration within the manufacturing domain.

Aim. An exploratory search system prototype (ESS) will be built and set for assessment using a robust evaluation methodology. The key focus of the evaluation is to ascertain the system's capability to enable a user to actively explore, learn, and understand a given subject. Emilie Palagi, in her PhD thesis [Pal18], suggested an evaluation model that is suitable for assessing the exploratory process rather than concentrating on a single key performance indicator such as recall, precision, or speed. Additionally, Al-Tawil et al. [ADT20] proposed a methodology for designing an exploratory search system evaluation using Bloom's taxonomy as a framework to assess the usefulness of an exploration. The evaluation methodology for the ESS prototype will be crafted based on the work of Palagi and Al-Tawil et al.

1.2 Methodology

This thesis is following the methodology outlined briefly in the subsequent description.

1. **Requirements Survey:** Informal interviews were conducted with engineering professionals from the two collaborating organizations. In these interviews, expectations towards the search system as well as specific requirements for their manufacturing domain were discussed to address RQI.
2. **Design & Implementation of ESS prototype:** Building upon the gathered requirements from the informal interviews and a literature study of exploratory search systems, an exploratory search system with the main goal of assisting engineers in exploring manufacturing knowledge was designed. The designed system assumes that an ontology-based data access is provided to the manufacturing data. It was implemented following a microservice architecture and it shall not only be flexibly adaptable to any manufacturing use case, but it shall also be agnostic to the legacy information management systems maintaining the manufacturing data.
3. **Evaluation:** An evaluation model was constructed for the designed exploratory search system based on the research of Emilie Palagi [Pal18], and Al-Tawil et al. [ADT20]. This model was applied to our system and the pilot factory use case (UC2). In order to get a meaningful result from the evaluation, 5 participants were invited to take part in user experiments.

1.3 Structure of Work

The remainder of this thesis is structured as follows:

Chapter 2

This chapter outlines the required background knowledge for the remaining thesis, and relevant related work for our three research questions. Section 2.1 takes a look at the Resource Description Framework (RDF) and Section 2.2 briefly describes ontologies, which are both fundamental technologies of the Semantic Web. Section 2.3 defines how the term knowledge graph is meant to be understood in this thesis. A deeper dive into exploratory search is taken in Section 2.4. The chapter is concluded with Section 2.5, which outlines related work about exploratory search in manufacturing.

Chapter 3

This chapter addresses the first research question, RQI. Firstly, Section 3.1 provides an in-depth description of the two manufacturing use cases, namely, the use case concerning the production plant manufacturer (UC1) and the use case concerning the pilot factory (UC2). Subsequently, Section 3.2 lays out the requirements for the exploratory search system, as gathered in interviews from stakeholders of both use cases. The chapter concludes with Section 3.3, where it underscores three manufacturing-specific requirements and delves into their details.

Chapter 4

This chapter aims to answer the second research question, RQII. Initially, in Section 4.1, it delves into common interface paradigms utilized in publicized exploratory search systems. Subsequently, certain interface paradigms are highlighted with the potential to meet the collected requirements for manufacturing. Building on these insights, Section 4.2 presents the design of an exploratory search system for the manufacturing domain. Finally, the concepts and architecture of the proposed exploratory search system (ESS) prototype are outlined in Section 4.3.

Chapter 5

This chapter addresses the third research question, RQIII. The ESS introduced in Chapter 4 was evaluated with five participants on UC2, focusing on qualitative analysis due to the small sample size. A specialized knowledge graph was created specifically for this use case including an ontology for smart manufacturing and collaborative robotics, which are discussed in Section 5.1. Subsequently, the design and results of the user study are outlined in Section 5.2.

Appendices A & B provide supplementary material for the conducted user study.

Chapter 6

This chapter brings this thesis to a close by revisiting the three research questions in Section 6.1. In Section 6.2, it details the potential avenues for future work, aiming at further refinement and optimization of the ESS.

CHAPTER 2

Background & Related Work

Knowledge graphs are seen as a promising solution for the digital transformation of multidisciplinary manufacturing knowledge, but the concept of a knowledge graph is not clearly defined. This thesis adheres to the simple definition that a knowledge graph is *"a structured dataset that is compatible with the RDF data model and has an ontology as its schema."* [VTGSR^{+}17]. Section 2.3 discusses how this concept has been defined by other authors.

The Resource Description Framework (RDF) is flexible data model to describe resources with statements, where a resource can be anything that is uniquely identifiable. This framework and related technologies of the Semantic Web stack are outlined in Section 2.1. While RDF facilitates the collaborative sharing of information in form of a collection of statements about resources, the need for a medium arises in which people can collaborate on models that allow us to organize shared knowledge [AH11]. This is enabled by ontologies, and Section 2.2 briefly describes common ontology languages.

Exploratory search systems over manufacturing knowledge graphs are suggested by this thesis as a solution to support engineers in their decision-making. Similarly to knowledge graphs, exploratory search is a loosely defined concept with an unstable definition. This thesis adheres to the simple definition that it is an open-ended, weakly defined information seeking task such as learning and sense-making. A deeper dive into exploratory search is taken in Section 2.4. The evaluation of exploratory search systems is a challenging task, and the aforementioned section also discusses proposed evaluation methodologies for such systems.

Finally, this chapter concludes with related work about exploratory search in manufacturing in Section 2.5. It outlines scientific works from the manufacturing domain that concern themselves with knowledge management and reusability of manufacturing knowledge as well as search and exploration in manufacturing.

2.1 Resource Description Framework (RDF)

The Resource Description Framework (abbreviated with RDF) offers a flexible data model that facilitates data sharing and data integration. It is one of the fundamental technologies to represent data on the Semantic Web and is recommend by W3C as such. It gained widespread momentum in the last decade resulting in a number of big public datasets. The DBpedia project, which is an attempt to extract facts from Wikipedia pages and make them machine-readable, offers an RDF dataset with billion pieces of information describing million of entities [LIJ+15]. The broad goal of RDF is to define a mechanism to describe resources. A resource can be anything that is uniquely identifiable, including a document, person, physical object or an abstract concept [SR14].

An RDF dataset consists of statements providing pieces of information about a resource in the form of $\langle subject, predicate, object \rangle$, which is commonly called a *triple*. The *subject* of a triple is a resource about which one wants to make a statement. It can be identified either globally by an Internationalized Resource Identifier (IRI) or a local identifier. The later is also known as blank node. A *predicate* puts the given *subject* in a certain relationship with the *object* of the triple. An *object* can be a literal or another resource. A literal is a value of a specific data type, whereas this data type is identified by an IRI as well. If no data type is explicitly assigned to a literal, it is considered to be a text string. Text strings have the distinct feature of language tags that can be attached to them. The *predicate* in a triple is a resource itself and ascribes a meaning to the relationship between the *subject* and *object*. Figure 2.1 depicts an example of an RDF graph.

Blank nodes are local identifier that allow for the representation of entities without the necessity of assigning explicit names in form of an IRI. These identifiers are not recognizable outside their local system application [HPS14]. Although blank nodes offer convenience to RDF authors and ontologists, they pose a challenge from an engineering perspective. According to the SPARQL specification [HSP13], an identifier of a blank node received in one query cannot be reused in subsequent SPARQL queries. This restriction complicates the tracking and unique identification of the entity represented by a blank node. Hence, the proposed ESS mandates skolemization, requiring all blank nodes to have an explicit IRI.

Collections and Containers are two distinct constructs in RDF to group resources. RDF collections can describe closed lists (known as `rdf:List`), which have a clearly specified beginning and end. Similarly to Lisp, a list can be constructed with the properties `rdf:first`, `rdf:next` and the instance `rdf:nil`. In contrast to collections, containers represent open collections, where "open" refers here to the fact that the RDF specification defines no mechanism for containers to state that there are no more members in the container [BGM14]. This is especially relevant for reasoning on-top of RDF data. RDF knows three classes of containers, namely `rdf:Bag`, `rdf:Seq` and `rdf:Alt`, each with its own unique semantics.

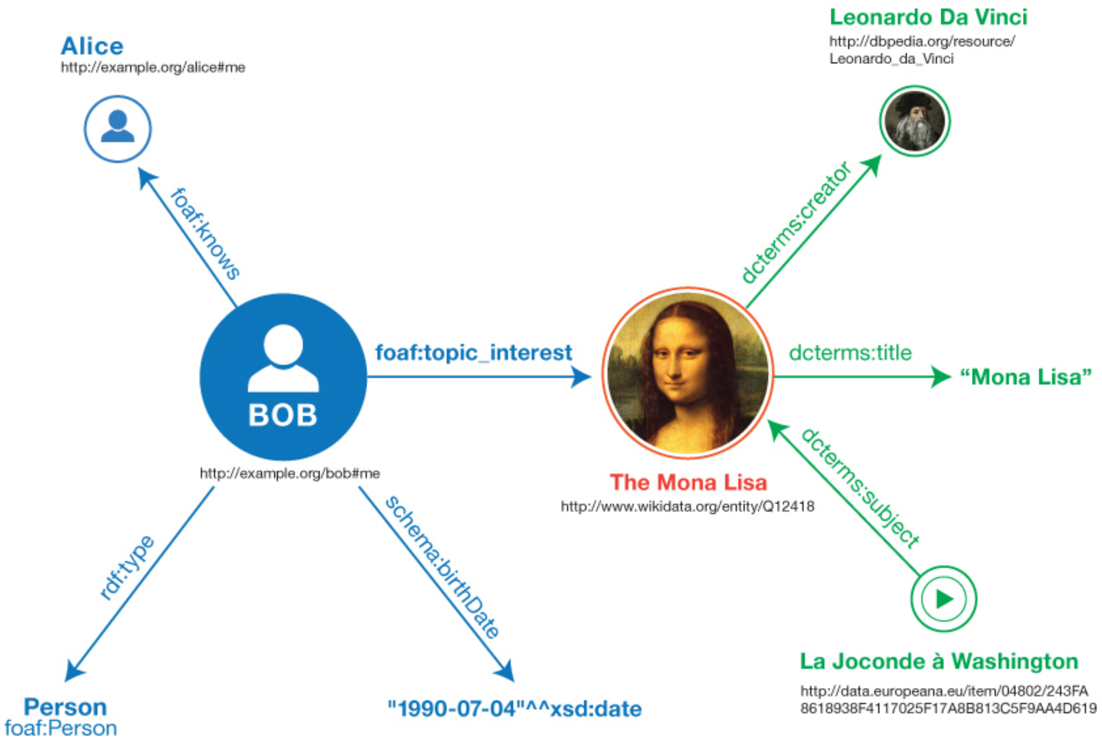


Figure 2.1: Example of an RDF graph [SR14].

Reification allows an RDF graph to act as metadata describing other RDF triples [HPS14] by introducing a special vocabulary with the class `rdf:Statement`, and the three predicates `rdf:subject`, `rdf:predicate`, as well as `rdf:object`. With this vocabulary an RDF triple such as '`ex:bob foaf:knows ex:alice`' can be formulated as identifiable resource (e.g. `_:s` in Listing 2.1) and hence, one can assert statements about this triple as about any other RDF resource.

Listing 2.1: Reification Example (Turtle syntax[1])

```
_:s a rdf:Statement ;
      rdf:subject ex:bob#me ;
      rdf:predicate foaf:knows ;
      rdf:object ex:alice#me .
```

Named Graphs are an extension of the RDF model [HPS14], which provide the ability to organize an RDF graph in a number of named subgraphs. A named subgraph is a non-empty collection of triples of the original RDF graph that is identified either by an IRI or a blank node. Thus, a named graph is itself an RDF resource and one can assert facts about it. An example for named graphs is shown in Listing 2.2.

[1]RDF 1.1 Turtle, `https://www.w3.org/TR/turtle/`

Listing 2.2: Named Graphs Example (TriG syntax[2]) [SR14]

```
GRAPH ex:dataset {
      ex:bob#me a foaf:Person ;
              foaf:knows ex:alice#me ;
              schema:birthDate "1990-07-04"^^xsd:date ;
              foaf:topic_interest wd:Q12418 .
}
GRAPH <https://www.wikidata.org/wiki/Special:EntityData/Q12418> {
      wd:Q12418 dct:title "Mona Lisa" ;
              dct:creator dbr:Leonardo_da_Vinci .
}
```

Triplestore is a name given to a database management system with the ability of storing and retrieving triples. Most *triplestores* support *SPARQL*, which is a semantic query language with which triples can be retrieved and manipulated (see Section 2.1.1).

2.1.1 SPARQL

SPARQL is the W3C recommended semantic query language for RDF data and is thus supported by most triplestores. The first language specification of SPARQL (referred to as version 1.0) [PS08] was published in January 2008 with basic patterns for graph pattern matching. SPARQL 1.1 [HSP13] extended this first specification with new features (e.g. path properties, sub-queries, aggregation) and was released in March 2013.

Basics

SPARQL provides four query forms, namely `SELECT`, `ASK`, `DESCRIBE` and `CONSTRUCT`. A `SELECT` query returns all or a subset of the variables bound in a query pattern match. `ASK` returns a boolean value indicating whether a query pattern matches (non-empty solution set) or not. `DESCRIBE` returns an RDF graph composed of available statements concerned with a given resource. With `CONSTRUCT` a new RDF graph can be constructed by specifying triple templates. Potential variables in the triple templates are substituted with the values of the corresponding bound variables in a query pattern match.

As stated in the specification of SPARQL [HSP13], *"Most forms of SPARQL queries contain a set of triple patterns called a basic graph pattern. Triple patterns are like RDF triples except that each of the subject, predicate and object may be a variable. A basic graph pattern matches a subgraph of the RDF data when RDF terms from that subgraph may be substituted for the variables."*.

Listing 2.3 shows a simple `SELECT` query with a basic graph pattern that consists of two triple patterns. The first triple pattern has a variable on the subject position, while the second one has the same variable on the object position. This query asks for all person that are

[2]RDF 1.1 TriG, `https://www.w3.org/TR/trig/`

known to Bob, and this would only be Alice (i.e. `http://example.org/alice#me`) considering the RDF example in Figure 2.1.

Listing 2.3: SELECT SPARQL Query

```
SELECT ?person WHERE {
        ?person a foaf:Person .
        <http://example.org/bob#me> foaf:knows ?person .
}
```

Shortcomings

Centrality and similarity metrics (see thesis of Armin Friedl [Fri17]) are crucial for constructing an exploratory search system. Centrality metrics such as Page Rank assign an "importance" value to entities in the knowledge graph. Similarity metrics on the other side evaluate the similarity of a pair of entities. However, we cannot solely rely on SPARQL for computing those metrics, because the specification of SPARQL 1.1 [HSP13] has some shortcomings.

Traversing an RDF Graph: SPARQL 1.1 introduces property paths as a new feature, which is defined as a possible route between two nodes of a graph. A triple pattern is a property path of length 1. It adds the ability to match the connectivity of two resources in a knowledge graph. However, computing the distance between a pair of resources is not possible, because we can only check whether a path exists and don't know how the path looks like. Hence, SPARQL is not well suited for tasks like computing the shortest path between a pair of resources.

Intractable Blank Nodes: Blank nodes are no stable identifiers and they can get an arbitrary name assigned to in the query result of a SPARQL query according to the specification of SPARQL 1.1, which makes it impossible to track entities referred to by a blank node over multiple SPARQL queries.

2.1.2 RDF-star and SPARQL-star

RDF-star and SPARQL-star are proposed extensions of RDF and SPARQL respectively [HCKS21]. In the original RDF 1.1 specification [HPS14], the subject of a triple can either be a resource or a blank node. The object of a triple can furthermore be either a resource, a blank node or a literal. However, with RDF-star a triple can be placed itself on the subject or object position of a triple. An RDF-star triple can then take the form `«ex:bob foaf:knows ex:alice» ex:confidence 0.75`, in which a confidence value is assigned to the assertion that Bob knows Alice. SPARQL-star is an extension of SPARQL, which allows users to query RDF-star triples.

At the moment of writing, RDF-star and SPARQL-star are not recognized as standard. Nonetheless, popular triplestores such as GraphDB[3] and the Java framework RDF4J[4] have already implemented these extensions.

Provenance , which is a crucial part of data management, are often stated as use case for RDF-star, because the original RDF 1.1 specification [HPS14] doesn't provide a convenient method to annotate statements. Harting identified in [Har17] three strategies for annotating RDF triples, which all have their shortcomings.

1. **RDF reification:** The reification vocabulary proposed in the RDF specification [HPS14] can be used to assert provenance statements about a triple. The problem of this approach is that suddenly four triples are needed instead of one to make an assertion and furthermore, querying of such data becomes cumbersome with common RDF query languages.
2. **Singleton properties:** Another approach would be to create a new property for each triple and to assert that it is a subclass of the corresponding property in the triple. Consequently, statements can be made about this individual relationship between subject and object as it can be done in property graphs. However, the occurrence of this many subproperties is untypical for RDF data and hence, can be an issue for certain storage models and query optimization strategies used in tripelstores (e.g. vertical partitioning [AMMH07]).
3. **Single-triple named graphs:** Triples can as aforementioned be assigned to named graphs and those named graphs can uniquely be identified by an IRI or a local blank node. Assertions about these triples can then be made in form of statements about this particular named graph. A shortcoming of using named graphs is that there is no explicitly, formally specified meaning to the relationship of being added as triple to a named graph, which makes it less convenient when used for more than one use case, e.g. for provenance information on one side and assigning confidence or trust values to triples on the other side.

The application of strategy (3) with conventional RDF is shown in Listing 2.2, where triples are assigned to the dataset resource from which they originated. Listing 2.4 shows how to achieve the same use case with RDF-star.

Listing 2.4: RDF-star example (Turtle-star syntax[5])

```
<<ex:bob#me a foaf:Person>> :statedBy ex:dataset .
<<ex:bob#me foaf:knows ex:alice#me>> :statedBy ex:dataset .
<<ex:bob#me schema:birthDate "1990-07-04"^^xsd:date>> :statedBy ex:dataset .
<<ex:bob#me foaf:topic_interest wd:Q12418>> :statedBy ex:dataset .

<<wd:Q12418 dct:title "Mona Lisa">> :statedBy d:Special:EntityData/Q12418> .
<<wd:Q12418 dct:creator dbr:Leonardo_da_Vinci>> :statedBy d:Special:EntityData/Q12418 .
```

[3]Ontotext GraphDB triplestore, https://www.ontotext.com/products/graphdb/

[4]Java framework for RDF from the Eclipse Foundation, https://rdf4j.org/

[5]Turtle-star from RDF-star draft [HCKS21]

2.1.3 Property Graphs

A property graph is composed of vertices and directed edges with labels, whereas both vertices and edges can have an arbitrary number of key/value pairs, also called properties, associated with them. RDF data can as well be seen as a directed, labelled graph (see Figure 2.1) and it is possible to represent RDF data as a property graph. Thus, making it possible to use a number of graph query languages on RDF data stored in form of a property graph.

At the moment of writing, no query language is accepted as a standard for property graphs. GQL is one of the emerging attempts to establish such a standard [ISO24]. In the following, two prominent query languages for property graphs are outlined, namely *openCypher* and *Gremlin*. The benefit of those query languages from the perspective of analysing RDF data is that they can compensate for the shortcomings of SPARQL.

openCypher: The graph database Neo4J[6] popularized Cypher as query language for property graphs, and the openCypher project intends to enable the use of Cypher as a standardized language capable of being implemented in other graph databases. Cypher as a declarative language was inspired by SPARQL and XPath. As stated in [FGG+18], *"A Cypher query takes as input a property graph and outputs a table. These tables can be thought of as providing bindings for parameters that witness some patterns in a graph, with some additional processing done on them. Cypher structures queries linearly. This allows users to think of query processing as starting from the beginning of the query text and then progressing linearly to the end. Each clause in a query is a function that takes a table and outputs a table that can both expand the number of fields and add new tuples. The whole query is then the composition of these functions."*. While some other graph databases besides Neo4J support this query language (e.g. Redis Graph, Agens Graph), it is not recognized as standard yet. Listing 2.5 shows the openCypher equivalent of the SPARQL query shown in Listing 2.3.

Listing 2.5: openCypher example

```
MATCH (bob)-[:'foaf:knows']->(friend)-[:'rdf:type']->(type)
WHERE id(bob) = 'ex:bob#me' AND id(type) = 'foaf:Person'
RETURN id(friend)
```

Gremlin: Gremlin is a functional graph traversal language for property graphs that can be executed on all Tinkerpop-enabled systems. Apache TinkerPop is an open source, vendor-agnostic, graph computing framework. As stated in [Fou18], *"Every Gremlin traversal is composed of a sequence of (potentially nested) steps. A step performs an atomic operation on the data stream. Every step is either a map-step (transforming the objects in the stream), a filter-step (removing objects from the stream), or a sideEffect-step (computing statistics about the stream)."* Listing 2.6 shows the Gremlin equivalent of the SPARQL query shown in Listing 2.3.

[6]Neo4J graph database, `https://neo4j.com`

Listing 2.6: Gremlin traversal example

```
g.V('ex:bob#me').out('foaf:knows')
        .where(__.out('rdf:type').has(T.id, 'foaf:Person'))
```

2.2 Ontologies

RDF facilitates the collaborative sharing of information on the Semantic Web in form of a collection of statements about entities. However, the need for a medium arises in which people can collaborate on models, that is, models that allow us to organize shared knowledge [AH11].

Computational ontologies are a means to formally organize knowledge [SS09]. Studer et al. defines an ontology as *"a formal, explicit specification of a shared conceptualization."* [SBF98]. A conceptualization is thereby *"an abstract, simplified view of the world that we wish to represent for some purpose. Every knowledge base, knowledge-based system, or knowledge-level agent is committed to some conceptualization, explicitly or implicitly"* [Gru93]. An ontology is in other words a domain model which is explicitly expressed formally in a machine-readable format. It should express a shared view among various parties, rather than being the perspective of an individual. Ontologies facilitate the communication of knowledge between computers themselves as well as with and among humans by representing the knowledge in a computer-processable format [Sab16].

The Semantic Web technology stack offers a number of modeling languages of varying expressiveness. RDF is a basic model that allows us to make simple statements about entities (see Section 2.1). The RDFS language provides the expressivity to describe basic notions of commonality and variability known from object-oriented programming languages and other class systems [AH11]. It introduces classes and properties as well as langauag elements for building hierarchies of classes and properties. We elaborate on RDFS in Section 2.2.1. The Web Ontology Language (OWL) furthermore introduces the expressiveness of logics, allowing us to define detailed constraints between instances, properties and classes. We elaborate on OWL in Section 2.2.2.

2.2.1 RDF Schema (RDFS)

The RDF Schema (RDFS) specification [BGM14] provides basic means for modeling RDF data. The schema itself is expressed in RDF. The elements of an RDF schema are defined as resources and statements (in form of triples) involving those resources. The key constructs of the RDFS vocabulary are listed in table 2.1.

RDFS makes it possibles to define classes (sets) of resources and to organize those classes in a hierarchy. Listing 2.7 shows the definition of the classes `:Motor` and `:ElectricMotor`, whereas `:ElectricMotor` was made a specialization of `:Motor`. Consequently, each instance of an electric motor would thus be also an instance of the more general class motor.

Modelling construct	Definition according to language specification
`rdfs:Resource`	All things described by RDF are called resources.
`rdfs:Class`	Declares a resource as a class for other resources.
`rdfs:subClassOf`	States that all the instances of one class are instances of another.
`rdf:type`	States that a resource is an instance of a class.
`rdf:Property`	Used to define an RDF property.
`rdfs:subPropertyOf`	States that all resources related by one property are also related by another.
`rdfs:domain`	Declares the class or datatype of the subject in triples whose second component is a certain predicate.
`rdfs:range`	Declares the class or datatype of the object in triples whose second component is a certain predicate.

Table 2.1: Overview of key constructs of RDF/RDFS as outlined in [Sab16]

Listing 2.7: Example for a class in RDFS (Turtle syntax)

```
:Motor rdf:type rdfs:Class .
:ElectricMotor rdf:type rdfs:Class; rdfs:subClassOf :Motor .
```

Additionally, we can define an `rdf:Property`, which relates a subject to an object, whereas a subject must be a resource and an object can either be a literal or another resource. Moreover, the domain of subjects and the range of objects for a property can be defined with `rdfs:domain` and `rdfs:range`. The values for domain and range of a property are expected to be classes of resources. Listing 2.8 outlines the definition of the property `:nominalVoltage`, which expects electrical objects as subjects and floating numbers as objects.

Listing 2.8: Example for a property in RDFS (Turtle syntax)

```
:nominalVoltage rdf:type rdf:Property ;
                rdfs:domain :ElectricalObject ;
                rdfs:range xsd:double .
```

However, not only classes can be organized in a hierarchy, but also properties with `rdfs:subPropertyOf`. If P is a subproperty of P' and there exists a resource pair that can be related with P, then this resource pair can also be related with P' [Sab16]. Listing 2.9 shows the utilization of `rdfs:subPropertyOf`, where `ex:machineA` hasn't only an engine and sensor, but also two general components by inference.

Listing 2.9: Example for a property hiearchy in RDFS (Turtle syntax)

```
:hasEngine rdfs:subPropertyOf :hasComponent .
:hasSensor rdfs:subPropertyOf :hasComponent .

ex:machineA :hasEngine ex:squirrelCageMotorD .
ex:machineA :hasSensor ex:ultrasonicProximityB .
```

While the RDFS language has limited expressiveness, it is well suited for expressing lightweight ontologies consisting of (hierarchies of) relations and classes [Sab16].

2.2.2 Web Ontology Language (OWL)

The first version of the Web Ontology Language (OWL) was introduced in 2004 [MvH04], but the W3C recommends to use the most recent release 2.0 [HPPsR12]. It introduces language elements to define detailed constraints between instances, properties and classes. While OWL can be expressed in different formats, it can also be represented under the semantics of RDF as it is the case for RDFS. A selection of key constructs of OWL is listed in table 2.2.

Modelling construct	Definition according to language specification
`owl:equivalentClass`	Relates two classes whose class extensions contain exactly the same set of individuals.
`owl:disjointWith`	Asserts that the class extensions of the two class descriptions involved have no individuals in common.
`owl:intersectionOf`	Defines a class that contains the same instances as the intersection of a specified list of classes.
`owl:unionOf`	Defines a class that contains the same instances as the union of a specified list of classes.
`owl:complementOf`	Defines a class as a class of all individuals that do not belong to a certain specified class.
`owl:ObjectProperty`	Defines a property that captures a relation between instances of two classes.
`owl:DatatypeProperty`	Defines a property that captures a relation between instances of classes and RDF literals/XML Schema datatypes.
`owl:inverseOf`	If a property, P_1, is owl:inverseOf P_2, then $\forall x, y : P_1(x, y) \iff P_2(y, x)$.
`owl:TransitiveProperty`	If a property, P, is transitive then $\forall x, y, z : P(x, y) \wedge P(y, z) \implies P(x, z)$.
`owl:ReflexiveProperty`	A reflexive property relates everything to itself.
`owl:SymmetricProperty`	If a property P is symmetric then if the pair (x,y) is an instance of P, then the pair (y,x) is also an instance of P.

Table 2.2: Overview of key constructs of OWL as outlined in [Sab16]

In RDFS (see Section 2.1), basic hierarchies of classes and properties can be defined. OWL on the other hand introduces additional language elements to model complex forms of classes. As a result, one can define classes based on set operations on other classes (i.e. union, intersection and complement). Listing 2.10 outlines how to define such a complex class as the union of three atomic classes.

Listing 2.10: Example for complex class in OWL (Turtle syntax)

```
:ArchitecturalElement owl:unionOf (:DesignPattern
                                   :DesignTactic
                                   :SEMethod) .
```

A class can furthermore be a restriction, which specifies constraints that all it‘s member instances have to satisfy. OWL knows different kinds of restrictions; some of which are concerned about the object types of properties (e.g. `owl:someValuesFrom`) and others are concerned with cardinality (e.g. `owl:minCardinality`). If a resource satifies the constraints of a restriction class, then it can implicetely be inferred that this resource is an instance of this class. Listing 2.11 shows how a property named `:scheduleRepairAt` is defined with a certain restriction class as domain. Consequently, this property can only be used on components that have the status of "broken".

Listing 2.11: Example for a restriction class in OWL (Turtle syntax)

```
:BrokenComponent a owl:Restriction ;
      owl:onProperty :hasStatus ;
      owl:hasValue :BrokenStatus .

:scheduleRepairAt a owl:DatatypeProperty ;
      rdfs:domain :BrokenComponent ;
      rdfs:range xsd:date .
```

In contrast to RDFS, OWL also categorizes properties into datatype and object properties (`owl:DatatypeProperty` and `owl:ObjectProperty` respectively). The former is linking resources to literals, and the later is linking resources to other resources. A number of axioms are introduced for object properties such as reflexive, symmetric, functional or transitive axioms. Listing 2.12 shows the definition of a property as transitive. Hence, it can implicetely be inferred that the squirrel cage motor as well as it‘s fan are part of `ex:machineA`.

Listing 2.12: Example for property axioms in OWL (Turtle syntax)

```
:isPartOf rdf:type owl:TransitiveProperty .

ex:fanB :isPartOf ex:squirrelCageMotorD .
ex:squirrelCageMotorD :isPartOf :machineA .
```

In order to find a balance between expressitivity and scalability, OWL has been divided into multiple profiles. The "OWL 2 Full" profile includes all language constructs of OWL 2 and has the highest expressivity, but it is also undecidable (under RDF-based semantics) [SS09]. Additionally, OWL 2 [HPPsR12] defines three subsets of "OWL 2 Full", namely "OWL 2 EL", "OWL 2 QL", and "OWL 2 RL". Each of these profiles was designed with an application domain in mind. These application domains correspond to a required level of expressitivity and scalability.

2.3 Knowledge Graph

The term "knowledge graph" was popularized back in 2012 by Google [Sin12], and it refers to their use of semantic knowledge in Web search [Pau17]. Nonetheless, there is no agreed upon and common definition for knowledge graphs in literature [EW16]. This section aims to outline how different authors defined knowledge graphs in the past and clarifies how this term shall be understood in the remaining part of this thesis.

Pujara et al. [PMGC13] lists a number of information extraction systems, *"which use a variety of techniques to extract new knowledge, in the form of facts, from the web. These facts are interrelated, and hence, recently this extracted knowledge has been referred to as a knowledge graph"*. This definition remains vague and defines knowledge graphs as a collection of interrelated facts generated from information extraction systems that are applied to the Web.

Ehrlinger and Wöß [EW16] draw a clear distinction between a knowledge base and a knowledge graph based on the usage of the term in literature. Their terminological analysis lead to the definition that *"a knowledge graph acquires and integrates information into an ontology and applies a reasoner to derive new knowledge"*. A knowledge base in contrast is missing the integration of new knowledge from multiple sources, and the reasoning engine to infer new knowledge. Bellomarini et al. [BFGS19] builds upon this definition of Ehrlinger et al., and defines a knowledge graph as a *"semi-structured data model characterized by three components: (i) a ground extensional component, that is, a set of relational constructs for schema and data (which can be effectively modeled as graphs or generalizations thereof); (ii) an intensional component, that is, a set of inference rules over the constructs of the ground extensional component; (iii) a derived extensional component that can be produced as the result of the application of the inference rules over the ground extensional component (with the so-called "reasoning" process)"*. A knowledge graph is in this definition essentially a knowledge base with a reasoning mechanism for inferring new knowledge.

According to Paulheim [Pau17], a knowledge graph has four defining characteristics in contrast to other collections of knowledge. A knowledge graph *"(1) mainly describes real world entities and their interrelations, organized in a graph, (2) defines possible classes and relations of entities in a schema, (3) allows for potentially interrelating arbitrary entities with each other, and (4) covers various topical domains"*. This definition is tailored towards encyclopedic projects in the LOD Cloud[7] such as Wikidata[8], but it ignores the use of this technology in (manufacturing) enterprises.

Färber et al. [FBMR18] defines a knowledge graph formally as an RDF graph. *"An RDF graph consists of a set of RDF triples where each RDF triple* (s, p, o) *is an ordered set of the following RDF terms: a subject* $s \in U \cup B$*, a predicate* $p \in U$ *, and an object* $U \cup B \cup L$*. An RDF term is either a IRI* $u \in U$ *, a blank node* $b \in B$*, or a literal* $l \in L$*"*. Consequently, any RDF dataset is a knowledge graph according to this definition.

