Leveraging Ontology-based Systems through Continuous Ontology Engineering

Jordana Sarmenghi Salamon[1,2], **Paola Espinoza Arias**[2], **Monalessa Perini Barcellos**[1]

[1]Ontology & Conceptual Modeling Research Group (NEMO)
Computer Science Department, Federal University of Espírito Santo, Vitória, ES, Brazil

[2]BASF Group
Berlin, Germany/Madrid, Espanha

jssalamon@inf.ufes.br, paola.espinoza@basf.com, monalessa@inf.ufes.br

Abstract. ***Research Context****: Ontologies have been recognized as key enablers of digital transformation in industry. They are used to assign semantics to information items, supporting data and knowledge management based on a common and shared understanding and representation. They play a paramount role in the Semantic Web and the implementation of Linked Open Data (LOD).* ***Scientific and/or Practical Problem****: Developing and maintaining ontologies may be complex, time-consuming, and resource-intensive. Additionally, traditional sequential ontology engineering methods often fail to address industrial needs such as changing requirements and time constraints. Iterative approaches mitigate some challenges but are not always sufficient.* ***Proposed Solution and/or Analysis****: To align ontology development with information systems development, both processes must be integrated. An end-to-end flow that integrates and supports ontology engineering from conception to operation is required to enable a continuous and industry-aligned process. We propose the use of a Continuous Ontology Engineering (COE) approach. We define principles that characterize COE, report on its adoption in a multinational chemical company, and share lessons learned.* ***Related IS Theory****: This work relates to the organizational knowledge creation theory. Ontologies structure and organize knowledge, help capture tacit knowledge and transform it into explicit, enabling knowledge integration into information systems and supporting open data publication.* ***Research Method****: We conducted a participative case study to investigate the use of COE.* ***Summary of Results****: Findings suggest that COE contributes to aligning ontology and information system development. However, challenges remain, such as the lack of comprehensive and integrated tool support.* ***Contributions and Impact to IS area****: For researchers, this work advances the state of the art in a novel and relevant topic for developing knowledge-based systems, Semantic Web, and LOD solutions. For practitioners, it offers knowledge about COE and an example that may inspire organizations to implement COE practices to align ontology and information systems development.*

1. Introduction

Semantic technologies like ontologies have been recognized as key enablers of digital transformation in industry [Ragavan et al. 2019, Jaskó et al. 2020, Sandkuhl et al. 2020,

Spoladore and Pessot 2022]. Ontologies can be used to assign semantics to information items, supporting data and knowledge management based on a common and shared understanding and representation [Jaskó et al. 2020, Taher et al. 2022]. By doing so, they enable advanced data integration [El Kadiri and Kiritsis 2015], big data analysis, and development of agent-based services and knowledge-based systems, among others [Salkin et al. 2018]. When combined with Artificial Intelligence, they form a powerful duo that turns data into knowledge and leverages business results [Hagedorn et al. 2020, Earley 2020].

Ontologies have been recognized as conceptual tools of great importance in Computer Science since the end of the 1960s, mainly in areas such as Data Modeling (conceptual modeling) and Artificial Intelligence [Davis 1998, Mealy 1967]. In the last two decades, the interest in ontologies has increased in various segments of Computer Science. This has been motivated by the recognition of the importance of the use of ontologies in semantic tasks (e.g., system integration, data integration, and development of knowledge-based systems), in general, and by its role in Semantic Web development, in particular. For instance, applications that process Linked Data available on the Web are programmed to understand well-known vocabularies (schemas, ontologies) at the time of their writing. Defining new vocabularies for every new data set we publish without properly linking to existing vocabularies makes this data unintelligible to existing applications. Reusing the appropriate vocabularies reduces heterogeneity by relying on ontological agreement [Heath 2011].

An ontology is an artifact that represents knowledge by describing a certain reality with some purpose. Like any artifact, ontologies have a life cycle. They are designed, implemented, evaluated, modified, reused, and so on [Gangemi and Presutti 2009]. Ontology engineers are supported by a wide range of Ontology Engineering (OE) methods and tools. However, building ontologies is still a complex task [Noppens and Liebig 2009, Reginato et al. 2022]. Furthermore, even though using ontologies has been recognized as a successful approach to dealing with semantic and knowledge-related issues, it has not been very common among practitioners yet. One possible explanation is that developing ontologies may be time-consuming and effort-demanding, and, in many cases, the development of a full-fledged ontology before building an ontology-based system is unfeasible [Yildiz and Miksch 2007].

