A Domain Reference Ontology for Design Science Research Knowledge Bases

Jean Paul Sebastian Piest[1,*,†], Victor Benoiston Jales de Oliviera[2,†], Patrício de Alencar Silva[2,†] and Manoel Ricardo da Cunha Junior[2,†]

[1] *University of Twente, Drienerlolaan 5, 7522 NB, Enschede, The Netherlands*
[2] *Federal University of the Semi-Arid Region (UFERSA), Mossoró, RN - Brazil*

Abstract

Knowledge bases play an important role in Design Science Research (DSR). Although some related work exists, no domain reference ontology is available for reuse and cumulative development of design knowledge. This paper presents a domain reference ontology for DSR knowledge bases using the SABiO methodology. The domain reference ontology is designed in Visual Paradigm based on the UFO and represented in OntoUML. Stakeholders have verified the results and syntactic consistency has been checked using the OntoUML plugin in Visual Paradigm and expert opinion. An operational ontology was designed and implemented in a DSR project in OWL using WebProtégé. The ontology was populated and validated using competency questions and expert opinion. The domain reference ontology provides an abstract blueprint for DSR knowledge bases and a conceptual foundation to develop operational ontologies to support the cumulative development of design knowledge. Current work aims to evaluate the use of the operational ontology in operational systems and implement an interactive Website for online collaboration. A user and working group will be established to manage the lifecycle, which is open for interested scholars and professionals to join.

Keywords

design science research, knowledge bases, domain reference ontology, SABiO, UFO, OntoUML, OWL.

1. Introduction

The field of Information Systems (IS) is concerned with "the use of information-technology artifacts in human-machine systems" [1]. Design Science Research (DSR) is an established research paradigm in the field of IS, focusing on "the design and investigation of artifacts in context" [2]. Knowledge Bases (KBs) are essential in DSR [3-5]. On the other hand, KBs inform DSR with existing and helpful knowledge. On the other hand, DSR produces novel artifacts, design knowledge, and design theory that contribute to advancing KBs.

Design knowledge lies scattered across literature and disciplines [1,3]. While DSR typically combines knowledge from different disciplines, developing an integrated KB

* Corresponding author.
† These authors contributed equally.
j.p.s.piest@utwente.nl (J.P.S. Piest)
0000-0002-0995-6813 (J.P.S. Piest); 0009-0000-7026-6929 (V. Benoiston Jales de Oliviera); 0000-0001-6827-1024 (P. de Alencar Silva); 0009-0009-9167-8469 (M. Ricardo da Cunha Junior)

requires a thorough literature review and integration of potentially conflicting perspectives regarding ontology and epistemology [3,4,6]. Although related work exists [3,4,6], no readily available domain reference ontology exists for DSR KBs. This hinders the reuse and cumulative development of design knowledge [5,6].

Building upon [7], this paper presents a domain reference ontology for DSR KBs based on the Systematic Approach for Building Ontologies (SABiO) [8].

This paper is structured as follows. **Section 2** discusses related work. **Section 3** is concerned with the methodology. **Section 4** presents the domain reference ontology for DSR KBs. **Section 5** discusses the operational ontology. **Section 6** contains a description of supporting processes. **Section 7** concludes the paper.

2. Related Work

This section discusses related work regarding Design Science Research (DSR) and its relationship with developing Knowledge Bases (KBs).

2.1. Design Science Research

The field of IS produced several DSR methodologies to support the development and evaluation of artifacts in context [2, 9-11]. A distinct feature of DSR is its connection to KBs and the application environment, respectively focusing on rigor and relevance [10]. In DSR, design knowledge contributions are informed by formal theories and rigorously evaluated.

The role of theory in IS and design theorizing in DSR are much-debated topics [1,12,13]. Important related work exists regarding the nature of theory in IS [1], different theory types [1,3,12], the relation between theory types and the design process [12], theory building in DSR [1,5,12,13], and the anatomy of a design theory [14]. These related works should be incorporated into a domain reference ontology for DSR KBs.

DSR projects typically result in different artifacts and outputs, which can be assessed by the level of abstraction, completeness, and knowledge maturity: (1) situated implementations of artifacts, (2) developing design theory, and (3) well-developed design theory about embedded phenomena [4]. The artifact (in development) is subject to verification, validation, and evaluation by stakeholders in a specific application environment or broader context [15].

2.2. Knowledge Bases in Design Science Research

Related work investigated the structure and contents of KBs in IS and DSR [3] and provided a DSR KB framework [4]. In the context of IS and DSR, KBs contain descriptive and prescriptive knowledge [3,4]. The former mainly consists of formal theories originating from natural and social sciences [3]. The latter involves design knowledge in constructs, models, methods, instantiations (abstract or situated artifacts), and design theories [3,4]. Although there is consensus that theories should inform DSR, there is no consensus regarding the role of justification knowledge in KBs [3]. DSR utilizes one or multiple KBs to produce and consume descriptive and prescriptive knowledge [3,4]. The framework of [5]

differentiates six modes for using KBs to support DSR projects and enable cumulative development of design knowledge.

3. Ontology Engineering Methodology

The SABiO methodology [8] was adopted to guide the domain reference ontology and operational ontology development for DSR KBs, as shown in **Figure 1**.

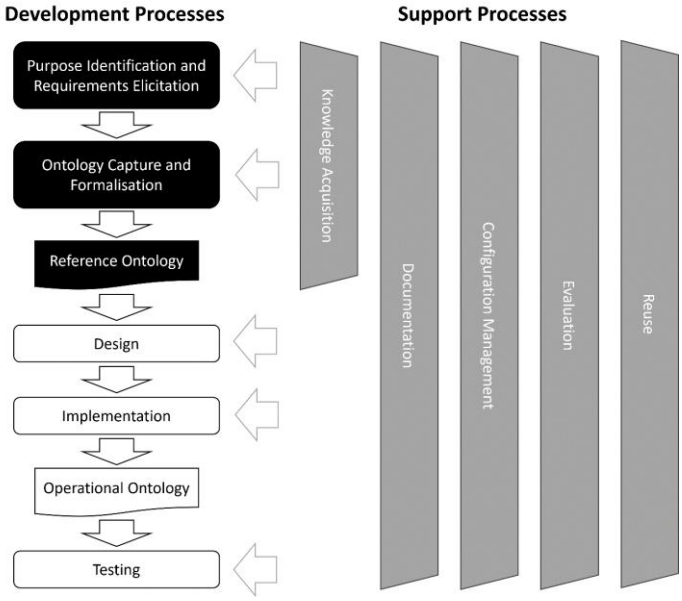

Figure 1: Overview of the SABiO development and support processes, adapted from [8].