[7]The Linked Open Data Cloud, `https://lod-cloud.net/`
[8]Wikidata, `https://www.wikidata.org`

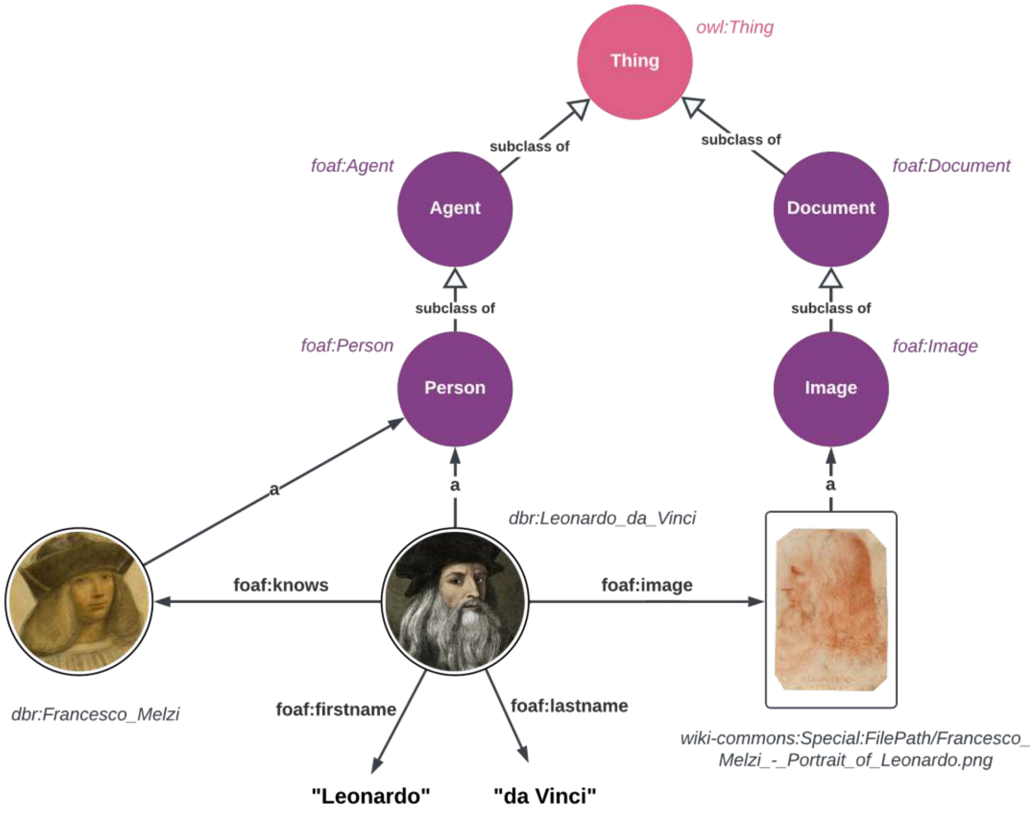


Figure 2.2: Example of a knowledge graph.

This thesis defines a knowledge graph as *"a structured dataset that is compatible with the RDF data model and has an ontology as its schema. A knowledge graph is not necessarily linked to external knowledge graphs; however, entities in the knowledge graph usually have type information, defined in its ontology, which is useful for providing contextual information about such entities"* [VTGSR+17]. This definition is not concerned with reasoning over RDF-based knowledge graphs, but rather sees the knowledge graph as the final product that is the union of explicetely stated and implicetely inferred relationships.

A small sample of a knowledge graph about Renaissance painters is shown in Figure 2.2. It makes use of FOAF[9] as lightweight ontology to ascribe a meaning to relationships between persons and related artifacts.

An exploratory search system can utilize the machine-readable contextual information about entities for the implementation of an adaptive user interface (see Chapter 4).

[9]Friend-of-a-Friend (FOAF) vocabulary, `http://xmlns.com/foaf/spec/`

2.4 Exploratory Search

Exploratory search is a particular information seeking activity, which we adopt as a solution concept to address the problem of information seeking in the manufacturing domain introduced in Section 2.5. As with the concept of knowledge graphs, there is at the moment of writing no stabilized or consensus definition of exploratory search in literature. It is rather a loosely defined concept with an unstable definition which continues to evolve [Pal18]. Section 2.4.1 outlines how different authors have tried to define exploratory search. Surveys about regular semantics-based search systems and exploratory search systems are summarized in Section 2.4.2. Section 2.4.3 is devoted to the challenging question of how to properly evaluate the ability of a search system to enable exploratory search.

2.4.1 Definition

A number of researchers have over the years discussed the defining characteristics of exploratory search. An early contribution has been made by Marchionini [Mar06], who highlights learning and investigating as pertinent activities of exploratory search. Considering the terminology of Blooms taxonomy of educational objectives [BK56], searches with the intention to learn involve multiple iterations and aim to achieve *"knowledge acquisition, comprehension of concepts or skills, interpretation of ideas, and comparisons or aggregations of data and concepts"* [Mar06]. Investigative searches on the other hand aim to achieve Bloom‘s highest level objective such as analysis, synthesis and evaluation. Knowledge is in this activity carefully examined before it is added to the personal knowledge base [Mar06]. Figure 2.3 illustrates their conceptual model of exploratory search.

While exploratory search and lookup searches are often contrasted, they should not be seen as mutually exclusive. Lookup based activities might be embedded in an activity of exploratory search. Lookup-based information retrieval is a predominant search paradigm commonly used by major search engines among others. This search model has been proven successful for well-defined information seeking tasks in which a user already knows what to exactly search for. A user issues a precise query, e.g. types "longest river in South America" into a search engine on the Web, and gets in response a well-structured answer with minimal need for result examination and aggregation. On the other hand, exploratory search is described as open-ended with an unclear information need [Pal18], e.g., I want to learn more about the country Honduras in South America. This kind of search activity is evolving and non-linear.

White et al. [WR09] defines exploratory search as the term *"can be used to describe an information-seeking problem context that is open-ended, persistent, and multi-faceted; and to describe information-seeking processes that are opportunistic, iterative, and multi-tactical. In the first sense, exploratory search is commonly used in scientific discovery, learning, and decision-making contexts. In the second sense, exploratory tactics are used in all manner of information seeking and reflect seeker preferences and experience as much*

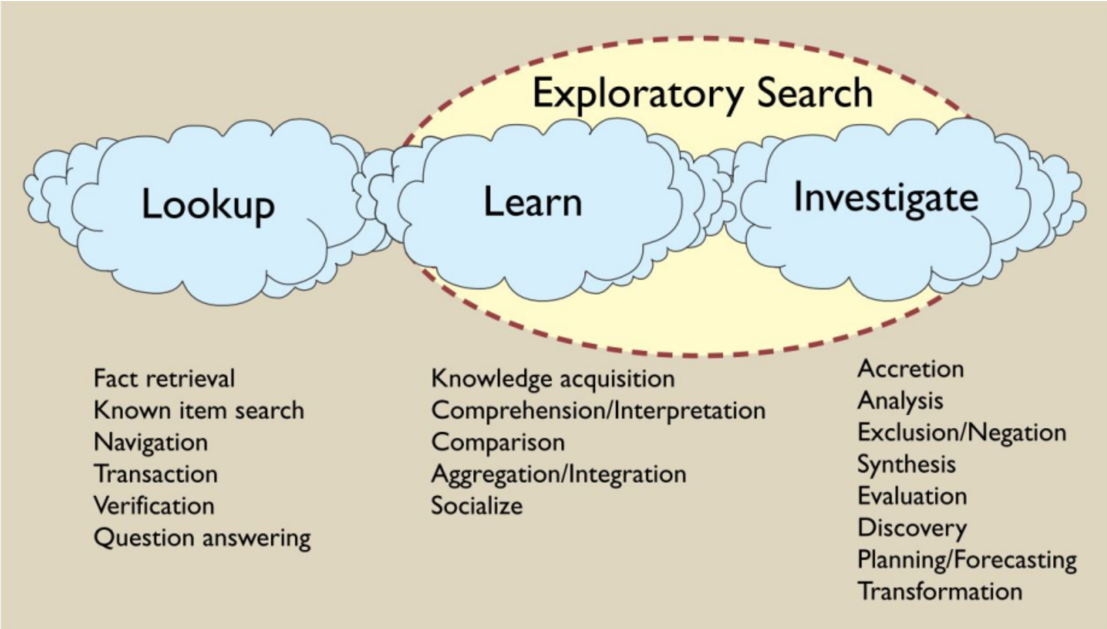


Figure 2.3: Taxonomy of search tasks proposed by Marchionini [Mar06].

as the goal". Exploratory search systems support aspects of *sense-making*, *information foraging*, and *berrypicking* [WR09].

The theory of *information foraging* was proposed by Pirolli et al. [PC95][PC99] and it draws an analogy between the behaviour of humans when seeking information and food foraging strategies in evolutionary anthropology. It analyses search activities from the point of view that all activities have resource currency returned and cost incurred. An academic has to invest energy to find valuable information from a wide range of sources for their project like a bird has to invest energy into finding prey. The theory introduces the concept of *information patches* as well as *information diet* and *scent following*. The concept of *information patches* refers to a situation in which relevant information might reside in collections of documents scattered over multiple sources and one has to decide on which documents to focus energy on. *Information scent* refers to the perception of cost and value of an information source obtained by proximal clues. An *information diet* is the careful selection of information sources to focus on. By analogy, the aim of an information predator is to select information prey so to maximize the information gain per invested energy unit [PC99].

A pictorial model of an exploratory search activity is given by the analogy of *berrypicking* in a forest, where *"berries are scattered on bushes, not in bunches, and the seeker must pick the berries singly"* [Bat89]. The information seeker is confronted with a landscape of information being scattered among a number of sources, and they have to navigate through this information space, gathering relevant pieces of information and clues that aid navigation decisions [WR09]. At each stage of the search, the finding of new pieces of information might lead to the information seeker getting new ideas and directions to follow. Hence, the information seeker might have different conceptualization of the query along the stages of the search in contrast to traditional models of information retrieval, in which a query is satisfied by a final retrieved set. In case of this model, it is rather satisfied by a collection of individual references and information fragments gathered along the evolving search [WR09].

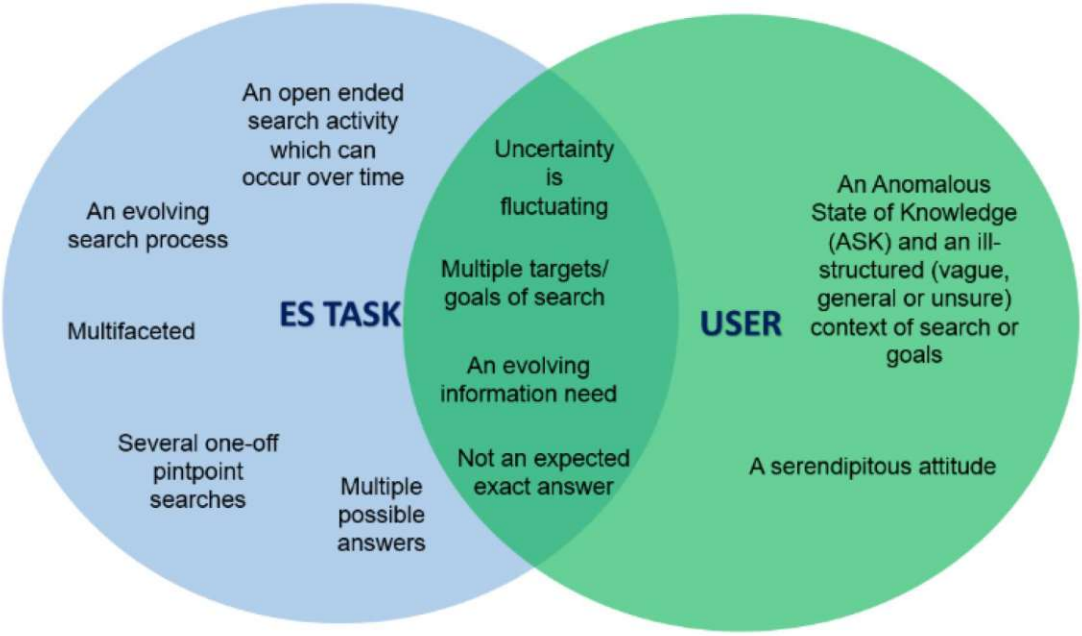


Figure 2.4: Venn diagram of characteristics of exploratory search [Pal18].

Palagi aims to define in her thesis [Pal18] an approximate model of exploratory search with the intention of covering an exploratory search process accurately enough to discuss evaluation methodologies for such search systems. The model defines characteristics of the exploration task and the inner state of the information seeker during the search. The model is based on an aggregation of characteristics commonly ascribed to exploratory search in literature and is depicted in detail in Figure 2.4.

According to this model, an exploratory search process typically starts with a loosely defined information need and is open ended as well as evolving. An information seeker might change their information need based on results discovered throughout the search session and perform a reformulation of current queries. An information seeker might not have one precise single goal in mind, but rather multiple goals or a loosely defined objective, which may evolve throughout the search session as well. In an exploratory search session, an information seeker is in general not looking for a correct answer to the goals or vague objective, but for relevant information that can be added to the personal knowledge base and further informs the ongoing search process. An serendipitous attitude can be ascribed to exploring information seekers, which means that they have the faculty to be surprised about a discovered element in a result set and the motivation to reformulate the search objectives based on the surprise. Moreover, an information seeker might perform several pinpoint searches to investigate a specific element of the result set or why in particular it was part of the result set given the issued query [Pal18].

This thesis simply defines exploratory search as an open-ended, weakly defined information seeking task such as learning and sense-making. A system is an exploratory search system, if it supports information seekers in their exploratory search tasks.

2.4.2 Exploratory Search Systems

An early survey of semantic-based exploratory search systems was published by Marie et al. [MG14]. It portrays the history of semantic-based search systems and characterizes them. The first generation of tools for navigating knowledge graphs were semantic browsers such as Tabulator [BlCC+06]. These tools were heavily inspired by the way we are browsing Web pages and provided a simple view of the data. A number of recommendation systems that make use of the semantic similarity of resources in a knowledge graph were proposed in the following years. An example is Yahoo SPARK[BCMT13], which was used for entity recommendation in Yahpoo‘s search interface. However, the most advanced systems in the field of exploring knowledge graphs are exploratory search systems [MG14], which are categorized by Marie et al. into either view-based or algorithm-based. Aemoo [NPG+17] is a view-based exploratory search system in which the neighborhood of a resource of interest is visualized as a graph. A user can then navigate through the neighborhood and explore it, whereas the neighborhood is restricted to a manageable number of neighbors based on Encyclopedic Knowledge Patterns[10] to not overwhelm users with irrelevant information. Lookup Explore Discover (LED) [MN10] is in contrast an algorithm-based exploratory search system in which a user can enter a list of topics of interest and gets a tag cloud of semantically similar resources in response. User can add topics from this tag cloud to their initial list in order to refine the query, because the specified topics are eventually used to issue a regular search with Google and the highest ranked results of this search are shown to users below their corresponding tag cloud.

Klimek et al. [KSN19] conducted a comprehensive survey of tools supporting the discovery, processing, and consumption of knowledge graphs (as defined in Section 2.3). 110 tools were identified and sixteen of them were evaluated based on a big set of criteria after they passed all three elimination rounds of the systematic literature study. While exploratory search was no concern of this survey, different forms of searching through Linked Data including full-text search, faceted search and SPARQL querying were part of the evaluation. Among the sixteen evaluated tools is SemFacet [AGK+16], which is an exploratory search system utilizing facets for the exploration of knowledge graphs. It can automatically generate these facets given that the metadata and domain knowledge of the graph is described with OWL 2 (see Section 2.2.2).

[10] Encyclopedic Knowledge Patterns, generated from DBpedia graph analysis [NGPC11].

Po et al. [PBDP20] discuss in their book different tools and methods to visualize and navigate knowledge graphs (as defined in Section 2.3). The tools are categorized into five categories: (1) *browsers and exploratory tools*, (2) *tools using multiple visualization types*, (3) *graph-based visualization tools*, (4) *domain and vocabulary-specific as well as device-oriented visualization tools*, and lastly, (5) *ontology visualization tools*. This thesis is mostly concerned with the first category of *browsers and exploratory tools*. These tools enable browsing and exploration of knowledge graphs in an intuitive way, but they lack or only have limited visual support in contrast to the other categories. The first generation of tools in this category were semantic browsers using tabular views for presentation of resources and links for navigation, whereas most of the newly publicized systems follow the faceted exploration paradigm to provide effective browsing functionalities [PBDP20].

These surveys offer a starting point for collecting search and interface paradigms that might be valuable for exploratory search systems in the manufacturing domain (as part of research question RQII). Section 4.1 discusses common interface paradigms and furthermore, paradigms of particular interest for manufacturing.

2.4.3 Evaluation Methodology for Exploratory Search Systems

In the previous section, we outlined a number of surveys for semantic-based exploratory search systems. Some of the publicized works that have been presented in those surveys conducted no evaluation of their system (e.g. [CFM14], [AGK+16]), whereas others have evaluated the search and recommendation results of their system using well-known metrics from information retrieval such as precision and recall (e.g. [MRNS10], [MGRR13], [NDC15]) or asked users to answer a questionnaire about the usability after some search tasks (e.g. [TRJ+17]). Marie et al. [MGRR13] claimed to have resorted to an evaluation of their recommendation engine by users, due to a lack of evaluation methodologies for exploratory search at that time. The evaluation of ranking and recommendation algorithms is however not sufficient, as is it the case for questionnaires about the usability of a system. In the following, we want to outline evaluation methodologies from literature with the aim of evaluating the exploratory search process.

Medlar et al. [MPG17] conducted a comprehensive user study for their search system to investigate how different exploration rates affect *"(1) the number of clicks and the reading time, (2) subjective satisfaction, and (3) overall correlation between the user's knowledge of a given research topic, the exploration rate and the user search behaviour"*. In their search system, a user gets a number of research papers as response to their query and has to give feedback to them in form of a "thumbs up" or a "thumbs down". Afterwards, the user gets other research papers recommended based on the initial feedback. Exploration rate refers here to the trade-off between exploitation and exploration. A low exploration rate supports exploitation, i.e. only closely related works are recommended, while high exploration rates support exploration with a more diverse set of recommendations. Students were recruited from computer science classes and given the task to explore with this search system a certain research topic for which they had to write an abstract at the end of the task. The designed tasks followed the template of Wildemuth et al. [WF12] for exploratory search tasks. Each student got a different exploration rate assigned, and got tracked during the search process with intermediate questionnaires about the satisfaction with the recommendations. However, the evaluation methodology is heavily tied to the specifics of their search system. Moreover, the ability of the system to support learning and sense-making was no concern of the study (e.g. the gain of knowledge was not assessed). Nonetheless, valuable lessons can be learned from it.

Emilie Palagi [Pal18] conducted a survey on a subset of published ES systems and the evaluation employed by the authors of the corresponding publication. She concluded that it was necessary to define an evaluation methodology for exploratory search systems. Palagi defines an exploratory search process that must be supported by an ES system. She suggests a user-centered as well as an inexpensive, quantitative model without users for evaluating how well an ES system supports this process. The later is an inspection method known from usability research. It is intended as a testing ground before undergoing a costly evaluation with users [Pal18]. For the user evaluation, Emilie Palagi outlines a guideline for creating exploratory search tasks and she formulates a model based on qualitative video analysis to detect usability flaws in the search system that have a negative effect on aforementioned exploration process.

In [ADT20], Al-Tawil et al. outline a methodology to construct an evaluation of an exploratory search system using the taxonomy of Bloom to conceptualize the utility of an exploration. *"It suggests linking knowledge to six cognitive processes: remember, understand, apply, analyze, evaluate, and create"* [ADT20], whereas only remember and understand are considered. Exploration tasks are designed following a guideline similar to Palagi's suggestions. However, users have to answer knowledge questions before and after they performed a task to facilitate a quantitative analysis. The answers are evaluated with a point system, and the utility of an exploration is assessed by the difference of the point score between answers given before the task and after the search system was used to perform the task.

Our evaluation methodology is going to be designed based on the work of Palagi [Pal18] and Al-Tawil et al. [ADT20]. Palagi provides an universal framework for the qualitative analysis of generic exploratory search systems, while Al-Tawil et al. outline a methodology to assess the knowledge gain of users for their specific system, which we adapted for our user study in Chapter 5.

2.5 Search and Exploration in Manufacturing

Several authors have investigated the need for search and exploration in industry. Manufacturing is a knowledge intensive field with the need for tools to explore this knowledge. He et al. [HJ19] formulate the need for reusing manufacturing knowledge in order to facilitate decision making in industrial enterprises. Similarly, Zhou et al. [ZZL$^+$20] see in the reuse of manufacturing knowledge a potential for better informing engineers, software or even machine tools in their decision making processes. Thus, they introduce a knowledge-driven framework for autonomous manufacturing cells.

As part of the EU FLEXINET project, Palmer et al. [PUJ+18] identify common problems with information management in manufacturing enterprises and propose a reference ontology that can be specialized by manufacturers to suit their needs. In manufacturing, the information for supporting decision making comes from many different sources, and many different types of decisions have to be carried out. Groups in manufacturing businesses can work on different types of information and might require multiple sources of knowledge to best support decisions. While they stress the need for constructing a knowledge base for manufacturing knowledge and propose techniques to construct it, human-friendly exploration of this knowledge plays a minor role in their discussion.

Biffl et al. [BMM+21] are specifically concerned about properly tracking change dependencies and coordination states of engineering artifacts in multi-disciplinary manufacturing environments. A failed coordination of engineering activities can lead to unplanned rework and project delays, which is costly for any manufacturing company. They propose a knowledge graph based on I4.0 assets[11] as a new coordination artifact. It shall represent change dependencies explicitly in a machine-readable format, and thus, allow a more efficient coordination of changes. However, a human-friendly search interface was not a focus of their feasibility study, rather, Cypher queries (see Section 2.1.3) were used to extract required information from the knowledge graph.

Apart from knowledge organization frameworks for manufacturing, a few search systems have been proposed for exploring knowledge graphs related to manufacturing and engineering. Sabou et al. [SEI+18] published an exploratory search system for assisting software engineers at Siemens in sieving through the central repository of architectural knowledge. In order to facilitate exploration, a faceted view of the direct neighborhood was rendered and a recommendation engine was introduced to recommend related architectural artifacts (e.g. software engineering design pattern) to the currently selected one.

At the same time, Kharlamov et al. [KSH+18] outlines the need of a big Norwegian oil and gas company named Equinor to facilitate exploration of new oil and gas fields while the valuable knowledge is dispersed over multiple isolated systems. The paper discusses the construction of an ontology based data access to those dispersed data sources and a query formulation tool for domain experts with the intent of simplifying the formulation of SPARQL queries and hence, the extraction of sought after information.

A platform for knowledge graph management named "Metaphactory" is presented by Haase et al. [HHK+19], and they briefly outline use cases in the engineering and manufacturing domain at Siemens for this platform. A use case involves the construction of a digital twin of a building including the static information about its composition and live data from sensors located in the building. While exploratory search is not discussed, it touches on faceted navigation, query building assistance and customizable search experience as well as result visualization. Nonetheless, no deeper insight is given into how engineers at Siemens actually interact with the platform.

[11] A physical or non-physical object, specified in ASS information model: `https://bit.ly/37A002I`

Despite of those preliminary systems, there is currently a lack in literature about typical requirements for exploratory search in manufacturing knowledge, which is why we will partially address it in Chapter 3 for research question RQI.

CHAPTER 3

Manufacturing Requirements

Most publicized exploratory search systems are designed to be applied on generic knowledge graphs for which common knowledge is sufficient, and not much attention has yet been dedicated to domains such as engineering and manufacturing. This thesis argues that within the complex environment of manufacturing new approaches are needed for assisting stakeholders in the exploration process of manufacturing knowledge.

In order to derive real-life requirements for exploratory search systems, two organizations from the domain of manufacturing collaborated with a use case in which the need for such a system arises. The first organization is a production plant manufacturer from the metallurgy domain. In this organization, a simulation expert has to collect and consolidate engineering artifacts of a designed plant for simulations, which is a challenge due to the fact that engineers from different disciplines work on such a plant design and sometimes provide inconsistent details about the same individual component. Secondly, researchers from the Aspern pilot factory collaborate, where researchers, industrial partners and students have access to collaborative robots and related equipment, but are not necessarily aware of their availability and potential. Section 3.1 elaborates on both use cases.

Informal interviews with members of the collaborating organizations were conducted to get an understanding about requirements that might not necessarily be unique to professionals in manufacturing, but are not considered for exploratory search systems on generic domains. The methodology and the results for these informal interviews are outlined in Section 3.2.

Section 3.3 proposes then three specific features based on the gathered requirements from these informal interviews. Firstly, adaptability of the system to multiple (engineering) perspectives. Secondly, visibility of provenance details to simplify investigative work. And finally, the ability to browse deep hierarchical structures. Exploratory search systems that are applied to the manufacturing domain should draw special attention to these three features in addition to the widespread ones for generic domains.

3.1 Manufacturing Use Cases

Two organizations from the domain of manufacturing collaborated to derive real-life requirements for exploratory search systems. The first organization is a production plant manufacturer from the metallurgy domain. Their use case is introduced in Section 3.1.1 in more detail. Secondly, researchers from the Aspern pilot factory collaborate. This pilot factory intends to provide researchers, industrial partners and students with space and equipment to experiment with new ways of shaping manufacturing work. Section 3.1.2 elaborates on this second use case.

3.1.1 Production Plant Manufacturing (PPM)

This section reports on a real-life use case supplied by a production plant manufacturer (PPM)[1] from the metallurgy domain with more than 10.000 employees. The goal of this use case is to assist simulation experts in their daily tasks of consolidating diverse views of production system designs.

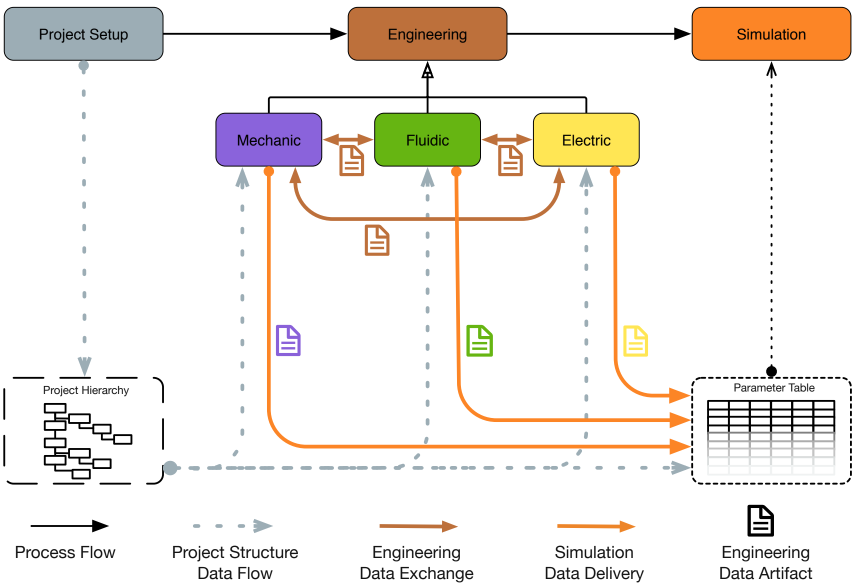


Figure 3.1: Engineering data exchange for simulation.

As depicted in Figure 3.1, representatives of several engineering disciplines work together during production system engineering (PSE) to build a production system. The PPM follows a specific process defined by the VDI[2] for plant engineering, where a functional

[1]Use case partner cannot be named, due to confidentiality agreements.
[2]Verein Deutscher Ingenieure e. V. (VDI), a German organization of engineers and natural scientists

hierarchy is created during project setup (see project hierarchy in bottom left of Fig. 3.1). This initiates the engineering activity across several disciplines such as mechanical and electrical engineering.

The interdisciplinary exchange of engineering data is challenging, however. Discipline-specific habits and vocabularies are hard to overcome in an organization, due to the education and socialization of engineers in their respective discipline [LPRB19]. The views on engineering artifacts in a production system might vary between engineers [BLRW19]. The characteristics of a motor could for instance be described in a impressively detailed manner by mechanical engineers, but only superficially by electrical engineers.

Nonetheless, engineers within the company have to necessarily exchange engineering information in order to make well-informed design decisions for a planned production plant. This is especially the case for simulation experts, who are tasked with the simulation of plant designs, and the optimization of these plant designs in respect to several key performance indicators. As an input for the simulation task, they have to collect engineering information about the individual components to fill the corresponding parameter table for simulations [BLRW19]. To that end, they:

1. identify similar component descriptions in various design files that are likely to refer to the same physical component;
2. select relevant parameters (individual disciplines send all their data); e.g. supply voltage for a motor
3. collect the values suggested by different disciplines; e.g. value A, value B, value C
4. choose the right value. A simulation expert either calls colleagues from relevant disciplines, or consults data sheets from device providers in order to decide on the right value.

This data collection and consolidation process performed by simulation experts is currently a manual process involving identification as well as inspection of relevant engineering artifacts. The details about individual components are scattered over multiple artifacts in a plethora of formats, e.g., Computer-Aided Design (CAD), Comma Separated Value (CSV), or Portable Document Format (PDF) [WRKB20]. Due to the fact that these artifacts are originating from different domain experts and sources, the simulation expert has to put additional effort into manual synchronization, which can be error-prone, and induces avoidable project risks [BLRW19].

An exploratory search system could support a simulation expert in their analysis of engineering artifacts, and help them in their decision-making process about what parameters should eventually be added to the parameter tables for simulations. This system would however require the engineering artifacts to be semantically-annotated and integrated into a central manufacturing knowledge graph, which is a challenging task.

3.1.2 Pilotfabrik 4.0: Collaborative Robots (COBOT)

The Aspern pilot factory is one of several Austrian "learning and experimentation factories" initiated and co-funded by the Austrian Federal Ministry of Transportation, Innovation, and Technology starting with 2015 [HSSW16]. The Aspern factory is administered by the Technical University of Vienna (TU Wien) and currently, several institutes lease certain areas in the factory for research, teaching, and industry contacts.

Industrial machines ranging from (collaborative) robot arms, 3D-printers, and CNC milling machines to virtual reality systems and real work pieces can be used by students, researchers, and companies to experiment with new ways of shaping manufacturing work. To better organize topics and presentations, the different projects are usually focused on creating demonstrators, which exist for a limited period, after which they are dismantled and the composing elements reused in other projects.

This thesis is concerned with collaborative robots and related equipment in the pilot factory. The Aspern factory currently hosts a series of valuable collaborative robotic arms as well as a wide range of supporting tools (grippers, 3D cameras, projectors, etc.), which are ready to be used by students, researchers, and companies.

The detailed capabilities of these machines are, however, unknown to a wider, potentially interested audience, because:

1. they are not publicly available,
2. they depend on the application-context (e.g., so-called robotic skills like assembling, drilling, screwing, etc. which are implemented in software),
3. they continuously evolve and change.

In this context, interested students, researchers, and companies are unaware of the availability and capabilities of the manufacturing technology in the Aspern factory, which leads to a relatively low usage degree of these expensive, state of the art production machines. Hence, the need arises for an exploratory search interface such that students, researchers and companies can explore and learn about the equipment at the pilot factory.

In Section 5.1, the process of constructing a knowledge graph for this use case is outlined in more detail, from the design of a domain ontology to the integration of heterogeneous data into a single knowledge graph. The resulting knowledge graph was used to evaluate the exploratory search system proposed in this thesis.

3.2 Stakeholder Interviews

Informal interviews were conducted with stakeholders of the collaborating organizations to get a better understanding of their requirements towards an exploratory search system. The interviews started with a brief introduction to the concept of exploratory search, and a showcase of an early prototype was presented. The stakeholders were then asked about their daily tasks and what requirements they would have towards such a system. The interviews would take up to one hour.

The remarks of the stakeholders were noted down during the interviews. Afterwards, all the notes were aggregated for each use case, and the following requirement lists were generated.

Collected Requirements

(UC1) PPM: One simulation expert of the production plant manufacturer was interviewed as well as one researcher who was closely connected to this use case.

Domain-Specific Lenses: As a domain expert, I want to see details about a component that are relevant to my discipline (electrical, mechanical, etc.), and not be overloaded with information from other disciplines. Moreover, I should be able to easily switch between discipline lenses and not be fixed to one. As an electrical engineer for example, I might be interested mainly in electrical properties and wiring details for an electrical motor, but not necessarily in mechanical properties.

Simulation Table: As a simulation expert, I am interested in finding parameter values for a physical component that can be found across different data sources (design files, datasheet, etc.). I want to see the values for parameters and the original data source in which this specific value for the parameter was stated (i.e., provenance information) in order to help me with choosing the correct value and constructing the simulation table. The provenance information for a value shall contain a link to the original data source, software that was used in the integration, agent that issued the integration, and timestamp of the integration.

Related Components: Given a specific component (e.g. squirrel cage motor employed in a certain project), as a domain expert, I want to get similar components in other projects, sites, areas, etc. suggested to me. Moreover, I want to be able to compare those components side-by-side.