To overcome these challenges, it is necessary to better align the OE process to dynamic, flexible, continuous, and fast processes, like the ones involved in digital transformation. It is needed to continuously gather and prioritize requirements from several users, keep domain experts engaged, deliver ontology modules according to time demands, respond to changing knowledge, and evolve the ontology as needed [Copeland et al. 2012]. Some of these challenges are similar to those faced when developing information systems. Hence, some methods and practices used in that context can be an inspiration.

In the last years, agile and continuous approaches have been used in information systems development. Agile methods enable responding to changing requirements, process adaptability, and customer involvement [Abdelaziz et al. 2017]. Continuous practices, in turn, go beyond and establish a continuous end-to-end flow between customer demands and the fast delivery of a product or service. They provide a more holistic view of the development process and require tight integration between activities and an effec-

tive flow between them [Fitzgerald and Stol 2017].

Some OE approaches have already considered agile practices (e.g., [Blomqvist et al. 2016], [Peroni 2017], [Abdelghany et al. 2019]). However, despite the advancements from a prescriptive approach to one iterative and more aligned with modern needs, they are not enough to address cases where OE activities have a continuous nature. For example, when ontologies are developed for an ontology-based information system whose development process is based on agile and continuous practices, there are unclear or changing ontology requirements and time constraints. In such cases, the information system is developed iteratively, in a continuous and automated flow that goes from the client's needs to the delivery of system modules and their use in the production environment. The process is continuously performed and repeated until the system is ready and fully working in production. Therefore, the OE process needs to be aligned with the system development process. For that, the ontology needs to be developed gradually, and the OE activities need to be integrated into a continuous and automated flow that covers from the system knowledge demands (which can change as the system development progresses) to the delivery of the ontology modules into the production environment (in this case, the ontology production environment is the system development environment).

Concerned with continuity and a better alignment between information systems and ontology development, Salamon and Barcellos (2022) raised the *Continuous Ontology Engineering* (COE) term and proposed the COnTE framework, which defines a set of interrelated processes that describe COE. However, they do not provide fundamental theoretical aspects that provide knowledge and support processes execution. Therefore, in this paper, we extend the knowledge about COE by defining a set of principles to characterize COE. This paper contributes in two ways: to the *state of the art*, by increasing knowledge about COE, a relevant topic for ontology-based systems, Semantic Web and LOD solutions; and to the *state of the practice* by reporting lessons learned from a case study in a large organization that has adopted COE practices to align information systems and ontology development. The lessons can help other organizations that want to apply COE practices.

The paper is organized as follows: Section 2 concerns continuous practices in information systems development and ontology engineering, providing the background for the paper; Section 3 increases knowledge about COE by discussing some core aspects; Section 4 reports on the experience of performing some COE practices in a real setting and discusses some lessons learned; Section 5 discusses related work, and Section 6 concludes the paper.

2. Background

In this section, we provide a theoretical foundation of continuous practices and present basic knowledge about Ontologies and Ontology Engineering.

2.1. Continuous Practices in Information System Development

Characteristics and demands of the modern and digital society have transformed the information system development scenario and presented new challenges, such as the need for faster deliveries, frequent changes in requirements, lower tolerance to failures, and the need to adapt to contemporary business models [Barcellos 2020]. Agile and continuous practices have been adopted to address these challenges.

The continuous and holistic perspective of system development (particularly its software component) has been explored in a recent subarea of Software Engineering called *Continuous Software Engineering* (CSE), which seeks to transform discrete development practices into more iterative, flexible, and continuous ones [Fitzgerald and Stol 2017]. CSE involves agile practices and goes beyond. In CSE, development is iterative and performed in short cycles, customers are proactive, and stakeholders are involved in the process, learning from usage data and feedback. Planning is continuous, as is requirements engineering, which focuses on features, modularized architecture and design, and fast response to changes. There is continuous integration of work, continuous delivery, and continuous deployment of releases. Code is carefully managed through version control, branching strategies, fast commits, code coverage, and code reviews. Quality assurance involves automated tests, regular builds, pull requests, audits, and run-time adaptation. Knowledge is shared through continuous learning, capturing decisions and rationale [Johanssen et al. 2019]. Tooling support is crucial to establish a continuous and fluid end-to-end flow from the customer demand to the delivery and use of the product or service [Fitzgerald and Stol 2017, Barcellos 2020].

