UNIVERSITY
INSTITUTE
OF LISBON

MyCODA: A Collaborative Knowledge Base Interface for the Many-Criteria Optimization and Decision Analysis Research Community.

Tiago Miguel Lopes Nunes

Master's in Computer Engineering,

Supervisor:
PhD Vítor Manuel Basto Fernandes, Associate Professor with Habilitation, Iscte-IUL

Co-supervisor:
PhD Michael Emmerich, Professor in Multiobjective Optimization, University of Jyväskylä, Finland

October 2024

TECHNOLOGY
AND ARCHITECTURE

Department of Information Science and Technology

MyCODA: A Collaborative Knowledge Base Interface for the Many-Criteria Optimization and Decision Analysis Research Community.

Tiago Miguel Lopes Nunes

Master's in Computer Engineering,

Supervisor:
PhD Vítor Manuel Basto Fernandes, Associate Professor with Habilitation, Iscte-IUL

Co-supervisor:
PhD Michael Emmerich, Professor in Multiobjective Optimization, University of Jyväskylä, Finland

October 2024

I dedicate this dissertation to my Parents, Professors, Colleagues, and Friends,
As well as to the Scientific Community, whom I seek to assist with this work.

Acknowledgements

I would like to express special thanks to my supervisor, Professor Vítor Basto Fernandes, for his valuable guidance, constant support, and availability throughout the development of this work. This acknowledgment also extends to my co-supervisor, Professor Michael Emmerich, for his valuable feedback and support.

I am grateful to my parents, Felisberto and Maria João, to my grandmother Judite, to my brother Daniel, and to the rest of my family, for the good example they have set for me, for the high expectations they have for me, and for everything they do for me.

I also thank my colleagues who were part of my project groups, especially my colleague Rodrigo Simões, with whom I spent many hours on online calls while we worked on our dissertation projects to mutually motivate each other, and with whom I thoroughly enjoyed collaborating throughout the Master's program.

I extend my thanks to ISCTE, for allowing me to host a server on a virtual machine within the university's infrastructure, and for providing me with the resources necessary to further develop my academic level.

I am also grateful to the company Bryj, where I worked for much of my Master's, and to all my colleagues at the company, to whom I wish the greatest success.

Finally, I thank all my friends who accompany me in life, for the positive influence they have on me.

Abstract

The growth in scientific knowledge production brings challenges in its management. As in other fields, in Multi-objective Optimization there are already incentives to use ontologies, within the context of the semantic web, to organize, store, and share knowledge. However, the community would benefit from an accessible way to consult knowledge and to contribute new content.

In this work, we develop a web-based knowledge management platform, named MyCODA. This platform operates on an ontology defined in OWL, which contains relevant knowledge for the domain of Many-Criteria Optimization and Decision Analysis. The platform allows users to browse the ontology's taxonomy, search for existing terms, and contribute to its improvement, aiming to make it simple and intuitive, so that even those unfamiliar with OWL ontologies can use it effectively.

To promote the contribution of new knowledge by the community, a new tool developed within the platform allows researchers to submit information about an article relevant to the knowledge area and obtain a contextual analysis that facilitates the understanding of existing knowledge. This offers researchers an intuitive and efficient way to introduce new contributions.

Keywords: Knowledge Engineering; Scientific Knowledge Base; Ontology; Multi-objective Optimization; Many-Criteria Optimization and Decision Analysis, OWL.

Resumo

O crescimento na produção de conhecimento científico traz desafios no âmbito da gestão do mesmo. Assim como noutras áreas, na Otimização Multiobjectivo já existem incentivos de utilização de ontologias, no contexto da web semântica, para organizar, armazenar e compartilhar conhecimento. No entanto, a comunidade beneficiaria de uma forma acessível de consultar o conhecimento e de contribuir com novo conteúdo.

Neste trabalho, é desenvolvida uma plataforma web de gestão do conhecimento, denominada MyCODA. Esta plataforma, trabalha sobre uma ontologia definida em OWL, que contem conhecimento relevante para o domínio de Otimização com Muitos Critérios e Análise de Decisão. A plataforma permite navegar sobre a taxonomia da ontologia, pesquisar por termos existentes e contribuir para o seu aprimoramento, procurando fazê-lo de forma simples e intuitiva, para que possa ser utilizada efetivamente, mesmo por aqueles que não sejam entendedores de ontologias OWL.

De forma a promover a contribuição de novo conhecimento pela comunidade, uma nova ferramenta desenvolvida na plataforma dá a possibilidade a investigadores de submeterem informações sobre um artigo que seja relevante na área do conhecimento, e obterem uma análise contextual que facilita a compreensão do conhecimento existente, oferecendo aos investigadores uma maneira intuitiva e eficiente de introduzir novas contribuições.

Palavras-chave: Engenharia do Conhecimento; Base de Conhecimento Científico; Ontologia; Otimização Multiobjectivo; Otimização com Muitos Critérios e Análise de Decisão, OWL.

Index

List of Figures

CHAPTER 1

Introduction

1.1. Background and Motivation

The rapid expansion of scientific knowledge and the growing number of research publications create unprecedented opportunities for knowledge dissemination and development on a global scale. However, this surge in information also brings significant challenges in knowledge management. The volume of scientific data is often poorly organized and difficult to access, which can lead to fragmentation and inconsistencies in how knowledge is categorized and interpreted (Borgman, 2015). These challenges include divergent taxonomies, inconsistent terminologies for similar concepts, and identical terms being used to refer to distinct ideas, complicating efforts to harness and apply scientific knowledge effectively.

Knowledge management is the process or practice involved in the creation, acquisition, capture, sharing, and utilization of knowledge, regardless of its location (Scarbrough et al., 1999). The primary objective is to enhance learning and performance for both organizations and individuals. Additionally, this approach plays a crucial role in extracting value from expertise within a specific knowledge domain.

Capturing and sharing domain knowledge with both machines and humans can be achieved using ontologies. Currently, ontologies are the most suitable method for formally representing concepts within a specific domain and articulating the relationships that exist between them. Ontologies not only establish a shared understanding of the structure of information but also facilitate the sharing and reuse of knowledge (Scarbrough et al., 1999). By using an ontology, a novice researcher or practitioner can easily explore details about an algorithm for a specific application or identify potential future research topics. This significantly reduces the effort required to acquire knowledge in a certain domain.

In the biomedical field, for example, ontologies have proven to be highly valuable for the research community. Notable examples of successful implementations include the Gene Ontology (Ashburner et al., 2000; The Gene Ontology Consortium et al., 2023), which encompasses more than 40,000 terms and the Uberon ontology (Mungall et al., 2012), which comprises over 25,000 terms. These ontologies have played a pivotal role in standardizing terminology and facilitating knowledge sharing within the domain.

In the field of multi-objective optimization, we are beginning to observe the development of various ontologies as well, such as the PMOEA ontology (Li et al., 2017), the OPTION ontology (Kostovska et al., 2022) and the MOODY ontology (Aldana-Martín et al., 2024).

Since the concept of ontologies is still relatively new to the multi-objective optimization community, it is essential to intensify our efforts to explore how we can maximize the utility of this technology. By doing so, we can try to replicate the success it has achieved in other fields where its impact is more pronounced, such as in the biomedical domain.

A key factor in the success of an ontology appears to be the availability of a platform that allows for easy browsing and querying of the ontology, along with a clear framework for contributing new terms. Notable examples include the AmiGO platform (Carbon et al., 2009) for the Gene Ontology project and the OLS (Ontology Lookup Service) platform (Côté et al., 2006, 2010), which hosts the Uberon ontology as well as other ontologies in the Open Biomedical Ontology (OBO) format (Golbreich et al., 2007; Tirmizi et al., 2011).

In chapter 13 of '*Many-Criteria Optimisation and Decision Analysis Book*' (Basto-Fernandes et al., 2023), the concept of the MyCODA platform is presented. This platform aims to enable users to easily access, learn about, and compare existing optimization methods, search for appropriate methods for specific problems, share new scientific knowledge, identify research gaps, promote collaboration among researchers in MACODA (Many Criteria Optimization and Decision Analysis) and manage knowledge within the domain.

Feedback on the MyCODA platform proposal, provided by a considerable number of respected researchers in MACODA during the Multi-Criteria Optimization and Decision Analysis Workshop (Lorentz Center, University of Leiden, Netherlands, September 16-21, 2019), revealed the need, relevance, value, and potential use of this platform by the MACODA research community, thereby motivating this master's dissertation work.

The present dissertation details the process of developing MyCODA, a web-based application to manage an ontology with knowledge in the domain of MACODA, to support the scientific community, taking into consideration the main factors that make other ontologies successful, to maximize the utility of this technology.

1.2. Objectives

The primary objective of this dissertation is to document the development of the MyCODA platform and figure out what are the essential features that can enhance its utility for the MACODA research community.

Finally, to understand if the scientific community could benefit from the platform, we'll gather feedback from the MACODA researchers' community.

1.3. Research Questions

Within the scope of the subject under study, the research questions that motivate the analysis prepared are the following:

Q1: How can we maximize the utility of ontologies within the field of MACODA research?

Q2: Does the software proposal resulting from the MACODA Workshop in the University of Leiden in 2019, and described in the book chapter (Basto-Fernandes et al., 2023), identify all the MACODA research community knowledge management needs?

1.4. Contribution Goals

The following contributions of this dissertation are aligned with the former objectives:

Contribution 1: A web-based application designed for managing an ontology, facilitating users in easily accessing, learning about, and comparing existing optimization methods. This application allows users to search for suitable methods for specific problems, share new scientific knowledge, identify research gaps, promote collaboration among researchers in MACODA, and effectively manage knowledge within the domain.

Contribution 2: A framework and standards for contributing to the ontology with new knowledge, curating proposed new knowledge, providing feedback, and taking part in the community' efforts for disseminating and aggregating useful knowledge within the MACODA domain.

1.5. Methodology

To address the subject of this dissertation and attain the outlined objectives from the previous chapters, the Design Science Research method (DSR) (Carstensen & Bernhard, 2019) was adpted. This method facilitated an organized, guided, and efficient research process.

Given that challenges often arise from intricate and distinctive designs and considering that Information Systems (IS) inherently consist of adaptable and flexible hardware, software, and human interfaces, they demand contemporary, modern, and creative ideas (Hevner et al., 2004). Therefore, DSR focuses on developing, designing, or "building" new artifacts, with evaluation primarily centered around the results of design science, corresponding IS Design Theories, and associated design artifacts. A comprehensive and rigorous approach to research is essential in DSR, necessitating the evaluation of the artifact's utility, quality, and effectiveness using appropriate evaluation methods. This aids in elucidating changes or improvements in the system, people, or organizational behavior (Venable et al., 2016).

As described in (Peffers et al., 2006), the DSR methodology incorporates six activities in a nominal sequence, which are presented in Figure 1.

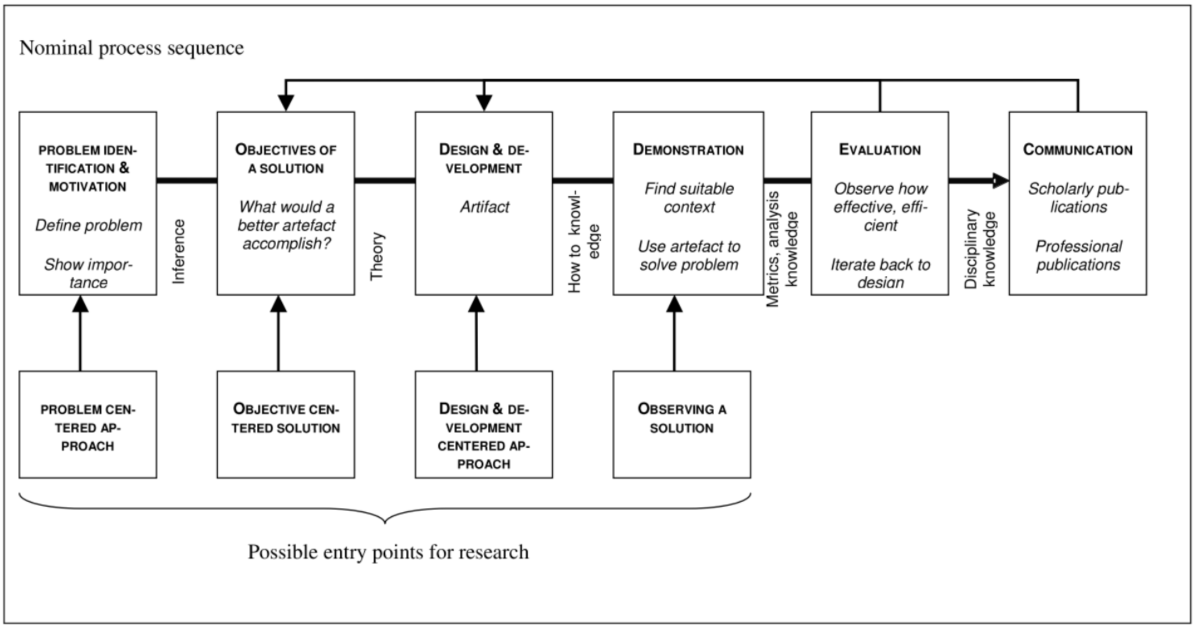


Figure 1 – DSR Methodology Process Model (Peffers et al., 2006)

1.6. Document Structure

The Introduction and Literature Review (Chapters 1 and 2) aim to provide a clear and detailed understanding of the subject matter. These chapters define the dissertation's objectives, introduce essential concepts related to the topic, highlight potential contributions to the field, and outline the proposed methodology for achieving the stated goals. Together, they establish a solid foundation for a comprehensive understanding of the context of this dissertation.

Following this, the Implementation Setup chapter (Chapter 3) provides a detailed explanation of the decisions made in selecting the resources and technologies used in the implementation of the platform.

Chapter 4 consists of an overview of the MyCODA platform, describing each feature in detail, and providing an insightful discussion about the options chosen during development and algorithms that were used to tackle the most difficult issues.

In the Validation chapter (Chapter 5), the methods used for validating the platform and its features are described and justified.

The last chapter of this dissertation is the Conclusion (Chapter 6), which provides a concise summary of the research findings, reflects on the achievement of the objectives, discusses the broader implications of the work, and offers recommendations for future research and practical applications. It also highlights the project's contributions to the field and addresses any limitations encountered.

For the citations and references in this dissertation, the American Psychological Association 7th edition style has been used. The guidelines for APA 7th edition can be found at the official website of the American Psychological Association: https://apastyle.apa.org.

CHAPTER 2

Literature Review

This chapter aims to gather and analyze information pertinent to the dissertation topic, covering topics such as ontologies and semantic web concepts in the context of knowledge representation and management, and describing previous efforts in knowledge management within the scope of MACODA research.

The chapter also highlights some examples of platforms that are currently being used, that have been developed with the same objective of managing knowledge for the scientific community, although in different scientific domains, such as the AmiGO platform (Carbon et al., 2009), and the OLS platform (Côté et al., 2006, 2010), that aim to aid the biomedical research community.

2.1. Ontology

Etymologically, ontology originates from Greek and essentially means "the study or theory of being or that which is". In simpler terms, ontology seeks to classify and explain entities.

In philosophy, ontology is defined as "the science of what is, of the kinds and structures of objects, properties, events, processes, and relations in every area of reality" (Smith, 2004). Over recent decades, ontologies have gained popularity in other fields such as Knowledge Management, Artificial Intelligence, and the Semantic Web, driven by the necessity for a shared and common understanding of domains.

In Computer Science, Gruber and Borst were pioneers in defining the concept of ontology (Borst, 1997; Gruber, 1993). Subsequently, Studer et al. presented the most widely accepted definition of ontology: "An ontology is a formal, explicit specification of a shared conceptualization" (Studer et al., 1998). Here, "conceptualization" denotes an abstract model of a knowledge domain representing concepts and their relationships. "Explicit specification" implies that the model should be represented using a coherent, unambiguous, and structured language. "Formal" suggests that the ontology should be interpretable by machines. "Shared" indicates that knowledge represented in an ontology should establish a common and agreed-upon vocabulary in each domain, enabling sharing across individuals and application systems.

Ontologies delineate the semantics of a knowledge domain by delineating concepts (or classes) representing existing 'entities' and their interrelationships, properties associated with each concept, constraints on concepts or properties, and axioms. An instance of a class is referred to as an individual.

The process of constructing an ontology is intricate and varied approaches exist to guide ontology development. A general framework proposed for the ontology-building process is provided by Noy (N. Noy & Mcguinness, 2001):

1. Determine the domain and scope of the ontology.
2. Consider reusing existing ontologies.
3. Enumerate important terms in the ontology.
4. Define the classes and the class hierarchy (taxonomy).
5. Define object properties.
6. Define data properties.
7. Create individuals.

The semantic structure offered by ontologies diverges from the organization of information provided by relational and XML databases. Ontologies establish an objective specification of domain information by embodying a consensus on the concepts and relationships that characterize the expression of knowledge within that domain.

By furnishing a formal and hierarchically structured representation of a knowledge domain with universally accepted definitions, ontologies mitigate misunderstandings and miscommunications while enabling reasoning.

Through the adoption of a shared underlying vocabulary, ontologies facilitate interoperability among computer agents, enabling them to comprehend incoming requests and furnish the requisite knowledge in return.

Additionally, their semantic structure streamlines the process of precise knowledge indexing and retrieval. A shared comprehension of a domain among individuals and application systems promotes knowledge sharing and reuse, not only within communities of experts but also among new learners. In this study, an ontology serves as the primary mechanism for representing and disseminating domain knowledge of interest.

2.2. Ontologies in Knowledge Management

An ontology plays a crucial role in knowledge management by facilitating the representation of knowledge. It accomplishes this by offering a shared vocabulary for a specific domain of interest. Through explicit knowledge representation, an ontology presents information in a format understandable by machines, enabling reasoning based on a defined set of facts and rules within the domain.

The advantages of employing an ontology for knowledge management in the MACODA domain are readily apparent. An ontology is particularly well-suited for organizing and processing vast amounts of information, offering the necessary capabilities to structure the scientific knowledge generated in this field systematically.

A significant portion of the MACODA knowledge domain can be effectively represented using formal logics, specifically predicate logics, supported by OWL ontologies knowledge representation standards. For instance, consider the following excerpt from the PMOEA ontology, which depicts a fragment of the PMOEA taxonomy (hierarchy of classes/subclasses) using the "isA" type of relation, along with the "canSolve" type of relation to indicate which algorithms can effectively address specific optimization problems:

- *JobShop isA SchedulingProblem*,
- *FlowShop isA JobShop*,
- *NSGA-II canSolve JobShop*.

This structured representation allows for clear delineation of relationships between entities, facilitating effective knowledge organization and retrieval within the MACODA domain.