The SABiO prescribes a development process that consists of five phases: 1) *purpose identification and requirements elicitation*, 2) ontology capture and formalization, 3) *design*, 4) *implementation*, and 5) *testing* [8]. The development process is supported by processes for knowledge acquisition, documentation, configuration management, evaluation, and reuse [8]. The authors used the SABiO guidelines for each phase and online collaborative modeling to complete the required activities. The first three phases related to the domain reference ontology are documented in **Section 4**. The fourth and fifth phases are discussed in **Section 5**. The support processes are described in **Section 6**.

4. Domain Reference Ontology

This section reports on developing the domain reference ontology for DSR KBs.

4.1. Ontology Requirements

Phase 1 of the SABiO is aimed at specifying four required aspects: 1) *purpose*, 2) *intended uses*, 3) *requirements*, and 4) *competency questions* (CQs), which together result in an Ontology Requirements Specification Document (ORSD). **Table 1** presents the ORSD.

Table 1
ORSD

Component	Description
Purpose and aim	The primary purpose is to develop an interactive KB system that supports (1) researchers, (2) designers, (3) developers, (4) end-users, and (5) system users with activities related to design theorizing and design processing in DSR projects. More specifically, the domain reference ontology for DSR KBs aims to create a common language and vocabulary, classify types of knowledge, and provide a foundation for conceptual modeling.
Intended uses	**Use 1a:** Researchers periodically review scientific databases and organize descriptive and prescriptive knowledge in the KB. **Use 1b:** Researchers create design knowledge maps by interrelating problem- and solution spaces, available technologies and alternatives, and evaluation in empirical research studies. **Use 1c:** Researchers initiate DSR projects, informed by the KB, to cumulatively develop design knowledge and verify, validate, and evaluate contributions with designers, developers, and end-users. **Use 1d:** Researchers review design knowledge contributions of fellow researchers from DSR projects for inclusion in the KB. **Use 1e:** Researchers review artifacts and design knowledge contributions of designers and developers from empirical research studies for inclusion in the KB. **Use 2a:** Designers search the KB for reusable design knowledge for the design of applications. **Use 2b:** Designers produce solution designs for applications to address/solve problems using the KB. **Use 2c:** Designers submit design artifacts and design knowledge contributions for review and inclusion in the KB. **Use 3a:** Developers search the KB for reusable artifacts and design knowledge for the construction and implementation of applications. **Use 3b:** Developers construct and instantiate applications based on solution designs of designers in **2b**, using the KB. **Use 3c:** Developers submit reusable applications, including related artifacts and implementation details, and design knowledge contributions for review and inclusion in the KB. **Use 4a:** End-users search the KB for existing applications and, if available, related case studies and empirical research studies. **Use 4b:** End-users participate in DSR projects and empirical research studies for verification, validation, and evaluation. **Use 5a:** Scientific databases provide publication data and meta-data (in .ris format) to include in the KB.

Component	Description
	Use 5b: Scientific databases provide APIs for automated data exchange with the KB. **Use 6a:** System users query the KB for inquiry (e.g., explore knowledge, search), analytical purposes (e.g., bibliometric analysis), and the development of software agents (e.g., recommender or assistant to support users 1-4). **Use 6b:** System users can integrate via APIs (e.g., to synchronize scientific databases). **Use 7:** All actors add their perspectives and ontological and epistemological positions in the KB.
Non-Functional Requirements	**NFR1:** The ontology must comply with, integrate, and reuse existing vocabularies as much as possible, if relevant for the purpose/scope. **NFR2:** The ontology distinguishes general concepts from task/application-specific concepts. **NFR3:** The ontology allows specializations and extensions. **NFR4:** The ontology allows deep modeling, thus connecting metamodels to instantiations. **NFR5:** The ontology must be at least available in the English language. **NFR6:** The ontology must be open-access. **NFR7:** The ontology must be based on FAIR principles.
Technical requirements	**TR1:** The ontology is represented in OntoUML. **TR2:** The ontology is implemented in OWL and instantiated using (Web)Protégé. **TR3:** The ontology is automatically checked for consistency and correctness. **TR4:** The ontology can be queried (e.g., using SPARQL).
CQs	**CQ1:** What constitutes a DSR project? **CQ2:** Which theory types and knowledge are used in DSR projects? **CQ3:** How are design theory and knowledge created in DSR projects? **CQ4:** Which applications and situated artifacts are built in DSR projects?

4.2. Ontology Capture

In phase 2 of the SABiO methodology, the ORSD captured and formalized the domain reference ontology for DSR KBs. SABiO recommends identifying and modeling concepts and relations based on a foundational ontology [8]. The selected representation language for conceptual modeling is OntoUML [16], which is grounded in the Unified Foundational Ontology (UFO) [17]. Based on the DSR project [7], the relevant concepts and relations were identified and organized using conceptual modeling in online sessions. **Table 2** presents the four conceptual models that were developed.

Table 2
ORSD

CQ	Conceptual model	Description	Knowledge acquisition
1	Agent taxonomy and DSR projects	Taxonomy of involved agents concerning the problem-solution space(s) in DSR projects	[4,5,6,9-11]
2	Types of knowledge	Classification of types of design knowledge	[1,3,4,7,12]
3	Design theorizing	Epistemological view on design theorizing	[3,5,12-14]
4	Cases and applications	Conceptual model to classify case studies and applications	[5,7]

SABiO prescribes a dictionary of terms and definitions and informal and formal axiom definitions [8]. The dictionary is based on the UFO [16], the ISO/IEC 2382:2015(en) Information Technology — Vocabulary [18], and disciplinary terms and definitions used in DSR. Following the informal axiom definition, the classes (e.g., (sub-)*kinds*, *categories*, *mixins*, *modes*, *qualities*) and formal axiom definitions (e.g., *characterization*, *mediation*, *materialization*) were modeled using UFO and OntoUML stereotypes. Each conceptual model will be described and justified in a separate subsection.