Hierarchical Navigation: As a domain expert, I want to navigate quickly through the different parts of projects by using a tree view commonly known from various engineering tools. Maximum usability shall be ensured by finding a minimal structure of the neighborhood, such that I am not overloaded with too much information and being able to focus on the important parts. Furthermore, I want to be able to search for components matching a partial identifier.

Export Search Results: As a domain expert, I must be able to export details about the currently displayed components in a CSV file. Per default all known properties about those components are exported, but I should be able to select the properties that are relevant to me, and deselect ones that I am not interested in. The CSV file (or Excel sheet) shall also contain the provenance information for the values. The provenance will indicate the original data source and should also provide a link with more information such as the trace can be requested.

SPARQL Querying: As a domain expert that is familiar with SPARQL, I should be able to formulate SELECT queries for the data using a SPARQL editor with auto-suggestions. The results of the formulated SPARQL query shall be displayed in a table. In case of resources, a snippet similar to the one presented in the search results (see picture in introduction) shall be displayed.

(UC2) COBOT: Two smart manufacturing researchers were interviewed, who were working in the pilot factory. They have an industry and university background respectively.

Research Overview: As a researcher, I want to see on what projects other researchers are working on in the pilot factory and what fields are those researchers focusing on in general by listing all their publications.

Provenance: As a domain expert, I want to be able to see the source of a piece of information and assign it a value of trust. I would like for example know whether a use case (i.e. the application of a collaborative robot for a particular task such as picking a transistor and placing it correctly into a printed circuit board) has been successfully tested by a trusted person, or is it only based on promotional material.

Familiarity: As a domain expert, I don't want to be overwhelmed by an overly complex user interface with a sharp learning curve. I shouldn't have to learn new complex interaction models with the search system, but I should rather be able to rely on established ones (e.g. known from major search engines on the Web).

Advanced Filtering: As a domain expert, I want to be able to filter search results about robots based on economical factors and safety standards. I should be able to filter robots on characteristics such as reach, maximum payload and degrees-of-freedom as well as price range and market availability. Moreover, I want to know whether a robot fits into safety concept of my project and filter based on this.

3.3 Special Manufacturing Requirements

A survey about widespread features in KG-based exploratory search systems was conducted by Marie et al. [MG14] (see left-side of Table 3.1). From the gathered requirements of the informal interviews outlined in the previous Section 3.2, three additional features were identified to be specifically important for the domain of manufacturing (see right-side

Common Features [MG14]	Manufacturing Features
(1) Overview and analysis feature (2) Faceted interface (3) Result clustering (4) Facilitator for back and forth navigation (5) Query-suggestions and refinement (6) Serendipitous discovery enforcement (7) Result explanation generator (8) Memorization feature	(9) Multiperspectival exploration (10) Provenance visibility (11) Hierarchical browsing

Table 3.1: Exploratory search features for the domain of manufacturing.

of Table 3.1). As outlined in our workshop paper [HESP22], the remaining part of this section elaborates on the three extracted features from these collected requirements.

Multiperspectival exploration accommodates diverse information needs by allowing stakeholders to choose their preferred perspective. Stakeholders can focus on the relevant aspects of engineering artifacts without being overwhelmed by irrelevant information that is not pertinent to their engineering discipline. In a manufacturing environment, stakeholders from various disciplines collaborate and contribute their conceptualizations and descriptions of engineering artifacts to the overall knowledge graph.

Provenance visibility is a crucial feature in a manufacturing environment that integrates interdisciplinary knowledge from diverse sources. Provenance visibility becomes particularly valuable when stakeholders need to investigate the properties of an engineering artifact and resolve ambiguities for informed decision-making.

By knowing the original data source of reported property values, stakeholders can assess the reliability and trustworthiness of the information. For instance, a parameter value might come from the manufacturer's promotional material or be measured by a local engineer. Provenance visibility allows stakeholders to quickly reason about the trustworthiness of the presented information, ensuring they can make proper decisions based on accurate and reliable data.

Hierarchical browsing is a prevalent search task within manufacturing knowledge graphs, especially when dealing with complex digital twins of production equipment and machines. These digital twins often exhibit deep containment relationships between components. Similarly, manufacturing processes and product bills of materials frequently involve deep hierarchies. To support stakeholders in efficiently navigating these hierarchies, it is crucial to enable them to focus on essential parts without being overwhelmed by the entire structure.

By providing stakeholders with proper hierarchical browsing capabilities, they can cognitively focus on the most relevant parts of the hierarchy, extracting valuable insights and making informed decisions.

In Chapter 4, three manufacturing-specific desired effects are derived from these requirements for exploratory search systems. The system must (M1) facilitate the traversal of deep hierarchical knowledge structures, (M2) make it easy to cognitively focus on relevant information, and (M3) make it transparent and easy to process where information comes from.

CHAPTER 4

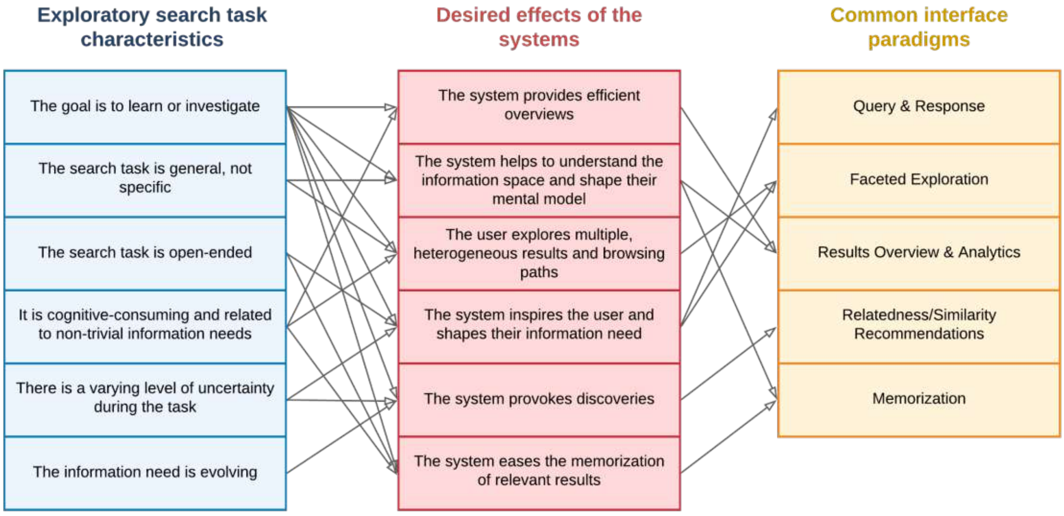
Design of Concepts & Implementation

Manufacturing is a knowledge intensive field with the need for tools to explore this knowledge for better decision-making by stakeholders. Chapter 3 discussed special requirements of engineers in manufacturing towards exploratory search systems. *Multiperspectival exploration* (1), *provenance visibility* (2), and *hierarchical browsing* (3) were presented as features that need special attention.

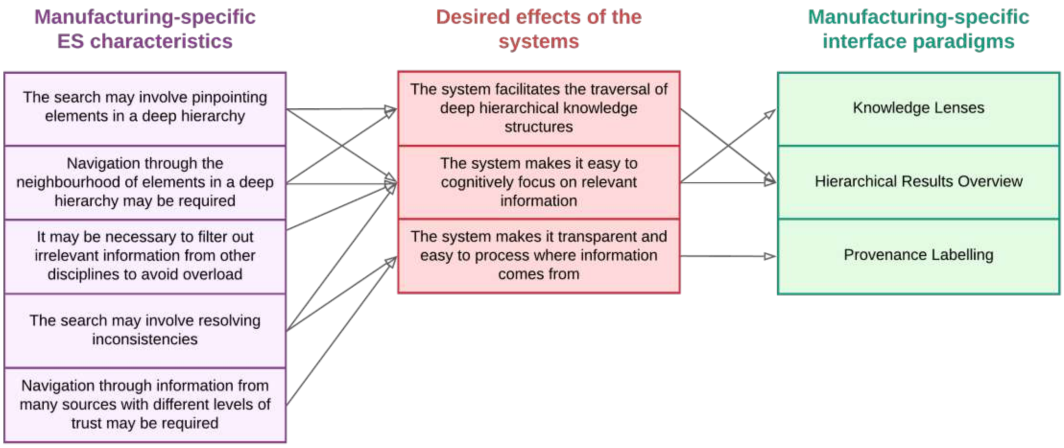
Additional desired effects for exploratory search systems are deduced from these requirements. The system must (M1) facilitate the traversal of deep hierarchical knowledge structures, (M2) make it easy to cognitively focus on relevant information, and (M3) make it transparent and easy to process where information comes from. In Section 4.1, this thesis initially discusses common interface paradigms applied in publicized exploratory search systems. Then, some interface paradigms are highlighted with the potential of satisfying the three additional desired effects for manufacturing.

Building on the insights of this section, the design of an exploratory search system tailored specifically for the manufacturing domain is presented in Section 4.2. The key element of the design is an adaptive user interface, which selects visualizations and underlying search algorithms based on the activated engineering perspective (e.g. mechanical or electrical engineer). Provenance information is passed down to the UI elements such that this information can be made visible. A simple tree view was employed to overcome the challenge of finding relevant information in deep hierarchical structures.

Finally, the concepts and architecture of the proposed ESS are outlined in Section 4.3. The objective of this prototype is to facilitate the integration of new interface paradigms without the need of major architectural restructuring of the core application. The ESS forms the basis of the evaluation in the user study, which is outlined in Chapter 5.



(a) Common interface paradigms, adapted from Marie et al. [MG14].



(b) Manufacturing-specific interface paradigms, constructed from Chapter 3.

Figure 4.1: Interface paradigms for exploratory search.

4.1 Interface Paradigms for Exploratory Search

General exploratory search tasks have a number of defining characteristics (see Section 2.4.1). As outlined by Gary Marchionini [Mar06], learning and investigating are pertinent to this kind of search task. The information seeker does not usually have a specific goal in mind, but the search is open-ended and the information need might evolve over time depending on newly discovered information and interests. Overall, a list of *desired effects* towards an exploratory search interface can be derived from these defining characteristics as depicted in Figure 4.1a. Exploratory search systems aim to achieve these *desired effects* by implementing specific *interface paradigms*. Section 4.1.1 summarizes commonly implemented *interface paradigms* that address one or multiple of these *desired effects* towards a general exploratory search interface.

While exploratory search tasks in the manufacturing domain adhere to the same general characteristics, this thesis wants to draw special attention to some specific characteristics that are not prominently discussed for general exploratory search. As outlined in more detail in the previous Chapter 3, information seeking engineers in manufacturing face the challenge of disciplinary-specific conceptualizations, information overload and inconsistencies. Consequently, a number of additional *desired effects* are expected from an exploratory search system as depicted in Figure 4.1b. Section 4.1.2 discusses manufacturing-specific *interface paradigms* that address these additional *desired effects*.

4.1.1 Common Paradigms

Marie et al. [MG14] identified a number of *desired effects* (see Section 4.1a) for exploratory search systems: (G1) the system provides efficient overviews, (G2) the system helps to understand the information space and shape their mental model, (G3) the user explores multiple, heterogeneous results and browsing paths, (G4) the system inspires the user and shapes their information need, (G5) the system provokes discoveries, and (G6) the system eases the memorization of relevant results.

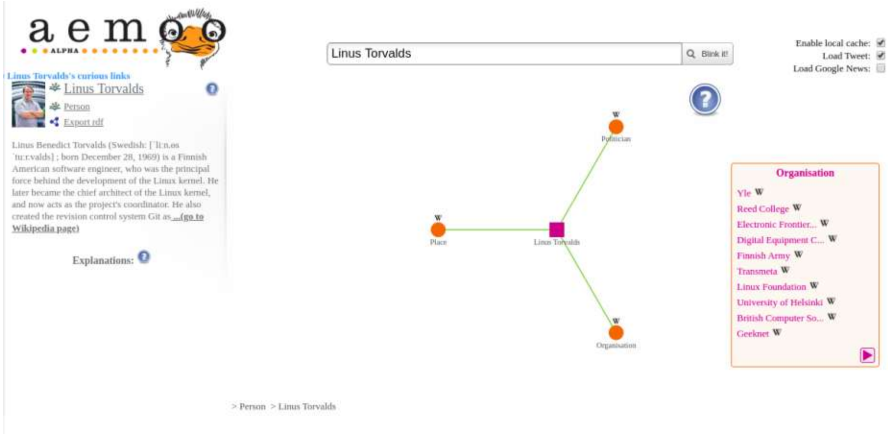
Over the last two decades, the publicized exploratory search systems tried to provide these *desired effects* using different approaches. The most common interface paradigms that address one or more of these *desired effects* are summarized in this section.

Query & Response

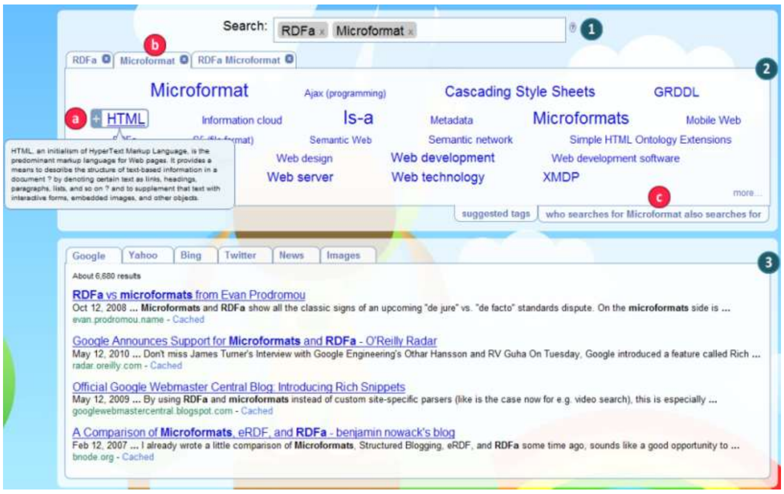
Lookup and keyword search are the most popular interaction paradigm among the surveyed exploratory search systems. It should allow the information seeker to shape their information need (G4). White refers to this paradigm as query-response, *"where queries are issued by the user, and a set of potentially relevant items are offered in response"* [WR09].

Certain exploratory search systems such as Aemoo [NPG^{+}17] (see Figure 4.2a) aim to identify an exact match for the specified search term, subsequently providing relevant information regarding this exact match. Similarly, this approach is applied for Discovery Hub [MGRR13] and inWalk [CFM14]. In case of Aemoo, an information box provides a high-level overview on a selected entity, while an interactive graph view allows the user to navigate through directly related entities.

In contrast, other exploratory search systems, such as Lookup Explore Discover (LED) [MN10] (see Figure 4.2b), operate differently by yielding a series of items that align with the inputted search phrase, similar to the functionality of most widely recognized search engines on the Web.



(a) Aemoo [NPG+17] (general domain)

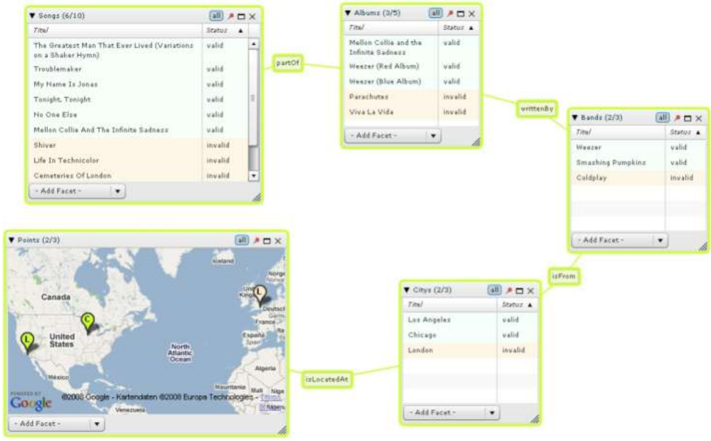


(b) Lookup Discover Explore (LED) [MN10] (general domain)

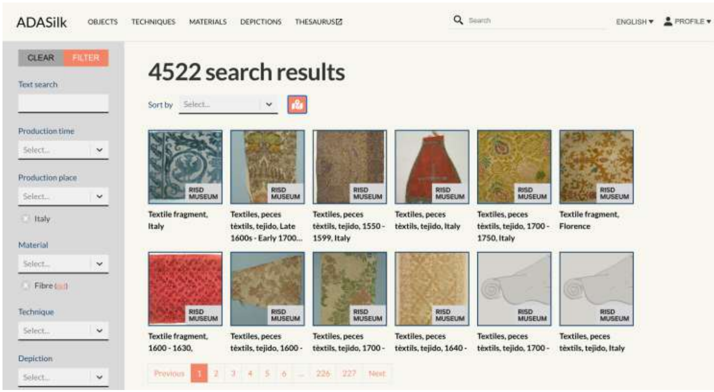
Figure 4.2: Selected exploratory search systems to showcase the query-response paradigm.

Faceted Exploration

Faceted exploration is an interface paradigm that enables information seekers to shape their information need (G4) and explore multiple heterogeneous results and browsing paths (G3). An information seeker can on their own behalf fluidly transition between browsing strategies, and thus, facets form an alternative to query reformulation [KCBS09]. As outlined by Heim et al. [HZL08], in faceted exploration, *"the data gets partitioned using orthogonal conceptual dimensions. One of the dimensions serves as the result set and the others are used as facets to filter the result set by different attributes that can be selected independently from each other"*. Facets are a well-known and popular interface paradigm. Moreover, facets have shown to play an important role in the exploratory search process for information seekers as outlined by Kules et al. [KCBS09].



(a) gFacet [HZL08] (general domain)



(b) AdaSilk [ELT21] (cultural heritage)

Figure 4.3: Selected exploratory search systems to showcase the faceted exploration paradigm.

Recognizing the significance of facets in exploratory search, several variants have been proposed throughout time. An example is gFacet [HZL08] (see Figure 4.3a), which incorporates a graph-based navigation system within hierarchical facets. This hierarchy within facets enables query refinement through selecting desired values for a specified chain of properties.

On the other hand, AdaSilk [ELT21] (see Figure 4.3b) utilizes the traditional facets in a sidebar layout, a design approach commonly known from e-commerce sites. Each facet serves as a filter for a particular attribute within the knowledge graph, giving users the option to confine the results to entities possessing the chosen property value. SemFacet [AGK+16] is capable of auto-generating this kind of facets given that metadata and domain knowledge is represented as OWL 2 (see Section 2.2.2) in the knowledge graph. Meanwhile, GraFa [MH18] enables responsive faceted navigation of this type for large-scale knowledge graphs on the scale of Wikidata.

Results Overview & Analytics

The presentation of the information space is as important as an intuitive interface for query refinement. Sense-making is a pertinent activity of exploratory search, and the user might want to do several one-off pinpoint searches. They might seek specific information to enhance their comprehension of a result or to discern the reasoning why a result was proposed for a given query [Pal18]. Hence, the search system must help the user to understand the presented information space and assist them in shaping their mental model (G2). Overall, the system must provide efficient overviews (G1). Typically, there are three main approaches employed by established exploratory search systems.

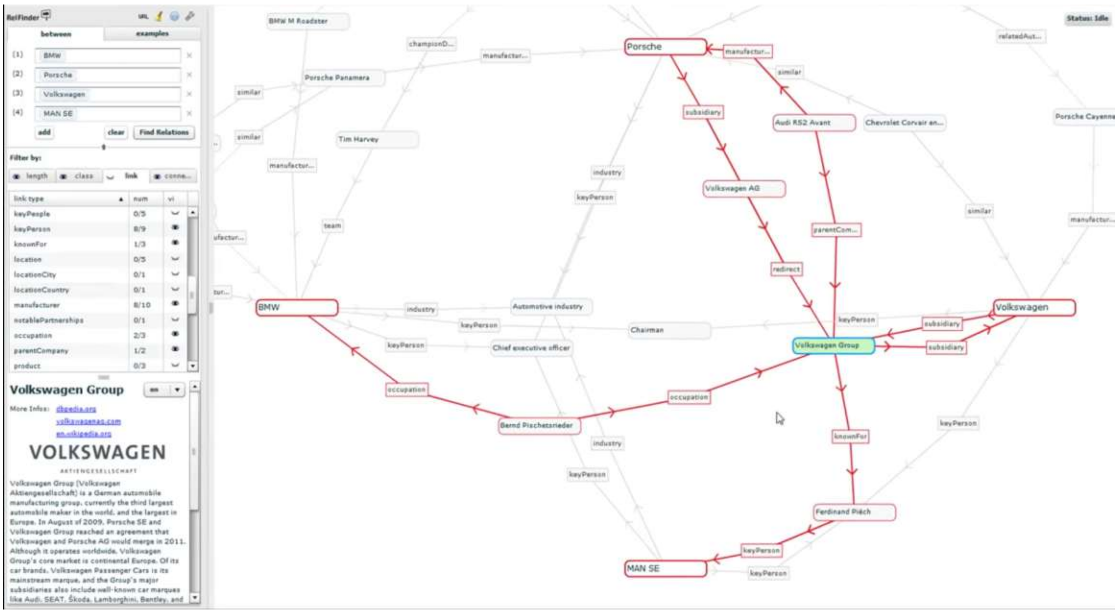
(1) Graph-based visualizations are the most common approach for presenting the information space in KG-based exploratory search systems. Given that the underlying data model of knowledge graphs resembles a directed, labeled graph, it is only natural to use a node-link layout. Po et al. [PBDP20] draw a distinction between *relation-oriented* and *incremental* graph visualizations.

In *relation-oriented* graph visualizations, the focus is on presenting the paths between entities in the knowledge graph [PBDP20]. RelFinder [HLS10] (see Figure 4.4a) for instance allows the user to select a number of entities, and then, highlights all paths in the knowledge graph that connect these selected entities.

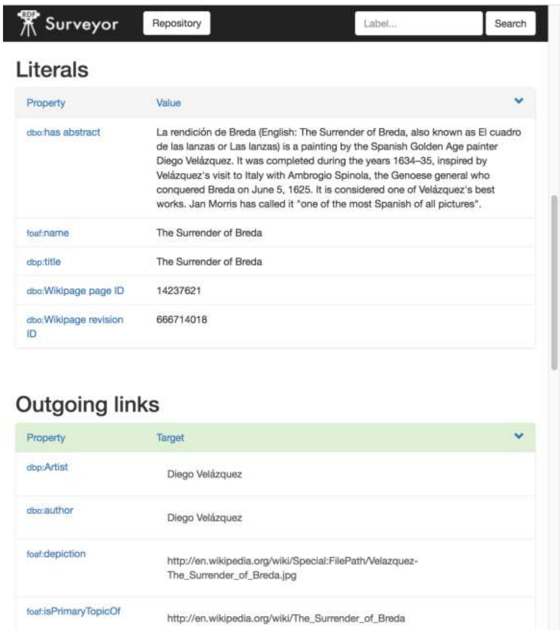
In *incremental* graph visualizations, the initial focus is on a starting point within the knowledge graph, as opposed to presenting the entire graph at once. Then, as the user interacts with the system, additional sections of the graph can be progressively unfolded [PBDP20]. The previously presented system, Aemoo [NPG+17] (see Figure 4.2a), lets the user jump to a specific entity in the knowledge graph with a lookup search, and then, the user can navigate through the neighboring entities from this starting point.

(2) Tabular-based visualizations started to be utilized by the early semantic browsers such as Tabulator [BlCC+06]. These tools drew significant inspiration from traditional Web browsing methods, offering a simplified tabular data view. Unfolding new sections of the knowledge graph is accomplished by following Web links embedded within the tabular presentation. RDF Surveyor [VGS17] (see Figure 4.4b) is such a modern semantic browser, which can be applied to any knowledge graph over a SPARQL endpoint (see Section 2.1.1). While graph-based exploration tools use an interactive graph-view for navigation instead, many of these tools also incorporate tabular visualizations, often implemented as compact information boxes (e.g. LODmilla [MTG14] or Aemoo [NPG+17]).

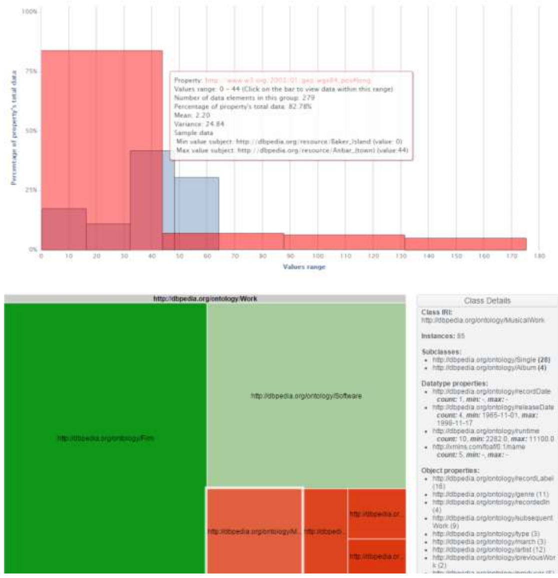
(3) Chart-based visualizations display information from the knowledge graph in various chart forms, including scatter charts, and tree maps among others. A detailed insight into these chart forms is out of scope for this thesis, but they are elaborated on by Po et al. [PBDP20]. Nonetheless, these forms of visualization are frequently overlooked in many exploratory search systems. SynopsViz [BPSS17] (see Figure 4.4c) is an exception as it introduces a generic tree model for exploring and analysing numeric as well as temporal data in a multilevel manner.



(a) Graph-based visualization by RelFinder [HLS10] (general domain)



(b) Tabular-based visualization by RDF Surveyor [VGS17] (general domain)



(c) Chart-based visualization by SynopsViz [BPSS17] (general domain)

Figure 4.4: Different visualization types across exploratory search systems.

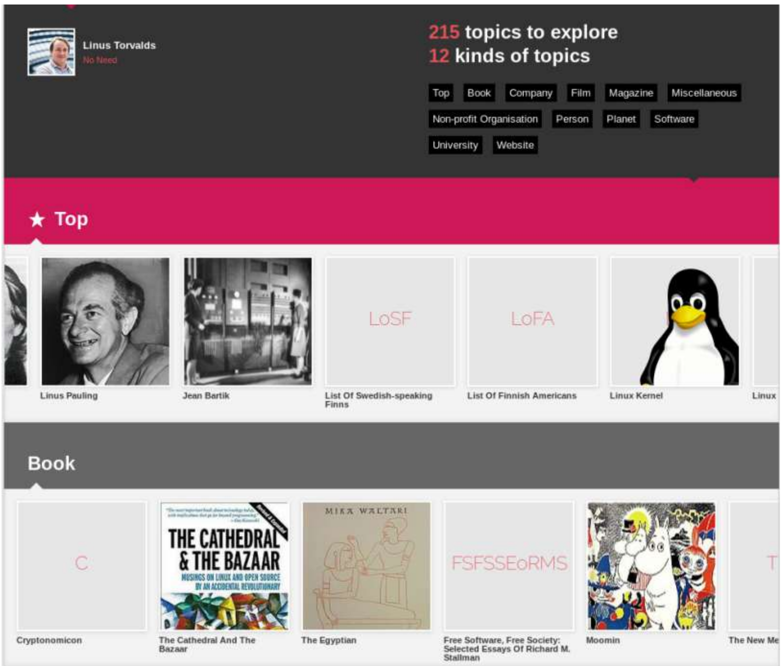


Figure 4.5: Recommendations by DiscoveryHub [MGRR13] (general domain)

Recommendations

Recommendations have become an integral part of (exploratory) search systems. Typically, recommendation engines provide the user with a list of recommended items they might be interested in, or the engine even predicts how much they might prefer each item [GSY22]. In context of exploratory search, recommendations suggest additional exploration paths to a user and hence, such a system provokes new discoveries (G5). While prediction accuracy and coverage are important metrics for recommenders, diversity and serendipity may play a more important role for exploratory search.

Generally, recommendation systems can be categorized into collaborative-filtering-based and content-based systems. However, KG-based exploratory search systems usually utilize a content-based approach, where either traditional graph algorithms are applied or more recently KG embeddings [PB18]. Lately, there's been an increase of proposed hybrid systems where user profiles are modeled in user-item knowledge graphs. Given the corresponding item knowledge graph, these systems employ algorithms to suggest items tailored to a specific user profile [GZQ$^+$22].

In terms of presenting recommendations, there is little variation across KG-based exploratory search systems. Recommended items are usually displayed in a simple list format, as it can be observed in DiscoveryHub [MGRR13] (see Figure 4.5) for instance. DiscoveryHub furthermore computes a number of related topics for an entity, and recommends specific items for each of these topics.

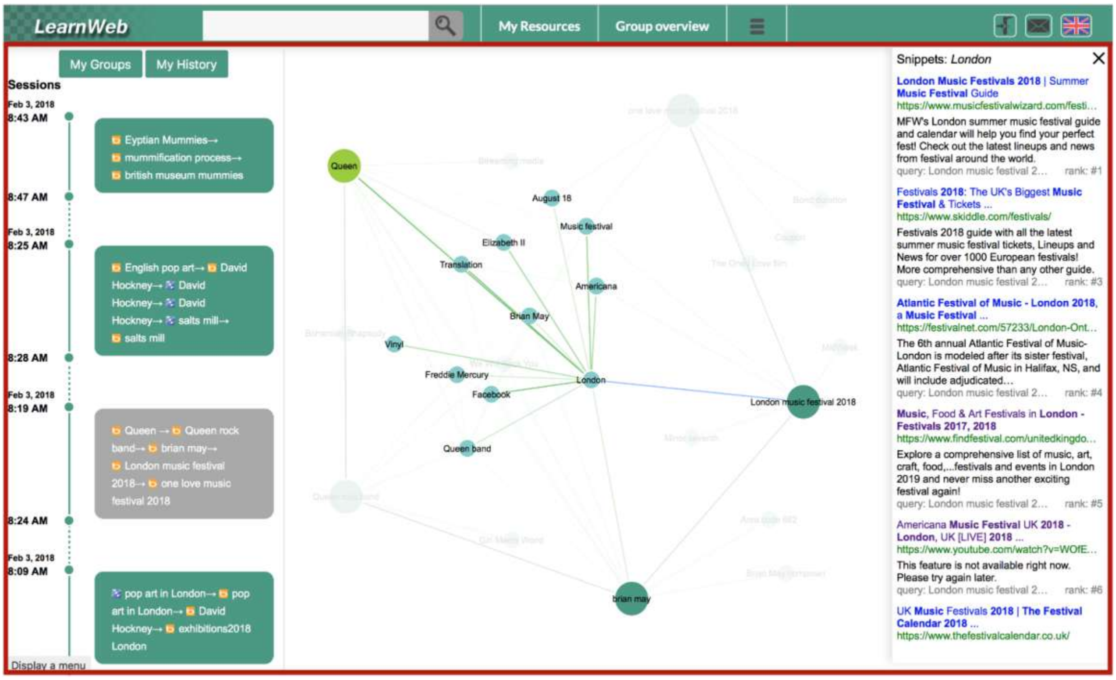


Figure 4.6: Search history (LogCanvas) by LearnWeb [XFZN18] (education)

Memorization

Exploratory search tasks can span hours, days, or even weeks, encompassing numerous search sessions as noted by White [WR09]. Consequently, it's crucial for the system to facilitate the memorization of queries and results (G6).

Maintaining a log of search sessions can aid users in retaining and revisiting their exploratory search journey, allowing them to quickly retrieve previously encountered information or insights. LogCanvas [XFZN18] is an advanced search history tool that groups search activities into sessions, as shown on the left side of Figure 4.6, and it identifies semantic entities within text queries and browsed content. A user can open an interactive graph visualization for a specific session, which displays the identified entities and their relationships within a target knowledge graph. LogCanvas aims to assist users in re-constructing the semantic relationships among their search activities.

Alternatively, the in-session memory feature (commonly referred to as breadcrumbs) eliminates the need for users to recall their browsing sequences, freeing up cognitive resources for more meaningful search tasks [MG14]. Aemoo [NPG$^+$17] (see Figure 4.2a) utilizes breadcrumbs, allowing users to navigate back and forth, retracing their exploration path.

Moreover, bookmarks are a frequently employed tool for revisitation [AJK05], enabling users to proactively add results or queries to their personal collection.

4.1.2 Manufacturing-specific Paradigms

In Chapter 3, this thesis delved into the unique requirements of engineers in manufacturing concerning exploratory search systems. Information seeking engineers in manufacturing face the challenge of disciplinary-specific conceptualizations, information overload and inconsistencies. Three features were highlighted as particularly essential: multiperspectival exploration (1), provenance visibility (2), and hierarchical browsing (3).