Furthermore, we have the capability to incorporate our own specific knowledge into the knowledge base. For instance, we could assert that '*mySchedulingProblem isA FlowShopProblem*'. By leveraging OWL ontologies for knowledge representation, we not only could query and retrieve explicit knowledge stored in the knowledge base (e.g., a query such as 'what are the algorithms that can solve *JobShop*?' would yield *NSGA-II* algorithm), but also benefit from the inference capabilities based on formal logics performed on the entire knowledge base by the querying engine. For example, a query such as 'what are the algorithms that can solve *mySchedulingProblem*?' would include *NSGA-II* algorithm, because *mySchedulingProblem* is a specific case of *JobShop*, and *NSGA-II* can solve *JobShop*. This demonstrates the power of leveraging ontologies for knowledge management and inference in the MACODA domain.

2.3. Semantic Web

Knowledge representation through OWL ontologies fosters a standardized and open portrayal of knowledge on a global scale, particularly within the World Wide Web ecosystem. A suite of standards established by the World Wide Web Consortium (W3C), which includes the OWL (Web Ontology Language) standard (Antoniou & Harmelen, 2003), constitutes what is recognized as the Semantic Web or Web of Knowledge (Berners-Lee, 2002), contrasting with the Web of HTML (HyperText Markup Language) Content.

According to Berners-Lee et al., the Semantic Web is defined as 'an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation' (Berners-Lee, 2002). Essentially, the Semantic Web, also known as Web 3.0, offers a collection of standards and technologies that empower computers to comprehend and manipulate data in a manner akin to humans. It emphasizes connecting pieces of information within documents or applications, rather than the documents or applications themselves—placing emphasis on semantics rather than the structure of the data.

The Semantic Web relies on a set of standards outlined by the World Wide Web Consortium (W3C) to formally represent metadata. These technologies establish a common framework for sharing information across diverse applications and systems, facilitating the collection, structuring, and retrieval of data in a cohesive manner.

The architecture of the Semantic Web is depicted in Figure 2.

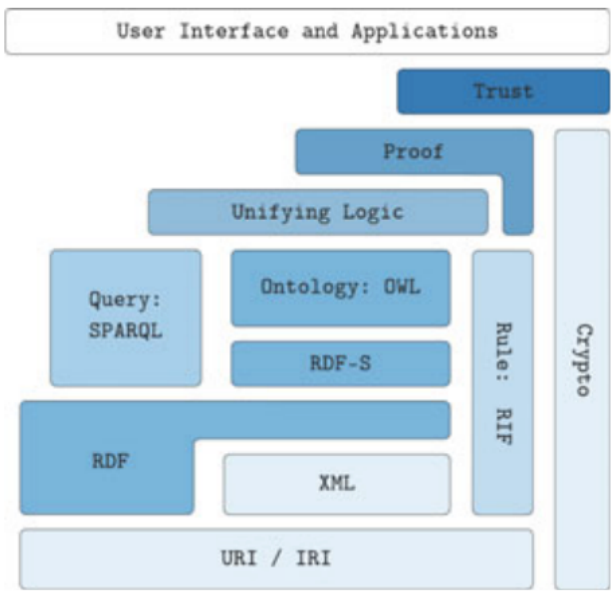


Figure 2 – Semantic Web Protocol Stack (Walker, 2011)

The lower layer standards of the Semantic Web Protocol Stack facilitate resource identification and basic forms of data representation. This includes the use of URI/IRI (Uniform Resource Identifier/Internationalized Resource Identifier) to identify OWL ontologies, classes, properties, etc., and the XML (Extensible Markup Language) family of standards to define lexical and syntactical structures and annotations of OWL ontologies.

On the other hand, the upper layer standards enable the representation of more abstract concepts and relations, allowing for the modeling of the knowledge domain of interest. This involves representing knowledge domain relations and semantics, accomplished through standards such as RDF (Resource Description Framework) and the OWL (Web Ontology Language) family of standards. Query languages for both data and knowledge representation layers are also defined and available in the Semantic Web standards stack, with SPARQL (SPARQL Protocol and RDF Query Language) (Angles & Gutierrez, 2008) and SQWRL (Semantic Query-enhanced Web Rule Language) (O'Connor & Das, 2009) being examples of a query languages.

In this work, our focus lies on OWL (Web Ontology Language), a knowledge representation language for ontologies. OWL encompasses three sub-languages: OWL Full, OWL Description Logic (DL), and OWL Lite (Antoniou & Harmelen, 2003). OWL DL is particularly suitable for our purposes due to its balanced trade-off between language expressiveness and formal logic reasoning features. For the sake of brevity and clarity, we will not delve into the specifics of the differences between the OWL sub-languages. Some of the relevant features of OWL DL for our work include:

- It enables the setting of cardinality restrictions to limit the number of distinct values a property may possess. This capability is useful for expressing constraints such as an algorithm can solve one or more types of optimization problems, an algorithm may have one or more authors, or an algorithm has only one creation year.
- It provides the ability to declare two classes as disjoint. This feature allows for the expression of concepts such as optimization problems being either combinatorial or continuous, without overlap.
- OWL allows for the definition of classes as logical combinations (intersections, complements, or unions) of other classes. This allows for the creation of complex class structures that accurately represent the relationships between different concepts.
- OWL defines various properties such as functional, reflexive, symmetric, inverse, and transitive properties. For example, it can express that the relation *isExtensionOf* is transitive, meaning that if algorithm X is an extension of algorithm Y, and algorithm Y is an extension of algorithm Z, then algorithm Z can be inferred as an extension of algorithm X. This capability enables inference and querying engines to process and reason about relationships between entities accurately.

OWL serves as an ontology language for the Semantic Web with formally defined meaning, enabling the use of a reasoner for maintaining a consistent and correct hierarchy of classes, as well as for formal logic inference and ontology querying.

Ontologies, represented as OWL documents, can be published on the Web and may refer to or be referred from other OWL ontologies, facilitating richer integration, sharing, and reuse of data.

In 2009, the W3C announced a new version of OWL, named OWL 2 (Grau et al., 2008, p. 2). OWL 2 retains a very similar structure to OWL but introduces new features, including increased expressive power for properties, extended support for datatypes, simple meta modelling capabilities, extended annotation capabilities, and keys. Additionally, it introduced three new profiles: OWL 2 EL, OWL 2 QL, and OWL 2 RL. OWL 2 EL is useful in applications employing large-scale ontologies. OWL 2 QL targets applications that handle very large volumes of instance data, where query answering is of paramount importance. OWL 2 RL is designed for applications requiring scalable reasoning without sacrificing too much expressive power.

An OWL ontology consists of classes, individuals, and properties. Classes can have subclasses that represent more specific concepts than their superclass. The hierarchy of classes establishes the taxonomy adopted in the ontology. Individuals represent instances of classes within the domain of interest. Properties are categorized into two types: object properties and data properties. Object properties are binary relations used to relate classes or individuals, while data properties associate classes or individuals with primitive data types (e.g., integer, string, boolean).

Several environments and tools are available for constructing ontologies, including OntoStudio (Weiten, 2009), Protégé (Gennari et al., 2003; Tudorache et al., 2013) and NeOn Toolkit (Erdmann & Waterfeld, 2012). Among these, Protégé has emerged as the most popular and widely used Semantic Web ontology editor, owing to the increasing adoption of OWL (Musen, 2015).

Protégé desktop (Gennari et al., 2003) is a free, open-source, Java-based ontology editor and framework designed for building both simple and complex ontology-based applications. It boasts a robust community comprising academic, government, and corporate users who leverage Protégé to develop knowledge-based solutions across diverse domains such as biomedicine, e-commerce, and organizational modeling. Protégé fully adheres to the latest OWL specifications and offers support for collaborative ontology editing, as well as annotation of ontology components and changes.

The Protégé editor screenshot provided in Figure 3 showcases a segment of the PMOEA ontology. It offers a glimpse into the Graphical User Interface (GUI) of the Protégé editor, featuring three panels: the MACODA taxonomy (left panel), instances of MACODA classes (middle panel), and relationships among MACODA domain concepts (right panel). Knowledge engineers can utilize the Protégé ontology editor for visualizing, comprehending, and modifying the PMOEA ontology, with assistance and input from domain experts (e.g., researchers/experts in the domain) who may lack knowledge engineering or semantic web standards expertise.

WebProtégé (Tudorache et al., 2013) is a lightweight ontology editor and knowledge acquisition tool for the Web, built on the Protégé infrastructure. It is accessible from any web browser, offering extensive support for ontology collaboration and featuring a highly customizable and pluggable user interface adaptable to users of varying expertise levels. Both Protégé and WebProtégé are employed in the present work for ontology design and editing.

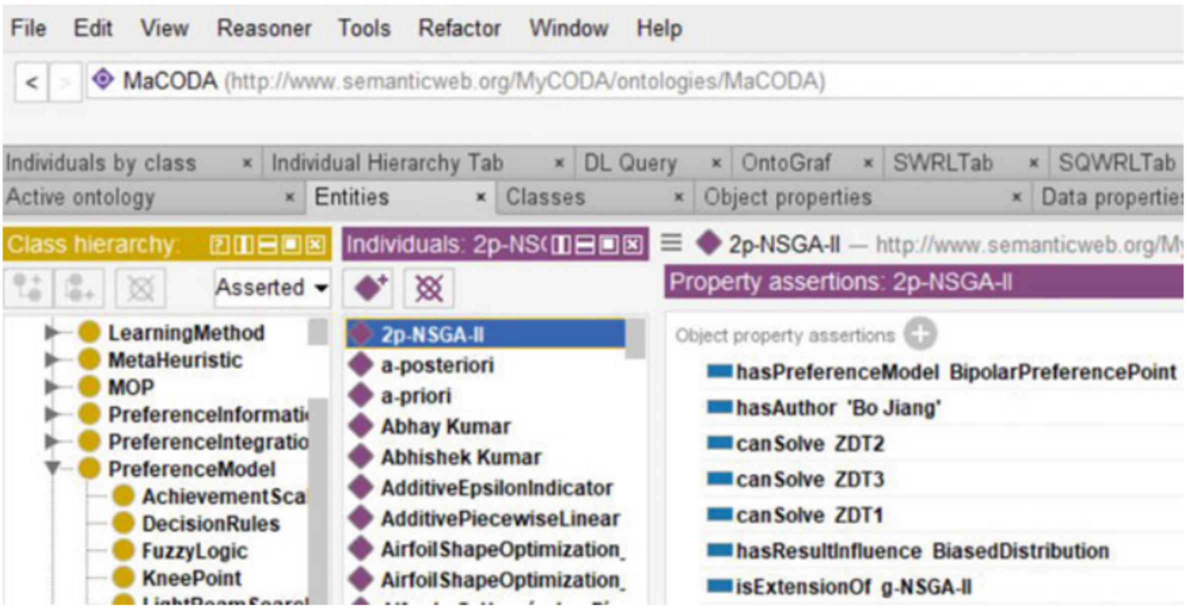


Figure 3 – View of PMOEA Ontology with Protégé Ontology Editor GUI (Basto-Fernandes et al., 2023)

2.4. Ontologies in Multi-Objective Optimization

Although it is still in an early phase, there are a few ontologies that have been built related to the scientific research field of Multi-Objective Optimization, which are relevant to the work being developed in this dissertation.

The Preference-based Multi-Objective Evolutionary Algorithms (PMOEA) Ontology (Li et al., 2017) is a structured framework designed to facilitate the organization of knowledge within the domain of preference-based multi-objective evolutionary algorithms. The ontology captures the key concepts, relationships, and attributes relevant to this area, aiming to enable users to understand and utilize these algorithms more effectively. This ontology lacks a framework to contribute and an easy way to access and browse the knowledge.

OPTION (Kostovska et al., 2022), is an ontology specifically developed to formalize knowledge around the benchmarking of optimization algorithms, with an emphasis on the formal representation of data related to performance metrics and problem landscape characteristics. It provides a comprehensive framework that describes key aspects of the benchmarking process and core entities involved, including optimization algorithms, benchmark problems, and evaluation measures. OPTION can be accessed and browsed via BioPortal (N. F. Noy et al., 2009), an ontology management platform supporting principled ontologies in biomedical science and clinical care. It includes its own framework for community contributions, allowing users to submit performance data from the BBOB benchmark suite within the COCO environment (Elhara et al., 2019).

MOODY (Multi-Objective Optimization ontologY) (Aldana-Martín et al., 2024) is an ontology developed to formalize multi-objective evolutionary algorithms, including their parameters, multi-objective continuous problems with search space landscape characteristics, and the quality indicators used to evaluate algorithm performance. MOODY does not provide a framework to contribute with new knowledge and provides a minimal way to directly browse the knowledge, via a static generated page, available in https://jfaldanam-phd.gitlab.io/moody/.

It is clear that ontologies within the Multi-Objective Optimization research field are lacking an ontology management platform designed specifically for the domain, in order to enhance their usefulness and maintainability. The current solutions for browsing, visualizing and querying the ontology are inexistent or lacking in functionality, in the case of the PMOEA and MOODY ontologies, or they are fit into a platform designed for a different domain, which is the case of the OPTION ontology. These ontologies also lack an easy framework for contributing to the knowledge base, except for the OPTION ontology, which has a very specific framework for contributing, which fundamentally would not fit in our use case for an ontology for the MACODA research domain.

2.5. Ontology Management Platform

Ontology management platforms are software tools designed to facilitate the visualization, editing and maintenance of ontologies. These platforms provide functionalities for organizing, managing, and updating knowledge bases, ensuring consistency and enabling effective use of ontologies. Several ontology management platforms are widely used, each serving specific purposes and domains.

In Life Sciences (LS), ontologies and knowledge management platforms have been especially prevalent and useful (Panzarella et al., 2023). Examples successful ontology management platforms are the AmiGO platform (Carbon et al., 2009), to manage the GO ontology (The Gene Ontology Consortium et al., 2023), the OLS (Côté et al., 2006, 2010) platform, which currently works with 267 ontologies, and the BioPortal (N. F. Noy et al., 2009), which currently includes 1171 ontologies.

Looking at the existing ontologies and management platforms, we have extracted which we conclude to be the main factors for the success of ontologies:

- Contributions:
    - Allow contributions to the ontology.
    - Clearly explain how to contribute to the ontology, have a documented framework with step-by-step instructions.
    - Make the contribution process easy for both contributors and curators.
- Visualization:
    - Allowing accessing the ontology data via a web-based platform with easy readability.
    - Show the data in various ways: Trees, Graphs, Tables.
    - Allow searching of terms, and querying based on rules defined by the ontology.
- Maintainability:
    - Make the most of integration with existing tools, such as GitHub Issues.
    - Work in synchrony with existing ontologies.
    - Use academic-standard tools.

Given this information, during the first phase of the DSR methodology, the problem identification and motivation phase, it was decided that it would be beneficial to develop the MyCODA Platform to manage the knowledge related to the MACODA research domain, taking ideas from the AmiGO, OLS and BioPortal platforms, and adding functionality based on the research and the feedback gathered from the evaluation after demonstrations, according to the DSR methodology. These functionalities would include:

- Browsing the ontology, taking inspiration from the OLS platform.
- Searching terms in the ontology, which is well implemented in AmiGO, OLS and BioPortal.
- Contributing to the knowledge base via GitHub Issues, which is the framework provided for most of the ontologies and platforms described previously.
- Contributing from an article submission, where the user could submit an article related to the research domain, and the platform would provide context on what terms contained in the article's title, abstract and keywords would be already in the knowledge base, suggest new additions, including the article itself, and provide an easy way to propose changes to the ontology. This is an innovative idea, which was introduced during one of the bi-weekly demonstrations meetings. The searching of existing terms from text functionality was inspired from the Annotator feature from BioPortal, where the user can get annotations for biomedical text with classes from the ontologies in the library, which is available in https://bioportal.bioontology.org/annotator.

CHAPTER 3

Implementation Setup

To ensure maximum accessibility and encourage broad engagement within the research community, the MyCODA platform was developed as a responsive, lightweight web application that can be accessed across a range of devices and operating systems. A key goal was to provide a seamless, user-friendly experience that would lower barriers to entry for all users, including those who may be less familiar with ontology-based systems.

Developing a web application involves numerous design decisions that influence both the user experience and the platform's overall performance. Careful consideration was given to the choice of programming framework, balancing flexibility with performance efficiency to meet the needs of an academic user base. Additionally, the application's infrastructure was designed to support scalability, ensuring that the platform can accommodate an increasing volume of contributions as it gains wider adoption in the MACODA community.

This chapter presents and discusses several aspects of the implemented solution, detailing our rationale for selecting specific tools, the underlying architectural framework, and how these choices serve the objectives of accessibility, usability, and robustness. By grounding these design decisions in both technical requirements and user-centered principles, the platform is positioned to maximize its impact as a knowledge management resource.

3.1. Data source considerations

A platform for managing a knowledge base requires data sources to serve as repositories for structured information. Such data sources should ideally support both efficient data retrieval and effective organization.

Ontologies, as highlighted in the literature, are particularly suited to this purpose because they offer a formal, structured approach to representing complex domains.

Selecting relevant ontologies or integrating existing ones provides a solid foundation for the platform, enabling it to serve as a reliable resource for the research community.

In the context of establishing a knowledge base for the MACODA domain, the following existing ontologies were evaluated as potential foundational resources for the knowledge managed by the platform.

3.1.1. The PMOEA ontology

The PMOEA Ontology (Li et al., 2017) has been used as the sole data source of the first prototype of the MyCODA platform described in chapter 13 of ‘Many-Criteria Optimization and Decision Analysis Book’ (Basto-Fernandes et al., 2023).

This is a great choice to be a data source for the platform, because the data is structured in a simple, concise and standardized way, the knowledge it contains is very relevant to the MACODA research domain, and there is not yet a framework for contributing to this knowledge base, nor is there a platform to browse or query the knowledge.

3.1.2. The OPTION ontology

OPTION (Kostovska et al., 2022) is not an ideal data source for this platform due to its complex base structure, which adheres to the Open Biological and Biomedical Ontology Foundry principles—standards primarily designed for the biological sciences.

Additionally, OPTION can already be accessed and browsed via BioPortal, an ontology management platform supporting principled ontologies in biomedical science and clinical care.

It also includes its own framework for community contributions, allowing users to submit performance data from the BBOB benchmark suite within the COCO environment.

3.1.3. The MOODY ontology

MOODY (Aldana-Martín et al., 2024) could be a strong candidate as a data source for the platform, given the high relevance of its content to the MACODA research domain and the current lack of a framework for contributing to this knowledge base.

However, using MOODY presents a few challenges. Firstly, MOODY contains numerous terms that duplicate those found in the PMOEA ontology, and mapping terms across both ontologies, although useful, is beyond the scope of this dissertation.