4.2.1. Agent Taxonomy and DSR Projects

Figure 2 depicts an agent taxonomy and representation of the concepts in DSR projects.

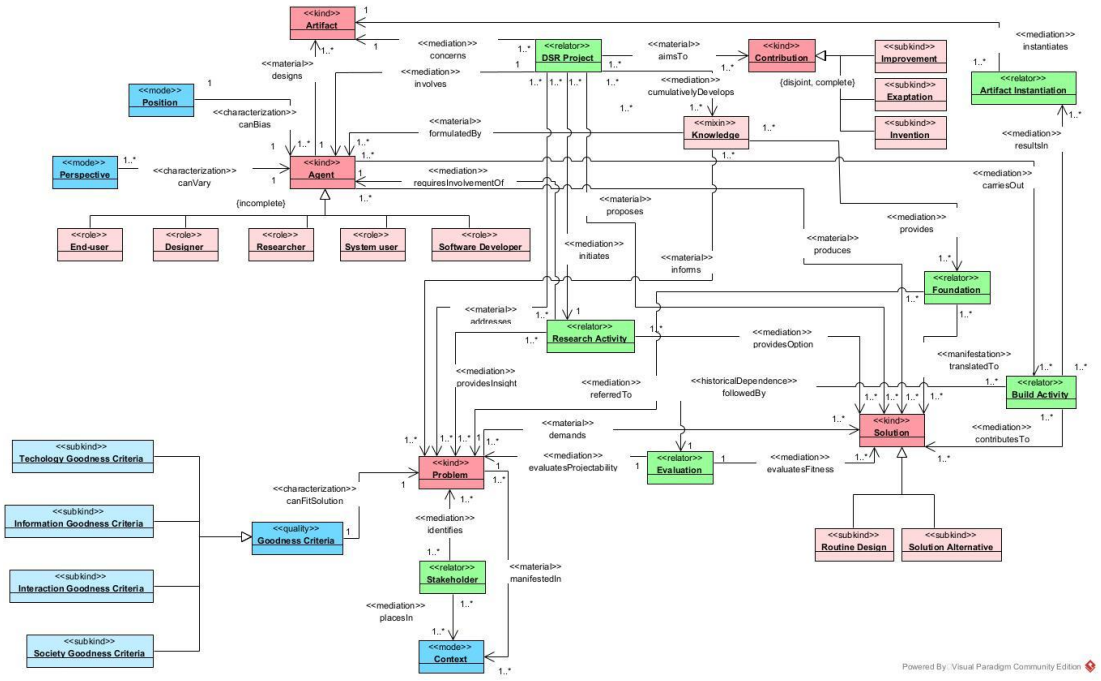

Figure 2: OntoUML model representing the agents in DSR projects.

Different human agents work together in DSR projects to develop knowledge contributions and artifacts [4,5]. Improvements, extensions, and exaptations are knowledge contributions [4]. Human agents have different perspectives related to epistemological or ontological positions [6]. The core DSR activities are related to searching (in both the problem and solution space) using appropriate research methods and applying the intertwined activities to build, instantiate, and evaluate artifacts [5]. DSR projects should be informed by prior knowledge [9-11]. Input knowledge informs both problem and solution space [5]. The three components of design knowledge include problem, solution, and evaluation [5]. More specifically, the context of the problem space is detailed using domain, stakeholder, time, and space [5]. The four goodness criteria are also related to the problem [5]. Conversely, the solution space is represented as an array of alternatives and, if known, routine designs to solve a (part of) the problem [4]. The projectability of the problem space, the solution fitness (for use and evolution), and the confidence form the basis for a design knowledge map [5]. Confidence is complemented by the core values in DSR: rigor and relevance [4,11].

4.2.2. Types of Knowledge

Figure 3 classifies types of knowledge that agents can use to connect to DSR projects.

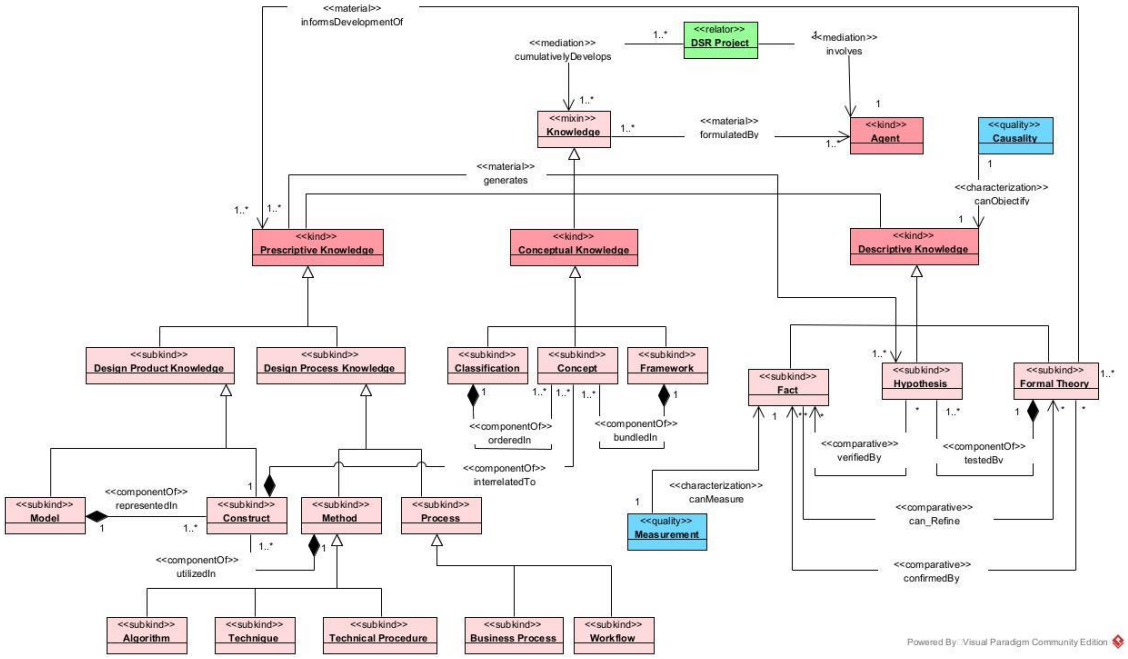

Figure 3: OntoUML model representing types of knowledge.