2.2. Ontology and Ontology Engineering

An ontology is a formal and explicit specification of a shared conceptualization [Studer et al. 1998]. An important distinction differentiates ontologies as conceptual models, called *reference ontologies*, from ontologies as computational artifacts, called *operational ontologies*. A reference ontology is a conceptual model constructed with the goal of making the best possible description of the domain in reality, regardless of computational properties. Operational ontologies, in turn, are designed with the focus on guaranteeing desirable computational properties and, thus, are machine-readable [Guizzardi 2007].

Developing ontologies involves activities such as requirements identification, ontology formalization, implementation, and evaluation. OE methods organize them and provide guidelines to assist ontology engineers. Some, such as SABiO [Falbo 2014], NeOn [Suárez-Figueroa et al. 2011], and METHONTOLOGY [Fernández-López et al. 1997], follow a prescriptive approach. Others, such as eXtreme Design [Blomqvist et al. 2016], SAMOD [Peroni 2017], and AMOD [Abdelghany et al. 2019], include agile practices (e.g., incremental development in short cycles).

A few works have pointed out the need to approach OE also from a holistic and continuous perspective. Cummings and Stacey (2018) propose Lean Ontology Development (LOD), which is inspired by Lean Startup [Ries 2011] and agile development. It defines a set of principles, such as Continuous Development, Minimum Viable Ontology via Prioritization, Community Evaluation, Ontology as API, Reuse, and Sustainability, to make development activities more flexible, continuous, and aligned with the user needs. Salamon and Barcellos (2022) advance the discussion by advocating that disregarding continuity hampers collaborative ontology development, which is particularly important to build ontologies to support information systems development, requiring integrated tools and automated workflow for an end-to-end flow. Thus, they set the *Continuous Ontology Engineering* (COE) term and propose COntE, a framework containing interrelated processes to define a workflow that enables agility, continuity, integration, and business

alignment in OE.

3. COE Principles

The COntE framework was introduced in [Salamon and Barcellos 2022], providing an overview of COE. However, it lacks foundational knowledge to characterize COE and help organizations adopt it to align ontology and information system development. Hence, in this section, we define a set of principles that characterize COE. COE is much inspired by CSE; thus, we draw on CSE foundations. To exemplify initiatives aligned with COE principles, we also refer to some works related to them.

Iterative Development: COE evolves agile development to achieve a more integrated and continuous process. Hence, COE includes agile practices. The OE process is performed iteratively, by cross-functional teams, in short time boxes, and considering a product backlog containing a prioritized list of work items to develop the ontology (e.g., ontology requirements). At the end of the iteration, a working product (e.g., an ontology module) is delivered to users, establishing close collaboration between them and ontology engineers. Frequent deliveries help ensure that the ontology meets stakeholder needs. Iterative development implies ontology modularization and continuous revision and (re)prioritization of requirements. In each iteration, new requirements are elicited or further detailed, implemented, tested, and a new version of the ontology is made available.

Alignment between development and operation: The recent increase in the number and frequency of deliveries revealed the need for a strong collaboration between development (Dev) and operations (Ops) to avoid long delays in releases to customers. DevOps [Debois et al. 2011] was proposed to align software development and deployment into production. It relies on full automation of build, testing, and deployment to achieve short lead times and, consequently, rapid feedback from users. It involves a set of activities (plan, code, build, test, release, deploy, operate, monitor) that can be organized in pipelines to integrate and accelerate tasks in a continuous and automated cycle. In OE, especially in industry, where ontologies are often developed not to be the end product, but to be coupled with information systems, engineers need to get development and operation closer to reduce lead time and deliver high-quality ontologies. Thus, it is necessary to establish an automated process integrating ontology build, test, and deploy activities (e.g., by dealing with ontology as code and using CI/CD pipelines). In recent years, some approaches have been proposed in this direction by providing a set of automation capabilities to support OE tasks. For example, ROBOT [Jackson et al. 2019] provides features for automated testing, such as reasoning, to ensure that the ontology is logically coherent; querying, to run SPARQL queries and ensure compliance to results; and verification, to ensure that the ontology conforms to a predetermined set of conditions. OnToology [Alobaid et al. 2019] provides automated support for documentation, evaluation, releasing, and versioning. OntoMaven [Paschke and Schäfermeier 2018] helps manage ontology artifacts in distributed repositories, to support ontology reuse from large repositories, ontology life cycle management, and transitive dependency management. Allemang et al. (2021) present a portable open-source infrastructure that manages ontologies as source code and offers features to automatically run ontology unit-level testing as part of the ontology integration and publication process.