Additionally, MOODY incorporates links to the DMOP ontology (Keet et al., 2015), a large and more complex ontology structured around DOLCE (Descriptive Ontology for Linguistic and Cognitive Engineering) (Borgo et al., 2022).

This complexity poses a disadvantage for this use case, as it complicates the platform design by introducing entities outside of an ontology directly editable by MACODA knowledge curators, which limits the usability of the platform’s contribution tools.

Furthermore, when navigating the knowledge base, it would be necessary to address the additional challenge of filtering out information irrelevant to the MACODA domain, originated from linked ontologies.

3.1.4. Conclusion: The MyCODA ontology, based on the PMOEA ontology

After careful consideration, it was determined that the best approach would be to create an ontology derived from the knowledge in the PMOEA ontology, the MyCODA ontology, with modifications to optimize its use within the platform and within the MACODA research domain.

This tailored ontology aims to specifically address the needs of the MACODA research community.

These modifications include:

- The addition of the concept of *Synonym* (IRI ending in *#altLabel*), an annotation property which is used to provide alternative labels to an entity.
- The renaming of labels to adhere with naming conventions principles described by Schober (Schober et al., 2007), such as replacing camel case with separators–in this case, spaces (“ ”), and replacing abbreviated labels and acronyms with explicit names, adding these abbreviations as synonyms instead.
- The addition of the *Article* class, with added properties *has keyword*, *has author*, and *has doi*.

In future work, the platform may be adapted to work with other data sources, depending on the context for which it will be used.

3.2. Ontology management API

In designing an ontology management solution for the MACODA platform, multiple ontology management APIs were evaluated to determine the best fit for handling ontology storage and querying functionalities.

1. *Apache Jena*: a popular Java-based framework that provides extensive support for managing RDF data and ontologies. It includes a suite of tools for creating, querying, and updating ontologies through SPARQL.

2. *OWL API*: A Java API specifically designed for creating, manipulating, and reasoning over OWL ontologies (Horridge & Bechhofer, 2011). The OWL API also integrates well with various reasoners–e.g., HermiT (Shearer et al., 2008) and Pellet (Sirin et al., 2007)–, allowing for powerful inference capabilities to deduce new relationships within the knowledge base. The SWRLAPI (O'Connor et al., 2008), a Java library designed for querying the knowledge within an ontology using SQWRL, is dependent on the OWL API, as they are both components of the Protégé Project.
3. *Neo4j*: Neo4j is a powerful graph database that is well-suited for handling ontological data with highly interconnected structures. Unlike traditional RDF or OWL ontologies, Neo4j's property graph model allows for the representation of complex relationships and supports highly performant, real-time queries through the Cypher query language. Although Neo4j is not natively based on OWL or RDF standards, it offers flexibility for modelling ontologies as graph structures, with tools available for importing RDF data.

According to the literature, working directly with OWL and SQWRL, rather than with RDF and SPARQL alone, provides a more expressive and specialized framework for ontology management. OWL (Web Ontology Language) enables richer semantic representation, supporting complex relationships and logical constructs that are essential for defining detailed domain-specific knowledge structures. SQWRL complements OWL by enabling powerful querying capabilities that go beyond SPARQL's traditional capabilities with RDF. Therefore, for our use case, it is preferred to use the OWL API instead over the Jena API for managing the ontology, in conjunction with the SQWRL API for querying the ontology.

Neo4j would also be a good option, as it provides a more complete suite for graph management applications and supports scalable performance optimization. However, using the OWL API and SQWRL API is more straightforward for this use case, as it relies solely on artifacts of academic works, rather than on commercial products and ecosystems.

3.3. Backend framework

Given this decision of using OWL API, which is a Java API, for the purpose of ease of integration, it would make sense for the backend server for the web application to be developed in a JVM-based programming language.

Existing JVM-based web server frameworks include Spring Framework (Spring, 2024), Micronaut (Micronaut Foundation, 2024), Ktor–a Kotlin Framework– (JetBrains, 2024), and many more.

Ktor offers a modern and straightforward API that emphasizes clarity, and utilizing Kotlin instead of Java enables developers to achieve the same results with significantly less code. This combined with the fact that the main developer of this platform has more professional experience with the Kotlin programming language, and with the Ktor framework, than with other JVM-based web server frameworks, is the reason that the platform's backend is written with this framework and programming language.

The backend server employs the Representational State Transfer (REST) protocol, an architectural style based on HTTP introduced by Roy Fielding (Fielding, 2000). REST is widely used for designing networked applications, particularly web services, and it establishes a standardized approach for different systems to communicate and exchange data over the internet. This protocol is centered on the concept of resources, where each component is treated as a resource accessible through a common interface that utilizes standard HTTP methods.

3.4. Internal database

The platform requires an internal database within the server, which stores:

1. Users' sensitive information used for communication purposes, such as e-mails from contributors, which should not be added to the ontology.
2. Any data submitted to the platform, which may be used for the purpose of debugging and improving the functionalities.
3. Feedback from users, such as answers to surveys and open suggestions for improvements.

The database framework used for this project is MySQL (Oracle Corporation, 2024), because of its ease to setup using a Docker Image (Docker Inc., 2024), and its high popularity. Therefore, the corresponding driver is used by the platform, the MySQL Connector-J, a JDBC– Java Database Connectivity– driver for connecting JVM applications to MySQL databases.

3.5. Frontend Framework

In selecting a frontend framework for this dissertation project, we opted for Vite (Vite Core Team, 2024) and Vue.js (Vue.js Core Team, 2024) due to their complementary strengths and modern features that enhance development efficiency and user experience.

Vite is a build tool that significantly optimizes the development process. It leverages native ECMAScript modules, which allows for instantaneous server start and hot module replacement (HMR) during development, providing a smoother experience compared to traditional bundlers. This focus on performance and simplicity makes it particularly appealing for projects that prioritize quick iterations and responsiveness.

Vue.js is a progressive JavaScript (JS) framework known for its ease of integration and flexibility. Its component-based architecture facilitates the development of reusable UI components, which enhances maintainability and scalability. The documentation for Vue.js is widely praised for being comprehensive and beginner-friendly, making it accessible to developers at all skill levels.

3.6. Version control

Git was chosen as the version control software. GitHub complemented Git by providing cloud-hosted storage, making the codebase accessible across devices and safeguarded against data loss. Each commit was clearly documented with messages that specified changes, contributing to an organized project history that facilitated debugging and tracking.

Together, Git and GitHub allowed for efficient development, high code quality, and resilient project management, creating an environment that streamlined the project's progression and maintained a high standard of organization and reliability.

3.7. Deployment

Deployment of the service is handled by SIIC, ISCTE's Information and Communication Infrastructure Services. The service is currently available via the link https://mycoda.iscte-iul.pt.

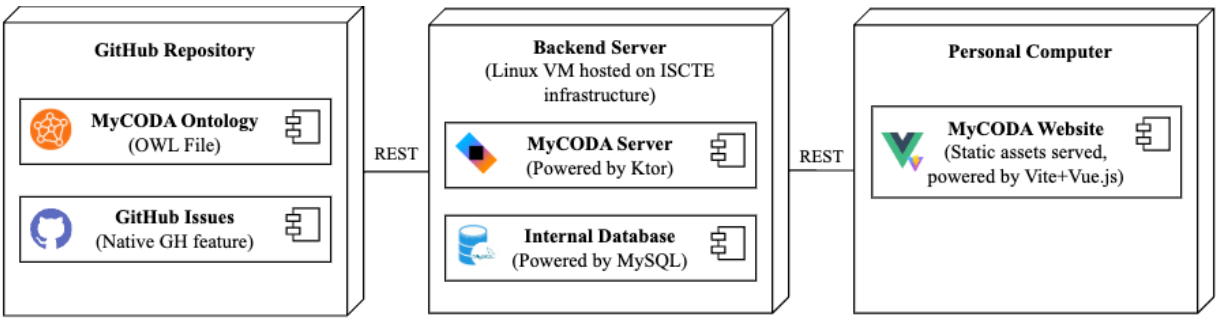


Figure 4 – MyCODA UML deployment diagram

Figure 4 represents the deployment diagram describing the architecture of the MyCODA software components. The MyCODA ontology component is where the knowledge base is stored, which comprises of an OWL file, stored in the MyCODA GitHub repository. GitHub Issues are a native GitHub feature, which will be used to document proposals of changes to the ontology or to the platform and track the development of these changes.

The MyCODA backend server is powered by the Ktor engine and is deployed in a Linux Virtual Machine (VM), hosted on the ISCTE infrastructure. The Ktor application connects to the internal database directly using the MySQL Connector-J driver. This internal database is powered by the MySQL engine, and it is also hosted in the backend server machine. As described in section 3.4, this database's primary function is to store user information and SUS form and feedback submissions. The server communicates with the GitHub repository via the GitHub REST API and via basic HTTP requests to fetch the required data. At startup, the server downloads the ontology file from the GitHub repository, so any information retrieved from the knowledge base is based on this copy. However, to keep the platform up to date with the latest ontology hosted in the repository, the platform is restarted daily, and a tool is provided for curators to manually trigger the ontology download by the server. This is described in more detail in the section 4.6.2.

The users' main interactions with the system are via the MyCODA website, the platform that is built using Vite and Vue.js frontend development technologies, exported to static files (e.g. HTML, CSS, JS files) which are served by the MyCODA backend server, and downloaded by the internet browser on the user's system. As the user interacts with the platform, requests are made to the server using a REST API provided by the Ktor server, to communicate with the MyCODA services, including fetching information from the ontology, storing data in the internal database, and generating GitHub Issues from contributions.

CHAPTER 4

MyCODA Platform

The final artifact resulting from this dissertation work, the MyCODA platform, comprises of multiple components, divided visually into tabs.

Following is a comprehensive view of each of these tabs, the tools and functionalities they contain, and instructions on how they are designed to work.

4.1. Home tab

The home tab is the default view of the platform. This page contains a summary about the project and its mission, to give some brief context to its users.

The content of the home page is static, rendering HTML generated from a Markdown file located in the MyCODA GitHub repository, with the intention of making it easier to edit. Therefore, apart from the anchor links, this page does not provide any interactive functionalities, as it is meant solely to display general information to the user.

This page also contains a reference to the very relevant *Many-Criteria Optimisation and Decision Analysis Book*, published as a result from the MACODA initiative, launched in September 2019 in the context of the MACODA Workshop, organized by the Lorentz Center at the Leiden Institute of Advanced Computer Science (LIACS), University of Leiden.

The home page is displayed in Figure 5.

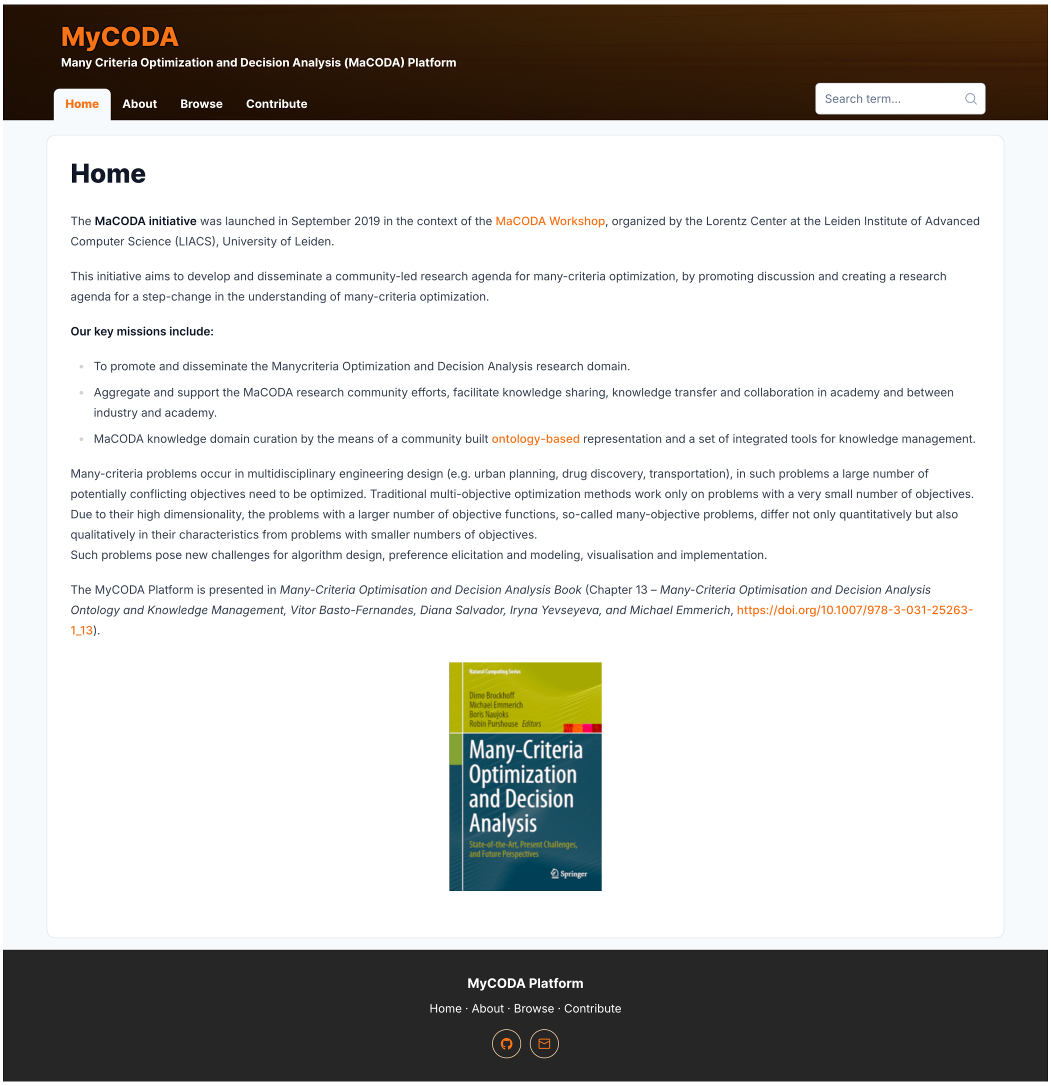


Figure 5 – MyCODA Platform Home tab

4.2. About tab

The about tab, like the home tab, is a simple static page, rendering HTML generated from a Markdown file located in the GitHub repository.

This page, displayed in Figure 6, contains useful contact information, links to educational resources relevant to the context of the platform and the MACODA domain, as well as a list of events that are related to the domain.

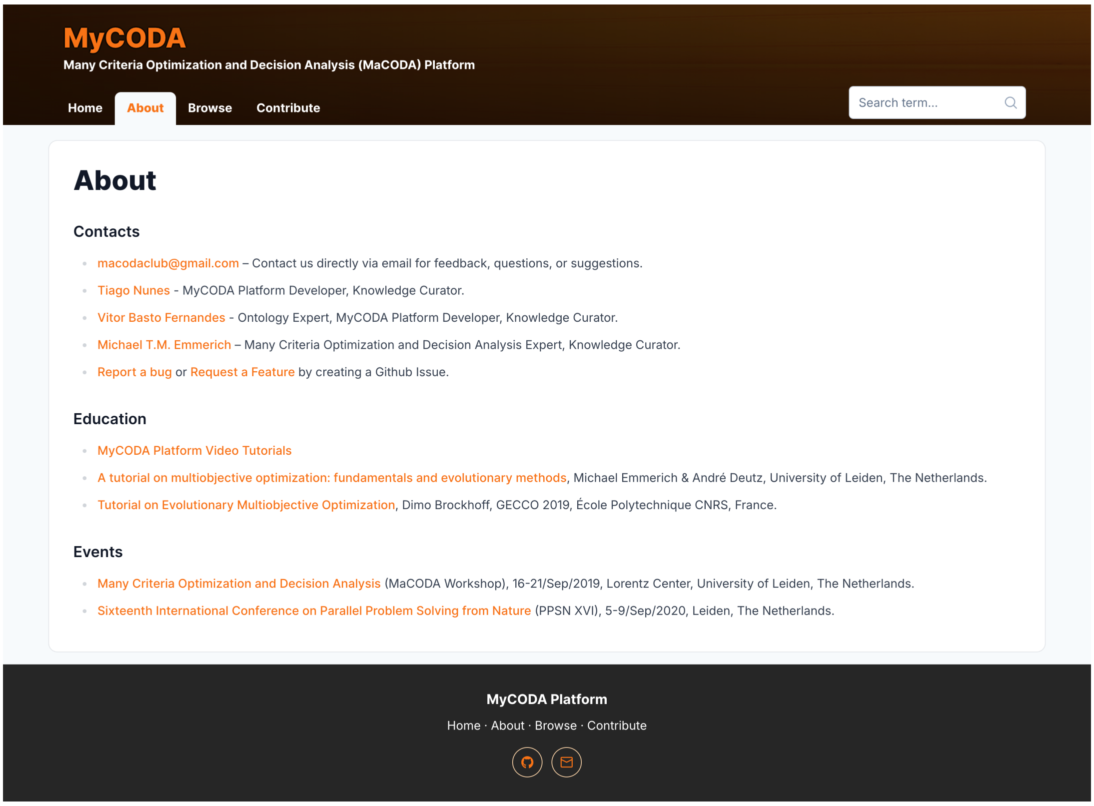


Figure 6 – MyCODA Platform About tab

4.3. Browse tab

The browse tab, displayed in Figure 7, contains a tool to navigate the ontology taxonomy. This tool consists of two panels positioned side by side.

Most of design of this functionality was inspired from the OLS platform, with some minor different design decisions. For example, while an entity is selected, in this platform it is possible to expand the hierarchy of specific siblings one by one, using the "…" buttons, while in the OLS platform the user would only have the options to have all siblings collapsed or all the siblings expanded. Expanding all siblings is often a very demanding task, which results in the user having to wait a long time for the action of expanding to take place, even if they only needed to expand a single sibling.