Following the DSR KB framework [4,7], knowledge is classified into descriptive and prescriptive knowledge [3,4]. The classification is differentiated by [3] with conceptual knowledge as a kind of knowledge to represent abstract forms of knowledge. As such, conceptual knowledge can be incorporated explicitly in the DSR project. Descriptive knowledge is based on causality and informs the development of prescriptive knowledge through formal theories [1,3,12]. Prescriptive knowledge is divided into product and

process-related knowledge and further refined based on the adopted classification [4]. Concepts can be classified and combined in a (conceptual) framework. A construct is a set of ideas with relationships. Constructs can be incorporated as part of a model or a method. Models can be characterized by representation, semantics, and syntax. Methods can be algorithms, techniques, or technical procedures. Processes can be business processes or workflows. Prescriptive knowledge can confirm or refine formal theories by linking facts (based on measurements) to hypotheses [12].

4.2.3. Design Theorizing

Figure 4 depicts an epistemological view of design theorizing.

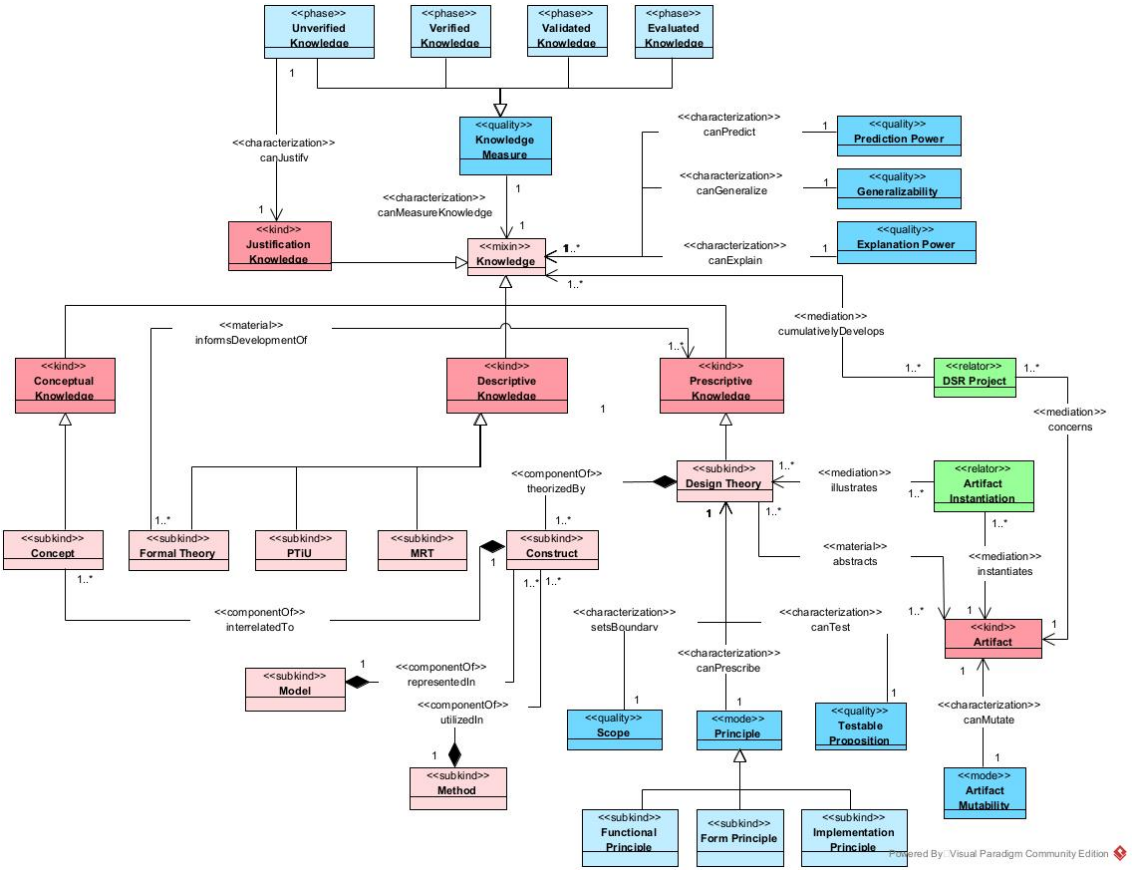

Figure 4: OntoUML model representing design theorizing.

This conceptual model builds upon **Figure 3** to develop an epistemological view of design theorizing. Formal theories inform the development of design theories [3,12,14]. A design theory consists of scope, construct (which can be part of a model or method), testable propositions, principles (e.g., form, function, implementation), and artifact mutability (e.g., state changes), eventually resulting in an expository instantiation [14]. The conceptual model of [12] is selected to interrelate design theorizing with the classification of theory types (e.g., formal theories, mid-range theories, practitioner-in-use theories, and design theories) and their qualities (e.g., explanation power, prediction power, and

generalizability). The knowledge measures are related to verification, validation, and evaluation in DSR [5]. Justification knowledge is defined as an unverified form of knowledge [13].

4.2.4. Case Studies and Applications

Figure 5 contains a conceptual model to represent case studies and applications.