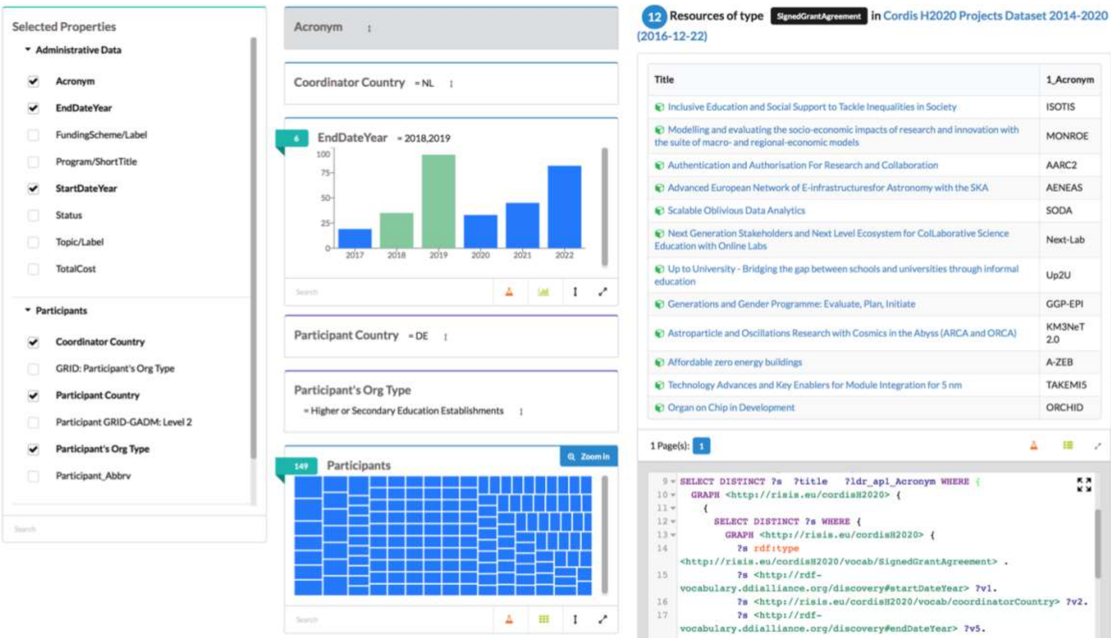
Beyond the general *desired effects* outlined in the preceding section, three specific *desired effects* were derived from these features: (M1) the system facilitates the traversal of deep hierarchical knowledge structures, (M2) the system makes it easy to cognitively focus on relevant information, and (M3) the system makes it transparent and easy to process where information comes from. This section highlights interface paradigms that address one or more of these three *desired effects*.

Knowledge Lenses

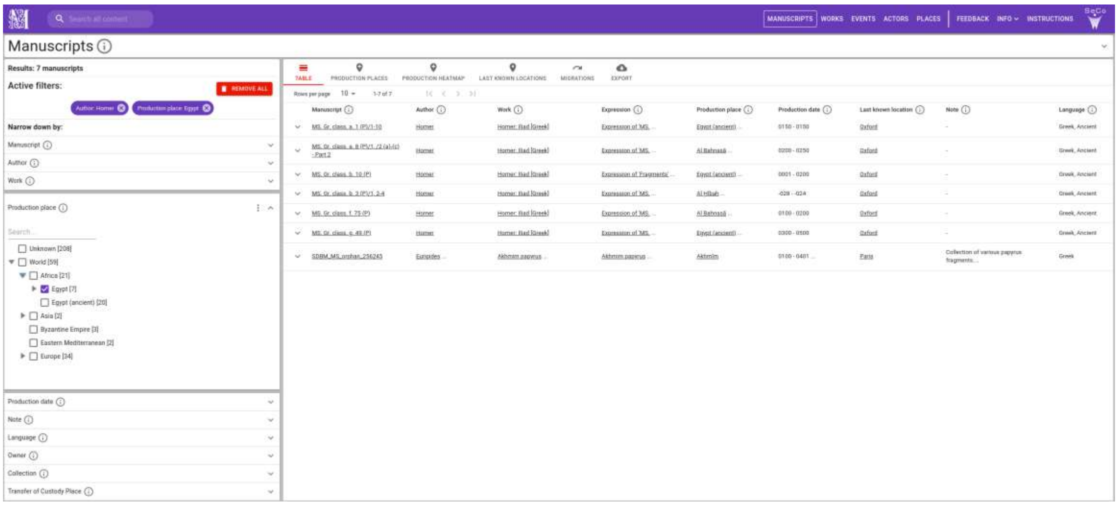
An integral part of exploratory search is the act of navigating the information landscape, or in context of this thesis, browsing entities within a knowledge graph. As Bates [Bat07] suggests, browsing involves a sequence of glimpses at entities. Some of these glimpses may lead to a more in-depth examination, while others prompt glimpses at additional entities. This underscores the need for the information seeker to quickly identify and target an entity of interest from a visually presented scene. As outlined in the information foraging theory by Pirolli et al. [PC99], the information seeker aims to select entities so to maximize their information gain per invested energy unit.

In the manufacturing domain, the conceptualization and views of engineering artifacts might vary between engineering disciplines. The characteristics of a motor could for instance be described in a impressively detailed manner by mechanical engineers, but only superficially by electrical engineers. Thus, it is crucial for engineers to focus on the relevant aspects of artifacts without being overwhelmed by irrelevant information that is not pertinent to their engineering discipline. The system must make it easy to cognitively focus on relevant information (M2).

Nuzzolese et al. proposed the Encyclopedic Knowledge Patterns (EKP) [NGPC11] to address the competency question: "*What are the most relevant entity types that provide an effective and intuitive description of entities of a certain type?*". In essence, they sought to determine the properties of an entity that should be showcased to users to provide an easily understandable overview of that entity. Their method involved analyzing the forward and backward links on Wikipedia pages, leading to the formulation of knowledge patterns for 84 different entity classes (e.g. persons or places). These patterns act as a filtering lens on the knowledge graph, ensuring users are not overwhelmed with extraneous properties when seeking information about an entity, a technique demonstrated in Aemo (see Figure 4.2a). Nonetheless, these pattern were defined for a general domain, operating under the assumption that all users would prefer the same knowledge pattern for an



(a) Adaptive user interface by Linked Data Reactor [KLvH16] (general domain)



(b) Multi-perspective portal by Sampo-UI [IHRK22] (cultural heritage)

entity class. This thesis argues however that this assumption is false for the domain of manufacturing.

Khalili et al. [KLvH16] proposed the Linked Data Reactor (LD-R), which suggests the use of adaptive and data-driven web components. These components can be configured and then compiled to create a customized user interface. As illustrated in Figure 4.7a, LD-R makes use of the faceted exploration paradigm to allow users to navigate the knowledge graph efficiently. In this system, both facets and the visualization of entities

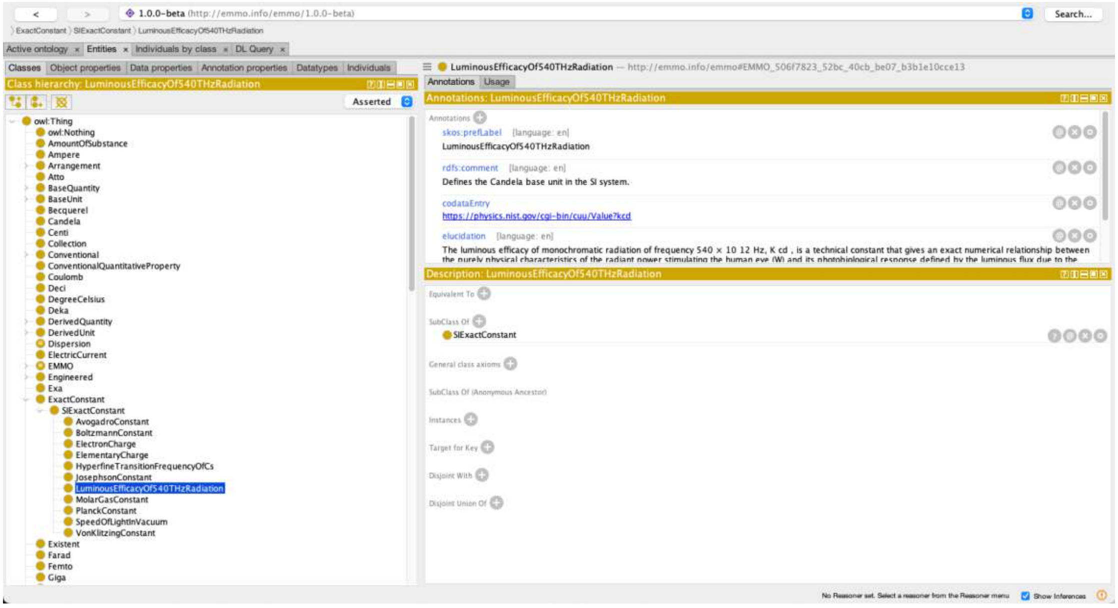


Figure 4.8: Tree view in Protégé ontology editor [Mus15] (knowledge engineering)

are configurable, allowing for a tailored user experience. While an application assembler would handle the initial configuration, users can potentially modify the user interface according to their preferences and requirements, ensuring it aligns with their interests and needs. The ESS prototype extends this approach by having a list of predefined perspectives (e.g. mechanical engineer) instead of individual user profiles. Section 4.2.1 elaborates on this extension.

Ikkala et al. [IHRK22] advocate for providing end-users with multiple application perspectives to the content of a knowledge graph. Similarly to LD-R, faceted exploration is the key paradigm for navigating the content of a knowledge graph. An application perspective then dictates the facets that are displayed for a specific faceted browsing experience. Figure 4.7b illustrates one faceted search perspective over 200,000 medieval and renaissance manuscripts within a cultural heritage knowledge graph. Additionally, a perspective defines the available data-analytic visualizations (see Section 4.4).

Hierarchical Results Overview

Hierarchical browsing is a common search task within manufacturing knowledge graphs, particularly when working with intricate digital twins of production equipment and machinery. These digital twins often display extensive containment relationships between components. Likewise, manufacturing processes and product bills of materials often encompass deep hierarchies. Hence, the system must generally make it easy to cognitively focus on relevant information (M2), and allow the information seeker to efficiently traverse deep hierarchical knowledge structures (M1).

Tree views are a commonly used interface paradigm for navigating hierarchical structures. They consist of top-level nodes, which can either branch into child nodes or stand as terminal nodes. Nodes with children can be toggled to reveal or conceal them, and visual cues often indicate whether a node can be expanded [Pap11]. Figure 4.8 depicts the conventional tree view employed in the Protégé ontology editor, which is used to navigate the class hierarchy in ontologies.

As Tominski et al. [TAvHS06] argue, a limitation of these textual tree views is their restricted capacity for displaying a limited number of entities simultaneously. While vertical and horizontal scroll bars allow exploration of a tree's breadth and depth by bringing different parts into view, obtaining a comprehensive overview of the tree structure remains a challenging task. Hence, they suggest a non-textual fisheye tree view, which presents the hierarchical structure through a top-down graphic representation. Additionally, this visualization allows for the application of lenses to magnify specific areas of interest.

Provenance Labeling

Provenance visibility is a key aspect in manufacturing settings where knowledge from various disciplines is merged. Its importance is amplified when stakeholders need to understand the characteristics of an engineering artifact and clear up uncertainties to make well-informed choices. Knowing where the data originated, stakeholders can evaluate how much they can trust the information. Hence, the system has to make it transparent and cognitively easy to process where information comes from (M3).

As outlined in Section 2.1.2, multiple mechanisms within RDF and its extensions are available to articulate provenance information about a specific statement in a knowledge graph. This diversity presents a significant challenge in developing a universal strategy for utilizing this provenance knowledge effectively in the design and implementation of user interfaces.

Transparent Fresnel as proposed by Rutledge et al. [RBLS23] is an extension of Fresnel to support RDF reification, which is one of the various mechanisms to provide provenance. Fresnel is a presentation vocabulary that serves as a bridge between the structured data model of RDF and its visualization on human-friendly web interfaces. It introduces the concepts of *lenses* and *format* together with a box-based presentation model. *Lenses* determine the properties of a class of resources in the knowledge graph to be displayed and their arrangement order, while *formats* dictate the styling and presentation of the content chosen by these *lenses* [PBKL06]. Transparent Fresnel wraps the boxes visualizing the property of a resource into a dedicated "reify" box, which allows to specify the visualization of provenance information.

Similarly, the ESS presented in this thesis shares the provenance knowledge with the corresponding UI components in order to enable the development of components visualizing this knowledge. It however assumes that the *singled triple named graph* approach is used in favor of RDF reification and other mechanisms.

Interface Paradigms	Desired Effects	Solution
Query & Response	G4	Keyword search with result list (see Fig. 4.9 δ_1, δ_2)
Faceted Exploration	G3, G4	
Results Overview & Analytics	G1, G2	Tabular visualizations with information boxes (see Fig. 4.9 α)
Recommendations	G5	List of items (see Fig. 4.9 γ)
Memorization	G6	Search history log (see Fig. 4.9 η)
Knowledge Lenses	M2	Adaptive UI (see Fig. 4.9)
Hierarchical Results Overview	M1, M2	Tree view (see Fig. 4.10a μ)
Provenance Labeling	M3	Provenance tables in adaptive UI (see Fig. 4.9 β)

Table 4.1: Design choices for the exploratory search system (ESS).

4.2 Exploratory Search Interface

Drawing upon requirements collected from stakeholder interviews associated with the two manufacturing use cases (see Chapter 3), coupled with a thorough literature review of publicized systems and common interface paradigms (see previous Section 4.1), an exploratory search system has been designed for the manufacturing domain. Table 4.1 summarizes the design choices made for the user interface. While faceted exploration is valuable for this domain, it hasn't been implemented for time reasons.

The interface is designed to resemble popular search engines on the Web, aiming to minimize the learning curve for novel users. The principal entry point to initiate an exploration is through a keyword search, as illustrated in Figure 4.9 δ_1. The outcomes of a keyword search are displayed in a vertical list (see Fig. 4.9 δ_2). An information box is presented for the top-ranked entry in the results list (see Fig. 4.9 α). As with the previously discussed semantic browsers, new sections of the knowledge graph can be unfolded by following the Web links in the information boxes or result lists.

At the upper-left corner (see Fig. 4.9 ε), the entry point for exploration can be switched from a keyword search to either a tree view or a SPARQL editor. Moving forward, additional entry points could be added. The tree view (see Fig. 4.10a μ) was incorporated to aid users in comprehending and navigating complex hierarchical structures more effectively. It can be controlled on the left panel, while an information box is shown on the right side for the currently selected entity in the tree view (see Fig. 4.10a α). The SPARQL editor (see Fig. 4.10b λ) was added for Semantic Web experts, but should be disabled or hidden for end users in the manufacturing domain.

A simple search history log (see Fig. 4.9 η) can be expanded at the upper-right corner. The log is organized such that the most recent activity is displayed at the top. Each entry represents a previous state and by clicking on the right "back" icon, the search system recovers this state and jumps back to it.

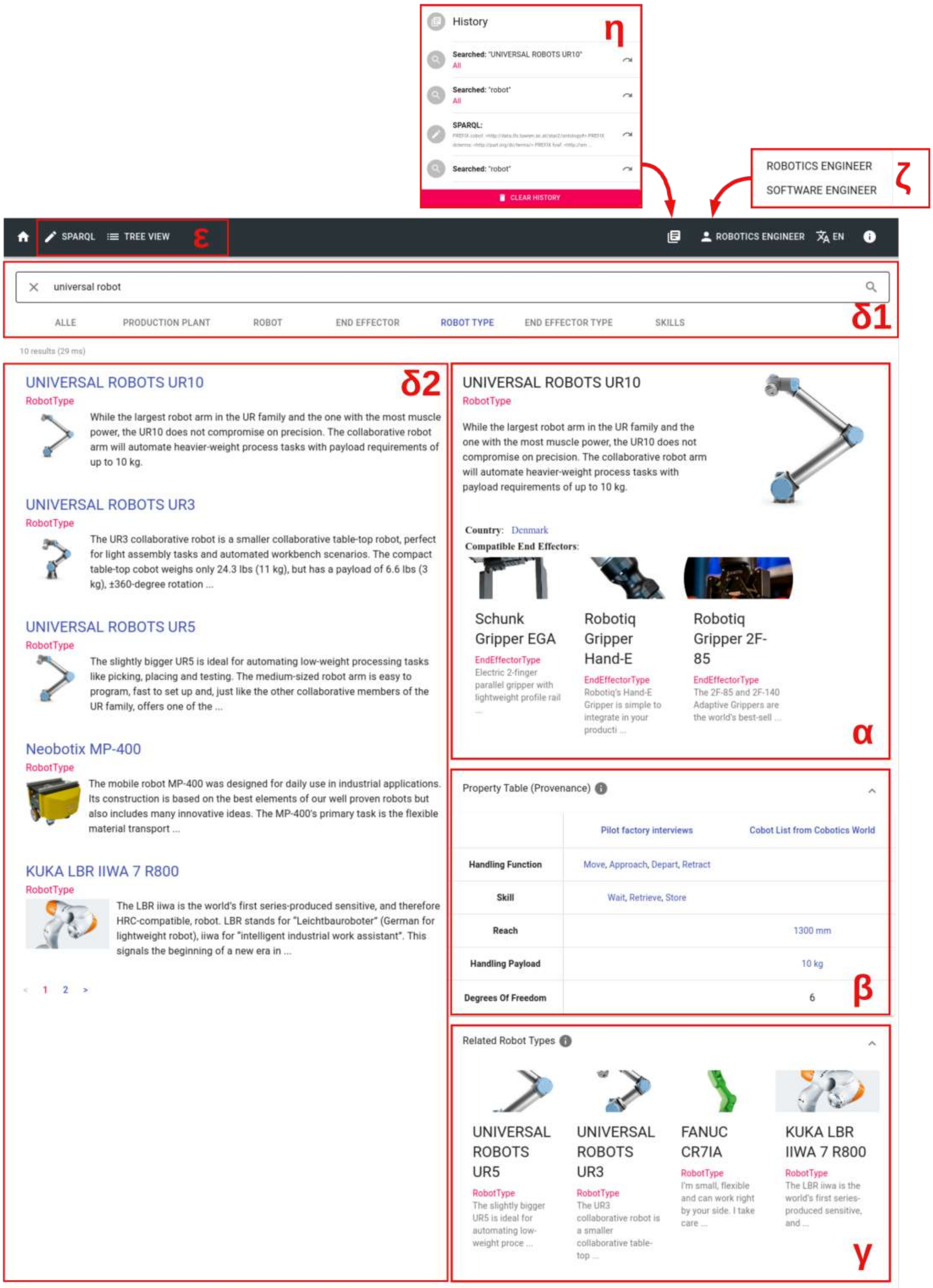
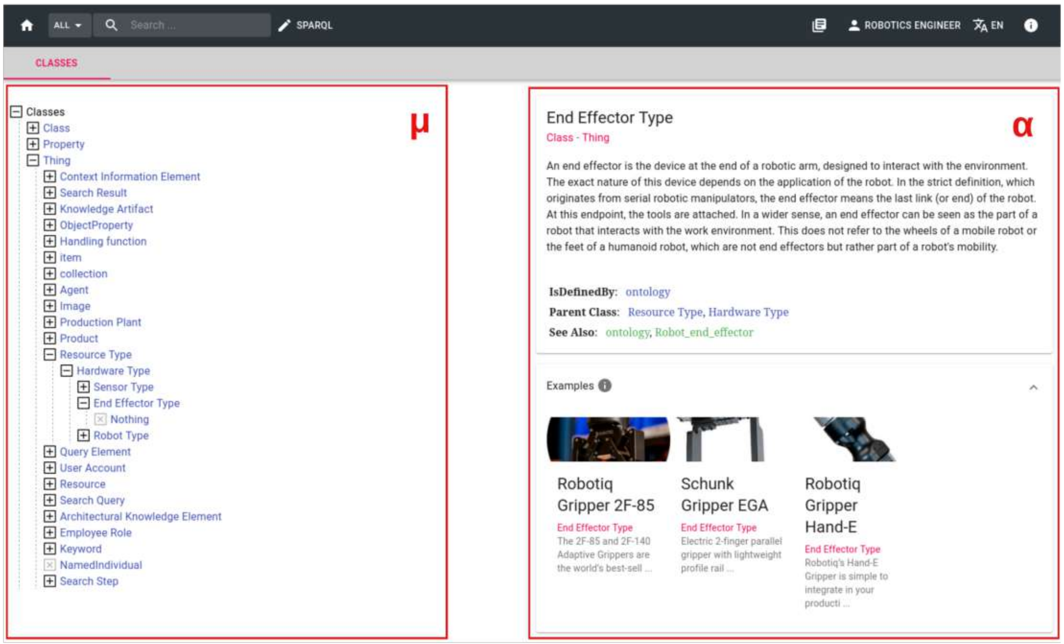
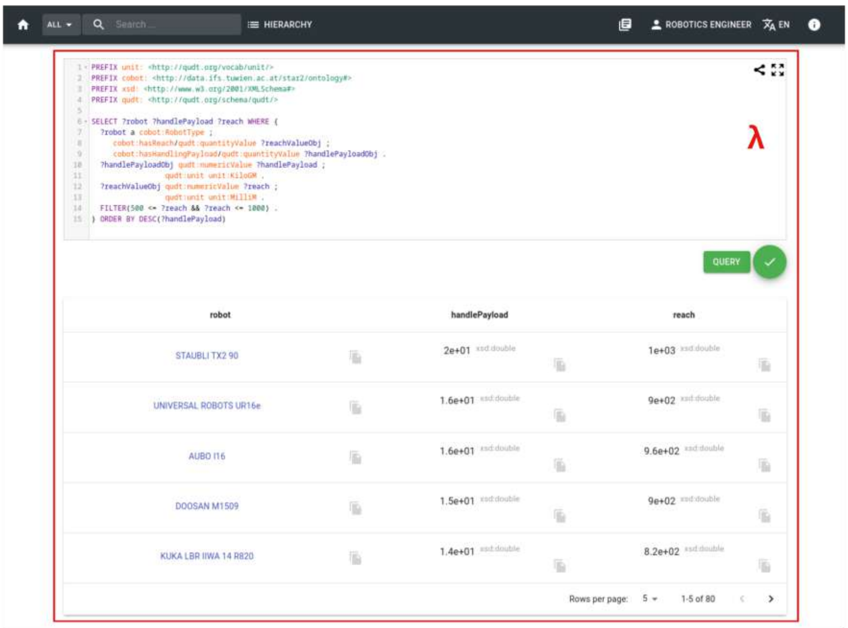


Figure 4.9: Exploratory Search System (ESS) [HESP22] (manufacturing).



(a) Tree view as entry point for exploration.



(b) SPARQL editor as entry point for exploration.

Figure 4.10: Other entry points for exploration in the ESS.

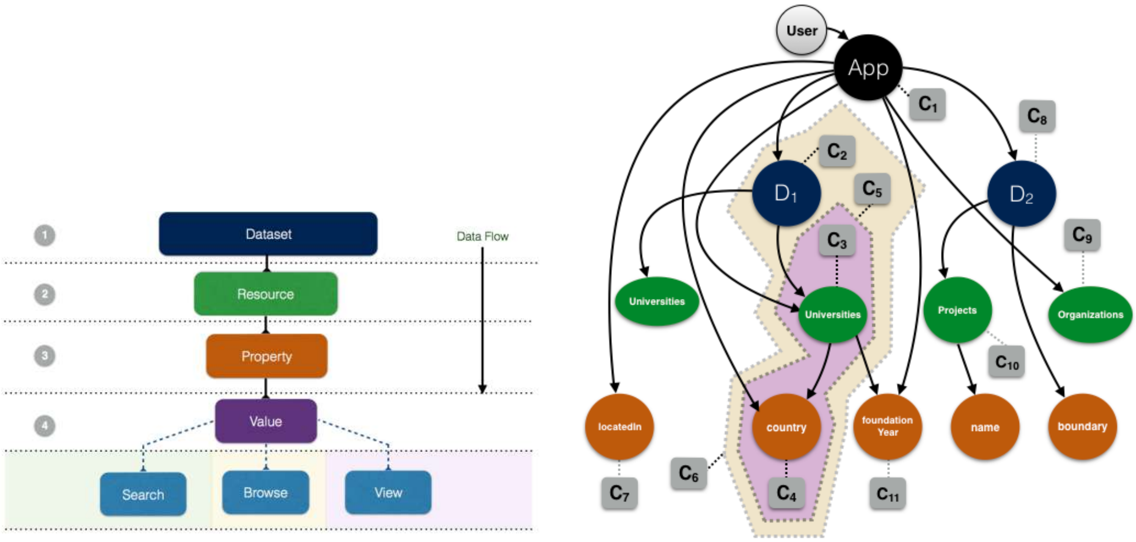


Figure 4.11: Scopes in the Linked Data Reactor (LD-R) [KLvH16].

Most importantly, the currently selected perspective can be switched in the upper-right corner (see Fig. 4.9 ζ). Depending on the defined configuration of a specific perspective, a switch can change the rendered UI and search experience quite considerably. This mechanism is elaborated on in the subsequent Section 4.2.1.

4.2.1 Adaptive UI

This adaptive search interface is based on the concept of *scopes* and *configurations*, which was introduced by the Linked Data Reactor (LD-R) [KLvH16]. LD-R proposes adaptive data-driven web components that can be compiled into a custom user interface and configured by an application assembler, thereby reducing the need to design such applications from the scratch.

A scope is in LD-R a hierarchical permutation of *dataset*, *resource*, *property* and *value*, where *dataset* is at the top and *value* at the bottom of this hierarchy, as depicted in Figure 4.11. Presentation templates for LD-R are written in JSON format for a particular scope, which tells the web application how to render entities in the knowledge graph that match this scope. Each scope has a specificity, and if entities belong to multiple scopes, then the configuration of the most specific scope overwrites the others.

Scopes

While the ESS adopts the mechanism from LD-R for it‘s adaptive search interface, the *scope components* and their hierarchy were changed to integrate multiperspectival exploration. The new organization of *scope components* for the ESS is illustrated in Figure 4.12. The remainder of this section outlines each scope component in more detail.

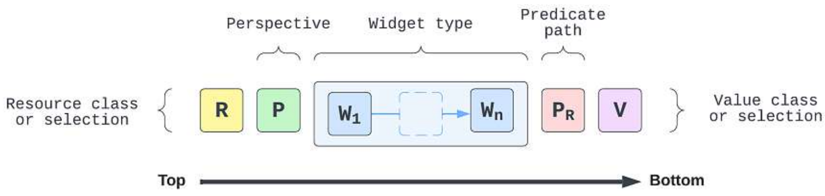


Figure 4.12: Scope hierarchy for the adaptive UI engine [HESP22].

Resource class or selection <u>R</u> is at the top of this hierarchy. This is either the IRI of a class, or the IRI of one single specific resource. The class `rdfs:Resource` can be used as a wildcard to match all resources in a knowledge graph.

Perspective <u>P</u> is a newly introduced *scope component* for the ESS. A perspective is identified by an unique name such as `"Software Engineer"` or `"Robotics Engineer"` (see Fig. 4.9 ζ). With this *scope component*, the presentation of widgets and entities as well as the algorithms powering the search experience can be customized for a specific perspective. The underscore (i.e. '_') is a wildcard and represents all perspectives.

Widget path <u>$\mathbf{W_1}$, ..., $\mathbf{W_n}$</u> is the subsequent component in the hierarchy. Widget refers here to an interface element with an unique name. The list always begins with the outermost widget and progresses towards the innermost one. This *scope component* enables the distinct presentation of entities dependent on the widget path context, e.g. different presentation of an entity in context of a bookmark widget compared to an information box (see Fig. 4.9 α).

Predicate path <u>$\mathbf{P_R}$</u> is one level below in the hierarchy and a subset of predicate paths in SPARQL, only allowing sequences (i.e. '/'), alternatives (i.e. '`|`') and inversion (i.e. '^'). A predicate path allows to define how the values of a matching property are presented.

Value class or selection <u>V</u> is at the bottom of the hierarchy. This is either the IRI of a class, or the IRI of one single specific resource. In contrast to the resource selector **<u>R</u>**, it makes sense here to also define the class `rdf:Literal` in order to distinguish between literals and other resources. This *scope component* enables the configuration of UI elements for matching values. Provenance metadata is passed to the UI elements of values, but no matching mechanism is proposed in this thesis to allow the adaptive rendering based on information in this metadata. Moreover, it is assumed that the *single-triple named graphs* approach is used to state provenance information in the knowledge graph, in favor of *RDF reification*, *singleton properties* or *RDF-star* (see Section 2.1).

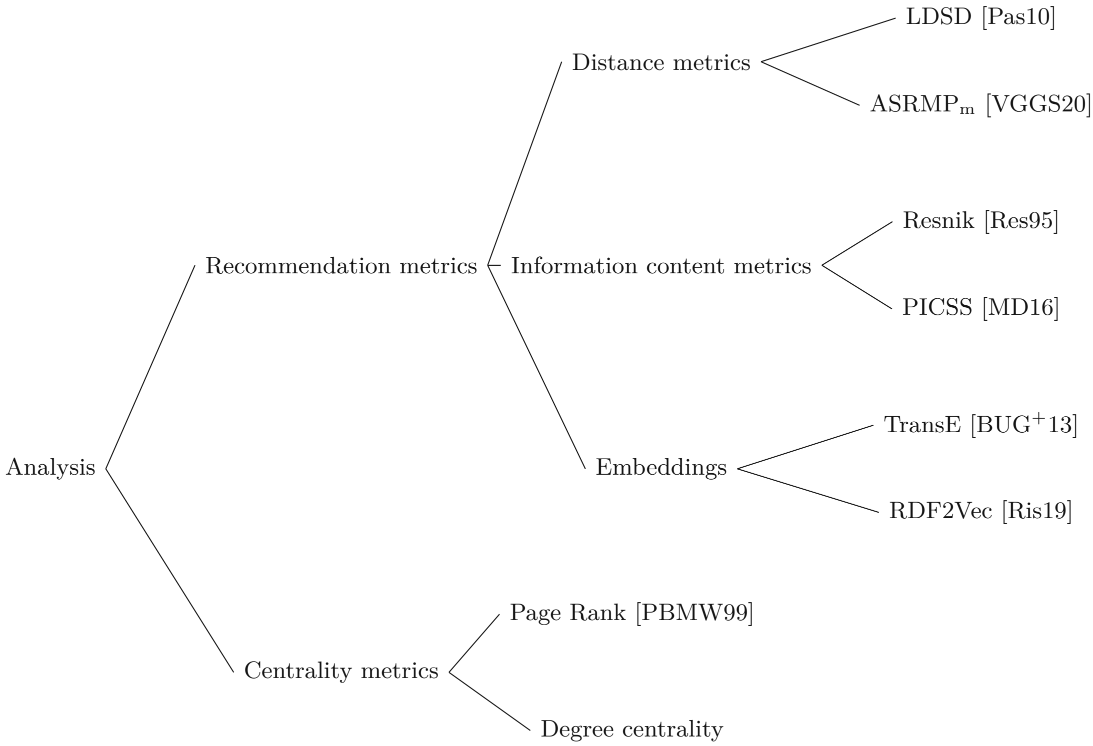


Figure 4.13: Centrality and similarity metrics for KG analysis.

Centrality & Recommendation Metrics

Knowledge graphs potentially consist of a large number of resources, and when displaying a bigger subset of these resources to the user, the questions arises, how to rank them. *Centrality metrics* help to identify the most "important" vertices in general graphs, and thus, can be used to rank resources of a knowledge graph based on their "importance".

A selection of centrality metrics M_c can be defined for a specific perspective within the adaptive user interface. A *centrality score* is computed from this selection M_c, which is then used to rank resources. A weight can be applied to each metric in order to allow prioritization of a metric over another.

Centrality score: When given a resource $v \in V(KG)$, the *centrality score* is defined as following, where w_i is the assigned weight to the metric f_i:

$$rank(v) = \sum_{f_i \in M_c} w_i * f_i(v) \tag{4.1}$$

A common interface paradigm among exploratory search systems is the recommendation of similar or related entities to a given entity (see Section 4.1.1). A *recommendation metric* assigns a score to a pair of resources $(i_1, i_2) : i_1, i_2 \in V(KG)$ in the knowledge graph.

The ESS prototype implements a number of metrics to evaluate either the similarity or relatedness of resources. Similarly to *centrality metrics*, a selection of *recommendation metrics* M_{rec} can be defined for a resource selector $\underline{\text{R}}$ within the adaptive user interface, and a weight can be applied to each metric. A *top-n recommendation* is then computed from this selection M_{rec} for a given resource, if this resource matches $\underline{\text{R}}$.