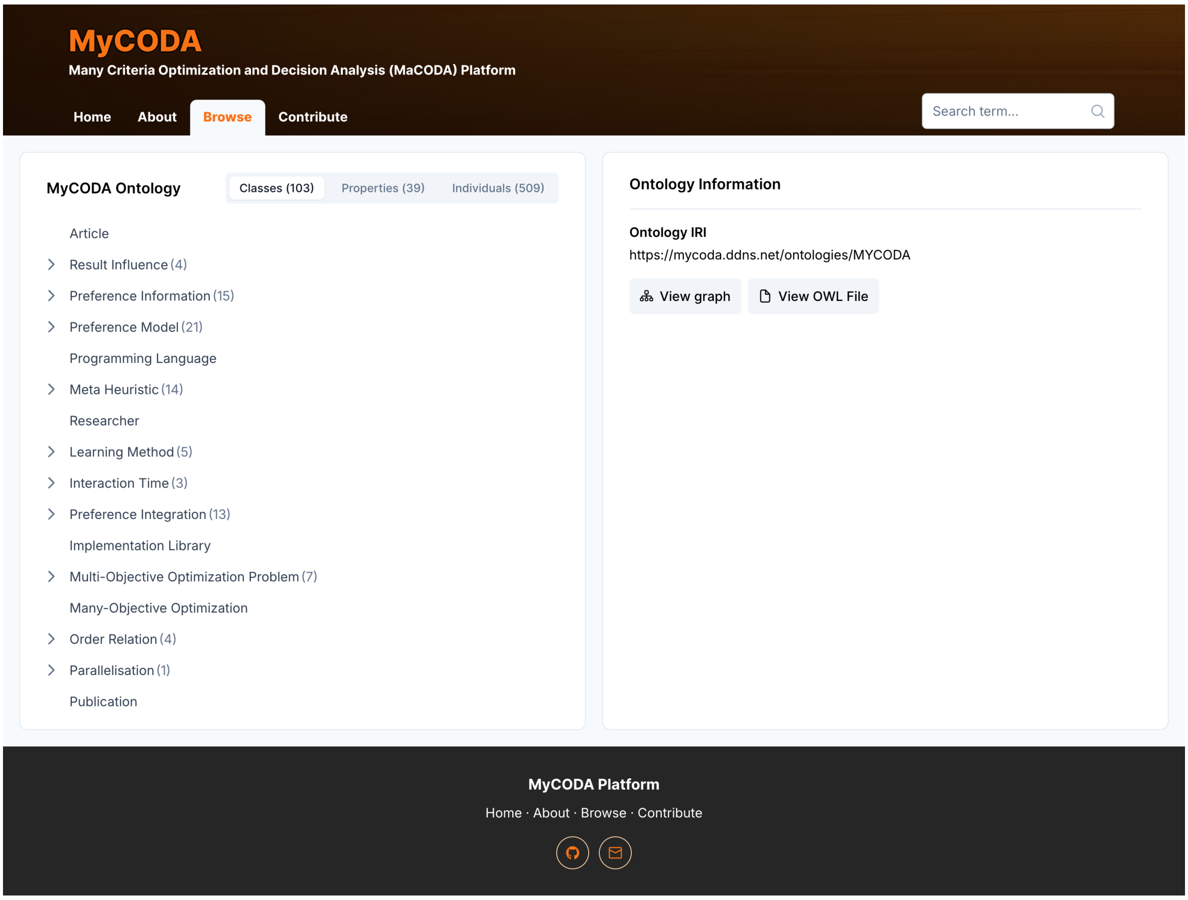


Figure 7 – MyCODA Platform Browse tab

4.3.1. Ontology taxonomy tree view (left panel)

The left panel, displayed in Figure 8, shows the ontology taxonomy tree view. Inside this panel, we can find the following components:

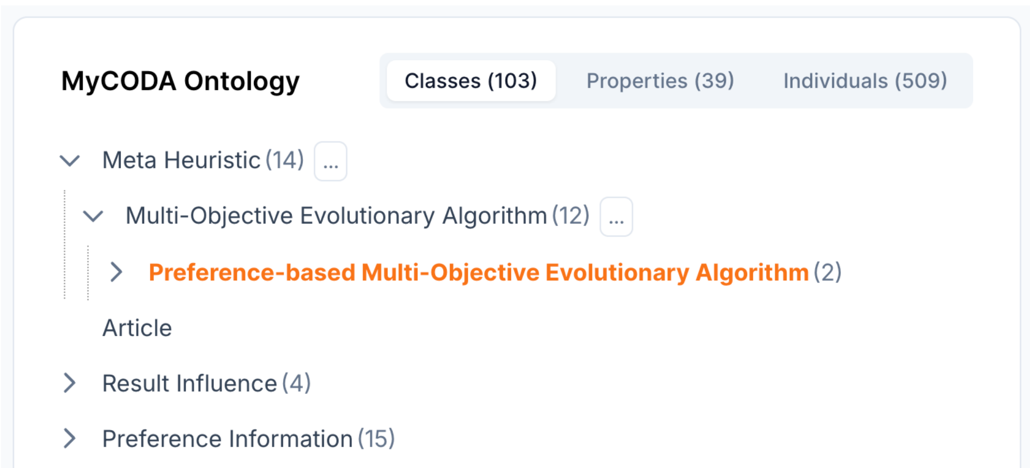


Figure 8 – MyCODA Platform ontology taxonomy tree view (left panel)

The left panel displays the title of the ontology taxonomy tree view, or the name of the ontology, which is *MyCODA Ontology*. Clicking on this title resets the view to its default state, which is a list of the ontology classes without any entity selected or expanded, and it shows the ontology information on the right panel.

On the right of the title is a button for each general type of entity–classes, properties and individuals–, followed by the number of existing entities of that type in the ontology between parentheses. Clicking on one of these buttons selects the entity type filter applied to the taxonomy tree view.

Under the title and entity type buttons, lies the taxonomy tree view, which displays a list of entities corresponding to the entity type selected from the entity type buttons. By default, it only lists top-level entities. These are classes without a superclass, properties that are not sub-properties, and any individuals. In this list, an entity is represented by a label, which is either its label annotation property, or the end of its corresponding IRI, after the '#' character. If the entity has any subclasses or sub properties, its label is followed by the number of nested subclasses or sub properties, between parentheses. Clicking on an entity label selects that entity, showing only that entity highlighted in orange and its ancestors in the taxonomy tree, and displaying detailed entity information in the right panel. Additionally, each tree level shows a button with text "…", which when clicked displays all the sibling entities within that tree level. Entities that have subclasses or sub properties have a button on their left with a caret symbol which when clicked expands their children within the tree, showing direct subclasses or sub properties.

4.3.2. Ontology/Entity information (right panel)

The right panel displays information either about the ontology or about the selected entity. In this panel, any entity that is referenced may be clicked, which selects that entity and updates both panels accordingly.

If no entity is selected, the ontology information is displayed in this panel, which is presented in Figure 9.

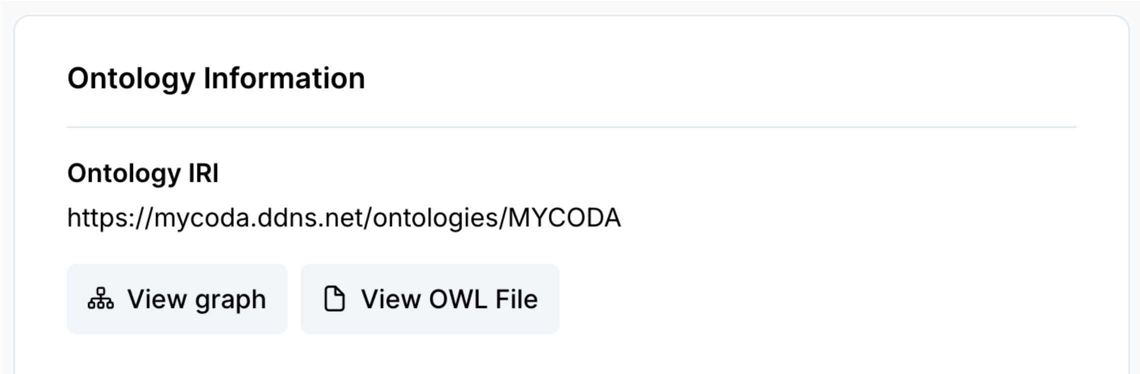


Figure 9 – MyCODA Platform Ontology/Entity information (right panel)

At the top of this panel a static label is displayed, indicating that the data in this panel refers to the ontology information.

Below the label, the ontology annotation properties are presented, which consist of pairs of property label on top and property value on the bottom. Currently, the only ontology annotation property is the *Ontology IRI*, but other annotation properties may be added in the future, such as the *Version IRI* and the *Author(s)*.

At the end of this panel, a button View graph is displayed. Clicking this button opens the WebVOWL view of the ontology in a new tab, navigating to https://service.tib.eu/webvowl/ -iri=https://raw.githubusercontent.com/macodaclub/MyCODA/refs/heads/main/ontologies/MaCODA.owl, providing a graph visualization tool for OWL ontologies, as displayed in Figure 10. This visualization is probably too complex for the average user of the platform that does not have much expertise in the field of ontologies. However, it is useful to have this functionality available for those users that have enough experience with ontologies, and that may want to see the ontology from a broader perspective.

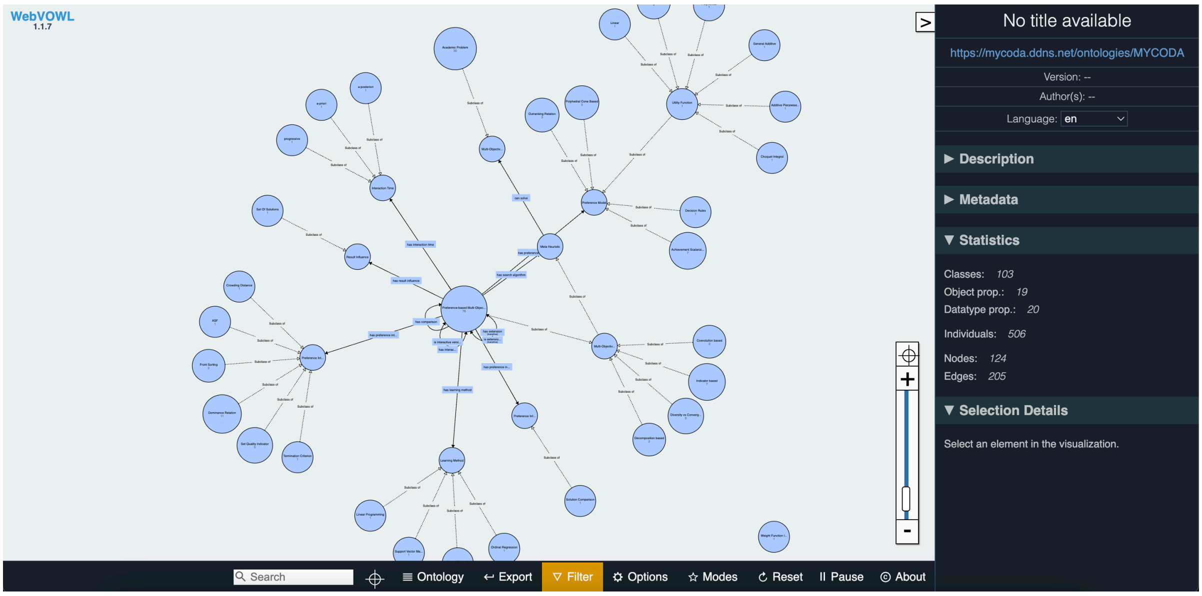


Figure 10 – WebVOWL view of the MyCODA ontology

Clicking on the View OWL File button this button shows the raw contents of the OWL file that describes de ontology, which is in the GitHub repository.

When an entity is selected the right panel will show information about the selected entity, as can be seen in Figure 11.

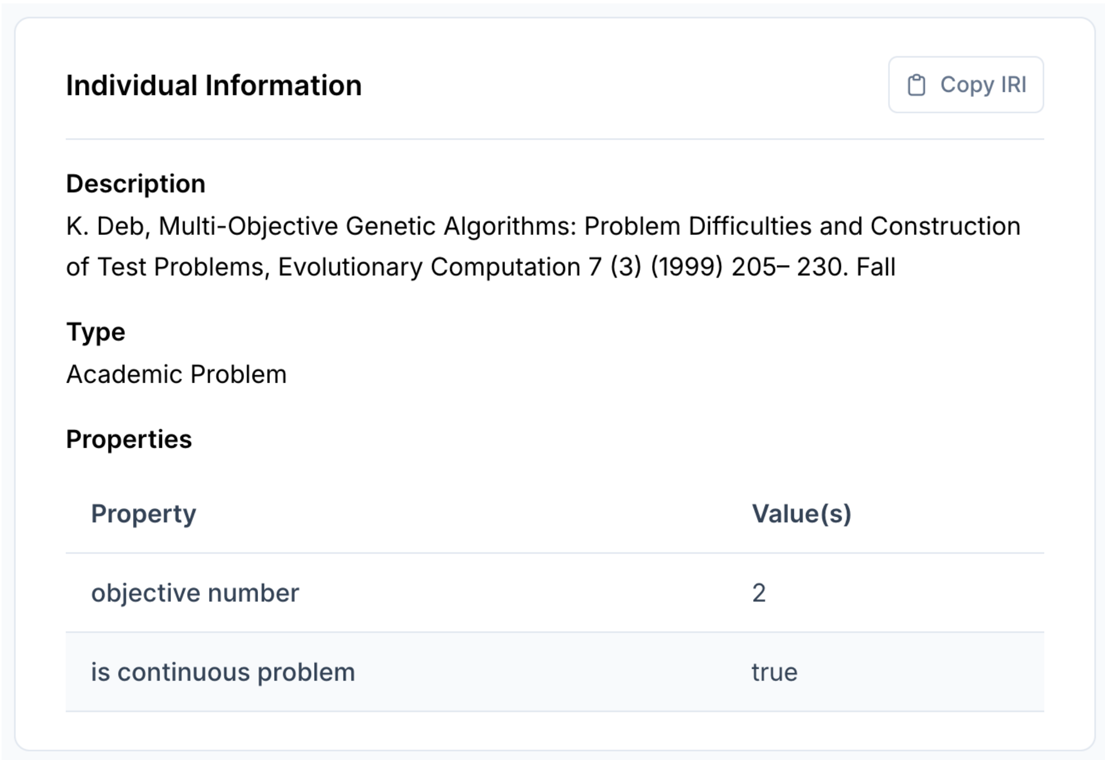


Figure 11 – MyCODA Platform entity information

At the top of this panel, a label is displayed, which specifies the type of entity, for example, “Individual Information”. On the right of the label, there is a button to Copy IRI. Clicking this button copies the IRI of the entity to the clipboard.

The description field corresponds to the *owl:comment* annotation property of the entity. If the entity does not contain this annotation property, the description is not displayed.

If the entity has any annotation properties, such as synonyms, a table is displayed with all the annotation properties’ labels and respective values. If the entity is a property, its domain and range are displayed. The domain corresponds to the type of individual that this property may be applied to, and the range corresponds to the type of value this property can take. If the entity is an individual, its type is specified, and below a list of properties assigned to this individual is displayed, paired with the value(s) of each property.

Performing a HTTP GET request to the IRI of an entity belonging to the MyCODA ontology navigates the user to the Browse tab of the MyCODA platform, with the respective entity selected.

4.4. Contribute tab

The contribute tab, displayed in Figure 12, relays instructions on how to contribute to the ontology, giving the options of contributing using a tool designed to enrich the ontology from an article, or proposing specific changes to the ontology by creating a GitHub Issue.

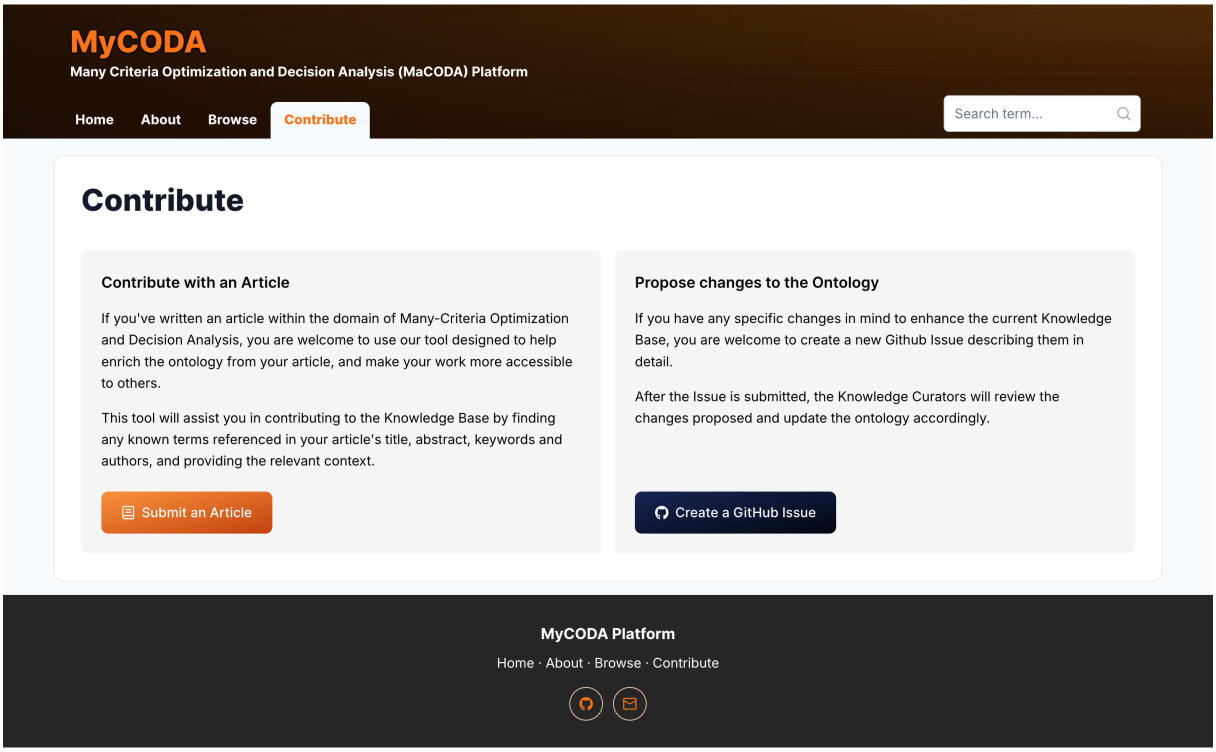


Figure 12 – MyCODA Platform Contribute tab

4.4.1. Contributing by creating a GitHub Issue

Clicking on the Create a GitHub Issue button leads the user to the GitHub Issues page of the project's GitHub repository, where they can choose between starting from a Bug report or Feature request template, or opening a blank issue. This page is displayed in Figure 13. The maintainers of the repository are notified when a new issue is created and may resolve the issue at their convenience.

This is a native feature provided by GitHub, and it is the way that most ontologies have available as the means of contributing to the knowledge base, although different templates are built. As more issues are created, we may update the existing or create new templates, as per the demand of our users.