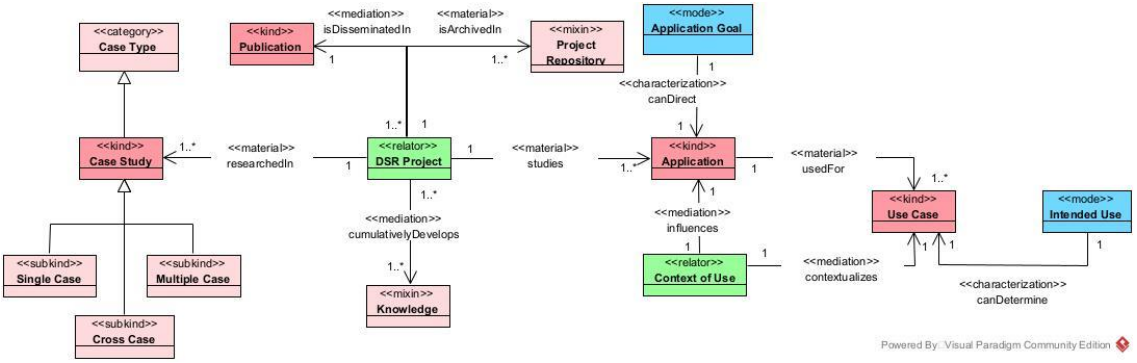

Figure 5: OntoUML model representing case studies and applications.

This conceptual model relates DSR projects to publications, empirical research studies, and applications [7]. The project repository will contribute to making the results of the DSR project FAIR and contribute to the reuse and cumulative development of design knowledge [5].

4.2.5. Verification

Using the ex-ante evaluation criteria, a stakeholder opinion survey (N=7) was conducted among three researchers, two designers, and two developers [15]. Here, the stakeholders' appreciation of the solution approach and feasibility of solving identified problems were evaluated. Furthermore, the relevant roles, intended future use, and contribution were assessed. The ex-ante evaluation provided support for the solution approach and input for the refinement of the conceptual models. After completing the conceptual modeling, each conceptual model was syntactically verified for consistency using the OntoUML plugin in Visual Paradigm. All four conceptual models are syntactically consistent with OntoUML.

5. Operational Ontology

This subsection documents the development of an operational ontology in a DSR project [7].

5.1. Design

Phase 3 of the SABiO methodology is related to the design of an operational ontology. It comprises the definition of the implementation environment, a high-level architecture design description, and a detailed design description.

Following the SABiO recommendation [8], OWL instantiated an implementation environment. For the implementation phase, the Ontology Lightweight Editor (OLED)

WebProtégé was selected to satisfy most requirements. WebProtégé is a free, open-source, OWL-based OLED. The primary limitations, compared to the desktop version of Protégé, are related to querying (WebProtégé uses web-based queries, the desktop version supports SPARQL) and integrations (WebProtégé provides webhooks and no APIs). Moreover, WebProtégé does not offer embedded reasoners.

Upon the UFO and OntoUML, an OLED can automatically generate an operational OWL ontology based on OntoUML, which can contribute to shortening the implementation phase. Another advantage of WebProtégé is that it facilitates online collaboration. Building Furthermore, WebProtégé offers several options for downloading ontologies to import into the desktop version of Protégé or other tools.

As part of detailed design, among other challenges, the problem of lower expressivity in operational languages needs to be assessed and resolved [8]. Three important design decisions were made: 1) Trim the "leaves" of the ontology to reduce the size and complexity of the ontology, 2) Rename all stereotypes relations to unique relations, and 3) Implement publication (meta)data as data properties. Most trimmed leaves were recreated in the operational ontology using object and data properties.

5.2. Implementation

Phase 4 of the SABiO is concerned with the implementation [8]. Based on the selected implementation environment, an application cooperation view was created in ArchiMate 3.2 [19], as depicted in **Figure 6**.

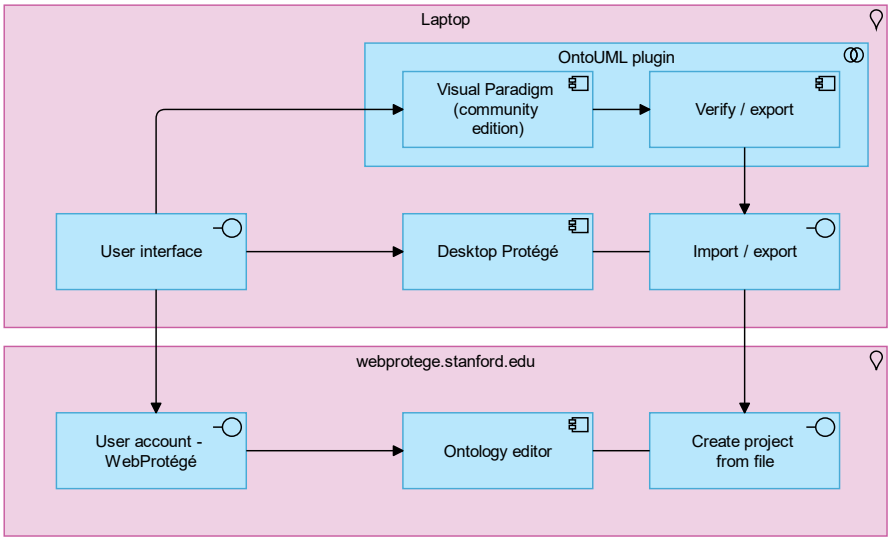

Figure 6: Application cooperation view.

Building upon the OntoUML models in Visual Paradigm, the gUFO export option generated a .ttl file. This file was imported into desktop Protégé and used to create an OWL ontology. Next, the OWL ontology was imported into WebProtégé, as shown in **Figure 7**.

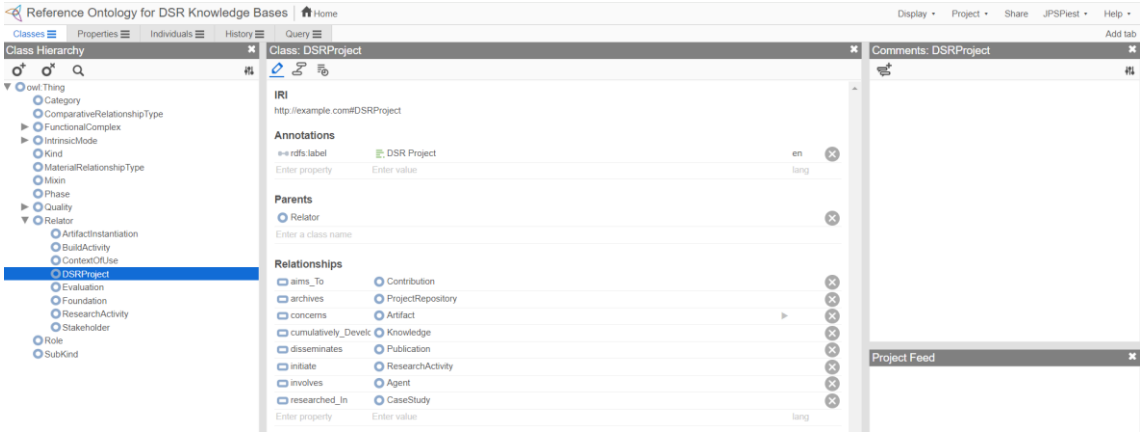

Figure 7: Screenshot of WebProtégé implementation.