Top-n recommendation: When given an resource $v \in V(KG)$ and a set of candidates $P \subseteq V(KG)$, the recommendation is defined as following, where w_i is the assigned weight to the metric f_i:

$$rec(n, v) = \underset{P' \subseteq P, |P'|=n}{\arg\max} \sum_{u \in P'} \sum_{f_i \in M_{rec}} w_i * f_i(v, u) \tag{4.2}$$

Centrality and *recommendation metrics* can have different domain ranges. The simple degree metric for instance maps a resource $v \in V(KG)$ to the range of $[0, \infty[$, whereas Page Rank [PBMW99] maps it to the range $[0, 1]$. Thus, feature scaling is required before computing the score. At the moment, the ESS prototype provides min-max normalization and standardization. New *centrality* and *recommendation metrics* can be added to the ESS using the plugin system (see Section 4.3).

Configuration Templates

Configuration templates are defined for each resource selector in separate HCL[1] files. The Hashicorp configuration language (HCL) was chosen, because it is easier to read and edit for humans than JSON or YAML. The *web application* of the ESS only interprets JSON, however. Thus, the HCL files are translated into JSON using an extension of the standard HCL compiler[2].

A snippet of the configuration file for the class `RobotType` is shown in Listing 4.1. Figure 4.10a (α,β,γ) showcases the rendered visualization of an instance of this class. UI elements are assigned to widgets, properties, and values with the '`handler`' keyword. Moreover, these UI elements may be customized by passing a property object with the '`config`' keyword.

Lines 29-39 in Listing 4.1 state that every value for the property `cobot:reach` of a `RobotType` shall be rendered as ordinary text literal as long as it is not a quantity value from the QUDT ontology[3], which needs some additional parsing. Hence, a specific UI element, which uniquely identifier by the name `QudtQuantityValue`, is specified in line 36. This element is capable of accurately parsing and visualizing quantities represented using the QUDT ontology.

[1]Hashicorp configuration language, `https://github.com/hashicorp/hcl`
[2]Extension of HCL compiler, `https://github.com/khaller93/esw-hcl-compiler`
[3]Semantic specifications for units of measure, quantity kind, dimensions and data types, `https://qudt.org/`

Lines 40-53 in Listing 4.1 configure the recommendation section (see Figure 4.9 γ) for instances of `RobotType` to generally use the Linked Data Semantic Distance (LDSD) [Pas10]. However, lines 54-60 state for the `"Robotics Engineer"` perspective that the recommendation candidates should be restricted to instances of the class `RobotType`, while lines 61-68 restrict it to software-related classes for the `"Software Engineer"` perspective.

Listing 4.1: Configuration of `RobotType` [HESP22].

```
(10)  class = "cobot:RobotType"
(11)
(12)  perspective _ widget infobox {
(13)    handler = "GeneralInfoBox"
(14)    config {
(15)      sections = ["prop_table",
(16)                  "recommendations"]
(17)    }
(18)  }
(19)  perspective _ widget infobox section prop_table {
(20)    handler = "ProvenanceTableSection",
(21)    config {
(22)      neighbourhood {
(23)        include = ["cobot:degreesOfFreedom",
(24)          "cobot:handlingPayload", "cobot:reach",
(25)          "cobot:skills"],
(26)      }
(27)    }
(28)  }
(29)  perspective _ widget infobox {
(30)    property "cobot:reach" {
(31)      handler = "LinkedProperty"
(32)      value _ {
(33)         handler = "TextValue"
(34)       }
(35)       value "qudt:Quantity" {
(36)        handler = "QudtQuantityValue"
(37)       }
(38)    }
(39)  }
(40)  perspective _ widget infobox {
(41)    section recommendations {
(42)      handler = "SimilaritySection"
(43)      config {
(44)        number = 4
(45)        ranking = {
(46)          ldsd {
(47)            step = "esm.exploit.sim.ldsd"
(48)            weight = -1.0
(49)          }
(50)        }
(51)      }
(52)    }
(53)  }
(54)  perspective RoboticsEngineer widget infobox {
(55)    section recommendations {
(56)      config {
(57)        classes = ["cobot:RobotType"]
(58)      }
(59)    }
(60)  }
(61)  perspective SoftwareEngineer widget infobox {
(62)    section recommendations {
(63)      config {
(64)        classes = ["cobot:HandlingFunction",
(65)                   "star:ArchitecturalElement"]
(66)      }
(67)    }
(68)  }
```

4.3 Exploratory Search Platform

The exploratory search platform is comprised of two distinct components. One component (A) is the exploratory search system (ESS), providing the exploratory search interface (as discussed in Section 4.2) for a given manufacturing knowledge graph. The other component (B) involves a process for acquiring and synthesizing a knowledge graph, which converts selected engineering and manufacturing artifacts into a cohesive knowledge graph representation. However, this thesis is primarily focused on the first component of the platform, and it will not delve far into the second component.

(A) Exploratory Search System (ESS) has three main components, which are going to be the focus of this section.

The *middleware*[4] is a Spring Boot application written in Java and it has two main tasks. Firstly, it provides an interpreter for *exploration flows*. Secondly, it orchestrates the computation of analytical services such as centrality and similarity metrics among others. Section 4.3.1 describes the concepts and architecture of this *middleware* in more detail.

An *exploration flow* is a sequence of steps which all execute a single operation. The goal of an *exploration flow* is to abstract the more complex parts of the Semantic Web technologies, and only expose the basic concepts of RDF to the *web application*. Section 4.3.2 elaborates on the concept of an *exploration flow*.

The *web application*[5] is designed to be a thin single-page application that implements the adaptive search interface presented in Section 4.2.1. ReactJS[6] is used to implement the UI components, and the state is managed with Redux[7]. The required data for rendering the UI components is then fetched by assembling corresponding *exploration flows*, and sending them to the *middleware*. The *web application* is only responsible for the correct rendering and is not handling any RDF data and SPARQL queries itself.

(B) Knowledge Graph Acquisition & Synthesis heavily relies on the specific information environment of the manufacturer. Consequently, the ESS only anticipates a set of well-defined interfaces to access the built knowledge graph, but it does not directly aid in its construction. The ESS aims however to be agnostic to the choices made by the manufacturing company with an extendable plugin system. A discussion about how to transform the information landscape of a manufacturing company into a unified knowledge graph is out of scope for this thesis, but readers are encouraged to take a look at the book of Pan et al. [VTGSR+17].

Section 5.1 briefly outlines the construction of a sample knowledge graph for the pilot factory use case (UC2), from the design of an ontology to the alignment of semi-structured data to said ontology.

[4] Source code of middleware: `https://github.com/khaller93/es-middleware`
[5] Source code of web application: `https://github.com/khaller93/es-web-app`
[6] JavaScript library for building user interfaces - `https://reactjs.org`
[7] State container for JavaScript applications - `https://redux.js.org`

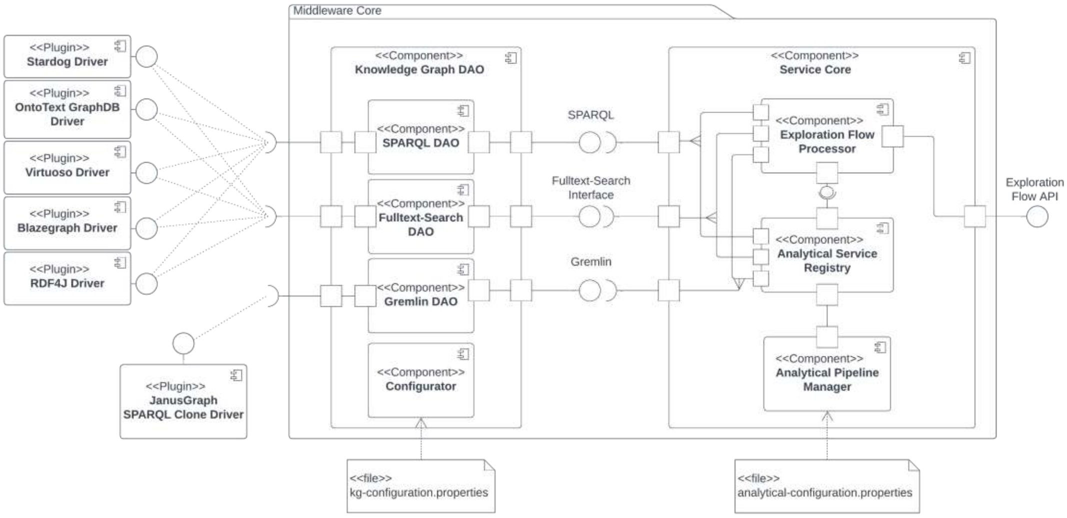


Figure 4.14: Component diagram of the ESS middleware.

4.3.1 Exploratory Search Middleware

The *middleware* is on one hand responsible for orchestrating the computation of knowledge graph analytics. It implements a range of information retrieval services, including centrality metrics for ranking entities and similarity metrics for recommendations. To that end, the *middleware* requires the target knowledge graph to be accessible for queries using SPARQL and Gremlin.

On the other hand, the *web application* shouldn't process any RDF data or SPARQL queries itself, which is why the *exploratory search flow* API (see subsequent Section 4.3.2) is provided by the *middleware* to the *web application*. This API must be expressive enough to implement all types of interface paradigms (see Section 4.1) in the *web application*.

The *middleware* is implemented in Java 8 and makes use of the Spring Boot framework[8]. It adopts a plugin-oriented architecture, where the *middleware core* implements the main business logic, and well-defined interfaces are established to facilitate seamless extension of the core with other storage solutions or analytical services. Plugins that conform to these well-defined interfaces can be effortlessly placed into the *middleware*'s library directory, enabling the extension of its functionality without the need to modify the source code or recompile the application.

The two major components of the *middleware core* are the *1) Knowledge Graph DAO*, and the *2) Service Core*, as depicted in the component diagram in Figure 4.14. While the *Knowledge Graph DAO* abstracts the storage solution, the *Service Core* uses the DAO to compute knowledge graph analytics and to process *exploration flows*.

[8]Spring Boot - `https://spring.io/projects/spring-boot`

1) Knowledge Graph DAO aims to abstract the storage solutions and tries to achieve this goal by introducing three interfaces. Moreover, Apache Commons RDF[9] is used in the definition of these interfaces in order to abstract over the different RDF frameworks in Java (e.g. RDF4J[10] or Apache Jena[11]).

1. **SPARQL 1.1**: The SPARQL interface defines two methods, one making it possible to issue a `SELECT`, `ASK`, `DESCRIBE` as well as `CONSTRUCT` query, and one method allowing it to issue update queries. Both of these methods must behave as expected from a standard SPARQL endpoint.
2. **Full-text search interface:** Among most of the popular triplestores, a full-text-search index for fast keyword searches can only be accessed with proprietary additions to SPARQL (differing from vendor to vendor), which is why a full-text search interface was additionally introduced. This interface has a single method for issuing a search given a keyword phrase. This single method is expected to return the results of the search in form of a list of resources (their IRI) and optionally a score assigned to them.
3. **Gremlin:** SPARQL, while useful for querying RDF-based knowledge graphs, has a notable limitation in terms of graph traversals. It can only confirm the existence of a path without providing specific details about the path itself. Therefore, a Gremlin query interface is introduced in order to provide advanced graph traversal capabilities to analytical services and *exploration flow* operators.

The *middleware* includes plugins, which implement these three interfaces for a number of popular triplestores (Blazegraph[12], GraphDB[13], Stardog[14] and Virtuoso[15]). And given that barely any triplestore supports the Gremlin query language, the *middleware* provides also a mechanism to clone the knowledge graph over the SPARQL interface into an embedded JanusGraph[16] instance.

Nonetheless, it is not mandatory for the storage solution to be a triplestore. An alternative approach, as demonstrated by Sabou et al. [SEI+18] for managing software-architectural knowledge, involved transforming MongoDB documents into RDF and then storing the resulting RDF data in a Sesame (now rebranded as RDF4J) triplestore. MongoDB, in this case, was utilized for text indexing by introducing a backward link in the form of an IRI. The presented *middleware* could also be applied to this storage solution, provided that a stakeholder develops a suitable plugin for its integration.

[9] Open source Java library for RDF, Apache Commons RDF - `https://commons.apache.org/proper/commons-rdf/`

[10] Open source Java library for RDF, Eclipse RDF4J - `https://rdf4j.org/`

[11] Open source Java library for RDF, Apache Jena - `https://jena.apache.org/`

[12] Open source triplestore - `https://blazegraph.com/`

[13] Commercial triplestore from ontotext - `https://www.ontotext.com/products/graphdb/`

[14] Commercial triplestore from Stardog - `https://www.stardog.com/`

[15] Commercial triplestore from OpenLink - `https://virtuoso.openlinksw.com/`

[16] Open source distributed graph database - `https://janusgraph.org/`

2) Service core carries out two primary functions: a) coordinating the computational analysis of the knowledge graph, and b) processing requests to the *exploration flow* API.

Analytical services focus on analyzing specific components of a knowledge graph's semantic structure. They can harness the capabilities of the three interfaces offered by the *Knowledge Graph DAO*, or depend on the outcomes of other analytical services. These services are organized and stored in a registry, each expected to include a method named `compute()` and to carry an `RegisterForAnalyticalProcessing` annotation. The registry searches for these services within the Java class path, ensuring it can identify and incorporate such services even from newly added plugins.

This `RegisterForAnalyticalProcessing` annotation details the unique identifier of the given analytical service, along with any other services upon which it depends. To illustrate this, the Resnik similarity metric requires that the class hierarchy and the information content values for classes be computed to generate any meaningful results. Consequently, these two services need to be indicated as dependencies within Resnik's annotation.

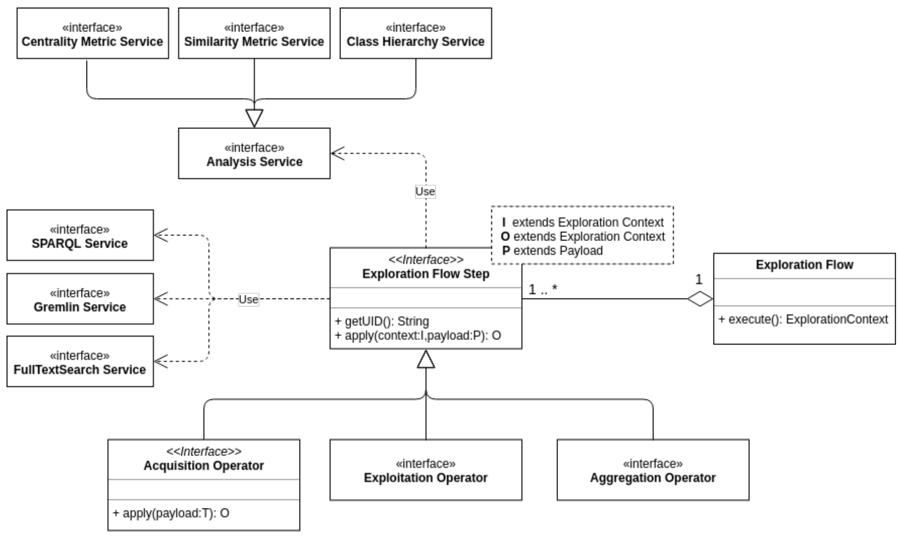
The analytical pipeline manager is equipped to autonomously formulate an appropriate pipeline incorporating all registered analytical services. The pipeline's execution can be triggered at system startups, during updates to the knowledge graph, or both. The output of the analysis can be accessed via the *exploration flow* API after a successful run.

Exploration flow processor serves as an interpreter for *exploration flows* transmitted to the *middleware's* API. It tries to find each operator defined in a given *exploration flow* within an operator registry, arranges the flow of operators along with their associated arguments, and subsequently executes them. Each operation can harness the capabilities of the three interfaces offered by the *Knowledge Graph DAO*, or can access the computed results of analytical services. The operator registry is assembled automatically at the start-up by scanning the Java class path of the application. Operators that can be used in an exploration flow request must be annotated with `RegisterForExplorationFlow`. Similarly to analytical services, plugins can add new operators to the *middleware*. The subsequent Section 4.3.2 discusses *exploration flows* in more detail.

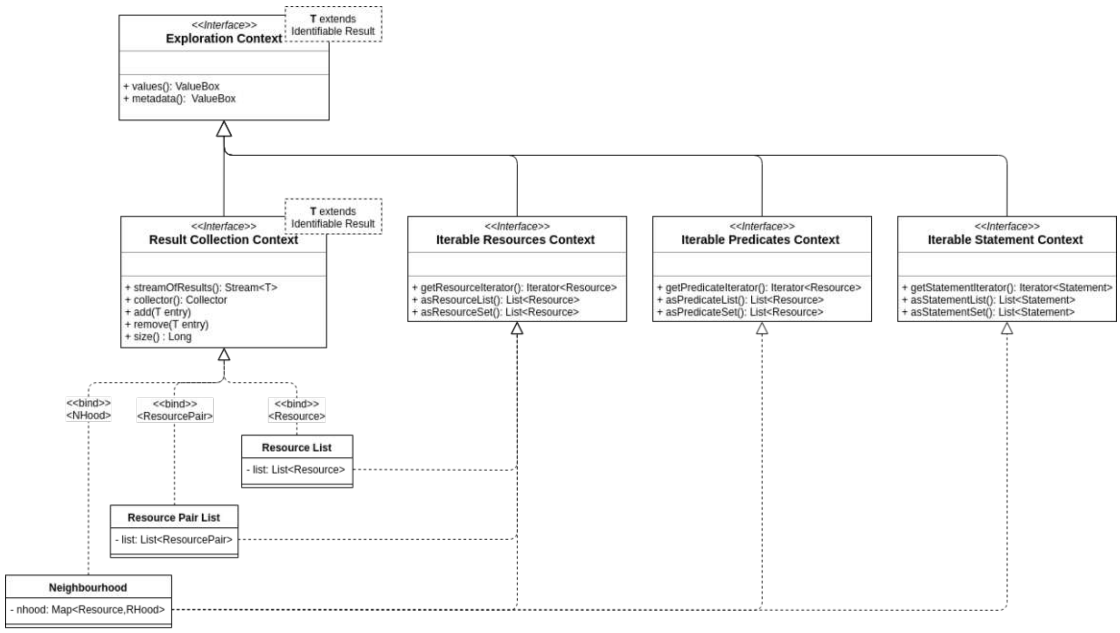
4.3.2 Exploration Flow

The introduction of the *exploration flow* interface is motivated by the limitations of SPARQL (see Section 2.1.1) and the missing flexibility of conventional HTTP+JSON APIs. An *exploration flow* consists of a sequence of steps (see Figure 4.15a), which we will thenceforward refer to as operators. Each operator has an `apply` method that expects an input of a certain type and returns a result of a specific type.

An interpreter assembles an execution plan for a given *exploration flow* and then, executes it. An executed operator can make use of any of the aforementioned three interfaces (namely SPARQL, Gremlin and full-text search) or access pre-computed values of metrics.



(a) Exploration Flow Step / Operator.



(b) `ExplorationContext` class hierarchy.

Figure 4.15: Exploration flow input/output types

A context is passed along from one step to the next step during the execution. The context can be manipulated by any operator or it can be newly created and passed on. An operator can be categorized into *a)* a source operator creating a new context, *b)* exploitation operator manipulating a context and *c)* aggregation operators. An `ExplorationContext` (see Figure 4.15b) manages an identifiable result, which can be a resource, a pair of resources, or a statement respectively. A context can take many forms. It can be among others a collection of resources (`ResourceList`), or a collection of predicates with their values for a given subject (i.e. the neighbourhood of a subject). A context does not only manage an identifiable result, it can also store meta information and values about an identifiable result.

Listing 4.2: Keyword search for "robot" (written in Python).

```
f = FTS(keyword='robot') \
  >> PageRank() \
  >> WeightedSum('sum', {Cent.pagerank: 2.0, FTS.score: 1.0}) \
  >> OrderBy('sum', strategy=Order.DESC) \
  >> Limit(n=10)
resp = FlowAPI("http://localhost:8080").execute(f)
```

Listing 4.3: Top 10 recommendations for a certain gripper (written in Python) [HESP22].

```
f = Single(resource='ex:gripper04') \
  >> PairWith(flow=All()) \
  >> (LDSD() | PeerPressure()) \
  >> WeightedSum('sum', {Sim.ldsd: -1, Sim.peerpressure: 1}) \
  >> OrderBy('sum', strategy=Order.DESC) \
  >> Limit(n=10)
resp = FlowAPI("http://localhost:8080").execute(f)
```

Example: Given the flow from Listing 4.2, firstly a full-text search with the phrase 'robot' is invoked. The result, which is a list of resources, is wrapped into a `ResourceList` context. Optionally, if a full-text score is given, the score will be stored in the value section of the context using the IRI of the resource as key. Then, this context is passed on to the next operator, which is computing the page rank for all the resources in the context. The result of the computation is stored in the value section of the context as the aforementioned score. In the next operation, a weighted sum is computed from the values given in the specified location (using JSON pointers[17]). The next operator ranks the results based on the values given at the specified pointer location and eventually, the `ResourceList` is limited to 10 entries and the rest is dismissed.

Similarly, Listing 4.3 describes a flow that computes the weighted sum of LDSD [Pas10] and a peer pressure metric for all possible pairs between a particular gripper and all other resources within the knowledge graph. This set of pairs is then sorted according to this weighted sum, with only the top 10 pairs ultimately being returned.

[17] https://tools.ietf.org/html/rfc6901

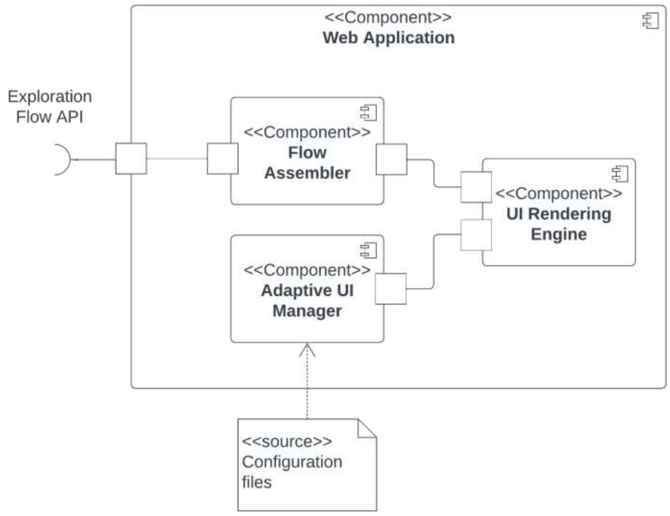


Figure 4.16: Component diagram of the ESS web application.

4.3.3 Exploratory Search Web Application

The *web application* is architectured as a lightweight single-page application that embodies the adaptive search interface outlined in Section 4.2. The primary components of this application are depicted in Figure 4.16.

Adaptive UI Manager reads the user-defined configurations of the adaptive UI in JSON format (i.e. HCL files are translated to JSON for the *web application*) and thenceforward maintains an index of what components to render in a given situation. The *UI Rendering Engine* component specifies a context and then asks the manager to return the corresponding UI element class.

Flow Assembler is asked to assemble *exploration flows* by the *UI Rendering Engine*, if certain data is missing for the rendering to complete, or the end user triggered new actions.

UI Rendering Engine implements all the UI elements, and contains the business logic for accurately assembling the user interface that is eventually shown to the end user. The UI elements are developed using ReactJS, while Redux is employed for state management.

CHAPTER 5

Evaluation

The ESS proposed in Chapter 4 was examined through a small-scale evaluation involving five participants on the COBOT use case (see Section 3.1.2 for details to the use case). Due to the limited participant number, this thesis primarily emphasizes the qualitative analysis of this prototype. In the experiments, participants assumed the role of information seekers exploring the domain of collaborative robotics. The objective of the evaluation was to determine the ESS's effectiveness in enabling participants to interactively explore, comprehend, and derive insights from a topic within the manufacturing domain.

For this purpose, a dedicated knowledge graph was developed from scratch specifically for the COBOT use case. As elaborated in Section 5.1, the process began with the design of an ontology tailored to the domain of smart manufacturing and collaborative robotics. Subsequently, data was collected from both interviews and public sources. This data was then aligned with the created ontology to construct the knowledge graph used in the evaluation, integrating also the software architectural knowledge from the STAR enterprise knowledge graph [SEI^{+}18].

Subsequently, Section 5.2 provides a detailed outline of the design of the user study and presents its results. The quantitative analysis of the evaluation follows the methodology of Al-Tawil et al. [ADT20], where participants answer knowledge questions before and after a task, with scores indicating "learning utility". Additionally, participants rate the usability of the ESS and the usefulness of specific interface paradigms. In the qualitative analysis Emilie Palagi's [Pal18] model is used, employing video analysis to identify usability issues impacting the exploratory search process.

The user study revealed that, generally, users successfully met their search objectives. However, it also highlighted potential areas for improvement. The final Chapter 6 discusses these topics in greater detail and presents considerations for potential future extensions.

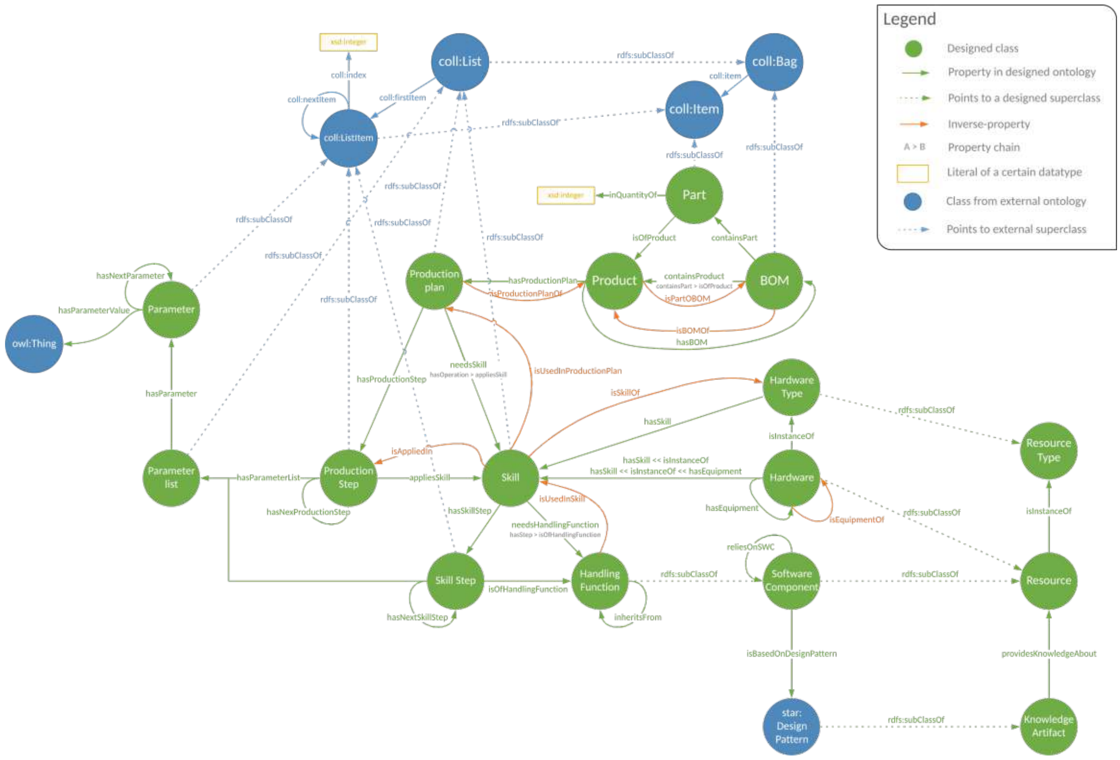


Figure 5.1: COBOT ontology.

5.1 COBOT Knowledge Graph Construction

The COBOT use case is concerned with the challenge of low utilization and limited awareness of manufacturing technology in the Aspern pilot factory, which is an Austrian learning and experimentation facility. The factory provides access to a range of industrial machines, including collaborative robots. However, the detailed capabilities of these robots are unknown to potential users due to factors such as limited public availability, contextual dependence, and continuous evolution.

To tackle this issue, this thesis proposes an exploratory search system that allows students, researchers, and companies to explore and learn about the available equipment. The key approach is the construction of a knowledge graph, which begins with designing a domain ontology and integrating heterogeneous data into a unified representation.

Section 5.1.1 elaborates on the design of the COBOT domain ontology, which aims to cover the field of collaborative robotics. This ontology was then used to align entities from several data sources into a unified representation. The process of this knowledge graph construction is outlined in Section 5.1.2 in more detail. The resulting knowledge graph was used to evaluate the exploratory search system ESS in the user study described in the subsequent Section 5.2.

5.1.1 Ontology

An OWL ontology[1] (see Figure 5.1) was designed for the COBOT use case based on accessible data, documentation and multiple interviews with a domain expert. It aims to cover the field of collaborative robots in manufacturing, from the bills of materials and process plans for manufacturing a product to the concrete implementation of a process with robots and their capabilities and equipment.

Besides the discipline of robotics and mechanical engineering, software engineers are needed for the implementation of handling functions for a robot (e.g. a simple pick & place routine). Thus, the COBOT ontology links to the STAR ontology designed by Sabou et al. [SEI+18], which is intended to cover the domain of architectures and design patterns in software engineering.

Description: The scope of this ontology ranges from manufactured products, manufacturing processes to robots and equipment used in those processes.

A manufactured `Product` may have a `BOM` (Bill of Materials) associated with it, which is a description of atomic parts and their required quantity for assembling the corresponding `Product`. A manufactured `Product` might additionally have a `Production Plan` associated with it. A `Production Plan` represents an ordered list of `Production Steps` for manufacturing the associated `Product`, whereas a `Production Step` represents a certain `Skill` being applied given a number of input `Parameters`.

A `Skill` is an activity of higher granularity, e.g. transporting a small object from location A to B, or picking and placing an object. Such a `Skill` could be part of the `Production Plan` of an electronic `Product`, where a transistor has to be picked from a storage box and placed into a certain drilling hole of a printed circuit board. A `Skill` is composed of atomic `Handling Functions` such as pick, place and approach. Software might be required to implement `Handling Functions` and `Skills` for a `Hardware` entity.

This thesis mainly focuses on `Robots` and hardware equipment for `Robots` such as `End Effectors` and `Sensors`. The COBOT ontology distinguish between the physically existing `Hardware` and abstract `Hardware Types`. The concrete Franka Emika[2] `Robot` at the pilot factory is of the `Robot Type` "Franka Emika Panda". A `Robot` itself has usually a limited number of `Handling Functions`, but those can be extended by equipping the concrete `Robot` with compatible `End Effectors` and `Sensors`. Consequently, a `Robot` might be able to perform more `Skills`. The Franka Emika `Robot` at the pilot factory can perform `Handling Functions` such as move and approach, which however are not sufficient to perform the aforementioned pick & place `Skill`. The `Robot` must be equipped with an `End Effector` that provides pick and place among other `Handling Functions`. Grippers (e.g. Robotiq 2F-85) and vacuum systems (e.g. Robotiq Airpick) are examples for such `End Effectors`.

[1] Ontology file and COBOT KG dataset at: `https://github.com/khaller93/cobot-ontology`
[2] Franka Emika collaborative robot, `https://www.franka.de/`

Figure 5.2: Class distribution of the COBOT KG (classes of the COBOT ontology are colored in red, classes of the STAR ontology [SEI+18] are colored in green and blue).

5.1.2 Knowledge Graph Acquisition & Synthesis

At the time of the study, the Aspern pilot factory didn't have an accessible information system dedicated to account maintained manufacturing technology, which is why the knowledge graph had to be manually curated. To that end, the following sources were utilized:

(A) Stakeholder interviews were conducted to compile a list of available equipment and associated skills using pen and paper. This data was subsequently digitized into Comma-separated values (CSV) files. For the equipment, datasheets were obtained, and properties were extracted from the tables within these PDFs using Tabula[3]. This extraction also yielded CSV files. The RDF extension[4] of the OpenRefine[5] tool was then employed to align this structured data with the COBOT ontology detailed in the preceding section.

[3] Open source tool for table extraction, `https://tabula.technology/`
[4] Open source RDF extension of OpenRefine, `https://github.com/AtesComp/rdf-transform`
[5] Open source cleansing and transformation tool, `https://openrefine.org/`

(B) Website scraping was performed on `coboticsworld.com`, a website that offers a comprehensive list of collaborative robots and their associated equipment. This data extraction was executed using BeautifulSoup[6]. Subsequently, the obtained structured data was mapped to the COBOT ontology utilizing rdflib[7].

(C) Integration of software architectural knowledge from the STAR enterprise knowledge graph [SEI$^+$18]. It encompasses design patterns and software engineering methods that are ready to be reused by robotics engineers in their specific use cases.

The COBOT KG[8], as a result, encompasses 2,682 distinct entities interconnected by 48,563 edges. This yields a density of $6.75 * 10^{-3}$. Notably, there are 116 unique relationship types observed between these entities. Figure 5.2 illustrates the class distribution of the knowledge graph.