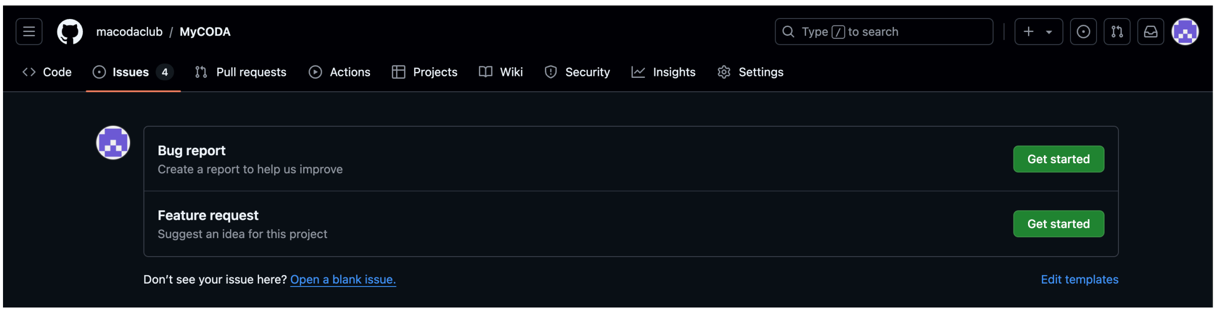


Figure 13 – MyCODA GitHub Issue creation

4.4.2. Contributing by submitting an article

From the contribution page, clicking on the Submit an article button leads the user to the article submission form, displayed in Figure 14, the first step of the tool designed to help contribute to the ontology from an article.

Along each step of the contribution, the user may click on the circular button with a question mark ('?') to find brief instructions on how to proceed, displayed in Figure 33, Figure 34, Figure 35 and Figure 36, accompanied by a video tutorial showcasing how to use the tool, displayed in Figure 37.

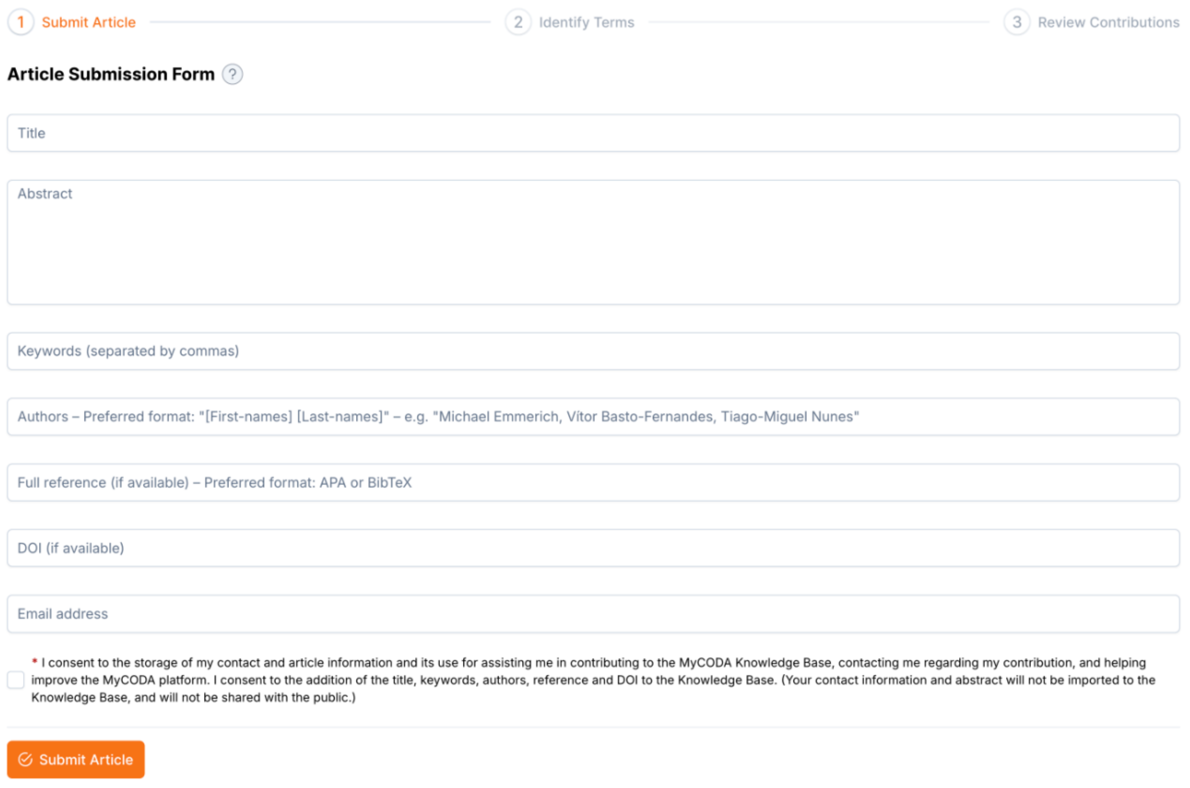


Figure 14 – MyCODA Article Submission Form

In this form, the user is prompted to add information about the article, as well as an e-mail address. To continue, they also required to check the box, consenting to the storage of their contact and article information, contacting them regarding their contribution. Checking this box, they consent to the addition of the title, keywords, authors, reference and DOI to the Knowledge Base.

After submitting the article, the user is lead to the next step, where existing terms/entities are identified, and using this information, they can add new terms or edit existing ones, to contribute to the Knowledge Base.

This page, displayed in Figure 32, consists of three sections, which are described below.

4.4.2.1. Submitted Article section

Any terms identified in the article that already exist in the knowledge base are highlighted in orange in this section, displayed in Figure 15.

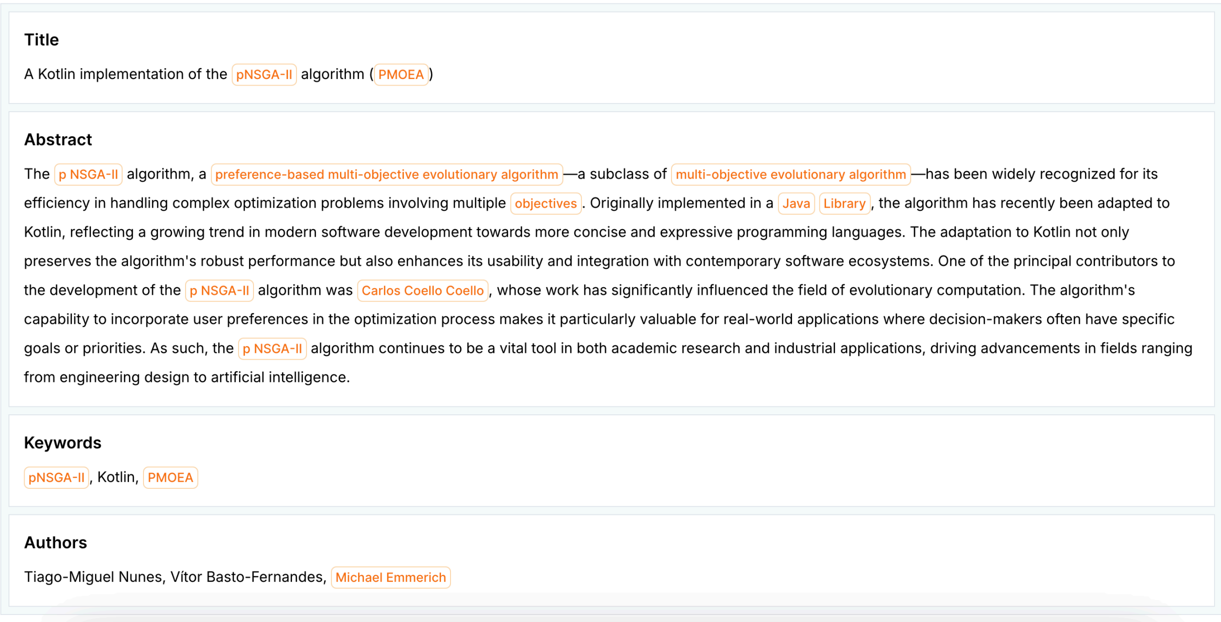


Figure 15 – MyCODA submitted article section

The searching of existing terms from text functionality was inspired from the Annotator feature from BioPortal, where the user can get annotations for biomedical text with classes from the ontologies in the library, which is available in https://bioportal.bioontology.org/annotator.

This feature works by splitting the texts in words, taking all combinations of grouped words and matching these strings with properties of entities stored in the ontology. These properties are label (*owl:label*), synonyms (*altLabel*), partial IRI (the ending of the IRI, after the "#" character), description (*owl:comment*), lenient label (the label disregarding any special characters) and lenient description (the description disregarding any special characters). If multiple entities are matched in an overlapping segment of the text, then the entity is chosen based on a priority setting. The entity that matches the group with the highest number of words takes the highest priority. After this calculation, if there is still a draw in priority, then the entity is chosen based on which property of the entity was matched, in which the priority follows the order by which these properties were previously described, label having the highest priority and lenient description having the lowest priority. There may be some cases where this algorithm will not work correctly. For this reason, the submissions are all recorded in the internal database, so we may check then for debugging purposes based on the feedback provided by the users. A more robust algorithm could later be incorporated into this feature, for example, a Natural Language Processing model, to detect the entities referenced in the text.

The user can then gather information about the current knowledge by clicking on any identified terms, which opens the Entity Preview popup, displayed in Figure 16, where they can navigate the same interface found in the Browse tab.

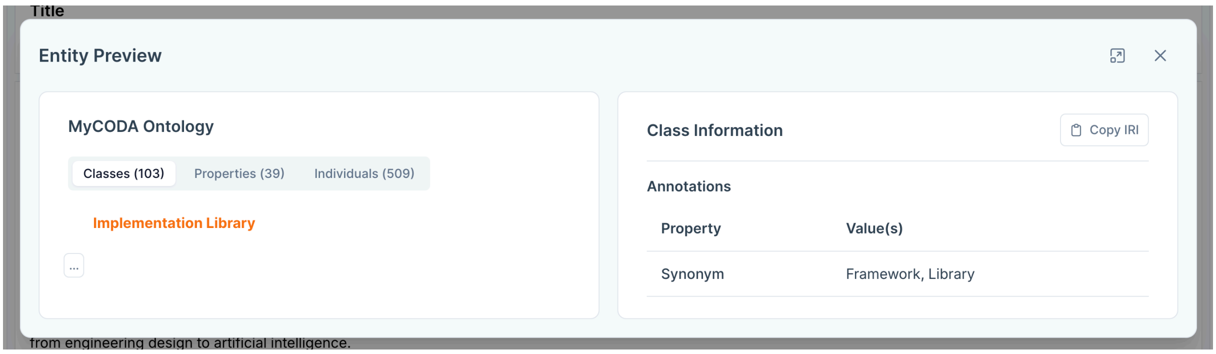


Figure 16 – MyCODA entity preview

The user can use these resources to understand how the knowledge around their topics is structured, and what information is missing.

4.4.2.2. Identified terms section

This section, displayed in Figure 17, contains a table that shows known terms referenced in the article, as well as terms the user has added or edited.

Identified Terms ?

Context	Type	Term	Actions
Referenced	Individual	p NSGA-II	Edit Browse
Referenced	Class	Preference-based Multi-Objective Evolutionary Algorithm	Edit Browse
Referenced	Class	Multi-Objective Evolutionary Algorithm	Edit Browse
Referenced	Class	Objectives	Edit Browse
Referenced	Individual	Java	Edit Browse
Referenced	Class	Implementation Library	Edit Browse
Referenced	Individual	Carlos Coello Coello	Edit Browse
Referenced	Individual	Michael Emmerich	Edit Browse
Added	Individual	New A Kotlin implementation of the pNSGA-II algorithm (PMOEA)	Edit Remove

Figure 17 – MyCODA identified terms section

Added terms are preceded by a badge indicating that term is new. Clicking new terms opens the popup to edit the term, while clicking on existing terms opens the Entity Preview popup.

Using the buttons in the Actions column, the user can edit or browse existing terms, and they can edit or remove added terms.

After the article is submitted, a new term is created automatically, an individual of type Article. The label of this term is the title of the article, and the keywords, authors, reference and DOI provided in the form are added as properties of the individual.

Clicking on Edit on this term opens the popup presented in Figure 18, displaying information about the newly added article, and allowing the user to edit any information.

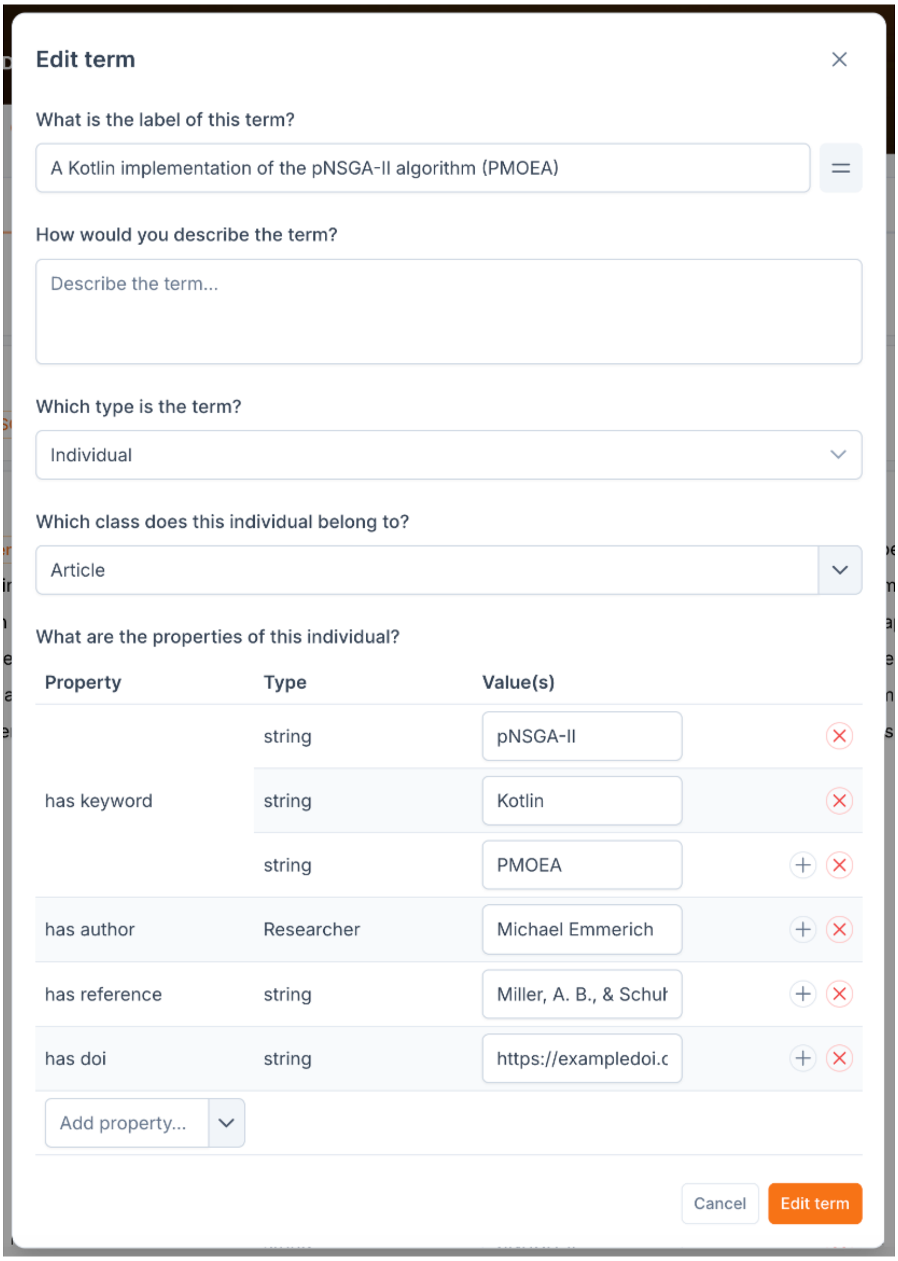


Figure 18 – MyCODA edit term view

4.4.2.3. Contribute section

The contribute section, displayed in Figure 19, consists of a button to add a new term, and a text box to select and edit an existing term.

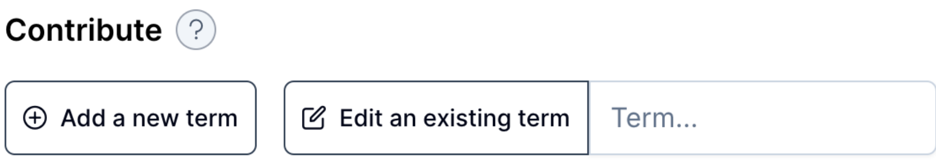


Figure 19 – MyCODA contribute section

Clicking in the Add a new term button opens the popup, displayed in Figure 20, asking information about the new term to introduce.

Figure 20 – MyCODA add new term popup

Firstly, the user is prompted to provide the label of the new term. They may also click the equal sign ('=') button to expand a table which enables them to add multiple synonyms for the term. This prompt is displayed in Figure 21.

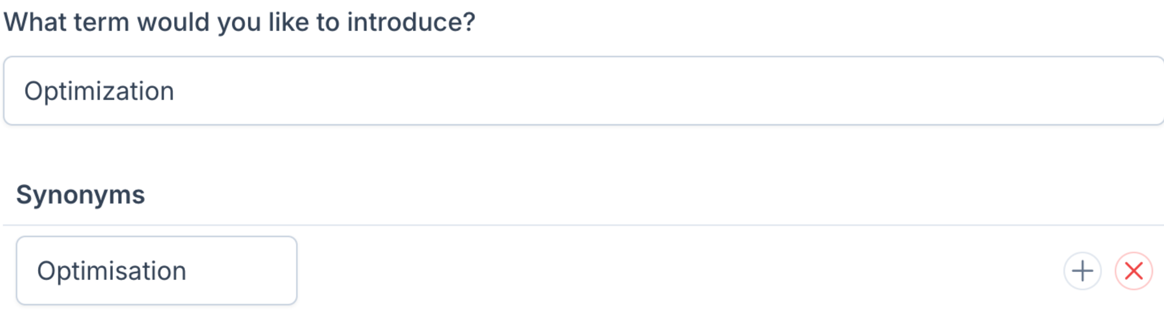


Figure 21 – MyCODA new term synonyms

After a label is provided, a search for suggested synonyms of the term being added is executed, presented in Figure 22, with the goal of avoiding the creation of duplicate terms.