For each class, the relationships were checked with the OntoUML models. Next, the domain and ranges of object properties were set. Following, data properties were created.

5.3. Testing

In phase 5 of the SABiO, the operational ontology was tested. The SABiO prescribes CQ-driven testing of sub-ontologies (not applicable), integrations, and the entire ontology [8].

The prescribed testing approach of the SABiO was adopted. First, the ontology was populated using the SLR results from 176 studies [7]. Second, the CQs' needs were transformed into machine-readable query languages (e.g., SPARQL). The SPARQL queries are included in **Appendix A**. Third, a finite set of test cases was defined based on the ORSD. These test cases were related to the intended uses and address NFRs and TRs. One integration test case was described to test the interoperability of the OWL ontology. Fourth, a test document was created to guide the testing process and report the results. This document includes the test cases and documented the results in tables (e.g., input-output of queries to answer CQs). The test results are available at [20].

5.3.1. Validation and Evaluation

The operational ontology was instantiated in WebProtégé, populated based on a DSR project [7], validated using the CQs and queries, and evaluated using expert opinion.

A single-case mechanism experiment was conducted in WebProtégé using the CQs. 4 out of the 4 CQs can be answered based on the populated OWL ontology using WebProtégé queries. The quality of the answers was low because WebProtégé uses a form-based approach. SPARQL queries were developed and validated in desktop Protégé based on the populated OWL ontology. This resulted in better answers. Regarding the NFRs, 6 out of the 7 NFRs have been satisfied. Satisfying the NFR related to deep modeling requires additional case study research. In terms of TRs, 4 out of the 4 requirements were satisfied.

The integration test case evaluated the reuse of the OWL ontology by downloading the populated OWL ontology from WebProtégé and importing it into desktop Protégé. Importing and exporting the results from Visual Paradigm into desktop Protégé functioned.

Exporting the OWL ontology from desktop Protégé and importing the OWL ontology in WebProtégé functioned as well. Lastly, the populated OWL ontology was downloaded from WebProtégé and imported into desktop Protégé. However, the structure was imported correctly, and there were issues regarding object and data property names. These issues were reported and resolved through manual renaming.

Finally, an expert opinion evaluation was conducted using the ex-post criteria of [15]. The main improvement is to make an explicit connection between the dictionary and the developed OntoUML models and OWL ontology. This will be included in the documentation [20]. Additionally, the expert recommended guiding the practical use of the OWL ontology. More specifically, the expert advised to focus on the implementation in an operational system. Based on the export opinion evaluation results, the decision was made to release the current version in beta to evaluate the use of the domain reference ontology and operational ontology in operation systems and multiple DSR projects.

6. Support Processes

This subsection describes the support processes. The SABiO prescribes supporting processes, including documentation, configuration management, evaluation, and reuse [8]. The support processes will contribute to making the domain reference ontology and operational ontology for DSR KBs FAIR.

SABiO recommends creating an online project repository to publish the source code, naming conventions, and rules for commenting on the code [8]. The Visual Paradigm project, containing the four OntoUML models, is made available with the populated OWL ontology [20]. Desktop Protégé provides functionality to generate interactive documentation as an OWL Doc. The generated OWL Doc is also available at [20].

The lifecycle of the domain reference ontology for DSR KBs and selected operational ontologies will be managed using major and minor releases, including new versions, changes, and supporting documentation. The release plan will be detailed based on the evaluation and feedback of stakeholders in DSR projects and made available at [20].

Initial verifications, validations, and evaluations have been conducted. The developed domain reference ontology for DSR KBs and operational ontology will be available in beta for a broader community evaluation. A user group will be formed to evaluate the use of operational systems in multiple DSR projects using the ex-post evaluation criteria (e.g., effectiveness, efficiency, external consistency) and evaluation framework [15].

A working group will be formed to translate the evaluation results into an official release and product roadmap in parallel with user evaluation. The working group will include the involved researchers and is open for interested scholars and IS professionals to join. After the official release, comments are registered using the comment function in WebProtégé and registered in the releases and documentation pages at [20].

After the official release, the WebProtégé project will also be made publicly available. The domain reference ontology can then be downloaded from WebProtégé in the OWL format for reuse and extended in operational ontologies. Guidelines will be defined to (re)use the domain reference for DSR KBs in DSR projects.

7. Conclusion

This paper presented a domain reference ontology for DSR KBs and operational ontology.

The domain reference ontology is developed using the SABiO and based on an ORSD. Following the SABiO recommendations, the domain reference ontology for DSR KBs is based on the UFO, represented in OntoUML, and implemented in OWL. Seven stakeholders verified the domain reference ontology for DSR KBs using ex-ante evaluation criteria. Four OntoUML models have been designed and syntactically verified in Visual Paradigm. Next, an operational ontology has been instantiated using WebProtégé as part of a DSR project. The operational ontology was validated using CQs and expert opinion using ex-post evaluation criteria. The developed domain reference ontology for DSR KBs is available in beta for community evaluation. It provides an abstract blueprint to develop DSR KBs and a conceptual foundation to develop operational ontologies to support DSR projects.

Some limitations need to be addressed. The operational ontology has been implemented in WebProtégé, which has some limitations compared to the desktop version regarding querying, integration, and reasoners. Another limitation is that the domain reference ontology for DSR KBs and the operational ontology have not been used and evaluated in operational systems. This is subject to current and future work.