5.2 User Study

Before the performance of the ESS can accurately be assessed, the properties of exploratory searches need to be clearly defined. As of the time of this writing, there is however no stabilized or consensus definition of exploratory search in literature. This thesis adheres to the definition of Marchionini [Mar06], which states that *"knowledge acquisition, comprehension of concepts or skills, interpretation of ideas, and comparisons or aggregations of data and concepts"* are pertinent activities of exploratory search. Hence, a system qualifies as an exploratory search system when it assists users in learning and making sense of information. Therefore, the main objective of the user study is to evaluate how well the ESS prototype aids participants in gaining insights and knowledge in the selected manufacturing domain of collaborative robotics.

To that end, a specialized knowledge graph was crafted from the ground up specifically for the COBOT use case as outlined in the preceding section. Additionally, a reproducible environment[9] for the user study was established using Docker.

Section 5.2.1 introduces the design of the user study. This section details the methodologies employed for both qualitative and quantitative evaluations of the ESS, and also explains the approach used in designing the exploratory search tasks. Following this, Section 5.2.2 elaborates on the the participant recruitment process for the user study, providing details on the selection criteria and the rationale behind these requirements. Lastly, Section 5.2.3 and Section 5.2.4 comprehensively present the quantitative and qualitative findings of the user study respectively.

[6] Open source Python library for pulling data out of HTML, `https://pypi.org/project/beautifulsoup4/`

[7] Open source Python library for handling RDF, `https://pypi.org/project/rdflib/`

[8] COBOT KG dataset, `https://github.com/khaller93/ess-cobot-playground`

[9] User study environment, `https://github.com/khaller93/ess-cobot-playground`

TASK A	**TASK B**
Imagine that you are a member of a team which is working on a manufacturing project with collaborative robots and you have access to the equipment located at the pilot factory in Aspern. In this project, you have to move a 4 kg heavy and cubic object with a length of 20cm from a conveyor belt to a manufacturing cell that is 60cm away.	Imagine that you are a member of a team which is working on a collaborative robot picking up a small transistor from a storage box and placing it on a certain position on a circuit board. The robot works in proximity of human workers and it might have to interrupt it‘s task, due to them coming too close. However, we don't want the robot to remain in this state, and proceed with the task as soon as possible.
Goal *You have been asked to design a hardware setup for this project with equipment that is available at the pilot factory. Your team would be glad about a brief presentation of your findings.*	**Goal:** *You have been asked to explore design patterns that could be lend from software engineering for this kind of error handling and eventually present promising design patterns to your team.*

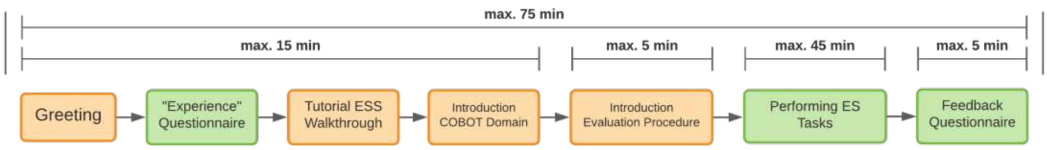
Table 5.1: Exploratory search tasks for evaluating the system.

5.2.1 Design

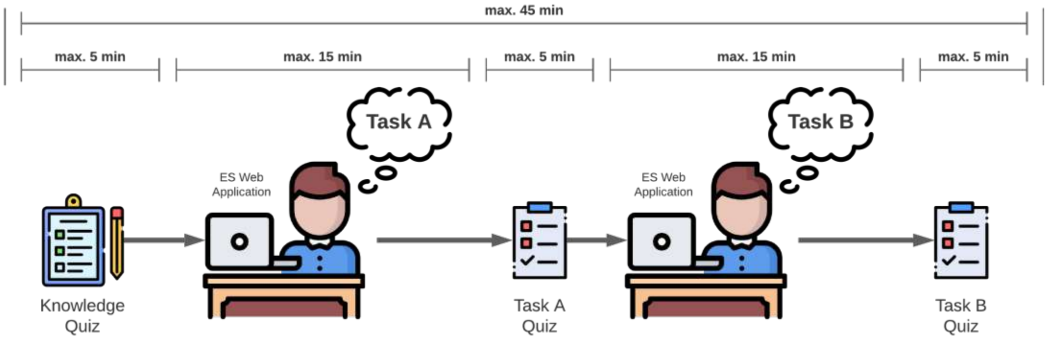
A system qualifies as an exploratory search system if it aids users in learning and making sense of information. Thus, the primary aim of our evaluation is to assess the effectiveness of the ESS prototype in helping participants acquire knowledge within the manufacturing domain. For the quantitative analysis (see Section 5.2.3 for results), the evaluation adheres to the methodology outlined by Al-Tawil et al. [ADT20]. Participants must answer knowledge questions before and after performing a task. Answers are scored using a point system. The "learning utility" is assessed by the difference in scores from answers given before and after completing the task with the ESS. Furthermore, participants are asked to rate the usability of the ESS and the "usefulness" of selected interface paradigms. For qualitative analysis (see Section 5.2.4 for results), the evaluation follows Emilie Palagi's model [Pal18], which employs video analysis to identify usability flaws in the search system that adversely impact the exploratory search process.

To that end, exploratory search tasks need to be specified. The design of these tasks is centered around work tasks aimed at learning, investigation, and decision making. It necessitates the selection of broad, open-ended topics in the domain of collaborative robotics that are poorly organized, embody some level of uncertainty, and involve the evolution of the search strategy. These topics should also be multi-dimensional and relatively challenging to search for. Moreover, the task scenario should encourage the information seeker to dedicate a substantial amount of time to search for and engage with various artifacts in the manufacturing knowledge graph [WF12].

Table 5.1 shows the two designed exploratory search tasks, which were crafted to elicit this behavior. Task A requires participants to research collaborative robots, understanding their capabilities and equipment, to determine a suitable assembly. Task B requires participants on the other side to identify relevant software engineering design patterns to



(a) Overall session flow.



(b) Performing exploratory search tasks.

Figure 5.3: Evaluation session flow.

address issues arising when a collaborative robot is interrupted during its task.

The whole evaluation procedure is illustrated in Figure 5.3, including the expected duration for each step. Steps highlighted in orange are carried out by the evaluation team, while those in green are executed by the participant. Due to COVID-19 concerns, all evaluations were conducted via Skype sessions. Participants received tasks and questionnaires through Google Documents and Forms respectively. Although participants were allowed to make notes in the Google Document while working on the tasks, they were prohibited from using the ESS post-task completion and after receiving the questionnaire. During the task execution, participants shared their screens, which were recorded by the evaluation staff for further analysis.

The remaining section delves into each step of the evaluation procedure in more detail.

Greeting. The evaluation starts with a cordial introduction, followed by a concise overview of the evaluation's objective. Participants are provided with an illustration of the session's structure, as shown in Figure 5.3, to familiarize them with the upcoming proceedings.

"Experience" Questionnaire. Subsequently, the participant is requested to complete the "experience" questionnaire (refer to Section B.1). This survey is designed to gather insights into the background of participants, as well as their familiarity with smart manufacturing, software architecture design, and Semantic Web technologies.

Tutorial. Next, participants will be introduced to the primary concepts of the ESS through a walkthrough. During this session, the system will be showcased on a knowledge graph focused on Pokémon[10], which diverges from the engineering context. If participants are unfamiliar with the Pokémon world, a concise introduction will be provided.

Domain Introduction. The participant receives an introduction to smart manufacturing and collaborative robots through a short presentation.

Evaluation Introduction. The participant is briefed on the main session's procedure, emphasizing that it's an evaluation of the ESS rather than a test of their abilities. They are informed that they will initially receive a skill questionnaire, which is not meant to be fully mastered. Tasks will then be provided sequentially. Participants are instructed to think aloud, verbalizing their thoughts as they navigate the user interface. Additionally, they are notified that their screen will be recorded during the task execution.

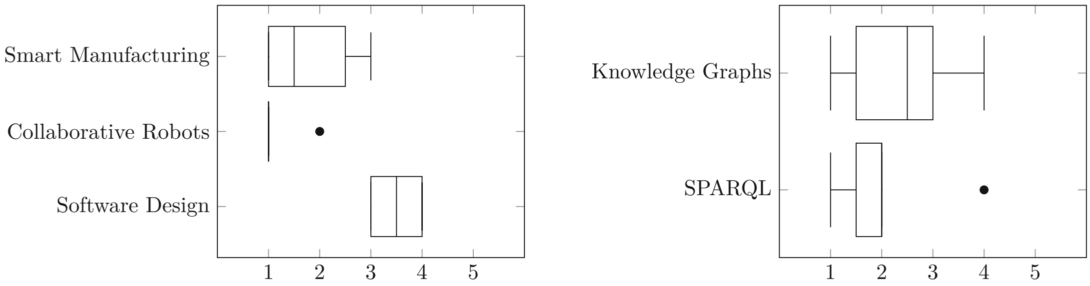
"You will be given tasks one at a time. Please read the task aloud once upon receipt. Afterwards you can try to fulfill the task and have up to 15 minutes for that. If you believe the task was completed successfully, please say so out loud."

Skill Questionnaire. Before tackling the assigned tasks, participants are prompted to answer a series of questions related to those tasks. This preliminary questionnaire (see Section A.2) aims to assess the a-priori domain knowledge of participants. To measure the learning utility of the system, the same questions will be posed again post-task (see Section A.3 for these questionnaires).

Performing Tasks. Participants are given tasks individually and are instructed to read the task out loud. This approach ensures participants don't preview subsequent tasks, which could bias their approach based on anticipated future challenges. Such foresight might skew evaluation results. Furthermore, tasks are presented in a randomized sequence for each participant to avoid any sequence-based bias. Each task is expected to take no more than 15 minutes, with a total of 2 tasks assigned per participant. After each task, participants are prompted to answer a few questions to assess the knowledge they've acquired (see Section A.3 for these questionnaires).

Feedback Questionnaire. Upon completion of the tasks, participants are provided with a link to a feedback questionnaire (see Section A.4). This questionnaire incorporates the System Usability Scale (SUS) and asks for ratings on selected

[10]Pokémon KG, `https://pokemonkg.org/`



(a) Domain-specific experience/familiarity. (b) Familiarity with Semantic Web technologies.

Figure 5.4: Results of the "experience" questionnaire.

features of the ESS. Additionally, participants are invited to offer open-ended feedback or suggestions for system enhancement and improvement.

5.2.2 Participants

The aim of this study is to evaluate the effectiveness of the ESS in facilitating participant‘s acquisition of knowledge on collaborative robotics, which is why the focus was on recruiting people with a technical inclination. The ecological state of the information seeker is a crucial aspect of the exploratory search process. Inviting participants with no affinity towards collaborative robots or with no interest in technical problem solving could potentially temper with the results of the evaluation.

Thus, the selection process incorporated three eligibility criteria to ensure quality participation. They had to (1) be able to understand documents written in English, (2) be somewhat familiar with technical problem solving, and (3) have a basic interest in learning about smart manufacturing / collaborative robotics. Overall, five participants have been recruited from the university campus, researchers and students alike.

Background

All five participants had a background in computer science. Two of the participants (40%) were male, and three (60%) were female. In terms of age distribution, three participants (60%) were aged between 25-34, one participant (20%) was in the 35-44 age bracket and another one (20%) in the 45-54 bracket. Four participants (80%) reported that their occupation was in research, and one (20%) stated to be a student. All participants reported a medium or high ability to understand documents written in English.

At the beginning of the experiments, participants were given an "experience" questionnaire (refer to Section A.1) to assess their familiarity with the use case domain and their experience with Semantic Web technologies. Detailed results from this questionnaire are provided in Section B.1.

Participant	Software/Pre	Software/Post	Cobot/Pre	Cobot/Post	Utility
P_1	0	6	0	5	+11
P_2	1	2	4	6	+3
P_3	2	4	4	4	+2
P_4	0	2	0	2	+4
P_5	2	2	1	1	+0
$ChatGPT_4$	16		8		-

Table 5.2: Learning utility scores of participants.

Domain experience. Participants were queried about their familiarity with smart manufacturing and collaborative robotics, as well as their experience in software design. A common trend emerged with all participants reporting limited familiarity with collaborative robots and to a slightly lesser degree, smart manufacturing. On the other hand, they considered themselves highly experienced in software design. Figure 5.4a presents a box plot diagram illustrating the distribution of responses to these three questions.

Semantic Web experience. Furthermore, participants were questioned regarding their experience with Semantic Web technologies. The responses indicated some familiarity with knowledge graphs among participants, but experience with the SPARQL query language was less commonly reported. Figure 5.4b provides a box plot diagram illustrating the distribution of responses to these two questions.

5.2.3 Quantitative Results

The quantitative evaluation followed the method defined by Al-Tawil et al. [ADT20]. Participants were required to respond to knowledge questions both prior to and after executing a task. The "learning utility" is then measured by the variance in scores from responses provided before and after the completion of a task using the ESS.

At the conclusion of the experiment, participants were furthermore asked for feedback after using the ESS prototype for approximately 30 minutes. They provided feedback using a form that included the standard SUS questionnaire and additional questions about the "usefulness" of selected interface paradigms.

Learning Utility

To measure the learning utility of the ESS, the well-established taxonomy of educational objectives by Bloom [BK56] is adopted. This taxonomy categorizes learning objectives into six cognitive processes: remember, understand, apply, analyze, evaluate, and create, with the first two (i.e. remember and understand) being most relevant to browsing and exploration activities. Specifically, "remember" involves retrieving knowledge from memory, including recognition and recall, while "understand" focuses on constructing meaning, particularly through categorization and comparison [ADT20].

The user study is following the methodology outlined by Al-Tawil et al. [ADT20]. To evaluate the understanding and retention of specific domain concepts, the participants are prompted in a skill questionnaire to identify and name concepts that belong into a specific category.

In the skill questionnaires, participants are posed open-ended questions like, *"What properties of a robot are important to consider when moving an object from A to B?"* both before and after completing a task. This question comes with a set of acceptable answers, such as "reach" or "maximum payload". Participants earn one point for each correct answer listed and lose a point for any incorrect ones, resulting in a final score. However, it is not possible to get a negative score for a question. The variation in scores before and after the task is used to measure the *learning utility* derived from the exploration conducted with the ESS during the task.

The detailed scores for each participant are presented in Table 5.2. These scores are divided into two categories, namely one for questions related to software engineering and another one for questions about collaborative robotics. All participants, except for one, experienced a positive increase in their knowledge about the domain following their use of the ESS.

Additionally, the table includes scores for responses provided by ChatGPT[11]. Given the large corpus of training data [BMR+20] for ChatGPT in comparison to the relatively small COBOT knowledge graph, the scores for the learning utility of the ESS are still discouraging. The responses of the conversational agent were sufficiently informative to quickly gain knowledge about the domain, and to quickly get reference points for further research. ChatGPT struggled specifically with providing accurate information regarding the narrow skill set of a gripper. The effectiveness of these agents for more specialized manufacturing knowledge is yet to be determined. Section 6.2 discusses this further.

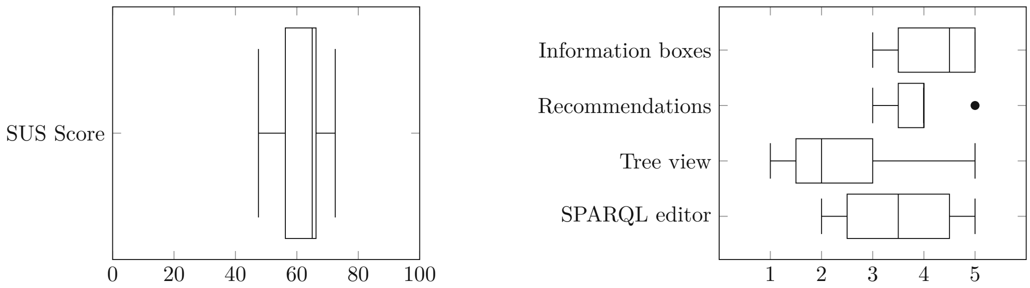
Nevertheless, all participants would have been able to answer the questionnaire with statistically significant improvement by solely using ChatGPT. However, this does not provide insights into how effectively the conversational agent aids participants in the learning activities of retention and understanding, which is the crucial part in evaluating how well a system supports the exploratory search process.

System Usability Scale

At the end of the experiment, participants provided feedback using the standard SUS questionnaire. The answers to its ten usability questions were converted into numerical values and summed based on the methodology proposed by Brooke et al. [Bro86]. The detailed responses from our five participants can be found in Section B.2.1.

Figure 5.5a presents the computed score for all participants ($n = 5$) using a box plot diagram. The ESS prototype has an average score of 63.5 with a standard deviation of

[11] Version 4, released on 3rd August of 2023, https://help.openai.com/en/articles/6825453-chatgpt-release-notes



(a) SUS score of the ESS.

(b) Rating of paradigm "usefulness".

Figure 5.5: Results of quantitative feedback questionnaires.

8.46. The 95% confidence interval for this score ranges from 52.95 to 74.05. According to the school-grade interpretation of these scores by Bangor et al. [BKM08], the usability of the ESS would be graded as a D.

Interface Paradigm Rating

Prior to the completion of the experiment, participants were asked to give feedback and the latter portion of the feedback questionnaire delved into the assessment of the "usefulness" of specific interface paradigms. "Usefulness" specifically refers here to how beneficial a paradigm was in accomplishing the tasks at hand, a point clarified during the questioning. To ensure clarity and avoid misinterpretation, participants were shown screenshots illustrating each paradigm in action when rating it. The detailed questionnaire can be found in Section A.4.

Figure 5.5b depicts the feature ratings of the ESS, evaluated on a Likert scale. A rating of 1 indicates strong disagreement regarding the feature's usefulness, while 5 indicates strong agreement. Participants found common interface paradigms (as surveyed in Section 4.1.1) such as information boxes and recommendations particularly useful. In contrast, the tree view was perceived as less beneficial for task completion. Interestingly, despite participants indicating limited familiarity with the query language SPARQL in the "experience" questionnaire, the SPARQL editor received high ratings.

5.2.4 Qualitative Results

The exploratory search sessions conducted by participants were analyzed using a video analysis grid [Pal18] derived from Palagi's model of exploratory search (see Section 2.4). This method enables the assessment of whether participants could successfully engage in all aspects of exploratory search without encountering difficulties. If obstacles were present, the analysis helps to pinpoint their origin, such as missing features or usability issues. At the conclusion of the study, participants were also invited to provide open-ended feedback on the ESS, offering suggestions for enhancements or expressing concerns about any features they found lacking.

A	***The user specifies their goals of search, expresses their lack of knowledge, their information needs. The user specifies how they decided to start the search.***
A_1	Does the user express their goals of search?
A_2	Does the user express their information need or lack of knowledge?
A_3	Does the user specify how they decided to start the search?
A_4	Does the user express their angle of attack of the problem?
B	***The user uses the search bar or the filter tools in order to formulate or reformulate their query.***
B_1	Does the user formulate the query with filter tools or search bar or any other means?
C	***The user says that an information is interesting. Never mind if this information is pertinent for the current problem and search.***
C_1	Does the user say or imply that some information is interesting?
D	***The user adds to bookmarks one or several results with some memorization features, from the list of results or one element description***
D_1	Does the user add to bookmarks or write down on a paper relevant information?
E	***The user wants to know more about an element (query result) or the link between a query and a result. They may use different descriptive information or features, e.g. a descriptive text, pictures, map, links, etc.***
E_1	Does the user examine one element to check an information about it?
E_2	Does the user examine one element to discover it?
E_3	Does the user examine one element to know more about the link between query and result?
F	***The user explains why the want to change the goal of search and maybe explain why.***
F_1	Does the user say that they don't find the current goal or the query relevant anymore?
F_2	Does the user say that they find something more interesting for their exploration?
F_3	Does the user simply say that they want to change the goal or the angle of attack?
G	***The user can use the browsing history, the breadcrumb, or the back/next buttons.***
G_1	Does the user uses the browsing history, the bread crumb or the back or next buttons?
G_2	Does the user step backward or forward?
G_3	Does the user go back to a previous screen?
H	***The user browses or scans the results given by the system.***
H_1	Does the user scan the result list?
H_2	Does the user want to know what kind of results are given by the system?
I	***The user analyses a result description. They evaluate its relevance based upon the goal of search.***
I_1	Does the user evaluate the relevance of the result for their goal of search?
J	***The user stops their search session.***
J_1	Does the user stop their search session?

Table 5.3: Video analysis grid by Palagi [Pal18]

Video analysis

This section analyzes the recorded videos of exploratory search sessions of the participants, in conjunction with their logged comments. For this purpose, this section systematically addresses the questions listed in Palagi's [Pal18] grid, as presented in Table 5.3. This approach enables the identification of features of the exploratory search process each participant engaged with, and to pinpoint areas where issues arose.

A) While participants were instructed to verbalize their thoughts, few explicitly articulated their goals and search strategies (A_1, A_3). None expressed their angle of attack to solve the search problem at hand (A_4). Some participants however expressed their lack of knowledge of the domain, and the need for learning about some unclear aspect (A_2).

B) While participants have the flexibility to choose different starting points for their exploration, the ESS initially presents a Google-like search bar. This was the chosen starting point for all participants, who began their search session by entering keyword phrases into this bar. The system allows for initiating new keyword searches through clickable links in the information box, a feature that participants frequently utilized. However, the tree view option was not popular, with many participants expressing difficulty in effectively using it. One specific task required the aggregation of robots based on certain properties, a function not directly supported by the interface due to the absence of facets. Consequently, some participants resorted to using the SPARQL query editor. It's noteworthy, however, that not all participants were familiar with this query language and got stuck with this task.

C) Throughout the session, every participant identified certain information as interesting and frequently proceeded to note it down.

D) Since the ESS lacks an integrated bookmarking feature for memorization, participants were permitted to take notes on a Google Sheet document. This option was utilized by all participants.

E) All participants actively engaged with the information box (E_1), frequently reviewing details about specific search results. They often utilized the links within this information box (E_2) to delve deeper into each result. However, none expressed interest in understanding the connection between their queries and the resulting outputs (E_3). Additionally, the ESS lacks a dedicated feature to effectively address this specific search requirement.

F) Similar to point A), few participants explicitly articulated their goals and search strategies. Consequently, questions F_1 to F_3 must be answered in the negative.

G) The ESS includes a history feature, accessible by clicking the designated icon in the top-right corner. However, none of the participants utilized this feature to revisit previous queries or results. Conversely, one participant, in particular, made extensive use of browser tabs, frequently switching back and forth between them. All participants revisited their queries and results, primarily utilizing the standard features of a conventional web browser (G_1 - G_3).

H) The ESS presents query results in a vertical list format, similar to that of major web search engines such as Google. All participants actively scanned through this result list, and they also clicked on specific entries to further explore information about those items (H_1, H_2). However, none of the participants navigated past the first page of results, which displayed the top 10 entries.

I) Each participant consistently examined individual results by reviewing the information provided in their respective information boxes. Additionally, they occasionally commented on the relevance of specific items in relation to their search goals.

J) Except for two instances, all search sessions were terminated due to the time limit rather than by the participants themselves.

Feedback

Participants were invited to provide open-ended feedback on the ESS.

One participant didn't use the SPARQL editor, so they were unable to provide detailed feedback on this aspect. However, they found the information boxes helpful, although the changing formats within these boxes were noted to add complexity.

Navigating from the selected class in the search results to the tree view posed a challenge for another participant. They suggested enhancing the system with more options to explore examples or recommendations beyond the current display limits.

Regarding the SPARQL editor, due to their lack of familiarity, this participant opted not to use it. They recommended that including basic guidelines or a tutorial for SPARQL novices could be beneficial. Such additions would enable users unfamiliar with SPARQL to try out and effectively utilize this feature.

CHAPTER 6

Conclusion & Future Work

Creative human exploration of manufacturing knowledge graphs through an exploratory search system can support engineers in their analysis and decision-making. However, most publicized exploratory search systems are designed with general knowledge graphs in mind, for which common knowledge is sufficient.

Based on interviews with domain experts, this thesis highlights three distinct features that boost the efficacy of exploratory searches for manufacturing environments. These encompass the system's ability to tailor itself to numerous engineering perspectives, the provision of clear provenance information that significantly aids stakeholders during their investigative process, and the necessity to traverse deep hierarchical structures of containment relationships prevalent in many manufacturing knowledge graphs.

By studying the common interface paradigms for general exploratory search systems, an exploratory search system (ESS) has been engineered with additional manufacturing-specific features. This includes a rendering engine for an adaptive user interface, allowing for a multi-faceted configuration of visualizations and underlying search algorithms. To tackle the issue of provenance visibility, the system was designed to share named graph information for each statement with relevant UI components, hence making possible the creation of UI components that manifest this knowledge. The challenge of navigating deep hierarchies was met with a simple tree view, reminiscent of the ontology editor Protegé, coupled with information boxes.

While the preliminary evaluation of the ESS system demonstrated its utility for users in carrying out search tasks in the domain of collaborative robotics, certain areas for improvement can be observed. The tree view design was perceived as not useful, and the memorization feature was often sidelined in favor of managing multiple browser tabs. Despite these areas for refinement, users were overall successful in achieving their search objectives. Section 6.1 discusses the three research questions in more detail. Section 6.2 elaborates on extensions for a contemporary exploratory search system.

6.1 Discussion

This thesis examined the three research questions that were previously outlined in the introduction (see Section 1.1). This section delves into each research question, offering a detailed discussion and analysis of their respective outcomes.

RQ I. What are special requirements for exploratory search in the manufacturing domain? Exploratory search systems are typically designed for general knowledge graphs, relying on common knowledge, and often overlook the specific requirements of professionals. To understand these specific needs better, informal interviews were conducted with stakeholders from partnering manufacturing organizations. The goal was to gather insights into potential requirements that are often missed in the design of exploratory search systems for the general domain. The following three features were extracted from the requirements collected in the interviews.

1. **Multiperspectival exploration** addresses the diverse information needs of stakeholders in a manufacturing environment. It allows them to select their preferred perspective, enabling them to concentrate on aspects of engineering artifacts that are relevant to their specific discipline.

2. **Provenance visibility** is identified as a key feature in manufacturing environments that integrate interdisciplinary knowledge from various sources. Its significance is particularly noted when stakeholders need to examine properties of an engineering artifact and clear up uncertainties for well-informed decision-making.

3. **Hierarchical browsing** is an essential search task, particularly in the context of complex digital twins representing production equipment and machines. To aid stakeholders in effectively navigating these complex structures, it is important to provide a mechanism that allows them to concentrate on critical parts without getting overwhelmed by the entire hierarchy. By equipping stakeholders with efficient hierarchical browsing tools, they can mentally focus on the most important sections of the hierarchy.

The informal interviews were limited in scope, involving a small number of participants from specific areas of manufacturing. These included a simulation expert from a production plant manufacturer and a researcher associated with this partner, as well as two smart manufacturing researchers active in a pilot factory. To gather a more representative collection of requirements, it would be beneficial to expand these interviews to include a wider range of sectors and companies within the manufacturing industry. This expansion should also involve a larger pool of participants to ensure a more comprehensive understanding of the diverse needs and challenges faced across different areas of manufacturing.

Additionally, the approach of conducting informal interviews ought to be refined by adopting a controlled interview format, adhering to an established methodology.

RQ II. Which of the common exploratory mechanisms are most suitable to address the needs of the two presented use cases? What interface paradigms are of particular interest for engineers in manufacturing? This thesis centered on adapting exploratory search specifically for manufacturing knowledge graphs. Through insights gathered from exploring research question RQI in collaboration with partnering organizations, it identified and emphasized interface paradigms to aid engineers in manufacturing settings.

Beyond the desired effects for general exploratory search systems, three specific desired effects were derived from the insights of RQI. First, (M1) an exploratory search system for manufacturing environments ought to enable easy navigation through complex hierarchical knowledge structures. Second, (M2) the system ought to make it easy to cognitively focus on relevant information, allowing it to effectively filter through the overwhelming amount of information about engineering artifacts arising from the integration of multidisciplinary data. Lastly, (M3) the system ought to provide clarity and ease in observing the sources of information.

Knowledge lenses as an interface paradigm serve to spotlight essential properties of an entity, offering information seekers a clear and concise overview for easy understanding (M2). Within the manufacturing context, knowledge lenses enable the creation of discipline-specific perspectives of engineering artifacts. They effectively filter out irrelevant details, significantly reducing information overload for engineers. The ESS presented in this thesis adapts the approach of the Linked Data Reactor [KLvH16] to implement knowledge lenses with an adaptive user interface.

Hierarchical result overviews allow the information seeker to efficiently traverse deep hierarchical knowledge structures (M1). The presented ESS implements a conventional textual tree view to realize this interface paradigm. A limitation of these textual tree views is however their restricted capacity for displaying a limited number of entities simultaneously [TAvHS06].

In manufacturing environments, the visibility of information provenance is crucial. The system must clearly and intuitively display the origins of information (M3), aiding engineers in their decision-making processes by making it easier to understand and trust the source of the data they are using. This diversity of possibilities to state provenance information in a knowledge graph is a significant challenge in developing a universal strategy for utilizing this knowledge in a user interface. The ESS presented in this thesis assumes that the *singled triple named graph* approach in favor other mechanisms is used. Hence, the ESS passes the named graph information to UI components. This can then be utilized by the UI components to visualize provenance knowledge to information seekers.

The interface paradigms outlined here were predominantly sourced from the scientific literature within the Semantic Web community. However, to gain a more comprehensive perspective on potential interface paradigms that could benefit engineers, it would broaden the scope to incorporate more insights from the Human-Computer Interaction (HCI) community.

RQ III. What effect do the selected interface paradigms have on the exploration utility of the search system, when applied on manufacturing knowledge? Drawing on the insights from RQII, a prototype of the ESS was developed, incorporating selected interface paradigms, followed by the conduct of a user study. The primary objective of the study was to determine the effectiveness of this system in facilitating active exploration, learning, and comprehension of a specific topic of the manufacturing domain. The user study involved gathering quantitative data, with a primary emphasis on qualitative video analysis and participant feedback. The user study yielded the following key takeaways:

1) Exploratory Search: The qualitative video analysis indicated that participants could effectively utilize most features of the exploratory search process with minimal obstacles. Nevertheless, there were some issues, which will be elaborated upon in the remaining section. Overall, participants were successful in gaining knowledge and learning about the domain using the ESS.

2) Tree View: Participants seldom used the textual tree views during their exploration, and they rated the usefulness of this interface element as very low. However, given that the COBOT knowledge graph lacks extensive hierarchical knowledge structures, it is understandable that this feature was infrequently used and perceived as less useful compared to other interface paradigms.

3) Faceted Exploration: The ESS did not include faceted search capabilities, and the video analysis showed that participants missed this feature. They attempted to use the SPARQL editor as a substitute for facets, but some participants were unfamiliar with this query language. Moreover, it should not be necessary for engineers from different disciplines to learn a query language for this purpose.

4) Memorization: Participants did not use the built-in history feature designed to facilitate revisiting queries and results. Instead, they relied on the standard navigation features of their web browser to move back and forth. Although this reliance on standard browser features is not inherently problematic, their limitations suggest that incorporating more advanced memorization features could enhance the learning utility of the ESS.

The user study was conducted on a relatively small scale, involving only five participants. While valuable insights were obtained from the participants verbal feedback and qualitative video analysis, the quantitative data gathered from this limited sample size is not representative. User studies in the exploratory search research community typically involve a larger group of participants, usually more than 32 individuals (e.g. [NPG$^+$17], [ADT20] or [MLTK21]). This larger sample size is crucial for obtaining more statistically significant quantitative results. The number of participants ought to be increased in a future study in order to align with these standards.

Additionally, the "think aloud" protocol of the user study was not adequately explained to and internalized by the participants, which rendered certain aspects of the video analysis ineffective and more challenging.

6.2 Future Work

While the previous section opens a number of avenues for the improvement of the user study and the current ESS, this section is going to focus on proposing three extensions of the ESS. These extensions are essential for a contemporary exploratory search system on (manufacturing) knowledge graphs.

Faceted exploration. In the user study (see Chapter 5), participants were tasked with locating robots capable of managing payloads over 4 kg and positioned within a specific manufacturing hall. However, the user interface of the ESS prototype proved challenging for gathering this information, due to insufficient interface elements for effective robot aggregation. Consequently, participants proficient in the SPARQL query language utilized the built-in SPARQL editor for manual information extraction. Nevertheless, those without familiarity with this language faced difficulties due to the user interface's limitations.