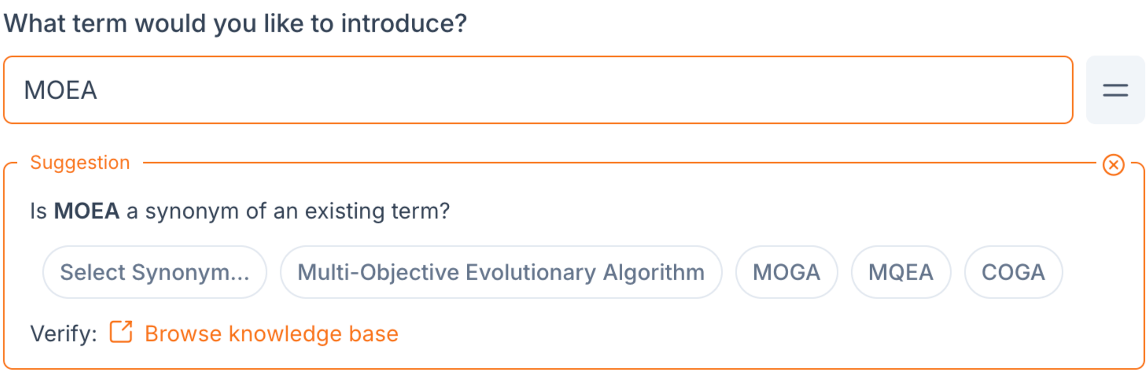


Figure 22 – MyCODA new term synonym suggestions

The terms suggested are found based on whether the new term label is close enough to an existing term's label or to existing synonyms, comparing using a hamming distance algorithm (with a hamming distance threshold of half of the length of the string provided), whether the label matches the initials of an existing term's label (to find acronyms in the ontology that could apply to the string provided), or whether an existing term's label matches the initials of the new term label (to find entities in the ontology that could be associated to an acronym provided).

The user may also click on the "Select Synonym…" button to manually input the existing term which this term would be a duplicate of.

After selecting an existing term from this suggestion box, the user is prompted to confirm whether they want to add the label provided as a synonym of the existing term. This prompt is displayed in Figure 23.

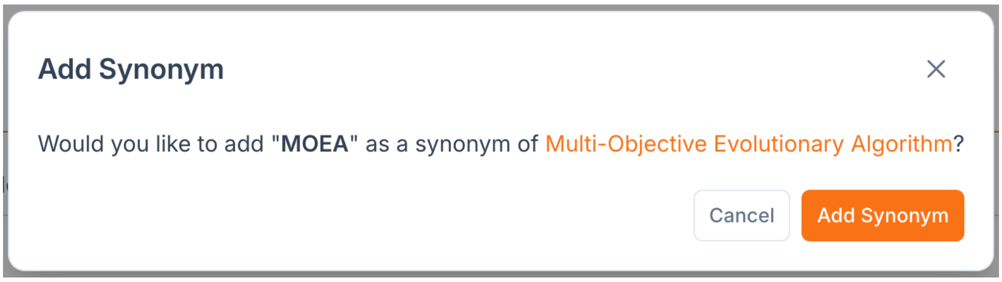


Figure 23 – MyCODA add synonym to existing entity

If the user accepts the prompt, they are redirected to the edit view of the existing term, with a new synonym added, corresponding to the label of the new term that was being added.

Otherwise, if the user chooses to continue adding the new term, they are prompted to describe the term and select the type of the term, as displayed in Figure 24.

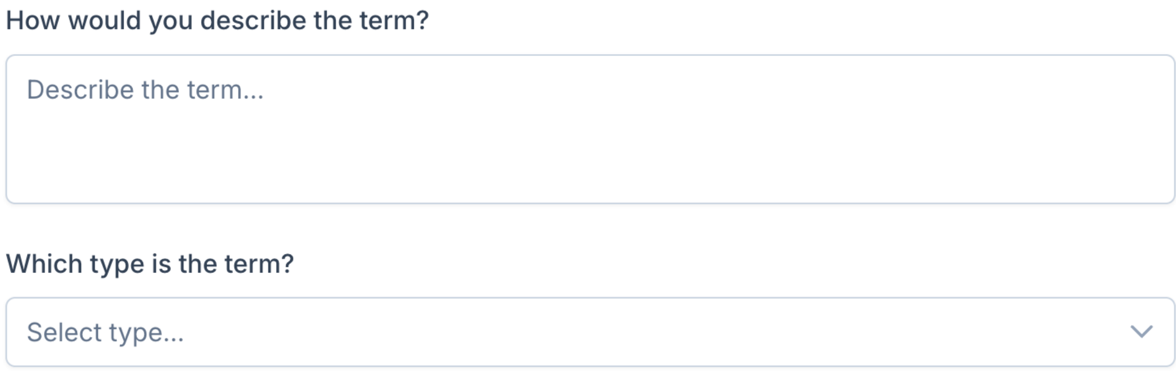


Figure 24 – MyCODA new term description and type prompt

The description is equivalent to the *owl:comment* annotation property of the resulting entity.

The type of term is one of the following:

- Class: A category of things.
- Individual: An instance of a certain class.
- Property: An attribute, or characteristic of something.

After the user provides the type, a new set of questions will follow.

If the term is a class, the user is asked to specify if the class has any super class, as displayed in Figure 25.

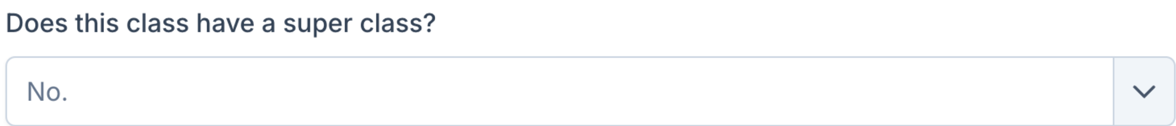


Figure 25 – MyCODA new term super class prompt

If the term is an individual, the user is asked to specify which class the individual belongs to, and which properties they want to assign to the individual, as displayed in Figure 26.

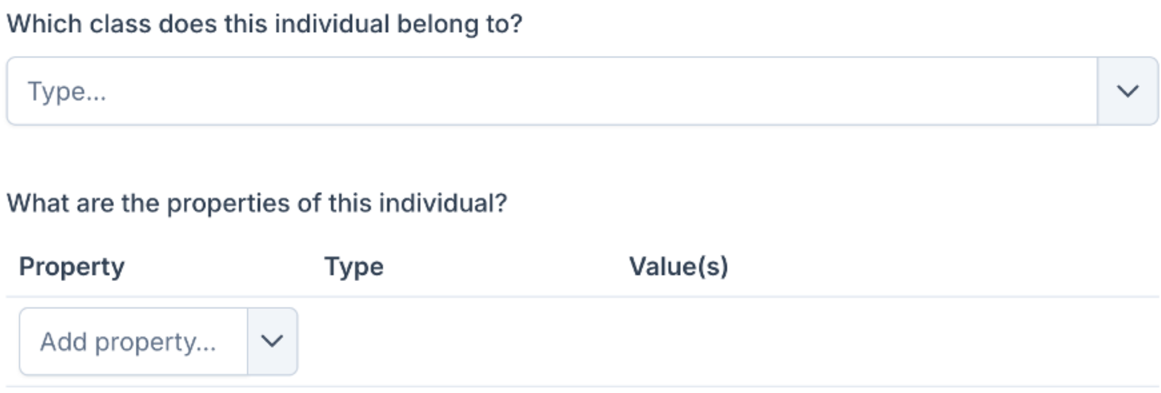


Figure 26 – MyCODA new individual information prompt

If the term is a property, the user is asked to specify which type of individuals may have this property, and which type is the value of the property, as displayed in Figure 27.

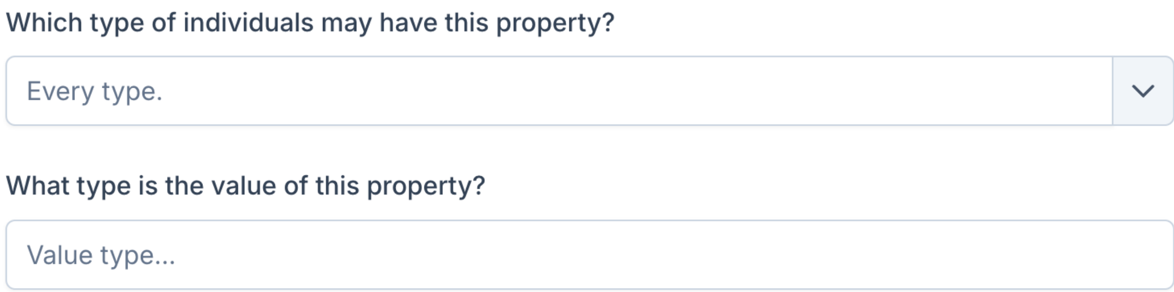


Figure 27 – MyCODA new property information prompt

After the user provides enough information and adds the term, the new term is added to the identified terms table and may be edited or removed at any point.

4.4.2.4. Finishing the contribution

Once the user decides that they have concluded their contribution, and they consent to the storage and sharing of all the information provided above, with the understanding that it will be publicly accessible in the MyCODA Knowledge Base, they may submit the contribution, where they are presented with the text displayed in Figure 28.

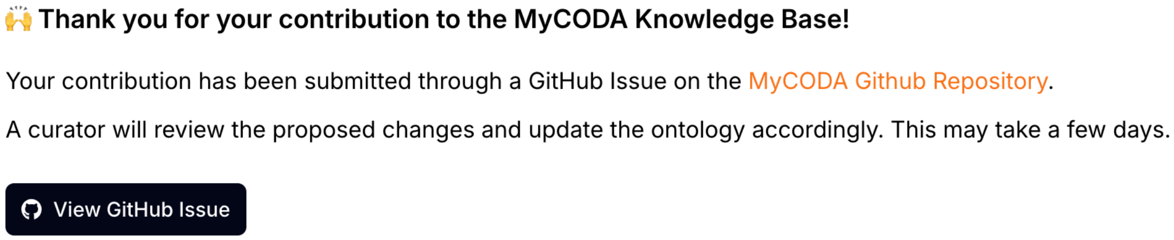


Figure 28 – MyCODA contribution completion

At this step, a GitHub issue is generated, listing all the additions and editions of entities proposed, and a button is displayed, linking the user to the resulting issue. An example of a GitHub issue resulting from an article submission is displayed in Figure 39.

The user is informed that the issue will be reviewed by the curators at their convenience.

Finally, a survey is displayed, where the user is asked to provide feedback, and answer the 10 questions from the System Usability Scale, with options ranging from 1 to 5. Answering this survey is optional. This is displayed in Figure 29 (feedback with SUS form collapsed) and Figure 38 (SUS form expanded).

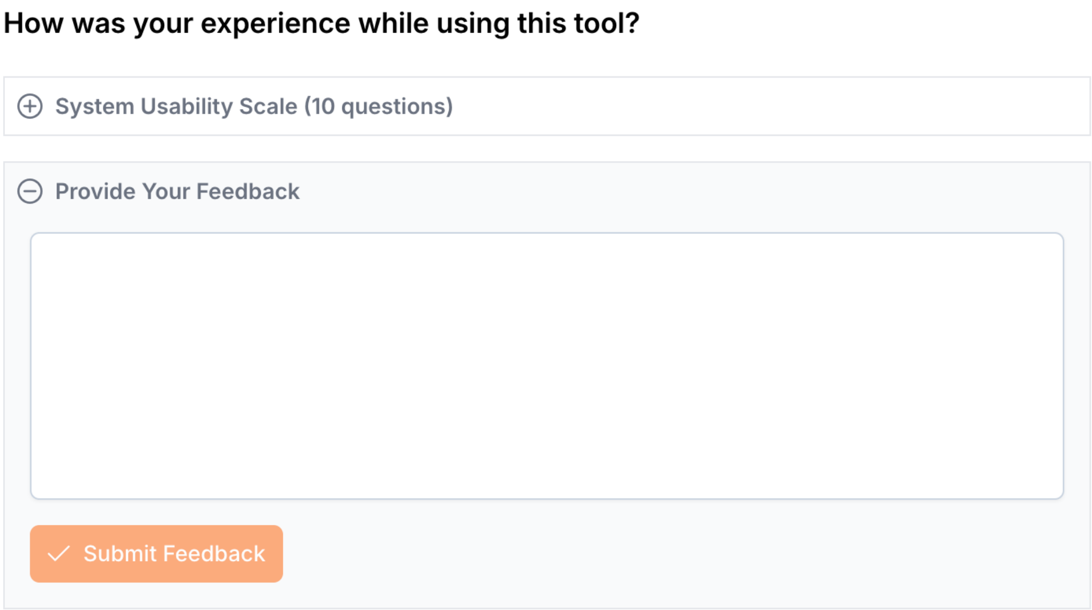


Figure 29 – MyCODA feedback after contribution

4.5. Search bar

The platform provides a search functionality, displayed in Figure 30, where the user can search for a term, by label or synonym. The results are a combination of entities with labels starting with the query string first, and entities with labels or synonyms containing the query string last.

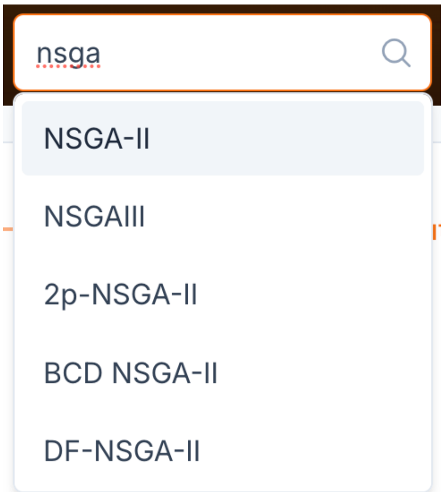


Figure 30 – MyCODA search bar

Clicking on a result navigates the user to the Browse tab, with the corresponding entity selected.

4.6. Curating contributions

Contributions are ontology changes proposed and logged in the repository's GitHub issues. These changes need to be curated by assigned curators, which should typically be domain experts, before they are applied to the ontology.

Curators are automatically notified whenever a new contribution is submitted and may curate contributions at their convenience.

4.6.1. Curation Process

Curating contributions is a simple process which typically consists of:

1. looking through the specified details of the proposed change described in the corresponding GitHub issue,
2. loading into the Protégé editor the OWL ontology file available here,
3. applying changes using the editor and saving changes to the file,
4. commiting and pushing the changes to the MyCODA GitHub repository.

4.6.2. Force ontology update

The MyCODA server service restarts daily at 4 a.m., which is defined in the *systemd timer* configuration here. Upon restarting, the server fetches the latest ontology in the repository, and the platform is updated accordingly.

However, the curators may want to update the platform immediately to reflect a new version of the ontology that was pushed to the repository. In this case, they may access the curator page, located at https://mycoda.iscte-iul.pt/curator, displayed in Figure 31, where they can force a live reload of the ontology, by providing a password that is distributed to curators. This password is stored securely on the server, using the *SHA-256* hashing algorithm with a salt string.

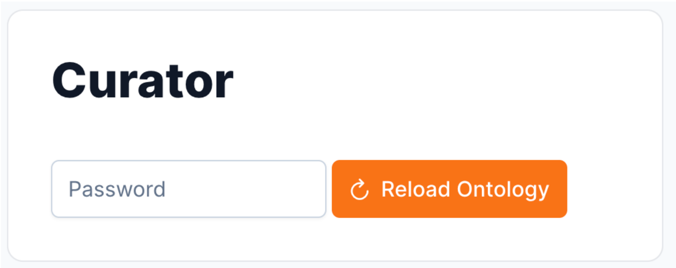


Figure 31 – MyCODA curator reload ontology tool

CHAPTER 5

Validation

This chapter describes the methods prepared to evaluate the effectiveness of the platform and to validate the implemented solution.

5.1. Bi-weekly presentations and discussions

Every two weeks, a meeting took place to present the current state of the platform, and to get feedback from the domain expert Professor Michael Emmerich, a Professor in Multi-Objective Optimization Faculty of Information Technology University of Jyväskylä Finland, organizer of the MACODA workshop, and co-supervisor of this dissertation. This corresponds to the demonstration phase in the DSR methodology.

This recurrent feedback shaped the development of the platform, and served to validate the functionalities and solutions with a respected member of the MACODA community.

5.2. Validation tests

Validation tests are designed to ensure that the platform meets the needs and requirements of its users. Validation tests answer the question, "Are we building the right product?", focusing on how well the product aligns with end-user expectations and intended purpose rather than just checking for functional correctness.

With this end, a questionnaire is present at the end of a contribution submission. This questionnaire consists of a System Usability Scale survey, and an open-ended feedback text box. The answering of this survey is optional.

5.3. SUS feedback evaluation

The System Usability Scale (SUS) (Brooke, 1995) is a widely used tool for assessing the usability of systems and interfaces through a standardized questionnaire. Developed by John Brooke in 1986, SUS comprises 10 items rated on a five-point Likert scale, between ‘Strongly disagree’ (equivaling 1 point) and ‘Strongly agree’ (equivaling 5 points), generating a single usability score between 0 and 100 that reflects the overall user experience. This score can be calculated by taking the sum of points from all the answers and multiplying the result by 2.5. Each item alternates between positive and negative statements, and respondents’ answers are converted into scores to indicate the ease or difficulty of using the system in question.

SUS is notable for its versatility and effectiveness across various domains, as it can assess everything from software applications to websites and even hardware. Research has consistently supported the reliability of SUS as a measure of usability, showing its ability to produce consistent results that correlate well with user satisfaction metrics (Sauro & Lewis, 2011). Furthermore, SUS has been validated in multiple settings as an efficient tool for both early-stage and summative usability testing, making it highly adaptable to iterative design processes (Bangor, Kortum, & Miller, 2008).

In practice, SUS scores are frequently interpreted using average benchmarks and grading scales to contextualize the user experience. Studies suggest that a score above 68 is generally considered above average (Bangor et al., 2009).

Following are the 10 statements asked to rate from 1 ('Strongly disagree') to 5 ('Strongly agree') in this survey:

1. I think that I would like to use this system frequently.
2. I found the system unnecessarily complex.
3. I thought the system was easy to use.
4. I think that I would need the support of a technical person to be able to use this system.
5. I found the various functions in this system were well integrated.
6. I thought there was too much inconsistency in this system.
7. I would imagine that most people would learn to use this system very quickly.
8. I found the system very cumbersome to use.
9. I felt very confident using the system.
10. I needed to learn a lot of things before I could get going with this system.

Answers to the SUS form are saved in the server's MySQL database, linked to the information provided in the first step of the contribution process (described in section 4.4.2), and to the GitHub issue link resulting from the contribution. These answers are at no point stored in the ontology.

So far, only two answers to the SUS form have been submitted, so it's not yet possible to gather any conclusions from this. However, the feedback form and SUS framework are present and ready on the platform, as displayed in Figure 38, and we expect to get a significant number of answers once we start getting a meaningful number of contributions to the knowledge base. This is more thoroughly expressed in the section 6.2 of the dissertation (Ongoing/Future work).

5.4. Open ended feedback

Unlike fixed-response questionnaires, open-ended feedback allows participants to articulate their thoughts in their own words, capturing the nuances of user sentiment and unmet needs that may not emerge in closed-question formats (Nielsen, 1994).

Studies suggest that open-ended responses are crucial for understanding contextual factors influencing user behavior, such as specific frustrations or unexpected positive aspects, and can reveal themes that are essential for iterative design processes (Kujala, 2003). This feedback often complements quantitative metrics by providing richer context, especially in areas where user emotions, preferences, and frustrations require interpretation beyond numerical scores (Patton, 2014).