Current work aims to evaluate the domain reference ontology for DSR KBs and operational ontology in operational systems and implement an interactive website for online collaboration. Future work can contribute to the experimental development of software agents that can augment DSR activities. More generally speaking, future research may contribute to developing methods and techniques for analyzing DSR KBs, for instance, to measure their current maturity, completeness, and quality.

References

[1] Gregor, S. (2006). The nature of theory in Information Systems. MIS Quarterly: Management Information Systems, 30(3), 611–642. DOI: https://doi.org/10.2307/25148742.

[2] Wieringa, R. J. (2014). Design science methodology for information systems and software engineering. Springer Berlin, Heidelberg. DOI: https://doi.org/https://doi.org/10.1007/978-3-662-43839-8.

[3] Gaß, O., Koppenhagen, N., Biegel, H., Mädche, A., & Müller, B. (2012). Anatomy of Knowledge Bases Used in Design Science Research: A Literature Review. In Design Science Research in Information Systems. Advances in Theory and Practice. DESRIST 2012. Lecture Notes in Computer Science, vol 7286. Springer, Berlin, Heidelberg. (pp. 328–344). DOI: https://doi.org/https://doi.org/10.1007/978-3-642-29863-9_24.

[4] Gregor, S., & Hevner, A. R. (2013). Positioning and presenting design science research for maximum impact. MIS Quarterly: Management Information Systems, 37(2), 337–355. DOI: https://doi.org/10.25300/MISQ/2013/37.2.01.

[5] Vom Brocke, J., Winter, R., Hevner, A., & Maedche, A. (2020). Special issue editorial – accumulation and evolution of design knowledge in design science research: A journey

through time and space. Journal of the Association for Information Systems, 21(3), 520–544. DOI: https://doi.org/10.17705/1jais.00611.

[6] Nguyen, A., Gardner, L., & Sheridan, D. (2019). Towards Ontology-Based Design Science Research for Knowledge Accumulation and Evolution. In Proceedings of the 52nd Hawaii International Conference on System Sciences (pp. 5755-5764). DOI: http://hdl.handle.net/10125/60011.

[7] Piest, J.P.S. (2024). Towards a Knowledge Base and Design and Action Theory for Intelligence Amplification. In: Sales, T.P., de Kinderen, S., Proper, H.A., Pufahl, L., Karastoyanova, D., van Sinderen, M. (eds) Enterprise Design, Operations, and Computing. EDOC 2023 Workshops. Lecture Notes in Business Information Processing, vol 498. Springer, Cham. DOI: https://doi.org/10.1007/978-3-031-54712-6_24.

[8] De Almeida Falbo, R. (2014). SABiO: Systematic Approach for Building Ontologies. In CEUR Workshop Proceedings (Vol. 1301). Retrieved from https://ceur-ws.org/Vol-1301/ontocomodise2014_2.pdf.

[9] Nunamaker, J. F., Chen, M., & Purdin, T. D. M. (1990). Systems Development in Information Systems Research. Journal of Management Information Systems, 7(3), 89–106. DOI: https://doi.org/https://www.jstor.org/stable/40397957.

[10] Hevner, A. R., Ram, S., March, S. T., & Park, J. (1996). Design science in Information Systems Research. AI and Society, 10(2), 199–217. DOI: https://doi.org/10.1007/BF01205282.

[11] Peffers, K., Tuunanen, T., Rothenberger, M. A., & Chatterjee, S. (2007). A design science research methodology for information systems research. Journal of Management Information Systems, 24(3), 45–77. DOI: https://doi.org/10.2753/MIS0742-1222240302.

[12] Kuechler, B., & Vaishnavi, V. (2008). On theory development in design science research: Anatomy of a research project. European Journal of Information Systems, 17(5), 489–504. DOI: https://doi.org/10.1057/ejis.2008.40.

[13] Venable, J. R. (2006). The Role of Theory and Theorising in Design Science Research. In Proceedings of the 1st International Conference on Design Science in Information Systems and Technology (DESRIST 2006) (pp. 1–18). DOI: https://doi.org/10.1.1.110.2475.

[14] Jones, D., & Gregor, S. (2007). The anatomy of a design theory. Journal of the Association for Information Systems, 8(5), 312–335. DOI: https://doi.org/10.17705/1jais.00129.

[15] vom Brocke, J., Hevner, A., & Maedche, A. (2020). Introduction to Design Science Research, (November), 1–13. DOI: https://doi.org/10.1007/978-3-030-46781-4_1.

[16] OntoUML - OntoUML Specification Documentation. Available online: https://ontouml.readthedocs.io/en/latest/intro/ontouml.html.

[17] Guizzardi, G. (2005). Ontological Foundations for Structural Conceptual Models. Retrieved from: http://www.loa.istc.cnr.it/Guizzardi/SELMAS-CR.pdf

[18] Available online: https://www.iso.org/obp/ui/en/#iso:std:iso-iec:2382:ed-1:v2:en.

[19] The Open Group (N.d.). ArchiMate ® 3.2 Specification. Available online: https://pubs.opengroup.org/architecture/archimate3-doc/.

[20] GitHub. DSR KB. Available online at GitHub repository: https://github.com/SebastianPiest/dsr-kb.

A. SPARQL Queries

CQ 1: What constitutes a DSR project?

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX : <http://www.dsr_knowledge_base#>

SELECT DISTINCT ?dsrProject ?artifact ?agent ?contribution ?solution ?problem
?researchActivity
WHERE {
  # Match instances of DSRProject
  ?dsrProject rdf:type :DSRProject .

  # Optional: Fetch artifacts that the DSR Project concerns
  OPTIONAL { ?dsrProject :concerns ?artifact . }

  # Optional: Fetch agents involved in the DSR Project
  OPTIONAL { ?dsrProject :involves ?agent . }

  # Optional: Fetch the contributions aimed by the DSR Project
  OPTIONAL { ?dsrProject :aimsTo ?contribution . }

  # Optional: Fetch the solutions proposed by the DSR Project
  OPTIONAL { ?dsrProject :proposes ?solution . }

  # Optional: Fetch the problems addressed by the DSR Project
  OPTIONAL { ?dsrProject :addresses ?problem . }

  # Optional: Fetch the research activities initiated by the DSR Project
  OPTIONAL { ?dsrProject :initiates ?researchActivity . }
}
```

This query fetches the concepts that are directly related to DSR Projects and can be extended to include other concepts.

CQ 2: Which theory types and knowledge are used in DSR projects?