Alignment between development and business: Product development should

take the business goals into account to bring the most value out of the developed solution. In CSE, analogous to DevOps, BizDev aims at alignment and continuity, but now between business and development [Fitzgerald and Stol 2017]. Therefore, it is necessary to continuously perform activities to ensure that the product being developed and delivered is aligned with the organization's needs and goals. In OE, the product is an ontology and is no different. In industry, ontologies are often developed involving stakeholders from different departments/areas to leverage IT solutions and bring value out of the data they handle. In COE, BizDev can be achieved by bringing the stakeholders closer to the OE process and giving them important tasks, such as domain knowledge gathering, requirements definition, prioritization and evaluation, and continuous feedback. This fosters collaboration between developers and domain experts. Besides, to add value, the ontology development project must be aligned with business constraints such as budget and schedule. For that, waste must be avoided. According to Lean Thinking [Womack and Jones 1997], any activity that does not add value is a waste and should be removed. Overproduction of unwanted or unneeded features leads to a waste of resources [Womack and Jones 1997, Ohno 1988]. Therefore, collaborating with stakeholders and getting their feedback is key to ensuring the alignment between ontology development and business.

Continuous feedback and experimentation: This aspect emphasizes continuous learning through data and experimentation and relies on the Lean Startups' Build-Measure-Learn principle [Ries 2011], which establishes a continuous feedback cycle comprising earlier implementation of ideas, user response evaluation, lessons learned identification, and idea refinement. The argument for keeping this cycle is that the faster we understand what constitutes value for the users, the better [Fitzgerald and Stol 2017]. In CSE, tests A/B are often used to get user feedback on new features and choose the ones to be added to the product. Moreover, user feedback is continuously and automatically collected during the software use in order to improve it. Continuous feedback is key to enabling continuous engineering. In COE, experiments can be conducted to evaluate the ontology and capture continuous feedback by considering user data and feedback. For example, users (e.g., domain experts, system engineers who use the ontology to develop a system, and data consumers who use the ontology to represent and analyze data) can be asked to evaluate different representations of a domain portion and provide feedback on them. The feedback will help ontology engineers decide which representation to use. Users can also be continuously stimulated to provide feedback on the ontology in use (e.g., "is any necessary concept missing?"). User feedback is key to identifying problems and improvement opportunities, and defining or adjusting plans considering the ontology modules to be developed or adjusted. Experiments can also be used to evaluate OE methods, tools, and practices, which helps to choose the most suitable ones to develop the ontology.

End-to-end flow: A core concept of CSE comes from Lean Thinking [Womack and Jones 1997]: flow. It refers to a connected set of value-creating actions. Establishing a continuous flow means leveraging an end-to-end flow that considers the organizational functions necessary to deliver the desired outcome (e.g., planning, development, deployment, maintenance, and operation) [Fitzgerald and Stol 2017]. This perception of the engineering process as an end-to-end process involving much more than development activities is a crucial difference between agile and lean (continuous) paradigms.

While the former is mostly focused on performing development activities iteratively, the latter aims to "see the whole" and has a very explicit focus on the end-to-end process from customer to delivery [Petersen 2011]. In COE, there must be a continuous flow involving all OE activities, from the ontology conception as the chosen strategy to achieve business goals, passing through development activities (e.g., requirements definition, conceptual model development, implementation), to the evaluation of the ontology use to verify whether the established goals have been achieved. It is important to notice that this flow goes beyond the repetition of development activities organized in iterations. There is a continuous concern with the motivations for developing the ontology and also with what happens after the ontology is delivered (e.g., how it has been used, and how it has impacted its users and the organization). This information is a vital input to the next iterations. Budget, schedule, and other organizational functions must also be involved in the flow. Moreover, in cases where ontologies are built in the context of system information development, the ontology and system development processes must be in harmony. Hence, it is necessary to consider them holistically to properly understand the whole flow between the production of the ontological artifacts and their use by the applications.