Open-ended feedback may be provided using the text box labeled "Provide Your Feedback" at the final step of a contribution submission–in which case the feedback is saved in the server's MySQL database, and linked to the contribution–, or via e-mail to macodaclub@gmail.com.

CHAPTER 6

Conclusion

6.1. Conclusions

This dissertation presents the development of the MyCODA platform, a web-based application to manage an ontology with knowledge in the domain of MACODA, to support the scientific community, taking into consideration the main factors that make other ontologies successful, to maximize the utility of this technology. For this purpose, two research questions are proposed, Q1 and Q2.

Regarding Q1, "How can we maximize the utility of ontologies within the field of MACODA research?", we delve into the existing literature, and we look at existing solutions that have been considerably successful in being adopted by the respective scientific communities, such as the Uberon ontology–integrated in the OLS platform– and the Gene Ontology–supported by the AmiGO platform. We conclude that the existence of a platform to support the community and to make it easier to visualize and leverage the ontology is essential. Another key factor for the success of a community-driven knowledge base is the existence of a clear framework for contribution and maintenance of the ontology.

Regarding Q2, "Does the software proposal resulting from the MACODA Workshop in the University of Leiden in 2019, and described in the book chapter (Basto-Fernandes et al., 2023), identify all the MACODA research community knowledge management needs?", we conclude that this software has great potential in aiding the research community, though the aim in developing specific functionalities has shifted in accordance to the literature review and feedback provided. For example, the focus on user registration and on the creation of a forum has lessened, in favor of tackling what are considered more important factors in the success of the platform and the ontology behind it, such as the refinement of a visualization and navigation system to easily browse the ontology, and the creation of a framework and tools to contribute to the knowledge base, and curate the resulting contributions.

The outputs produced from the work undertaken are the MyCODA platform artifact, available at https://mycoda.iscte-iul.pt/ and this dissertation document, which provides a thorough description of the platform and its development process, and a framework for contributing and maintaining the MyCODA platform and the MyCODA ontology.

6.2. Ongoing/Future work

The artifact has recently been deployed in the ISCTE infrastructure, on the 25th of October 2024.

Professor Michael Emmerich has been submitting articles to the contribution tool, and this effort is planned to continue. So far, four articles have been submitted, and over thirty entity additions or changes have been proposed resulting from the usage of this tool. These can be found here.

The following action planned is to communicate this with the organizers of the EMO 2025 conference, and potentially ask for authors of papers submitted to the conference, related to the MACODA research domain, to use the contribution tool developed in the MyCODA platform to propose the addition of new knowledge into the ontology. Authors may provide useful feedback, and the usage statistics may be analyzed to better assess the validity and effectiveness of the platform, as well as to prepare it for further improvements.

Another component of this platform aimed for future work is the Query tab, where the user may perform queries to the Knowledge Base, using SQWRL. This feature idea is presented in the chapter 13 of '*Many-Criteria Optimisation and Decision Analysis Book*', and an iteration of this feature was developed in the accompanying prototype artifact.

Bibliography

Aldana-Martín, J. F., Roldán-García, M. del M., Nebro, A. J., & Aldana-Montes, J. F. (2024). MOODY: An ontology-driven framework for standardizing multi-objective evolutionary algorithms. *Information Sciences*, *661*, 120184. https://doi.org/10.1016/j.ins.2024.120184

Angles, R., & Gutierrez, C. (2008). The Expressive Power of SPARQL. Em A. Sheth, S. Staab, M. Dean, M. Paolucci, D. Maynard, T. Finin, & K. Thirunarayan (Eds.), *The Semantic Web—ISWC 2008* (pp. 114–129). Springer. https://doi.org/10.1007/978-3-540-88564-1_8

Antoniou, G., & Harmelen, F. (2003). Web ontology language: OWL. *Handbook on Ontologies*. https://doi.org/10.1007/978-3-540-92673-3_4

Ashburner, M., Ball, C. A., Blake, J. A., Botstein, D., Butler, H., Cherry, J. M., Davis, A. P., Dolinski, K., Dwight, S. S., Eppig, J. T., Harris, M. A., Hill, D. P., Issel-Tarver, L., Kasarskis, A., Lewis, S., Matese, J. C., Richardson, J. E., Ringwald, M., Rubin, G. M., & Sherlock, G. (2000). Gene Ontology: Tool for the unification of biology. *Nature Genetics*, *25*(1), 25–29. https://doi.org/10.1038/75556

Bangor, A., Kortum, P., & Miller, J. (2009). Determining What Individual SUS Scores Mean: Adding an Adjective Rating Scale. *J. Usability Stud.*, *4*, 114–123.

Basto-Fernandes, V., Salvador, D., Yevseyeva, I., & Emmerich, M. (2023). Many-Criteria Optimisation and Decision Analysis Ontology and Knowledge Management. Em D. Brockhoff, M. Emmerich, B. Naujoks, & R. Purshouse (Eds.), *Many-Criteria Optimization and Decision Analysis: State-of-the-Art, Present Challenges, and Future Perspectives* (pp. 337–354). Springer International Publishing. https://doi.org/10.1007/978-3-031-25263-1_13

Berners-Lee, T. (2002). *The Semantic Web: A New Form of Web Content that is Meaningful to Computers Will Unleash a Revolution of New Possibilities*. Scientific American.

Borgman, C. L. (2015). *Big Data, Little Data, No Data: Scholarship in the Networked World*. The MIT Press. https://doi.org/10.7551/mitpress/9963.001.0001

Borgo, S., Ferrario, R., Gangemi, A., Guarino, N., Masolo, C., Porello, D., Sanfilippo, E. M., & Vieu, L. (2022). DOLCE: A Descriptive Ontology for Linguistic and Cognitive Engineering. *Applied Ontology*, *17*(1), 45–69. https://doi.org/10.3233/AO-210259

Brooke, J. (1995). SUS: A quick and dirty usability scale. *Usability Eval. Ind.*, *189*.

Carbon, S., Ireland, A., Mungall, C. J., Shu, S., Marshall, B., Lewis, S., the AmiGO Hub, & the Web Presence Working Group. (2009). AmiGO: Online access to ontology and annotation data. *Bioinformatics*, *25*(2), 288–289. https://doi.org/10.1093/bioinformatics/btn615

Carstensen, A.-K., & Bernhard, J. (2019). Design science research – a powerful tool for improving methods in engineering education research. *European Journal of Engineering Education*, *44*(1–2), 85–102. https://doi.org/10.1080/03043797.2018.1498459

Côté, R., Jones, P., Apweiler, R., & Hermjakob, H. (2006). The ONTOLOGY lookup service, a lightweight cross-platform tool for controlled vocabulary queries. *BMC bioinformatics*, *7*, 97. https://doi.org/10.1186/1471-2105-7-97

Côté, R., Reisinger, F., Martens, L., Barsnes, H., Vizcaino, J. A., & Hermjakob, H. (2010). The Ontology Lookup Service: Bigger and better. *Nucleic Acids Research*, *38*(Web Server issue), W155-160. https://doi.org/10.1093/nar/gkq331

Docker Inc. (2024). *Docker* (Versão 24.0) [Software]. Docker Inc. https://www.docker.com

Elhara, O., Varelas, K., Nguyen, D., Tusar, T., Brockhoff, D., Hansen, N., & Auger, A. (2019). *COCO: The Large Scale Black-Box Optimization Benchmarking (bbob-largescale) Test Suite* (No. arXiv:1903.06396). arXiv. https://doi.org/10.48550/arXiv.1903.06396

Erdmann, M., & Waterfeld, W. (2012). Overview of the NeOn Toolkit. Em M. C. Suárez-Figueroa, A. Gómez-Pérez, E. Motta, & A. Gangemi (Eds.), *Ontology Engineering in a Networked World* (pp. 281–301). Springer. https://doi.org/10.1007/978-3-642-24794-1_13

Fielding, R. T. (2000). Architectural Styles and the Design of Network-based Software Architectures. *Chapter 2: Network-Based Application Architectures, University of California, Irvine*.

Gennari, J. H., Musen, M. A., Fergerson, R. W., Grosso, W. E., Crubézy, M., Eriksson, H., Noy, N. F., & Tu, S. W. (2003). The evolution of Protégé: An environment for knowledge-based systems development. *International Journal of Human-Computer Studies*, *58*(1), 89–123. https://doi.org/10.1016/S1071-5819(02)00127-1

Golbreich, C., Horridge, M., Horrocks, I., Motik, B., & Shearer, R. (2007). OBO and OWL: Leveraging Semantic Web Technologies for the Life Sciences. Em K. Aberer, K.-S. Choi, N. Noy, D. Allemang, K.-I. Lee, L. Nixon, J. Golbeck, P. Mika, D. Maynard, R. Mizoguchi, G. Schreiber, & P. Cudré-Mauroux (Eds.), *The Semantic Web* (pp. 169–182). Springer. https://doi.org/10.1007/978-3-540-76298-0_13

Grau, B. C., Horrocks, I., Motik, B., Parsia, B., Patel-Schneider, P., & Sattler, U. (2008). OWL 2: The next step for OWL. *Journal of Web Semantics*, *6*(4), 309–322. https://doi.org/10.1016/j.websem.2008.05.001

Hevner, A., R, A., March, S., T, S., Park, Park, J., Ram, & Sudha. (2004). Design Science in Information Systems Research. *Management Information Systems Quarterly*, *28*, 75.

Horridge, M., & Bechhofer, S. (2011). The OWL API: A Java API for OWL ontologies. *Semantic Web*, *2*(1), 11–21. https://doi.org/10.3233/SW-2011-0025

Keet, C. M., Ławrynowicz, A., d'Amato, C., Kalousis, A., Nguyen, P., Palma, R., Stevens, R., & Hilario, M. (2015). The Data Mining OPtimization Ontology. *Journal of Web Semantics*, *32*, 43–53. https://doi.org/10.1016/j.websem.2015.01.001

Kostovska, A., Vermetten, D., Doerr, C., Džeroski, S., Panov, P., & Eftimov, T. (2022). *OPTION: OPTImization Algorithm Benchmarking ONtology* (No. arXiv:2211.11332). arXiv. https://doi.org/10.48550/arXiv.2211.11332

Kujala, S. (2003). User involvement: A review of the benefits and challenges. *Behaviour & Information Technology*, *22*(1), 1–16. https://doi.org/10.1080/01449290301782

Li, L., Yevseyeva, I., Basto-Fernandes, V., Trautmann, H., Jing, N., & Emmerich, M. (2017). Building and Using an Ontology of Preference-Based Multiobjective Evolutionary Algorithms. Em H. Trautmann, G. Rudolph, K. Klamroth, O. Schütze, M. Wiecek, Y. Jin, & C. Grimme (Eds.), *Evolutionary Multi-Criterion Optimization* (pp. 406–421). Springer International Publishing. https://doi.org/10.1007/978-3-319-54157-0_28

Micronaut Foundation. (2024). *Micronaut* (Versão 4.7.3) [Software]. Micronaut Foundation. https://micronaut.io

Mungall, C. J., Torniai, C., Gkoutos, G. V., Lewis, S. E., & Haendel, M. A. (2012). Uberon, an integrative multi-species anatomy ontology. *Genome Biology*, *13*(1), R5. https://doi.org/10.1186/gb-2012-13-1-r5

Musen, M. A. (2015). The Protégé Project: A Look Back and a Look Forward. *AI matters*, *1*(4), 4–12. https://doi.org/10.1145/2757001.2757003

Nielsen, J. (1994). *Usability Engineering*. Morgan Kaufmann.

Noy, N. F., Shah, N. H., Whetzel, P. L., Dai, B., Dorf, M., Griffith, N., Jonquet, C., Rubin, D. L., Storey, M.-A., Chute, C. G., & Musen, M. A. (2009). BioPortal: Ontologies and integrated data resources at the click of a mouse. *Nucleic Acids Research*, *37*(Web Server issue), W170–W173. https://doi.org/10.1093/nar/gkp440

Noy, N., & Mcguinness, D. (2001). Ontology Development 101: A Guide to Creating Your First Ontology. *Knowledge Systems Laboratory*, *32*.

O'Connor, M., & Das, A. (2009, janeiro 1). *SQWRL: A query language for OWL*.

O'Connor, M., Shankar, R., Musen, M., Das, A., & Nyulas, C. (2008). *The SWRLAPI: A Development Environment for Working with SWRL Rules.*

Oracle Corporation. (2024). *MySQL* (Versão 8.0) [Software]. Oracle Corporation. https://www.mysql.com

Panzarella, G., Veltri, P., & Alcaro, S. (2023). Using ontologies for life science text-based resource organization. *Artificial Intelligence in the Life Sciences*, *3*, 100059. https://doi.org/10.1016/j.ailsci.2023.100059

Patton, M. Q. (2014). *Qualitative Research & Evaluation Methods: Integrating Theory and Practice*. SAGE Publications.

Peffers, K., Tuunanen, T., Gengler, C., & Rossi, M. (2006). The design science research process: A model for producing and presenting information systems research. *Proceedings Design Research Information Systems and Technology DESRIST'06*, *24*.

Scarbrough, H., Swan, J., & Preston, J. (1999). *Knowledge Management: A Literature Review*. Institute of Personnel and Development.

Schober, D., Kusnierczyk, W., Lewis, S., Lomax, J., Mungall, C., Rocca-Serra, P., & Sansone, S.-A. (2007, janeiro 1). *Towards naming conventions for use in controlled vocabulary and ontology engineering*. ISMB/ECCB Special Interest Group (SIG) Meeting Program Materials, Bio-Ontologies SIG Workshop Vienna, Austria.

Shearer, R., Motik, B., & Horrocks, I. (2008). *HermiT: A Highly-Efficient OWL Reasoner*. OWL: Experiences and Directions. https://www.semanticscholar.org/paper/HermiT%3A-A-Highly-Efficient-OWL-Reasoner-Shearer-Motik/96286cb50a109e50fad743570184ea1fa55f2ba9

Sirin, E., Parsia, B., Grau, B. C., Kalyanpur, A., & Katz, Y. (2007). Pellet: A practical OWL-DL reasoner. *Journal of Web Semantics*, *5*(2), 51–53. https://doi.org/10.1016/j.websem.2007.03.004

Smith, B. (2004). Ontology. Em *The Blackwell Guide to the Philosophy of Computing and Information* (pp. 153–166). John Wiley & Sons, Ltd. https://doi.org/10.1002/9780470757017.ch11

Spring. (2024). *Spring Framework* (Versão 6.2) [Software]. VMware. https://spring.io/projects/spring-boot

Studer, R., Benjamins, V. R., & Fensel, D. (1998). Knowledge engineering: Principles and methods. *Data & Knowledge Engineering*, *25*(1), 161–197. https://doi.org/10.1016/S0169-023X(97)00056-6

The Gene Ontology Consortium, Aleksander, S. A., Balhoff, J., Carbon, S., Cherry, J. M., Drabkin, H. J., Ebert, D., Feuermann, M., Gaudet, P., Harris, N. L., Hill, D. P., Lee, R., Mi, H., Moxon, S., Mungall, C. J., Muruganugan, A., Mushayahama, T., Sternberg, P. W., Thomas, P. D., … Westerfield, M. (2023). The Gene Ontology knowledgebase in 2023. *Genetics*, *224*(1), iyad031. https://doi.org/10.1093/genetics/iyad031

Tirmizi, S. H., Aitken, S., Moreira, D. A., Mungall, C., Sequeda, J., Shah, N. H., & Miranker, D. P. (2011). Mapping between the OBO and OWL ontology languages. *Journal of Biomedical Semantics*, *2*(Suppl 1), S3. https://doi.org/10.1186/2041-1480-2-S1-S3

Tudorache, T., Nyulas, C., Noy, N. F., & Musen, M. A. (2013). WebProtégé: A Collaborative Ontology Editor and Knowledge Acquisition Tool for the Web. *Semantic web*, *4*(1), 89–99. https://doi.org/10.3233/SW-2012-0057

Venable, J., Pries-Heje, J., & Baskerville, R. (2016). FEDS: A Framework for Evaluation in Design Science Research. *European Journal of Information Systems*, *25*(1), 77–89. https://doi.org/10.1057/ejis.2014.36

Vite Core Team. (2024). *Vite* (Versão 4.1) [Software]. Vite. https://vitejs.dev

Vue.js Core Team. (2024). *Vue.js* (Versão 3.3) [Software]. Vue.js. https://vuejs.org

Walker, A. (2011). A Wiki for Business Rules in Open Vocabulary, Executable English. *Computing Research Repository - CORR*.