As stated in the discussion of Section 4.1, faceted exploration is a valuable interface paradigm for exploratory search in general, but in particular for manufacturing. However, it was not implemented, due to time constraints. The introduction of facets would simplify aggregation and filtering tasks.

Intuitive querying. In a vision paper for 2043, Ilkou et al. [IGD23] foresee user-friendly and intuitive querying tools to ease adoption of knowledge graphs, but what might an intuitive method for querying (manufacturing) knowledge graphs look like? Some publications have suggested the creation of a block-based visual programming language for SPARQL using Blockly[1] [BC15][CB17]. However, these methods introduce users to RDF and SPARQL constructs, which ought to be circumvented. An exploratory search system should not necessitate proficiency in programming and Semantic Web technologies, a prerequisite that many individuals in the manufacturing domain may not satisfy.

Beyond the incorporation of faceted exploration, a visual programming interface could be an effective tool to bypass the constraints of a limited user interface. Blockly provides a framework to create your own domain-specific visual programming language based on blocks, whereas the vocabulary (i.e. syntax) and the runtime (i.e. semantics) have to be specified by the integrator of this framework [PFM17]. CoBlox is for instance a programming platform that aims to make robot programming accessible to a broader demographic with a block-based interface. In a user study with 67 adult novices, the designers of CoBlox reported that the majority of participants could correctly implement a simple pick & place routine using the visual language CoBlox, whereas logical errors such as not lifting the robot arm before moving it horizontally were prevalent [WAS$^+$18].

An area for future research and extension of the ESS could be the design of such a block-based interface for exploring knowledge graphs.

[1] Blockly, Open Source library from Google, `https://developers.google.com/blockly`

Question answering. The knowledge questionnaire was not only given to the participants of the user study (see Chapter 5), but also to ChatGPT, which is a conversational agent build upon a particular Large Language Model (LLM) [RNS+18]. While the answers to the questionnaire were not perfect (as documented in Appendix B.3), they would have been decent enough for the participants to quickly gain knowledge about the domain, and to quickly get reference points for further research.

Consequently, one might naturally question the necessity of experimenting with different interface paradigms for exploratory search systems when ChatGPT could seemingly suffice as an conversational assistant. While modern large language models are an impressive advancement, they don't provide a human-analogous understanding of natural language, but rather offer a statistical framework for generating sequences of word forms [BK20]. As of this writing, the existing models lack a reliable grounding to the factual world. They often produce information that appears correct at first glance, but proves to be inaccurate upon deeper examination, a phenomenon termed "hallucination". Martino et al. [MIT23] propose an injection technique from knowledge graphs to reduce "hallucinations" in generated text by LLMs. However, in professional settings such as manufacturing, the expectation for a search system is to deliver accurate and dependable results.

Yet, another potential research question could explore how conversational agents, based on LLMs, can be incorporated into an exploratory search system and tap into a private manufacturing knowledge graph, ensuring their limitations are clear to the information seeker.

APPENDIX A

Evaluation Questionnaires

This chapter documents all the questionnaires provided to the participants throughout the experiment. Section A.1 presents the initial questionnaire, filled out at the beginning of the experiment to gather more insight into the background of participants. The knowledge questionnaire, distributed prior to the participants starting their tasks, is outlined in Section A.2. The questionnaires given post completion of tasks A and B are respectively illustrated in Section A.3. Finally, participants were requested to complete a feedback form, exhibited in Section A.4.

A.1 Experience Questionnaire

At the beginning of the evaluation, participants were provided with an "experience" questionnaire. The aim was to capture their level of familiarity with the domain of collaborative robotics as well as smart manufacturing, and assess their proficiency in Semantic Web technologies.

1. What is your occupation (developer, researcher, etc.)?

   ____

2. Language Skills

   a) How do you rate your ability to understand English documents?

bad	☐	☐	☐	☐	☐	excellent
	1	2	3	4	5	

3. Domain Experience

a) How familiar are you with smart manufacturing?

not familiar ☐ ☐ ☐ ☐ ☐ familiar
1 2 3 4 5

b) How familiar are you with collaborative robots?

not familiar ☐ ☐ ☐ ☐ ☐ excellent
1 2 3 4 5

c) How do you rate your experience in designing software?

no experience ☐ ☐ ☐ ☐ ☐ excellent
1 2 3 4 5

d) How do you rate your experience with design patterns in software engineering?

no experience ☐ ☐ ☐ ☐ ☐ excellent
1 2 3 4 5

4. Semantic Web

a) How do you rate your experience with knowledge graphs?

no experience ☐ ☐ ☐ ☐ ☐ excellent
1 2 3 4 5

b) How do you rate your experience with the query language SPARQL?

no experience ☐ ☐ ☐ ☐ ☐ excellent
1 2 3 4 5

A.2 Knowledge Questionnaire

This questionnaire is given to participants before they are tasked with the search activities. Its purpose is to evaluate the baseline knowledge of participants prior to task execution.

1. What design patterns come into your mind, when you hear the term "error handling"?

   ____

   ☐ I don't know

2. What architectural qualities are in your mind related to "error handling"?

   ____

   ☐ I don't know

3. What handling functions (e.g. drill, screw) come into your mind, when you think about a gripper?

   ____

   ☐ I don't know

4. Which properties of a robot are in your mind important to consider, when moving a heavy object from A to B?

   ____

   ☐ I don't know

A.3 Task Questionnaires

This section presents the questionnaires that were distributed to participants following their engagement in the task for a duration of up to 15 minutes.

A.3.1 Task A

Task A focused on identifying suitable design patterns for a robot to recover from an error state. The following questions were posed to the participant subsequent to the execution of this task.

1. What architectural qualities are in your mind related to "error handling"?

   ____

   ☐ I don't know

2. What architectural qualities are in your mind related to "error handling"?

   ____

   ☐ I don't know

3. What is the most promising design pattern that you would present to your team?

   ____

   ☐ I don't know

A.3.2 Task B

Task B aimed at sourcing appropriate hardware at the Pilotfabrik in Aspern for transferring an object from point A to point B. The subsequent questions were presented to the participant after the completion of this task.

1. What handling functions (e.g. drill, screw) come into your mind, when you think about a gripper?

   ____

   ☐ I don't know

2. Which properties of a robot are in your mind important to consider, when moving a heavy object from A to B?

   ____

   ☐ I don't know

3. What hardware would you present to your team to solve this task?

   ____

   ☐ I don't know

A.4 Feedback Questionnaire

Upon the conclusion of the experiment, participants were invited to share their feedback. The feedback form consisted of a standard System Usability Scale (SUS) questionnaire along with inquiries regarding the "usefulness" of certain features of the ESS prototype (see Chapter 4 for more details). To avoid confusion and ensure clarity, illustrations of the features in question were provided to the participants.

1. System Usability Scale (SUS)

   a) I think that I would like to use this system frequently?

strongly disagree	☐	☐	☐	☐	☐	strongly agree
	1	2	3	4	5	

   b) I found the system unnecessarily complex?

strongly disagree	☐	☐	☐	☐	☐	strongly agree
	1	2	3	4	5	

c) I thought the system was easy to use?

strongly disagree ☐ ☐ ☐ ☐ ☐ strongly agree
1 2 3 4 5

d) I think that I would need the support of a technical person to be able to use this system?

strongly disagree ☐ ☐ ☐ ☐ ☐ strongly agree
1 2 3 4 5

e) I found the various functions in this system were well integrated.

strongly disagree ☐ ☐ ☐ ☐ ☐ strongly agree
1 2 3 4 5

f) I thought there was too much inconsistency in this system.

strongly disagree ☐ ☐ ☐ ☐ ☐ strongly agree
1 2 3 4 5

g) I would imagine that most people would learn to use this system very quickly.

strongly disagree ☐ ☐ ☐ ☐ ☐ strongly agree
1 2 3 4 5

h) I found the system very cumbersome to use.

strongly disagree ☐ ☐ ☐ ☐ ☐ strongly agree
1 2 3 4 5

i) I felt very confident using the system.

strongly disagree ☐ ☐ ☐ ☐ ☐ strongly agree
1 2 3 4 5

j) I needed to learn a lot of things before I could get going with this system.

strongly disagree ☐ ☐ ☐ ☐ ☐ strongly agree
1 2 3 4 5

k) If you want to give some free floating feedback about the usability of the system, please enter it below (optional).

____

2. Feature Rating

a) Info Boxes were useful for completing the given tasks.

strongly disagree ☐ ☐ ☐ ☐ ☐ strongly agree
1 2 3 4 5

b) Recommendations were useful for completing the given tasks.

strongly disagree	☐	☐	☐	☐	☐	strongly agree
	1	2	3	4	5	

c) Tree View was useful for completing the given tasks.

strongly disagree	☐	☐	☐	☐	☐	strongly agree
	1	2	3	4	5	

d) SPARQL editor was useful for completing the given tasks.

strongly disagree	☐	☐	☐	☐	☐	strongly agree
	1	2	3	4	5	

e) If you want to give some free floating feedback about any of these features, please enter it below (optional).

APPENDIX B

Evaluation Results

This chapter encapsulates findings of the evaluations, which may not have been explicitly laid out in Chapter 5. Detailed insights gathered from the "experience" questionnaire can be found in Section B.1. We delve into the outcomes of the feedback questionnaire in Section B.2, while responses from ChatGPT to the knowledge questionnaire are thoroughly documented in Section B.3.

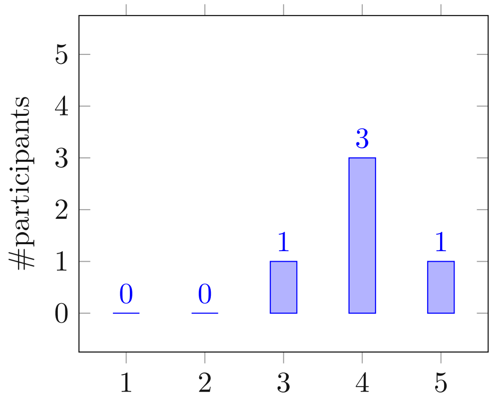
B.1 Experience Questionnaire

At the beginning of the evaluation, participants were given an "experience" questionnaire (see Section A.1). This was designed to collate data concerning their familiarity with collaborative robotics and smart manufacturing, as well as their proficiency level in relation to Semantic Web technologies.

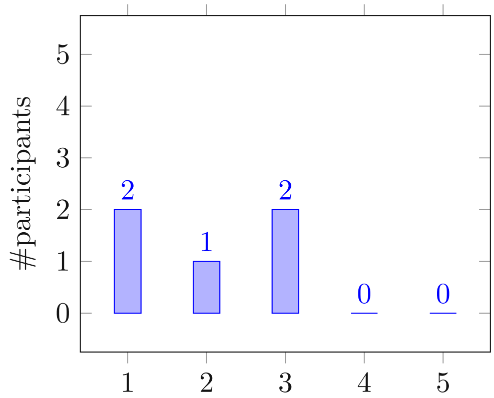
Demographics: Questions about age and gender were not part of the "experience" questionnaire, but these characteristics were estimated. Two of the participants (40%) were male, and three (60%) were female. In terms of age distribution, three participants (60%) were aged between 25-34, one participant (20%) was in the 35-44 age bracket and another one (20%) in the 45-54 bracket.

Occupation and English proficiency: Four participants (80%) reported that their occupation was in research, and one (20%) stated to be a student. Fig. B.1a presents the selected proficiency levels of English, 1 refers to "bad" and 5 to "excellent".

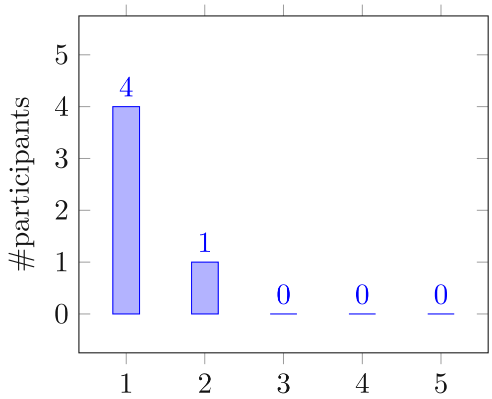
Familiarity and experience scale: Participants were queried about their familiarity with smart manufacturing and collaborative robotics, as well as their experience in software design. Fig. B.1b and Fig. B.1c present the selected levels of familiarity with smart manufacturing and collaborative robots, 1 refers to "not familiar" and 5 to "strongly familiar". The selected levels of experience in software design are illustrated in Fig. B.1d, 1 refers to "no experience" and 5 to "strong experience".



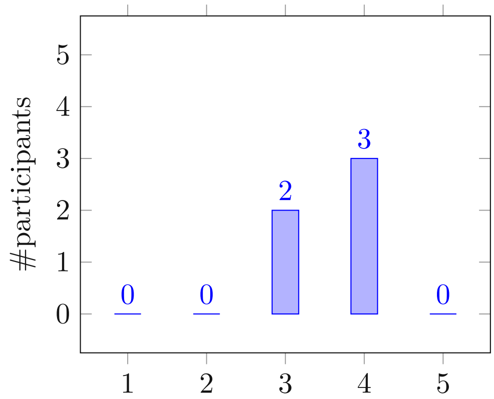
(a) How do you rate your ability to understand English documents?



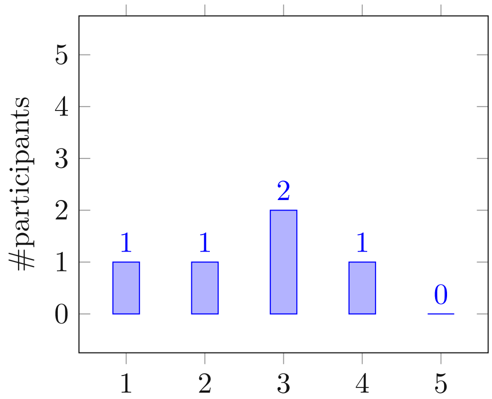
(b) How familiar are you with smart manufacturing?



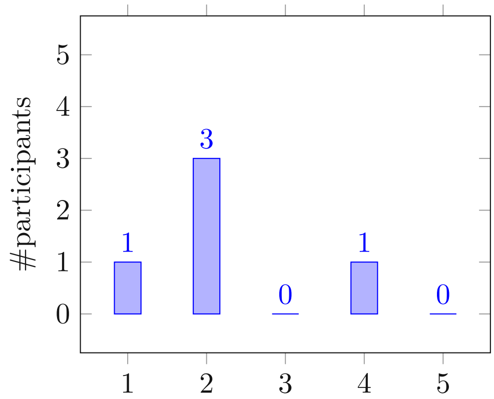
(c) How familiar are you with collaborative robots?



(d) How do you rate your experience in designing software?



(e) How do you rate your experience with knowledge graphs?



(f) How do you rate your experience with the query language SPARQL?

Figure B.1: Information from the "experience" questionnaire (see Section A.1).

B.2 Feedback Questionnaire

Upon conclusion of the experiment, participants were requested to provide their feedback. The feedback form was split into a standard System Usability Scale (SUS) questionnaire and inquiries regarding the "usefulness" of certain features of the ESS prototype (see Chapter 4 for more details).

B.2.1 System Usability Scale

Figure B.2 illustrates the findings from the System Usability Scale (SUS) section of the feedback questionnaire. The scoring system ranges from 1, denoting "strongly disagree", to 5, symbolizing "strongly agree". The free-form feedback was as follows:

- URI copy, data model schema/diagram for overview
- For me it is a little bit confusing about the type in the robotic engineering part, some of them it is the same, maybe we can put a description on it, or maybe we can put it both the class, so it easy to understand. but so far it is easy to use.
- the similarity to search machines make the first introduction to the system easy, however there are still some design flaws (diverging textual description, description needs to be reloaded extra) so I cannot give full rating for usability or confidence . But I think for context-aware searching it could be helpful after fixing these bugs.

B.2.2 Feature Rating

Figure B.3 showcases the outcomes of the feature rating. The assessment scale spans from 1, representing "strongly disagree", to 5, indicating "strongly agree". The open-ended feedback provided was as follows:

- explore more on examples or recommendation (not limited to what can be displayed)
- Switching between selected class in search result to tree view
- in the sparql part, since im not to familiar i will prefer to not used it, otherwise, maybe you can put some guidelines for someone who is not familiar with this sparql, so i can try it out to use this feature.
- didn't use the Sparql editor so cannot elaborate. Info boxes were helpful however again changing formats made it more difficult

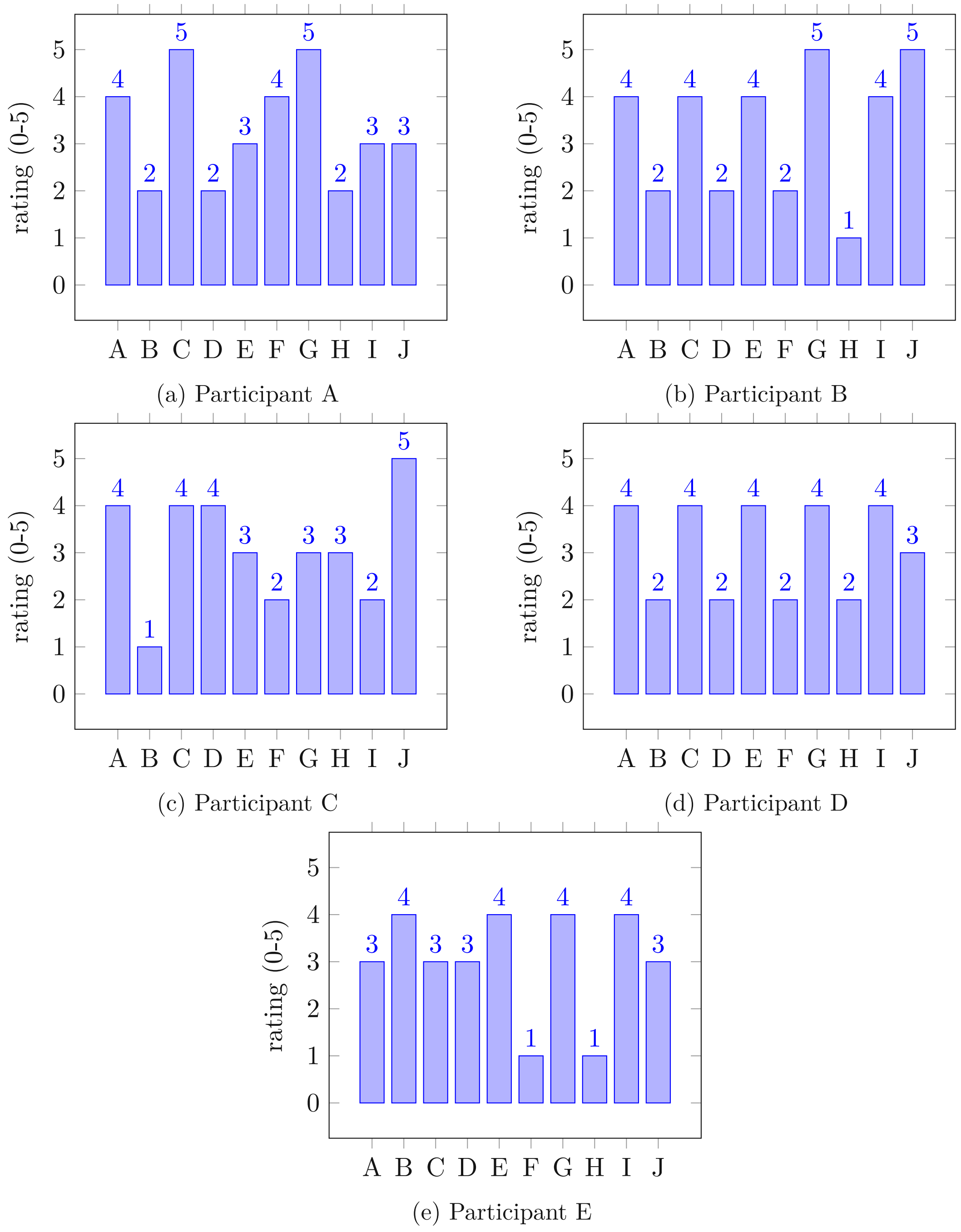
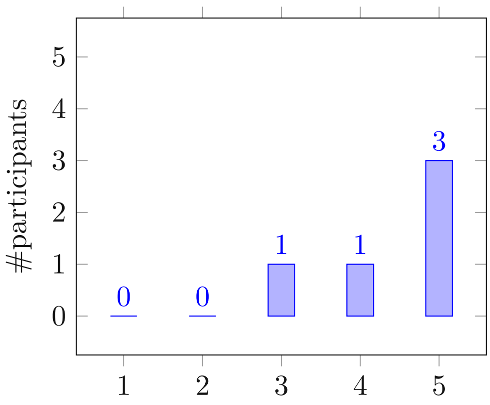
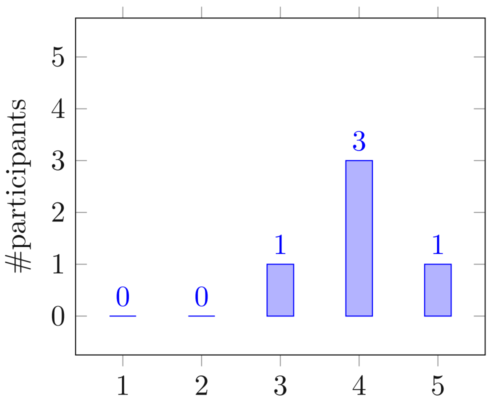


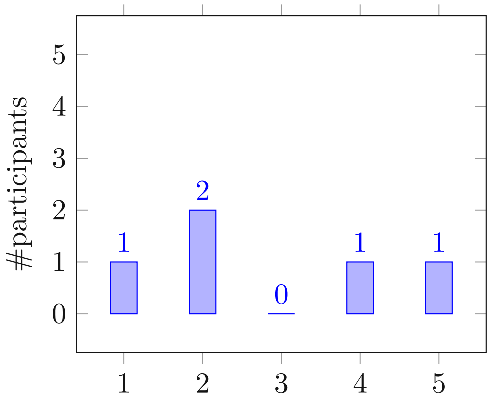
Figure B.2: Evaluation results of System Usability Scale.



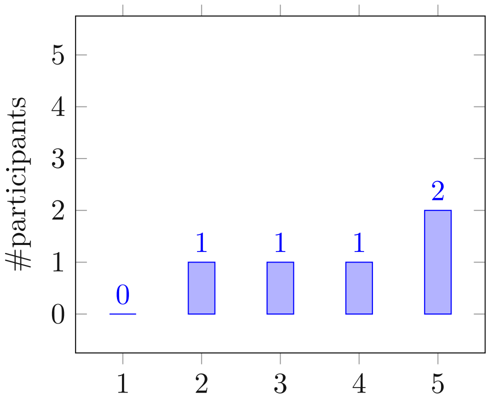
(a) Info Boxes were useful for completing the given tasks.



(b) Recommendations were useful for completing the given tasks.



(c) Tree View was useful for completing the given tasks.



(d) SPARQL editor was useful for completing the given tasks.

Figure B.3: Evaluation results of feature rating.

B.3 ChatGPT Answering Knowledge Questionnaire

This section outlines the answers of ChatGPT[1] to all questions in the knowledge questionnaire (see Section A.2). Each question was posed separately within its own session, with a brief context provided prior to each inquiry.

B.3.1 What design patterns come into your mind, when you hear the term "error handling"?

Interviewer: In context of software design, what design patterns come into your mind, when you hear the term "error handling"?

[1]Version 4, released on 3rd August of 2023, `https://help.openai.com/en/articles/6825453-chatgpt-release-notes`

ChatGPT: Error handling is a critical aspect of any application. In the context of software design, certain patterns come to mind to help us systematically manage, handle, and mitigate errors in a way that the application remains stable and users get a good experience.

Exception handling: The most basic error handling pattern is the use of try/-catch blocks for exception handling. When you foresee that a particular block of code can throw an error, you put it in a try block and then catch the error, so the program doesn't crash, and you can provide meaningful error messages to the user.

Retry Pattern: If a certain operation fails due to transient issues (like network issues), then you can use the Retry Pattern to attempt the operation again after a certain amount of time or a certain number of times.

Circuit Breaker Pattern: If a system fails repeatedly, the circuit breaker pattern can stop the system from constantly trying to execute a potentially failing operation. After a "cool-down" period, the system tries again, and if it's still failing, it continues the cool-down.

Fallback Pattern: When a call to a remote service fails, an application can use a pre-defined fallback operation. This pattern is especially common in microservices-based architectures.

Null Object Pattern: Rather than using a null reference when an object is not available, using an object which implements the expected interface but whose method body is empty can help to avoid null reference exceptions.

Fail-fast: The idea here is that when an error occurs, the system should fail immediately and loudly. This is often used in systems that require high reliability to prevent error propagation throughout the system.

Error Accumulation: In scenarios where there are multiple error possibilities, rather than failing on the first error, accumulate all the errors and then throw them together, so the user gets a full picture of what went wrong.

Logging and Monitoring: While not a design pattern per se, logging and monitoring play a crucial role in error handling. They provide insights into where and why errors are happening, helping in identifying patterns and in debugging and resolving issues.

Remember that these are patterns, not strict rules. You should decide which pattern to use based on the specific context and requirements of your application.

B.3.2 What architectural qualities are in your mind related to "error handling"?

Interviewer: In context of software design, what architectural qualities are in your mind related to "error handling"?

ChatGPT: In software design, error handling is an important aspect of system resilience and usability. This directly relates to several architectural qualities:

Robustness: This is the ability of a system to cope with errors during execution. In a robust system, exceptions or errors don't cause system crashes but are instead handled gracefully, preserving system functionality as much as possible.

Resilience: Similar to robustness but at a larger scale, resilience is the ability of the system to recover from failures and continue to function. This often involves mechanisms to restart failed components or fallback strategies to ensure essential services remain available.

Reliability: If a system can consistently perform its required functions under stated conditions, it's considered reliable. Good error handling helps to increase system reliability by addressing potential problems before they escalate to a system-wide failure.

Fault Tolerance: This is the property that enables a system to continue operating properly in the event of the failure of (or one or more faults within) some of its components. If its operating quality decreases at all, the decrease is proportional to the severity of the failure, as compared to a naively designed system, in which even a small failure can cause total breakdown.

Maintainability: Exception handling and error reporting have a significant effect on maintainability. Errors should be logged in a way that facilitates diagnosis and correction of problems. Clear and descriptive error messages, warnings, and logs can make it easier to understand the root causes of problems and fix them.

Usability: Good error handling includes providing meaningful error messages to users, allowing them to understand what went wrong and possibly how to correct the issue, improving the overall user experience.

Testability: Proper error handling makes the software more testable. When errors are thrown and caught properly, they can be tested to ensure the system handles all exceptions as expected.

Security: Proper error handling prevents the exposure of sensitive system information when an error occurs. For example, system error messages shouldn't reveal system internals that could be exploited by attackers.

Proper error handling involves anticipating potential problems, handling them appropriately when they occur, logging them for review, and communicating them clearly to users and developers. It is an essential aspect of high-quality software.

B.3.3 What handling functions (e.g. drill, screw) come into your mind, when you think about a gripper?

Interviewer: In context of collaborative robotics, what handling functions (e.g. drill, screw) come into your mind, when you think about a gripper?

ChatGPT: In the field of collaborative robotics, a gripper is a critical part of the system that is used for picking up, manipulating, and setting down objects. This tool is designed to perform a range of handling functions, depending on the specific needs of the task. Below are some handling functions that could be associated with a gripper:

Picking and Placing: This is one of the most basic functions of a robotic gripper. It's used in applications such as sorting, packaging, and assembly where items need to be picked up from one location and placed in another.

Screwing: Some grippers are designed to be able to screw or unscrew items, such as in the assembly of electronics or other products.

Drilling: There are grippers designed with the ability to hold drill bits and perform drilling operations, typically in manufacturing contexts.

Cutting: In some cases, grippers may also be used to hold and manipulate cutting tools for processes such as trimming, deburring, or machining.

Soldering/Welding: Some grippers can hold soldering irons or welding torches and perform these functions in an automated fashion.

Painting or Spraying: Grippers can also hold paint brushes or spray nozzles to perform tasks such as painting or coating objects.

Buffing or Polishing: Certain types of grippers can hold and manipulate buffing or polishing tools to finish off products in a manufacturing line.

Dispensing: Grippers can also be used to hold and control dispensing tools, which could be used to dispense glue, sealant, or other substances onto a product.

Inspection: Grippers can hold cameras or other inspection equipment to inspect parts or assemblies for quality control.

Testing or Measurement: Grippers can be used to hold testing or measuring devices to evaluate parts or products.

Keep in mind that the exact functions a gripper can perform depend greatly on its design and the specific robotic system it's part of. Some grippers are general-purpose, while others are designed for very specific tasks. The materials the gripper is made from, the power of the robotic arm it's attached to, and the programming of the system can all affect what tasks a gripper can perform.

B.3.4 Which properties of a robot are in your mind important to consider, when moving a heavy object from A to B?

Interviewer: In context of collaborative robotics, which properties of a robot are in your mind important to consider, when moving a heavy object from A to B?

ChatGPT: Collaborative robotics, or "cobots," are designed to work alongside human workers in a shared workspace, and they have important properties that are particularly significant when moving a heavy object from point A to point B. Here are some of them:

Safety: This is the most critical feature to consider when dealing with heavy objects. Collaborative robots must be equipped with advanced safety features such as force sensing or vision systems to avoid colliding with humans or other objects in the workspace. They must also have the ability to automatically stop if they encounter an unexpected obstacle.

Payload Capacity: This refers to the maximum weight that a robot can lift or handle. If the object being moved is heavy, it is crucial that the robot has a sufficient payload capacity.

Precision and Accuracy: Especially in industrial applications, the robot must be able to place the heavy object at point B with high precision. This may involve complex manipulations, which require high accuracy in robot control.

Range of Motion: The robot must be able to reach both point A and point B, which could potentially be located in complex or tight spaces. This requires a suitable range of motion and possibly also a high level of maneuverability.

Speed: Depending on the context, the speed at which the robot can safely move the object may be important. However, speed must always be balanced against safety considerations.

Endurance: The robot's ability to perform the task repetitively without degradation of performance over time is crucial in industrial settings.

Ease of Programming: Depending on the complexity of the task, the robot may need to be easily programmable to adapt to different objects or tasks, especially in environments where the tasks change frequently.

Human-Robot Interaction: Since these are cobots, they are meant to work alongside humans. Therefore, how well they interact with their human counterparts is of utmost importance. This includes easy communication of the robot's intentions, the ability to learn from human demonstration, and the capacity for seamless handovers of objects or tasks.

Compliance and Force Control: In order to safely interact with the environment and human workers, cobots need to have compliance and force control. This allows them to adapt to the variable forces during lifting or placing the heavy object.

Robustness: The robot must be able to perform under different conditions and should be resistant to wear and tear, especially when handling heavy objects.

Remember that the importance of these properties can vary depending on the specifics of the task and the environment in which the robot will be used. Therefore, these factors should be taken into account when designing or choosing a robot for a particular task.

List of Figures

List of Tables

Bibliography

[ADT20] Marwan Al-Tawil, Vania Dimitrova, and Dhavalkumar Thakker. Using knowledge anchors to facilitate user exploration of data graphs. *Semantic Web*, 11(2):205–234, 2020.

[AGK+16] Marcelo Arenas, Bernardo Cuenca Grau, Evgeny Kharlamov, Sarunas Marciuska, and Dmitriy Zheleznyakov. Faceted search over RDF-based knowledge graphs. *Journal of Web Semantics*, 37-38:55–74, 2016.

[AH11] Dean Allemang and James Hendler. *Semantic Web for the Working Ontologist: Effective Modeling in RDFS and OWL*. Morgan Kaufmann, 2nd edition, 2011.

[AJK05] Anne Aula, Natalie Jhaveri, and Mika Käki. Information search and re-access strategies of experienced web users. In Allan Ellis and Tatsuya Hagino, editors, *Proceedings of the 14th international conference on World Wide Web, WWW 2005, Chiba, Japan, May 10-14, 2005*, pages 583–592. ACM, 2005.

[AMMH07] Daniel J. Abadi, Adam Marcus, Samuel R. Madden, and Kate Hollenbach. Scalable semantic web data management using vertical partitioning. In *33rd International Conference on Very Large Data Bases (VLDB) - Conference Proceedings*, 2007.

[Bat89] Marcia J. Bates. The design of browsing and berrypicking techniques for the online search interface. *Online Review*, 13:407–424, 1989.

[Bat07] Marcia J. Bates. What is browsing - really? A model drawing from behavioural science research. *Inf. Res.*, 12(4), 2007.