Integrated tool support: Continuous engineering depends heavily on automation to enable end-to-end flow and built-in quality. Different from non-continuous approaches, isolated tools are not enough for continuous engineering. The tools need to be integrated in such a way that outcomes produced in a task are inputs for the next ones, in a continuous and automated flow. Moreover, it is necessary to closely monitor quality during the production process (e.g., software organizations often have an indicator that turns red as soon as the build fails and proofing mechanisms to help eliminate errors and identify problems as soon as possible) [Fitzgerald and Stol 2017]. In OE, some tools have been proposed to help built-in quality. For example, Fonseca et al. (2021) propose a plugin in Visual Paradigm[2] that aids in developing conceptual models in OntoUML [Guizzardi 2005, Guizzardi et al. 2021] and relies on proofing mechanisms that avoid or eliminate errors. Chávez-Feria et al. (2022) , in turn, provide a drawing library and a diagram converter to OWL language, helping find and fix errors faster. Besides tools like these, in COE, an integrated suite of tools is needed to support the entire process. However, achieving full automation in OE is challenging. There are some promising initiatives towards an integrated suite. They have tackled mostly implementation, testing, and delivery activities (i.e., DevOps). For example, Allemang et. al (2021) propose a collaborative ontology development platform based on software tools, including CI platforms, version control systems, testing platforms, and review workflows. Matentzoglu et. al (2022), in turn, present the Ontology Development Kit (ODK), which contains a set of executable ontology-engineering workflows and a toolbox to aid in the ontology life cycle management. The workflows are delivered as a Git repository, while the toolbox is delivered as a Docker image and includes the tools necessary to execute the workflows (from command-line utilities to ontology pipeline tools such as ROBOT [Jackson et al. 2019] and dosdp-tools [Osumi-Sutherland et al. 2017]). Similarly, VoCol[3][Halilaj et al. 2016] supports collaborative ontology development and uses Git repositories to maintain ontology-related files. It supports ontology evaluation, documentation, visualization, and publication.

[2]https://www.visual-paradigm.com/

[3]https://github.com/vocol/vocol

Holistic continuous improvement: To produce better outcomes, a continuous improvement cycle is needed. Improvements can happen incrementally or through radical steps to start the transformation and, thus, follow with incremental improvements. The improvement must be holistic, i.e., consider all the organizational functions involved in the product development, as well as other stakeholders. Otherwise, tension points with the not considered parts can cause barriers to improvements [Fitzgerald and Stol 2017]. Therefore, the focus of improvement actions must go beyond ontology development activities and take into account all other activities, their interactions, and the involved parties. Continuous improvement depends on continuous feedback from stakeholders (about both product and process). Hence, it is crucial to regularly check if stakeholders feel themselves as an important part of the process. Moreover, holistic continuous improvement relies on communication among all the organizational functions involved in product development, so that strengths and weaknesses in the OE process can be identified.

4. Investigating COE in Practice

In Section 3 we presented the main COE principles. Now, we discuss COE in practice. In this section, we report a participative case study carried out at BASF. Participative case study was selected as the research method in this study because two researchers work at BASF and were participants in the process being observed [Baskerville 1997].

BASF is a European multinational company and the largest chemical producer in the world. Its headquarters are located in Germany, with over 111.000 employees worldwide. Like many large organizations, in BASF there are data silos resulting from data stored in different systems. Data silos can lead to inefficiencies, inconsistencies, and missed opportunities for data analysis and insights. They pose a challenge for organizations that want to leverage their data assets and gain a competitive edge [Kumar 2023]. To deal with this scenario, BASF has developed and used ontologies for describing data across the organization and interoperating systems to leverage decision-making and improve data quality.

The study **goal** was to investigate the adoption of COE principles in practice. Aligned with this goal, the study seeks to answer the following **research question**: How have COE principles been adopted in an organization?

The **procedure** followed in the study included four steps: (i) explaining the COE principles to the study participants (a team of employees, who work developing ontologies to support the development of computational solutions); (ii) gathering information about the ontology development approach adopted at BASF; (iii) applying a questionnaire and perform interviews to obtain feedback from the participants regarding the adopted practices; (iv) analyzing collected data and recording lessons learned.