Weiten, M. (2009). OntoSTUDIO® as a Ontology Engineering Environment. Em J. Davies, M. Grobelnik, & D. Mladenić (Eds.), *Semantic Knowledge Management: Integrating Ontology Management, Knowledge Discovery, and Human Language Technologies* (pp. 51–60). Springer. https://doi.org/10.1007/978-3-540-88845-1_5

Yang, S.-Y. (2009). OntoPortal: An ontology-supported portal architecture with linguistically enhanced and focused crawler technologies. *Expert Systems with Applications*, *36*(6), 10148–10157. https://doi.org/10.1016/j.eswa.2009.01.004

Attachments

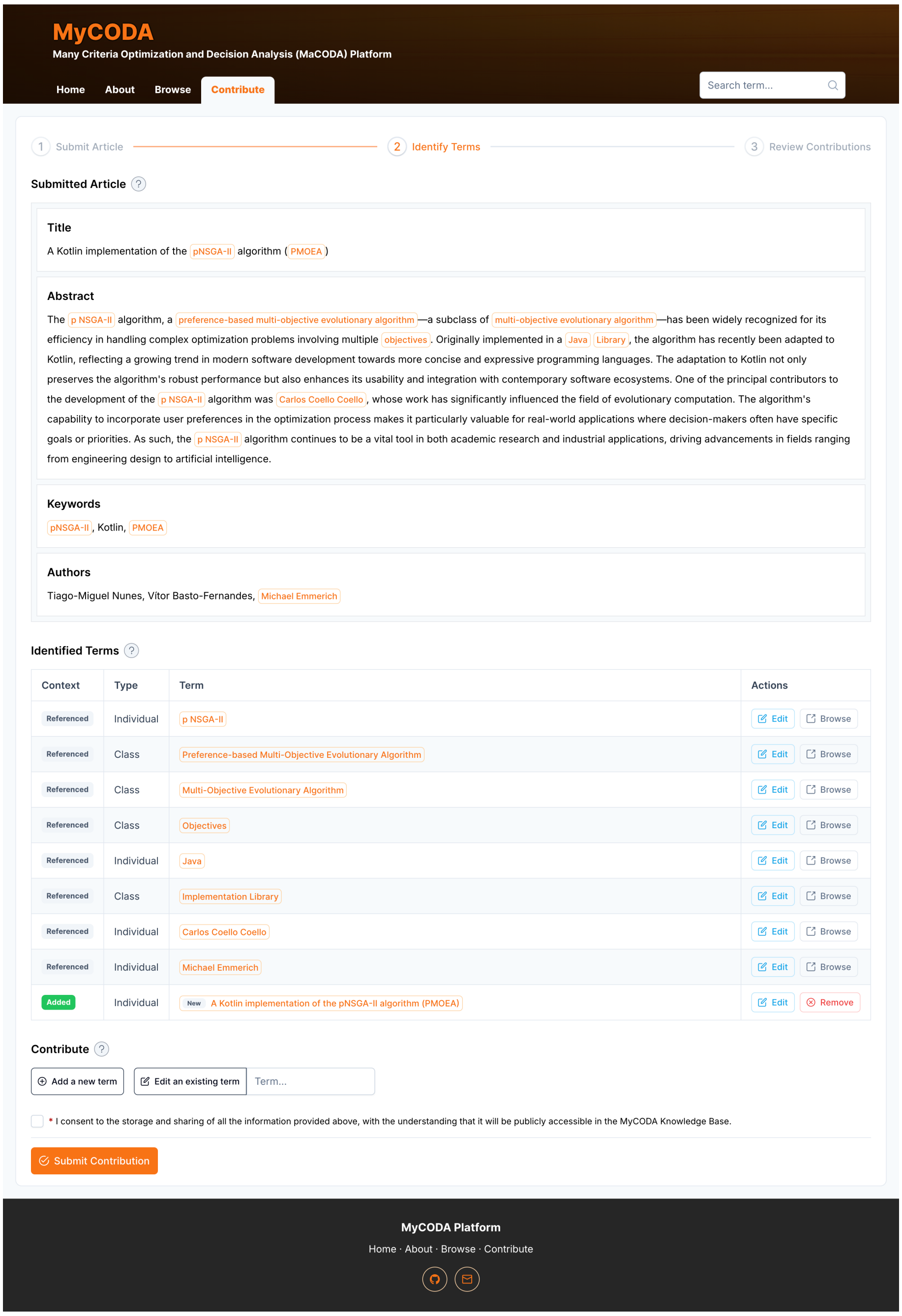


Figure 32 – MyCODA full article contribution page

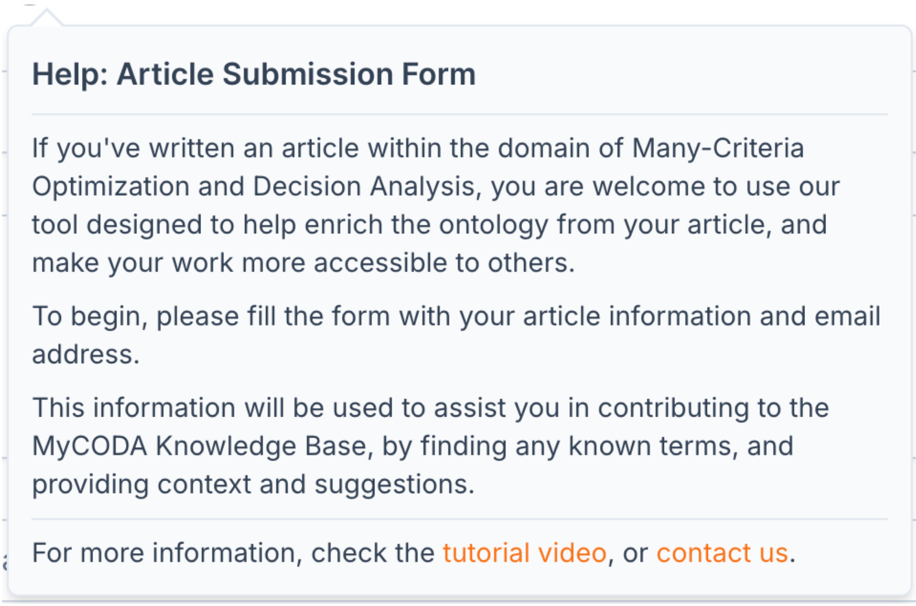


Figure 33 – MyCODA article submission form help tooltip

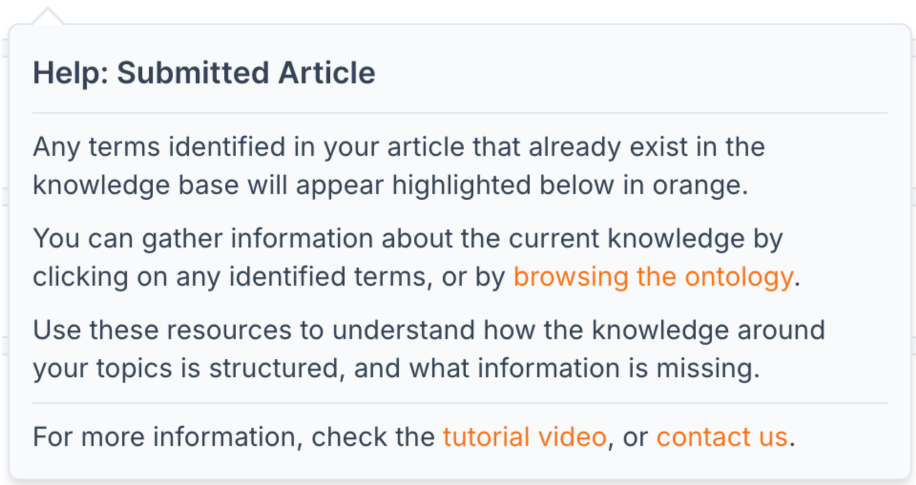


Figure 34 – MyCODA submitted article help tooltip

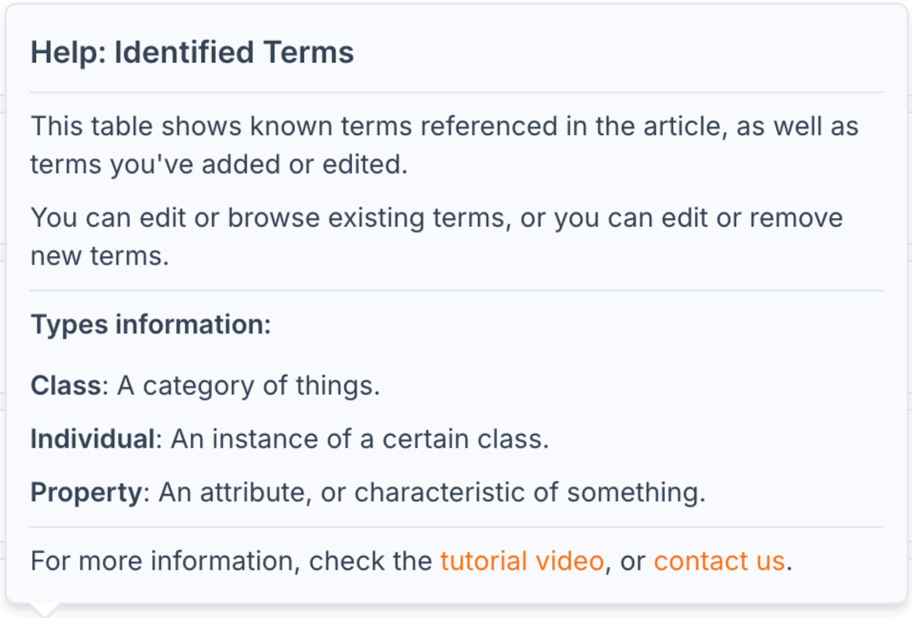


Figure 35 – MyCODA identified terms help tooltip

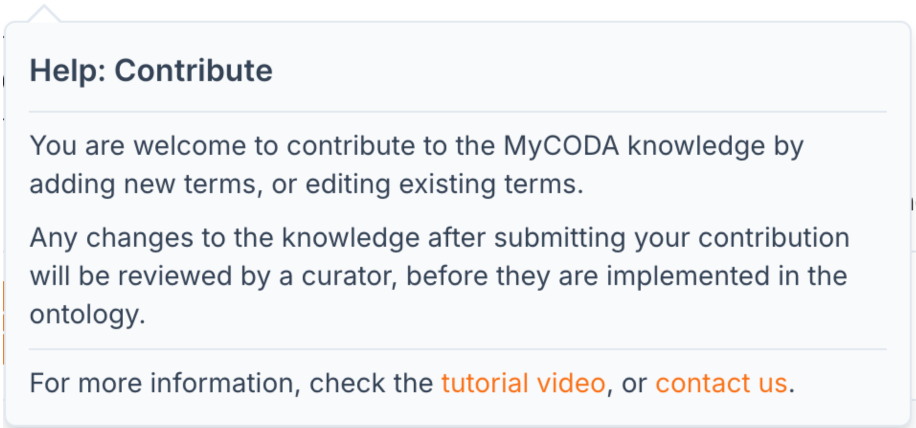


Figure 36 – MyCODA contribute help tooltip

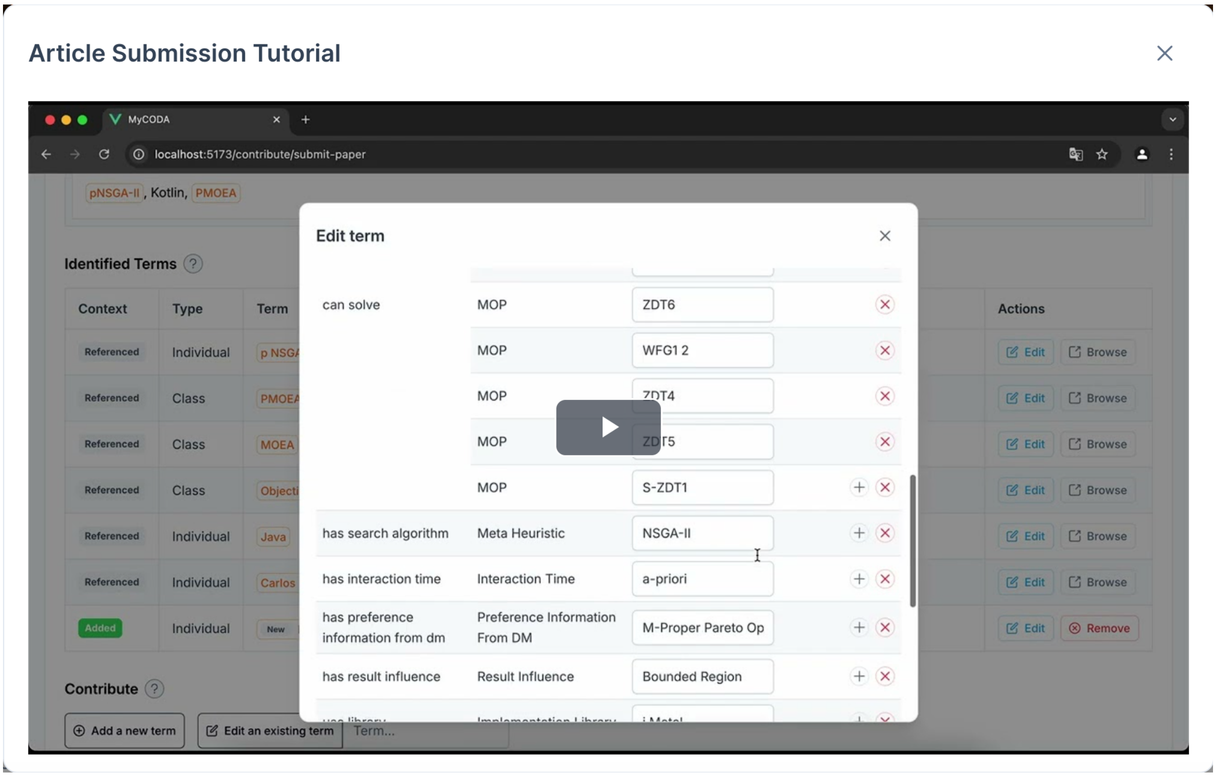
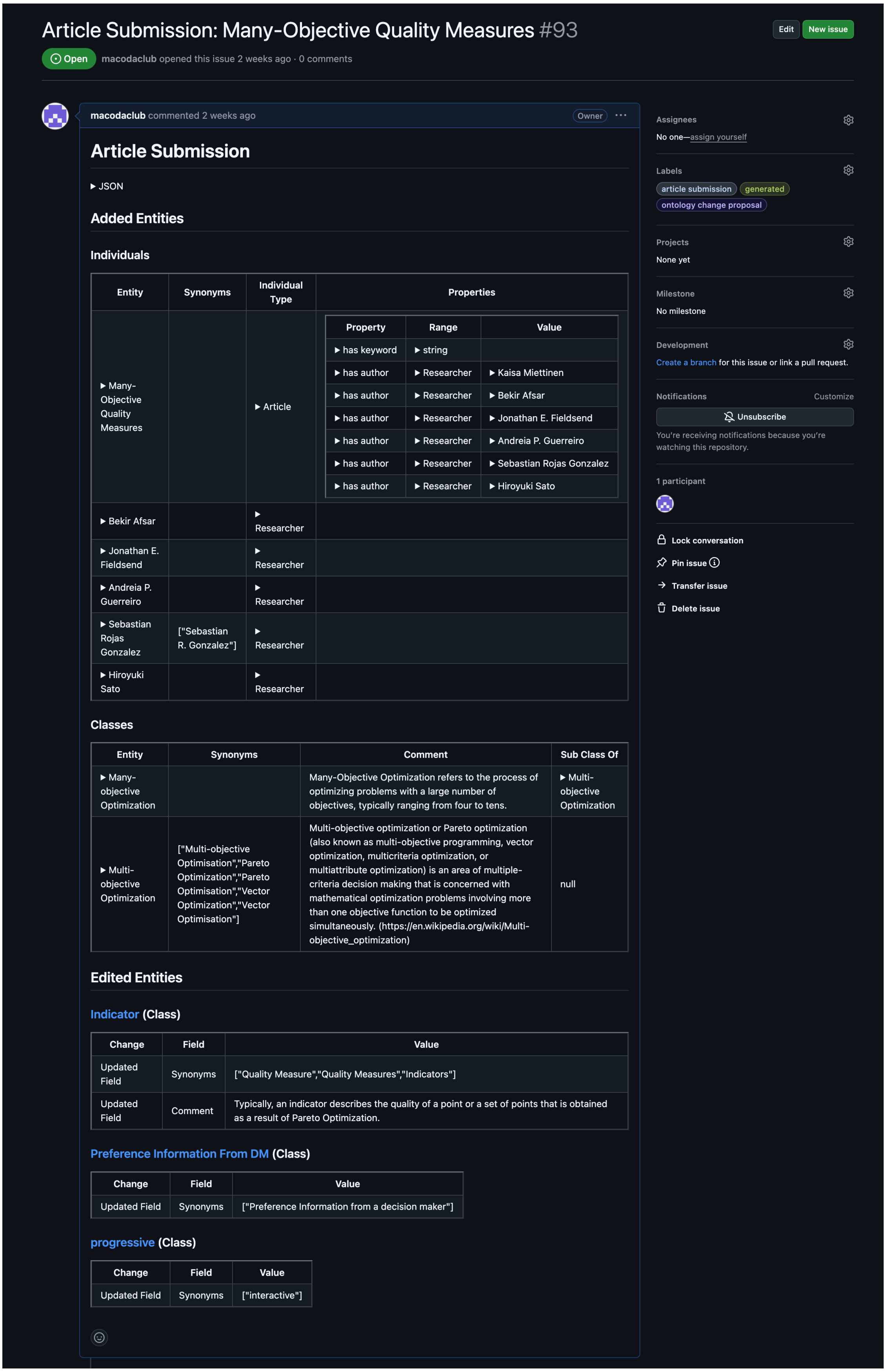


Figure 37 – MyCODA article submission tutorial video

Figure 38 – System Usability Scale Survey



Article Submission: Many-Objective Quality Measures #93

Edit | New issue

Open macodaclub opened this issue 2 weeks ago · 0 comments

macodaclub commented 2 weeks ago · Owner

Article Submission

▸ JSON

Added Entities

Individuals

Entity	Synonyms	Individual Type	Properties
▸ Many-Objective Quality Measures		▸ Article	Property / Range / Value: ▸ has keyword / ▸ string / ; ▸ has author / ▸ Researcher / ▸ Kaisa Miettinen; ▸ has author / ▸ Researcher / ▸ Bekir Afsar; ▸ has author / ▸ Researcher / ▸ Jonathan E. Fieldsend; ▸ has author / ▸ Researcher / ▸ Andreia P. Guerreiro; ▸ has author / ▸ Researcher / ▸ Sebastian Rojas Gonzalez; ▸ has author / ▸ Researcher / ▸ Hiroyuki Sato
▸ Bekir Afsar		▸ Researcher	
▸ Jonathan E. Fieldsend		▸ Researcher	
▸ Andreia P. Guerreiro		▸ Researcher	
▸ Sebastian Rojas Gonzalez	["Sebastian R. Gonzalez"]	▸ Researcher	
▸ Hiroyuki Sato		▸ Researcher	

Classes

Entity	Synonyms	Comment	Sub Class Of
▸ Many-objective Optimization		Many-Objective Optimization refers to the process of optimizing problems with a large number of objectives, typically ranging from four to tens.	▸ Multi-objective Optimization
▸ Multi-objective Optimization	["Multi-objective Optimisation","Pareto Optimization","Pareto Optimisation","Vector Optimization","Vector Optimisation"]	Multi-objective optimization or Pareto optimization (also known as multi-objective programming, vector optimization, multicriteria optimization, or multiattribute optimization) is an area of multiple-criteria decision making that is concerned with mathematical optimization problems involving more than one objective function to be optimized simultaneously. (https://en.wikipedia.org/wiki/Multi-objective_optimization)	null

Edited Entities

Indicator (Class)

Change	Field	Value
Updated Field	Synonyms	["Quality Measure","Quality Measures","Indicators"]
Updated Field	Comment	Typically, an indicator describes the quality of a point or a set of points that is obtained as a result of Pareto Optimization.

Preference Information From DM (Class)

Change	Field	Value
Updated Field	Synonyms	["Preference Information from a decision maker"]

progressive (Class)

Change	Field	Value
Updated Field	Synonyms	["interactive"]

Assignees
No one—assign yourself

Labels
article submission · generated · ontology change proposal

Projects
None yet

Milestone
No milestone

Development
Create a branch for this issue or link a pull request.

Notifications · Customize
Unsubscribe
You're receiving notifications because you're watching this repository.

1 participant

Lock conversation
Pin issue
Transfer issue
Delete issue

Figure 39 – Example of a GitHub issue generated from an article submission