```
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX : <http://www.dsr_knowledge_base#>

SELECT DISTINCT ?theory ?theoryType ?relatedKnowledge ?knowledgeType
WHERE {
  # Step 1: Match theories and their types
  {
    ?theory rdf:type ?theoryType .  # Match theories and their types
    FILTER(?theoryType = :FormalTheory)  # Filter for FormalTheory
  }
  UNION
  {
    ?theory rdf:type ?theoryType .  # Match theories and their types
    FILTER(?theoryType = :MRT)  # Filter for MRT
  }
```

```
  UNION
  {
    ?theory rdf:type ?theoryType .  # Match theories and their types
    FILTER(?theoryType = :PTiU)  # Filter for PTiU
  }

  # Step 2: Optional matching of related knowledge
  OPTIONAL {
    {
      ?theory :confirmedBy ?relatedKnowledge .  # Match theories confirmed by related
knowledge
      ?relatedKnowledge rdf:type ?knowledgeType .  # Match related knowledge types
      FILTER(?knowledgeType = :Fact || ?knowledgeType = :Hypothesis)  # Filter for Fact
or Hypothesis
    }
    UNION
    {
      ?theory :generates ?relatedKnowledge .  # Match theories generating related
knowledge
      ?relatedKnowledge rdf:type ?knowledgeType .  # Match related knowledge types
      FILTER(?knowledgeType = :Fact || ?knowledgeType = :Hypothesis)  # Filter for Fact
or Hypothesis
    }
  }
}
```

CQ 3: How is design theory and knowledge created in DSR projects?

```
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX : <http://www.dsr_knowledge_base#>

SELECT DISTINCT ?dsrProject ?designTheory ?formalTheory ?scope ?construct
?testableProposition ?principle ?artifactInstantiation
WHERE {
  # Match instances of DSRProject and its relation to DesignTheory
  ?dsrProject rdf:type :DSRProject .
  ?designTheory rdf:type :DesignTheory .
  ?dsrProject :cumulativelyDevelops ?designTheory .

  # Optional: Match Formal Theories that inform the development of Design Theories
  OPTIONAL { ?formalTheory :informsDevelopmentOf ?designTheory . }

  # Optional: Fetch characteristics of a Design Theory
  OPTIONAL { ?designTheory :hasBoundarySetBy ?scope . }
  OPTIONAL { ?designTheory :theorizes ?construct . }
  OPTIONAL { ?designTheory :isTestedBy ?testableProposition . }
  OPTIONAL { ?designTheory :isPrescribedBy ?principle . }
  OPTIONAL { ?designTheory :isIllustratedBy ?artifactInstantiation . }
}
```

Design Theory Instances: The query starts by selecting instances of DesignTheory using the type relationship rdf:type and the related DSR projects in which knowledge was cumulatively developed. After that, how the design theory was created.

Formal Theories Related to Design Theories: It retrieves any FormalTheory that contributes to the development of a DesignTheory. The predicate :informsDevelopmentOf links FormalTheory to DesignTheory.

Characteristics of Design Theories: :hasBoundarySetBy: Retrieves the scope that sets the boundaries of the design theory. :theorizes: Fetches the constructs, which can be part of models or methods within the theory. :isTestedBy: Gets testable propositions linked to the theory, indicating elements that are empirically testable. :isPrescribedBy: Obtains principles that might define form, function, implementation strategies, etc. :isIllustratedBy: Retrieves artifact instantiations, which are practical demonstrations or exemplars of the theory.

CQ 4: Which applications and situated artifacts are built in DSR projects?

```
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX : <http://www.dsr_knowledge_base#>

SELECT DISTINCT ?dsrProject ?artifact ?application ?agent ?designTheory
?applicationGoal ?designRepresentation ?description ?requirement ?useCase
WHERE {
  # Match DSR projects
  ?dsrProject rdf:type :DSRProject .
  # Match Artifacts concerned by DSR projects
  OPTIONAL {
    ?dsrProject :concerns ?artifact .
    ?artifact rdf:type :Artifact .

    # Optional: Match agents that design the artifact
    OPTIONAL { ?agent :designs ?artifact . }

    # Optional: Match design theories that abstract the artifact
    OPTIONAL { ?designTheory :abstracts ?artifact . }
  }
  # Match Applications studied by DSR projects
  OPTIONAL {
    ?dsrProject :studies ?application .
    ?application rdf:type :Application .
    # Optional: Match ApplicationGoal that directs the application
    OPTIONAL { ?applicationGoal :canDirect ?application . }

    # Optional: Retrieve data properties for the application
    OPTIONAL { ?application :hasApplicaitonDesignRepresentation ?designRepresentation
. }
    OPTIONAL { ?application :hasApplicationDescription ?description . }
    OPTIONAL { ?application :hasApplicationRequirement ?requirement . }

    # Optional: Match UseCase associated with the application
    OPTIONAL { ?application :usedFor ?useCase . }
```

```
  }
}
```

DSR Projects as the Central Focus: The query starts by identifying instances of DSRProject.

Fetching Artifacts and Applications Related to DSR Projects: The query retrieves any artifacts that are "concerned" by these DSR projects using the relationship ?dsrProject :concerns ?artifact. This identifies the artifacts that are being developed or utilized within these projects. Similarly, the query retrieves applications that are "studied" by the DSR projects using the relationship ?dsrProject :studies ?application. This fetches applications that are either the focus of these research projects or are being developed as part of them.

Optional Details for Artifacts and Applications: For artifacts, it optionally fetches the agents who design them (?agent :designs ?artifact) and the design theories that abstract them (?designTheory :abstraction ?artifact). For applications, it optionally retrieves various descriptors such as the application's goal (?applicationGoal :can_Direct ?application), design representation, description, requirements, and use cases. The use of OPTIONAL clauses ensures that the query will still return DSR projects even if some information about related artifacts or applications is missing. This means the query is robust against incomplete data.