The **instrument** used in the study consisted of a form created in Microsoft Forms to collect the participants' answers. It was composed of a consent term for participation in the study and two sections of questions. The first section contained six closed questions to characterize the participant. The second section was composed of fourteen questions related to the research question. Eleven were closed questions. The participant was asked to justify their answer or provide clarifications in three of them. There was also an open question for the participant to provide general comments and suggestions.

The study's **participants** needed to be people from the organization who have

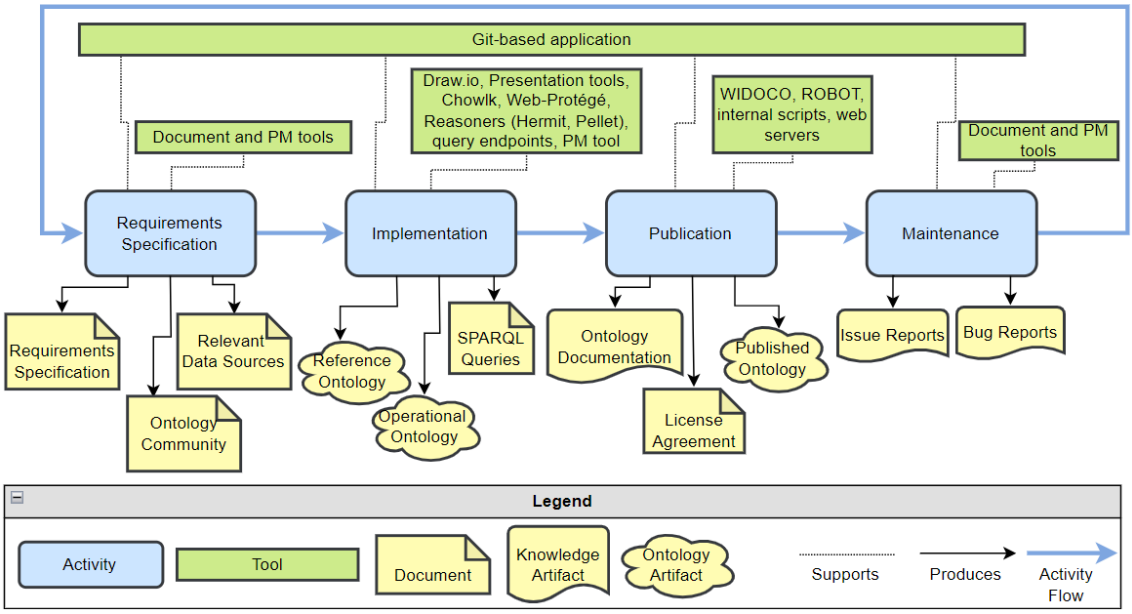

Figure 1. Overview of the OE process at BASF.

worked in the organization's ontology engineering process and its interaction with the system development process. Nine employees participated in the study. Regarding the academic background, four have a doctoral degree, three have a master's degree, and two are doctoral students. Regarding practical experience, four have worked with ontologies for 5 to 10 years, two for 3 to 5 years, two for 1 to 3 years, and one for less than a year.

Next, we present an overview of the OE process adopted in BASF (Section 4.1), analyze it in light of COE principles (Section 4.2), and summarize the main lessons derived from the collected data, experience at the organization, and feedback from participants (Section 4.3). Due to confidentiality reasons, the collected data cannot be published.

4.1. OE process adopted at BASF

To ensure that ontology development follows common rules, BASF defined a process model that provides a set of principles, standards, and best practices to guide ontology development. When following it, BASF has implemented some practices towards COE. Figure 1 presents an overview of the model. The figure was created by the authors based on the steps taken by the practitioners when following the company's process model. For simplification reasons, we omit some steps (e.g., in the first iteration, there is a kickoff step to launch the project, present the starting points and initial timeline, and create the project repository on a git-based application). Following, we briefly describe the macro-steps.

Requirements Specification: In the first iteration, ontology engineers identify the ontology community, which comprises the project stakeholders. Initial conversations result in an overview of the domain to be covered. Requirements are defined through use cases and competency questions (CQs) that are recorded in a document. The ontology portion to be developed in the iteration and the resources to be considered are identified. In each iteration, the use cases and CQs to be addressed are selected, reviewed, and new ones are defined according to the stakeholders' needs. Documents are stored on the git repository and a project management tool is used to store requirements and track their

status and prioritization.