[BC15] Paolo Bottoni and Miguel Ceriani. SPARQL playground: A block programming tool to experiment with SPARQL. In Valentina Ivanova, Patrick Lambrix, Steffen Lohmann, and Catia Pesquita, editors, *Proceedings of the International Workshop on Visualizations and User Interfaces for Ontologies and Linked Data co-located with 14th International Semantic Web Conference (ISWC 2015), Bethlehem, Pennsylvania, USA, October 11,* *2015*, volume 1456 of *CEUR Workshop Proceedings*, page 103. CEUR-WS.org, 2015.

[BCMT13] Roi Blanco, Berkant Barla Cambazoglu, Peter Mika, and Nicolas Torzec. Entity Recommendations in Web Search. In Harith Alani, Lalana Kagal, Achille Fokoue, Paul Groth, Chris Biemann, Josiane Xavier Parreira, Lora Aroyo, Natasha F. Noy, Chris Welty, and Krzysztof Janowicz, editors, *The Semantic Web - ISWC 2013 - 12th International Semantic Web Conference, Sydney, NSW, Australia, October 21-25, 2013, Proceedings, Part II*, volume 8219 of *Lecture Notes in Computer Science*, pages 33–48. Springer, 2013.

[BFGS19] Luigi Bellomarini, Daniele Fakhoury, Georg Gottlob, and Emanuel Sallinger. Knowledge Graphs and Enterprise AI: The Promise of an Enabling Technology. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, pages 26–37, 2019.

[BGM14] Dan Brickley, Ramanathan V Guha, and Brian McBride. RDF Schema 1.1. *W3C Recommendation*, `https://www.w3.org/TR/rdf-schema/`, 2014. Accessed: 2023-01-14.

[BHVH14] Thomas Bauernhansl, Michael Ten Hompel, and Birgit Vogel-Heuser. *Industrie 4.0 in Produktion, Automatisierung und Logistik: Anwendung-Technologien-Migration*. Springer, 2014.

[BK56] Benjamin S Bloom and David R Krathwohl. *Taxonomy of Educational Objectives: The Classification of Educational Goals*. Longmans, Green, 1956.

[BK20] Emily M. Bender and Alexander Koller. Climbing towards NLU: on meaning, form, and understanding in the age of data. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel R. Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 5185–5198. Association for Computational Linguistics, 2020.

[BKM08] Aaron Bangor, Philip T. Kortum, and James T. Miller. An Empirical Evaluation of the System Usability Scale. *International Journal of Human–Computer Interaction*, 24(6):574–594, 2008.

[BlCC+06] Tim Berners-lee, Yuhsin Chen, Lydia Chilton, Dan Connolly, Ruth Dhanaraj, James Hollenbach, Adam Lerer, and David Sheets. Tabulator: Exploring and Analyzing linked data on the Semantic Web. *International Semantic Web User Interaction*, 2006.

[BLRW19] Stefan Biffl, Arndt Lüder, Felix Rinker, and Laura Waltersdorfer. Efficient Engineering Data Exchange in Multi-disciplinary Systems Engineering. In Paolo Giorgini and Barbara Weber, editors, *Advanced Information Systems*

Engineering - 31st International Conference, CAiSE 2019, Rome, Italy, June 3-7, 2019, Proceedings, volume 11483 of *Lecture Notes in Computer Science*, pages 17–31. Springer, 2019.

[BMM+21] Stefan Biffl, Juergen Musil, Angelika Musil, Kristof Meixner, Arndt Lüder, Felix Rinker, Danny Weyns, and Dietmar Winkler. An Industry 4.0 Asset-Based Coordination Artifact for Production Systems Engineering. In *2021 IEEE 23rd Conference on Business Informatics (CBI)*, volume 01, pages 92–101, 2021.

[BMR+20] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020.

[BPSS17] Nikos Bikakis, George Papastefanatos, Melina Skourla, and Timos Sellis. A hierarchical aggregation framework for efficient multilevel visual exploration and analysis. *Semantic Web*, 8(1):139–179, 2017.

[Bro86] John Brooke. System usability scale (SUS): a quick-and-dirty method of system evaluation user information. *Reading, UK: Digital Equipment Co Ltd*, 43, 1986.

[BUG+13] Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. Translating Embeddings for Modeling Multi-relational Data. In Christopher J. C. Burges, Léon Bottou, Zoubin Ghahramani, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, pages 2787–2795, 2013.

[CB17] Miguel Ceriani and Paolo Bottoni. Sparqlblocks: Using blocks to design structured linked data queries. *J. Vis. Lang. Sentient Syst.*, 3:1–21, 2017.

[CFM14] Silvana Castano, Alfio Ferrara, and Stefano Montanelli. inWalk: Interactive and Thematic Walks inside the Web of Data. In Sihem Amer-Yahia, Vassilis Christophides, Anastasios Kementsietsidis, Minos N. Garofalakis, Stratos Idreos, and Vincent Leroy, editors, *Proceedings of the 17th International Conference on Extending Database Technology, EDBT 2014, Athens, Greece, March 24-28, 2014*, pages 628–631. OpenProceedings.org, 2014.

[ELT21] Thibault Ehrhart, Pasquale Lisena, and Raphaël Troncy. KG Explorer: a Customisable Exploration Tool for Knowledge Graphs. In Patrick Lambrix, Catia Pesquita, and Vitalis Wiens, editors, *Proceedings of the Sixth International Workshop on the Visualization and Interaction for Ontologies and Linked Data co-located with the 20th International Semantic Web Conference (ISWC 2021), Virtual Conference, 2021*, volume 3023 of *CEUR Workshop Proceedings*, pages 63–75. CEUR-WS.org, 2021.

[EW16] Lisa Ehrlinger and Wolfram Wöß. Towards a Definition of Knowledge Graphs. In *Posters&Demos@SEMANTiCS 2016 and SuCCESS'16 Workshop*, volume 1695. CEUR Workshop Proceedings, 2016.

[FBMR18] Michael Färber, Frederic Bartscherer, Carsten Menne, and Achim Rettinger. Linked data quality of DBpedia, Freebase, OpenCyc, Wikidata, and YAGO. *Semantic Web*, 9:77–129, 2018.

[FGG+18] Nadime Francis, Alastair Green, Paolo Guagliardo, Leonid Libkin, Tobias Lindaaker, Victor Marsault, Stefan Plantikow, Mats Rydberg, Petra Selmer, and Andrés Taylor. Cypher: An evolving query language for property graphs. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2018.

[Fou18] The Apache Software Foundation. The Gremlin Graph Traversal Machine and Language. `https://tinkerpop.apache.org/gremlin.html`, 2018. Accessed: 2023-01-14.

[Fri17] Armin Friedl. Exploratory search on expert knowledge graphs. Diploma thesis, TU Wien, Vienna, Austria, 2017.

[Fuk18] Mayumi Fukuyama. Society 5.0: Aiming for a new human-centered society. *Japan Spotlight*, 27:47–50, 2018.

[Gru93] Thomas R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–220, 1993.

[GSY22] Asela Gunawardana, Guy Shani, and Sivan Yogev. Evaluating Recommender Systems. In Francesco Ricci, Lior Rokach, and Bracha Shapira, editors, *Recommender Systems Handbook*, pages 547–601. Springer US, 2022.

[GZQ+22] Qingyu Guo, Fuzhen Zhuang, Chuan Qin, Hengshu Zhu, Xing Xie, Hui Xiong, and Qing He. A Survey on Knowledge Graph-Based Recommender Systems. *IEEE Trans. Knowl. Data Eng.*, 34(8):3549–3568, 2022.

[HA18] Sebastian Haag and Reiner Anderl. Digital twin – Proof of concept. *Manufacturing Letters*, 15:64–66, 2018. Industry 4.0 and Smart Manufacturing.

[Har17] Olaf Hartig. RDF* and SPARQL*: An Alternative Approach to Annotate Statements in RDF. In *16th International Semantic Web Conference (Posters, Demos & Industry Tracks)*. Springer, 2017.

[HCKS21] Olaf Hartig, Pierre-Antoine Champin, Gregg Kellog, and Andy Seaborne. RDF-star and SPARQL-star. *W3C Community Group Report*, `https://w3c.github.io/rdf-star/cg-spec`, 2021. Accessed: 2023-01-14.

[HESP22] Kevin Haller, Fajar J. Ekaputra, Marta Sabou, and Florina Piroi. Enabling exploratory search on manufacturing knowledge graphs. In Bo Fu, Patrick Lambrix, and Catia Pesquita, editors, *Proceedings of the Seventh International Workshop on the Visualization and Interaction for Ontologies and Linked Data co-located with the 21st International Semantic Web Conference (ISWC 2022), Virtual Event, Hangzhou, China, October 23, 2022*, volume 3253 of *CEUR Workshop Proceedings*, pages 16–28. CEUR-WS.org, 2022.

[HHK+19] Peter Haase, Daniel M. Herzig, Artem Kozlov, Andriy Nikolov, and Johannes Trame. metaphactory: A platform for knowledge graph management. *Semantic Web*, 10(6):1109–1125, 2019.

[HJ19] Longlong He and Pingyu Jiang. Manufacturing Knowledge Graph: A Connectivism to Answer Production Problems Query With Knowledge Reuse. *IEEE Access*, 7:101231–101244, 2019.

[HLHH18] Thomas Hubauer, Steffen Lamparter, Peter Haase, and Daniel Markus Herzig. Use Cases of the Industrial Knowledge Graph at Siemens. In *ISWC 2018 P&D/Industry/BlueSky Tracks*, volume 2180. CEUR Workshop Proceedings, 2018.

[HLS10] Philipp Heim, Steffen Lohmann, and Timo Stegemann. Interactive Relationship Discovery via the Semantic Web. In Lora Aroyo, Grigoris Antoniou, Eero Hyvönen, Annette ten Teije, Heiner Stuckenschmidt, Liliana Cabral, and Tania Tudorache, editors, *The Semantic Web: Research and Applications, 7th Extended Semantic Web Conference, ESWC 2010, Heraklion, Crete, Greece, May 30 - June 3, 2010, Proceedings, Part I*, volume 6088 of *Lecture Notes in Computer Science*, pages 303–317. Springer, 2010.

[HPPsR12] Pascal Hitzler, Bijan Parsia, Peter F Patel-schneider, and Sebastian Rudolph. OWL 2 Web Ontology Language Primer (Second edition). *W3C Recommendation*, `https://www.w3.org/TR/owl2-primer/`, 2012. Accessed: 2023-01-14.

[HPS14] Patrick J Hayes and Peter F Patel-Schneider. RDF 1.1 Semantics. *W3C Recommendation*, `https://www.w3.org/TR/rdf11-mt/`, 2014. Accessed: 2023-01-14.

[HSP13] Steve Harris, Andy Seaborne, and Eric Prud'hommeaux. SPARQL 1.1 Overview. *W3C Recommendation*, `https://www.w3.org/TR/sparql11-query/`, 2013. Accessed: 2023-01-14.

[HSSW16] Trude Hausegger, Christian Scharinger, Jürgen Sicher, and Friederike Weber. Qualifizierungsmaßnahmen im Zusammenhang mit der Einführung von Industrie 4.0. *Studie im Auftrag der Austria Wirtschaftsservice GmbH - aws, der Arbeiterkammer Wien und des Bundesministeriums für Verkehr, Infrastruktur und Technologie, bmvit*, 2016.

[HZL08] Philipp Heim, Jürgen Ziegler, and Steffen Lohmann. gFacet: A Browser for the Web of Data. In Sören Auer, Sebastian Dietzold, Steffen Lohmann, and Jürgen Ziegler, editors, *Proceedings of the International Workshop on Interacting with Multimedia Content in the Social Semantic Web (IMC-SSW'08) Koblenz, Germany, December 3, 2008*, volume 417 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2008.

[IGD23] Eleni Ilkou, Martina Galletti, and Daniil Dobriy. EduMultiKG attains 92% accuracy in K-12 user profiling! In *The Semantic Web: 20th International Conference, ESWC*, 2023.

[IHRK22] Esko Ikkala, Eero Hyvönen, Heikki Rantala, and Mikko Koho. Sampo-UI: A full stack JavaScript framework for developing semantic portal user interfaces. *Semantic Web*, 13(1):69–84, 2022.

[ISO24] ISO/IEC JTC 1/SC 32. Information Technology — Database Languages — GQL. Standard ISO/IEC CD 39075, International Organization for Standardization, 2024.

[KCBS09] Bill Kules, Robert Capra, Matthew Banta, and Tito Sierra. What do exploratory searchers look at in a faceted search interface? In Fred Heath, Mary Lynn Rice-Lively, and Richard Furuta, editors, *Proceedings of the 2009 Joint International Conference on Digital Libraries, JCDL 2009, Austin, TX, USA, June 15-19, 2009*, pages 313–322. ACM, 2009.

[KLvH16] Ali Khalili, Antonis Loizou, and Frank van Harmelen. Adaptive Linked Data-Driven Web Components: Building Flexible and Reusable Semantic Web Interfaces - Building Flexible and Reusable Semantic Web Interfaces. In Harald Sack, Eva Blomqvist, Mathieu d'Aquin, Chiara Ghidini, Simone Paolo Ponzetto, and Christoph Lange, editors, *The Semantic Web. Latest Advances and New Domains - 13th International Conference, ESWC 2016, Heraklion, Crete, Greece, May 29 - June 2, 2016, Proceedings*, volume 9678 of *Lecture Notes in Computer Science*, pages 677–692. Springer, 2016.

[KSH+18] Evgeny Kharlamov, Martin G. Skjæveland, Dag Hovland, Theofilos Mailis, Ernesto Jiménez-Ruiz, Guohui Xiao, Ahmet Soylu, Ian Horrocks, and Arild Waaler. Finding data should be easier than finding oil. In *IEEE International Conference on Big Data (IEEE BigData 2018), Seattle, WA, USA, December 10-13, 2018*, pages 1747–1756. IEEE, 2018.

[KSN19] Jakub Klímek, Petr Skoda, and Martin Necaský. Survey of tools for Linked Data consumption. *Semantic Web*, 10(4):665–720, 2019.

[LIJ+15] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, et al. DBpediaa large-scale, multilingual knowledge base extracted from Wikipedia. *Semantic Web*, 6:167–195, 2015.

[LLW+21] Xinyu Li, Mengtao Lyu, Zuoxu Wang, Chun-Hsien Chen, and Pai Zheng. Exploiting knowledge graphs in industrial products and services: A survey of key aspects, challenges, and future perspectives. *Computers in Industry*, 129:103449, 2021.

[LPRB19] Arndt Lüder, Johanna-Lisa Pauly, Felix Rinker, and Stefan Biffl. Data exchange logistics in engineering networks exploiting automated data integration. In *24th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2019, Zaragoza, Spain, September 10-13, 2019*, pages 657–664. IEEE, 2019.

[Mar06] Gary Marchionini. Exploratory search: from finding to understanding. *Commun. ACM*, 49(4):41–46, 2006.

[MD16] Rouzbeh Meymandpour and Joseph G. Davis. A semantic similarity measure for linked data: An information content-based approach. *Knowl. Based Syst.*, 109:276–293, 2016.

[MG14] Nicolas Marie and Fabien Gandon. Survey of linked data based exploration systems. In Dhavalkumar Thakker, Daniel Schwabe, Kouji Kozaki, Roberto García, Chris Dijkshoorn, and Riichiro Mizoguchi, editors, *Proceedings of the 3rd International Workshop on Intelligent Exploration of Semantic Data (IESD 2014) co-located with the 13th International Semantic Web Conference (ISWC 2014), Riva del Garda, Italy, October 20, 2014*, volume 1279 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2014.

[MGRR13] Nicolas Marie, Fabien Gandon, Myriam Ribière, and Florentin Rodio. Discovery hub: on-the-fly linked data exploratory search. In Marta Sabou, Eva Blomqvist, Tommaso Di Noia, Harald Sack, and Tassilo Pellegrini, editors, *I-SEMANTICS 2013 - 9th International Conference on Semantic Systems, I-SEMANTICS '13, Graz, Austria, September 4-6, 2013*, pages 17–24. ACM, 2013.

[MH18] José Moreno-Vega and Aidan Hogan. GraFa: Scalable Faceted Browsing for RDF Graphs. In Denny Vrandecic, Kalina Bontcheva, Mari Carmen Suárez-Figueroa, Valentina Presutti, Irene Celino, Marta Sabou, Lucie-Aimée Kaffee, and Elena Simperl, editors, *The Semantic Web - ISWC 2018 - 17th International Semantic Web Conference, Monterey, CA, USA, October 8-12, 2018, Proceedings, Part I*, volume 11136 of *Lecture Notes in Computer Science*, pages 301–317. Springer, 2018.

[MIT23] Ariana Martino, Michael Iannelli, and Coleen Truong. Knowledge Injection to Counter Large Language Model (LLM) Hallucination. In *Industry track, The Semantic Web: 20th International Conference, ESWC*, 2023.

[MLTK21] Boxuan Ma, Min Lu, Yuta Taniguchi, and Shin'ichi Konomi. Exploration and explanation: An interactive course recommendation system for university environments. In Dorota Glowacka and Vinayak R. Krishnamurthy, editors, *Joint Proceedings of the ACM IUI 2021 Workshops co-located with 26th ACM Conference on Intelligent User Interfaces (ACM IUI 2021), College Station, United States, April 13-17, 2021*, volume 2903 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2021.

[MN10] Roberto Mirizzi and Tommaso Di Noia. From exploratory search to web search and back. In Anisoara Nica and Aparna S. Varde, editors, *Proceedings of the Third Ph.D. Workshop on Information and Knowledge Management, PIKM 2010, Toronto, Ontario, Canada, October 30, 2010*, pages 39–46. ACM, 2010.

[MPG17] Alan Medlar, Joel Pyykkö, and Dorota Glowacka. Towards Fine-Grained Adaptation of Exploration/Exploitation in Information Retrieval. In George A. Papadopoulos, Tsvi Kuflik, Fang Chen, Carlos Duarte, and Wai-Tat Fu, editors, *Proceedings of the 22nd International Conference on Intelligent User Interfaces, IUI 2017, Limassol, Cyprus, March 13-16, 2017*, pages 623–627. ACM, 2017.

[MRNS10] Roberto Mirizzi, Azzurra Ragone, Tommaso Di Noia, and Eugenio Di Sciascio. Ranking the Linked Data: The Case of DBpedia. In Boualem Benatallah, Fabio Casati, Gerti Kappel, and Gustavo Rossi, editors, *Web Engineering, 10th International Conference, ICWE 2010, Vienna, Austria, July 5-9, 2010. Proceedings*, volume 6189 of *Lecture Notes in Computer Science*, pages 337–354. Springer, 2010.

[MTG14] András Micsik, Sándor Turbucz, and Attila Györök. LODmilla: a Linked Data Browser for All. In Harald Sack, Agata Filipowska, Jens Lehmann, and Sebastian Hellmann, editors, *Proceedings of the Posters and Demos Track of 10th International Conference on Semantic Systems - SEMANTiCS2014, Leipzig, Germany, September 4-5, 2014*, volume 1224 of *CEUR Workshop Proceedings*, pages 31–34. CEUR-WS.org, 2014.

[Mus15] Mark A. Musen. The protégé project: a look back and a look forward. *AI Matters*, 1(4):4–12, 2015.

[MvH04] Deborah L McGuinness and Frank van Harmelen. OWL Web Ontology Language Overview. *W3C Recommendation*, `https://www.w3.org/TR/owl-features/`, 2004.

[NDC15] Thi-Nhu Nguyen, Duy-Thanh Dinh, and Tuan-Dung Cao. Empowering Exploratory Search on Linked Movie Open Data with Semantic Technologies. In Huynh Quyet Thang, Le Anh Phuong, Luc De Raedt, Yves Deville, Marc Bui, Truong Thi Dieu Linh, Thi-Oanh Nguyen, Dinh Viet Sang, and Nguyen Ba Ngoc, editors, *Proceedings of the Sixth International Symposium on Information and Communication Technology, Hue City, Vietnam, December 3-4, 2015*, pages 296–303. ACM, 2015.

[NGPC11] Andrea Giovanni Nuzzolese, Aldo Gangemi, Valentina Presutti, and Paolo Ciancarini. Encyclopedic Knowledge Patterns from Wikipedia Links. In Lora Aroyo, Chris Welty, Harith Alani, Jamie Taylor, Abraham Bernstein, Lalana Kagal, Natasha Fridman Noy, and Eva Blomqvist, editors, *The Semantic Web - ISWC 2011 - 10th International Semantic Web Conference, Bonn, Germany, October 23-27, 2011, Proceedings, Part I*, volume 7031 of *Lecture Notes in Computer Science*, pages 520–536. Springer, 2011.

[NPG+17] Andrea Giovanni Nuzzolese, Valentina Presutti, Aldo Gangemi, Silvio Peroni, and Paolo Ciancarini. Aemoo: Linked Data exploration based on Knowledge Patterns. *Semantic Web*, 8(1):87–112, 2017.

[Pal18] Émilie Palagi. *Evaluating exploratory search engines: designing a set of user-centered methods based on a modeling of the exploratory search process. (Evaluation des moteurs de recherche exploratoire: élaboration d'un corps de méthodes centrées utilisateurs, basées sur une modélisation du processus de recherche exploratoire).* PhD thesis, Université Côte d'Azur, France, 2018.

[Pap11] Lisa Pappas. Tree View. *W3C Wiki*, `https://www.w3.org/wiki/TreeView`, 2011. Accessed: 2023-10-01.

[Pas10] Alexandre Passant. dbrec - Music Recommendations Using DBpedia. In Peter F. Patel-Schneider, Yue Pan, Pascal Hitzler, Peter Mika, Lei Zhang, Jeff Z. Pan, Ian Horrocks, and Birte Glimm, editors, *The Semantic Web - ISWC 2010 - 9th International Semantic Web Conference, ISWC 2010, Shanghai, China, November 7-11, 2010, Revised Selected Papers, Part II*, volume 6497 of *Lecture Notes in Computer Science*, pages 209–224. Springer, 2010.

[Pau17] Heiko Paulheim. Knowledge graph refinement: A survey of approaches and evaluation methods. *Semantic Web*, 8:489–508, 2017.

[PB18] Guangyuan Piao and John G. Breslin. A study of the similarities of entity embeddings learned from different aspects of a knowledge base for item recommendations. In Michael Cochez, Thierry Declerck, Gerard de Melo, Luis Espinosa Anke, Besnik Fetahu, Dagmar Gromann, Mayank Kejriwal, Maria Koutraki, Freddy Lécué, Enrico Palumbo, and Harald Sack, editors, *Proceedings of the First Workshop on Deep Learning for Knowledge Graphs and Semantic Technologies (DL4KGS) co-located with the 15th Extended Semantic Web Conerence (ESWC 2018), Heraklion, Crete, Greece, June 4, 2018*, volume 2106 of *CEUR Workshop Proceedings*, pages 2–13. CEUR-WS.org, 2018.

[PBDP20] Laura Po, Nikos Bikakis, Federico Desimoni, and George Papastefanatos. *Linked Data Visualization: Techniques, Tools, and Big Data*, chapter Linked Data Visualization Tools, pages 47–72. Synthesis Lectures on the Semantic Web. Morgan & Claypool Publishers, 2020.

[PBKL06] Emmanuel Pietriga, Christian Bizer, David R. Karger, and Ryan Lee. Fresnel: A Browser-Independent Presentation Vocabulary for RDF. In Isabel F. Cruz, Stefan Decker, Dean Allemang, Chris Preist, Daniel Schwabe, Peter Mika, Michael Uschold, and Lora Aroyo, editors, *The Semantic Web - ISWC 2006, 5th International Semantic Web Conference, ISWC 2006, Athens, GA, USA, November 5-9, 2006, Proceedings*, volume 4273 of *Lecture Notes in Computer Science*, pages 158–171. Springer, 2006.

[PBMW99] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank Citation Ranking: Bringing Order to the Web. Technical Report 1999-66, Stanford InfoLab, November 1999.

[PC95] Peter Pirolli and Stuart K. Card. Information Foraging in Information Access Environments. In Irvin R. Katz, Robert L. Mack, Linn Marks, Mary Beth Rosson, and Jakob Nielsen, editors, *Human Factors in Computing Systems, CHI '95 Conference Proceedings, Denver, Colorado, USA, May 7-11, 1995*, pages 51–58. ACM/Addison-Wesley, 1995.

[PC99] Peter Pirolli and Stuart Card. Information foraging. *Psychological Review*, 106:643–675, 1999. Publisher: American Psychological Association.

[PFM17] Erik Pasternak, Rachel Fenichel, and Andrew N. Marshall. Tips for creating a block language with blockly. In *2017 IEEE Blocks and Beyond Workshop (B&B)*, pages 21–24, 2017.

[PMGC13] Jay Pujara, Hui Miao, Lise Getoor, and William Cohen. Knowledge graph identification. In *The Semantic Web – ISWC 2013*, pages 542–557. Springer Berlin Heidelberg, 2013.

[PS08] Eric Prud'hommeaux and Andy Seaborne. SPARQL Query Language for RDF. *W3C Recommendation*, `https://www.w3.org/TR/rdf-sparql-query/`, 2008. Accessed: 2023-01-14.

[PUJ+18] Claire Palmer, Zahid Usman, Osiris Canciglieri Junior, Andreia Malucelli, and Robert I M Young. Interoperable manufacturing knowledge systems. *International Journal of Production Research*, 56:2733–2752, 2018.

[RBLS23] Lloyd Rutledge, Brent Berghuis, Kelvin Lim, and Mark Soerokromo. A Web-Based Approach for Traceability in Rule-Based Business Information Systems. In Boris Shishkov, editor, *Business Modeling and Software Design - 13th International Symposium, BMSD 2023, Utrecht, The Netherlands, July 3-5, 2023, Proceedings*, volume 483 of *Lecture Notes in Business Information Processing*, pages 308–318. Springer, 2023.

[Res95] Philip Resnik. Using Information Content to Evaluate Semantic Similarity in a Taxonomy. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, IJCAI 95, Montréal Québec, Canada, August 20-25 1995, 2 Volumes*, pages 448–453. Morgan Kaufmann, 1995.

[Ris19] Petar Ristoski. *Exploiting Semantic Web Knowledge Graphs in Data Mining*, volume 38 of *Studies on the Semantic Web*. IOS Press, 2019.

[RLR+22] Jože M Rožanec, Jinzhi Lu, Jan Rupnik, Maja Škrjanc, Dunja Mladenić, Blaž Fortuna, Xiaochen Zheng, and Dimitris Kiritsis. Actionable cognitive twins for decision making in manufacturing. *International Journal of Production Research*, 60:452–478, 1 2022.

[RNS+18] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.

[Sab16] Marta Sabou. An introduction to semantic web technologies. In Stefan Biffl and Marta Sabou, editors, *Semantic Web Technologies for Intelligent Engineering Applications*, pages 53–81. Springer International Publishing, 2016.

[SBF98] Rudi Studer, V. Richard Benjamins, and Dieter Fensel. Knowledge Engineering: Principles and methods. *Data & Knowledge Engineering*, 25(1):161–197, 1998.

[SEI+18] Marta Sabou, Fajar J Ekaputra, Tudor Ionescu, Juergen Musil, Daniel Schall, Kevin Haller, Armin Friedl, and Stefan Biffl. Exploring Enterprise Knowledge Graphs: A Use Case in Software Engineering. *The Semantic Web – ESWC 2018*, pages 560–575, 2018.

[Sin12] Amit Singhal. Introducing the Knowledge Graph: things, not strings. *Google blog*, `https://blog.google/products/search/`

introducing-knowledge-graph-things-not/, 2012. Accessed: 2023-01-14.

[SR14] Guus Schreiber and Yves Raimond. RDF 1.1 Primer. *W3C Working Group Note*, https://www.w3.org/TR/rdf11-primer/, 2014. Accessed: 2023-01-14.

[SS09] Steffen Staab and Rudi Studer. *Handbook on Ontologies*. Springer Berlin, Heidelberg, 2nd edition, 2009.

[STF+19] Jannik Strötgen, Trung-Kien Tran, Annemarie Friedrich, Dragan Milchevski, Federico Tomazic, Anika Marusczyk, Heike Adel, Daria Stepanova, Felix Hildebrand, and Evgeny Kharlamov. Towards the Bosch Materials Science Knowledge Base. In *ISWC 2019 Satellite Tracks*, volume 2456, pages 323–324. CEUR Workshop Proceedings, 2019.

[TAvHS06] Christian Tominski, James Abello, Frank van Ham, and Heidrun Schumann. Fisheye tree views and lenses for graph visualization. In *10th International Conference on Information Visualisation, IV 2006, 5-7 July 2006, London, UK*, pages 17–24. IEEE Computer Society, 2006.

[TRJ+17] Raphaël Troncy, Giuseppe Rizzo, Anthony Jameson, Óscar Corcho, Julien Plu, Enrico Palumbo, Juan Carlos Ballesteros Hermida, Adrian Spirescu, Kai-Dominik Kuhn, Catalin-Mihai Barbu, Matteo Rossi, Irene Celino, Rachit Agarwal, Christian Scanu, Massimo Valla, and Timber Haaker. 3cixty: Building comprehensive knowledge bases for city exploration. *Journal of Web Semantics*, 46-47:2–13, 2017.

[VGGS20] Cheikh Brahim El Vaigh, François Goasdoué, Guillaume Gravier, and Pascale Sébillot. A novel path-based entity relatedness measure for efficient collective entity linking. In Jeff Z. Pan, Valentina A. M. Tamma, Claudia d'Amato, Krzysztof Janowicz, Bo Fu, Axel Polleres, Oshani Seneviratne, and Lalana Kagal, editors, *The Semantic Web - ISWC 2020 - 19th International Semantic Web Conference, Athens, Greece, November 2-6, 2020, Proceedings, Part I*, volume 12506 of *Lecture Notes in Computer Science*, pages 164–182. Springer, 2020.

[VGS17] Guillermo Vega-Gorgojo, Martin Giese, and Laura Slaughter. Exploring semantic datasets with RDF surveyor. In Nadeschda Nikitina, Dezhao Song, Achille Fokoue, and Peter Haase, editors, *Proceedings of the ISWC 2017 Posters & Demonstrations and Industry Tracks co-located with 16th International Semantic Web Conference (ISWC 2017), Vienna, Austria, October 23rd - to - 25th, 2017*, volume 1963 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2017.

[VTGSR+17] Boris Villazon-Terrazas, Nuria Garcia-Santa, Yuan Ren, Alessandro Faraotti, Honghan Wu, Yuting Zhao, Guido Vetere, and Jeff Z. Pan. Knowledge graph foundations. In *Exploiting Linked Data and Knowledge Graphs in Large Organisations*, pages 17–55. Springer Cham, 2017.

[WAS+18] David Weintrop, Afsoon Afzal, Jean Salac, Patrick Francis, Boyang Li, David C. Shepherd, and Diana Franklin. Evaluating coblox: A comparative study of robotics programming environments for adult novices. In Regan L. Mandryk, Mark Hancock, Mark Perry, and Anna L. Cox, editors, *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems, CHI 2018, Montreal, QC, Canada, April 21-26, 2018*, page 366. ACM, 2018.

[WF12] Barbara M. Wildemuth and Luanne Freund. Assigning search tasks designed to elicit exploratory search behaviors. In *Human-Computer Information Retrieval Symposium, HCIR 2012, Cambridge, MA, USA, October 4-5, 2012*, page 4. ACM, 2012.

[WMZ+16] Jost Wübbeke, Mirjam Meissner, Max J Zenglein, Jaqueline Ives, and Björn Conrad. Made in China 2025. *Mercator Institute for China Studies. Papers on China*, 2:74, 2016.

[WR09] Ryen W. White and Resa A. Roth. *Exploratory Search: Beyond the Query-Response Paradigm.* Synthesis Lectures on Information Concepts, Retrieval, and Services. Morgan & Claypool Publishers, 2009.

[WRKB20] Laura Waltersdorfer, Felix Rinker, Lukas Kathrein, and Stefan Biffl. Experiences with technical debt and management strategies in production systems engineering. In Clemente Izurieta, Matthias Galster, and Michael Felderer, editors, *TechDebt '20: International Conference on Technical Debt, Seoul, Republic of Korea, June 28-30, 2020*, pages 41–50. ACM, 2020.

[XFZN18] Luyan Xu, Zeon Trevor Fernando, Xuan Zhou, and Wolfgang Nejdl. Logcanvas: Visualizing search history using knowledge graphs. In Kevyn Collins-Thompson, Qiaozhu Mei, Brian D. Davison, Yiqun Liu, and Emine Yilmaz, editors, *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, SIGIR 2018, Ann Arbor, MI, USA, July 08-12, 2018*, pages 1289–1292. ACM, 2018.

[ZZL+20] Guanghui Zhou, Chao Zhang, Zhi Li, Kai Ding, and Chuang Wang. Knowledge-driven digital twin manufacturing cell towards intelligent manufacturing. *International Journal of Production Research*, 58:1034–1051, 2020.