Implementation: Ontology engineers gather domain knowledge from experts and develop the reference ontology for the module to be addressed. The conceptual models (diagrams) are created, for example, in Draw.io[7] and Chowlk tools[8][Chávez-Feria et al. 2022]. Once the reference ontology is validated (the requirements and the conceptual patterns that answer them are evaluated by domain experts in meetings using, for example, slide presentations), the operational ontology is implemented. For that, the diagrams are exported and converted into a machine-readable ontology language. Alternatively, the ontology can be implemented on an ontology editor (e.g., Web-Protégé[9][Musen 2015]), or programmatically (e.g., via Python scripts). The ontology module produced is integrated with those already developed. Modular development can be made in a single file (i.e., integration involves one master ontology) or using different files and repositories (i.e., integration involves one master ontology and other ontology branches/files). Tests are carried out to ensure fulfillment of requirements, compliance with standards, consistency regarding non-contradiction, and syntax, semantics, and modeling correctness. SPARQL queries and reasoners (e.g., Hermit[10]) are used to evaluate the operational ontology. Diagrams and the operational ontology are stored on the git repository to allow automated tasks. The SPARQL queries derived for each CQ are recorded on the project management tool for traceability and unit testing.

Publication: A license document is produced defining the terms of the use of the ontology. Tools like WIDOCO [Garijo 2017] are used to automatically generate the ontology documentation. A merge request is made and the files are evaluated by the ROBOT tool [Jackson et al. 2019] and some internally developed scripts. Verification includes analysis of ontology metadata and quality assurance. Validation involves the execution of SPARQL queries validating the requirements and execution of reasoning tools. When the artifacts pass the evaluations, the ontology is published as an *ontology-as-a-service*, which enables the use of the ontology in different use cases (it can be searched for, downloaded, or accessed via API).

Maintenance: Once the ontology is published, users can use it and report bugs that require corrective maintenance, which are recorded as Bug Reports. Users can also request minor changes (e.g., add comments, synonyms, alternative labels in another language), which are recorded as Issue Reports on the git repository. Moreover, based on user feedback, ontology engineers can identify the need to address new requirements in new versions of the ontology.

4.2. Analyzing BASF OE practices in light of COE

In this section, we discuss the OE practices adopted in BASF in light of the COE principles introduced in Section 3.

BASF adopts *iterative development*, with small and cross-functional teams (e.g., including ontology engineers and domain experts) building modular ontologies based on an ontology backlog that contains prioritized CQs. Time boxes are short but may have

[7]https://www.drawio.com/

[8]https://chowlk.linkeddata.es/notation.html and https://chowlk.linkeddata.es/

[9]https://protege.stanford.edu/

[10]http://www.hermit-reasoner.com/

different durations depending on the complexity of the module to be developed and the availability of domain experts to answer questions and participate in sessions to evaluate the knowledge represented in the ontology module. Requirements are reviewed and re-prioritized by the experts at each iteration to accommodate new understandings of the domain. At the end of the iteration, the built module is made available through a service that enables ontology use.

To ensure *alignment between development and business*, ontologies are developed considering use cases that describe specific applications (e.g., data analytics, software development) desired by the organization. Domain experts and people from other organizational functions (e.g., system engineers who use the ontologies in their projects) are given responsibilities in the OE process to foster closer collaboration. For example, domain experts participate in meetings to prioritize requirements and evaluate the produced artifacts. System engineers can be asked to evaluate the ontology in use. This allows identifying expectation misalignments as soon as they occur and taking action to realign them. Especially when developing very complex ontologies, which may take many iterations to complete, BASF finds it imperative to keep in touch with domain experts and application owners to ensure the ontology remains a priority and there is a clear plan for its use. To meet budget and time constraints, the OE process follows a standard workflow that aims at reducing waste with overproduction. Moreover, as the ontology development progresses, alignment with expected costs and time is monitored and reviewed if necessary.

To achieve *alignment between development and operations*, BASF implements DevOps practices. For example, to ensure fast delivery of ontology modules and feedback, a CI/CD pipeline is used. Quality is monitored through internally defined metrics. A culture of shared responsibility for the delivery of quality ontologies is institutionalized.

Despite BASF capturing user feedback, this is not done automatically and continuously. Hence, BASF has not yet implemented *continuous feedback and experimentation*. Feedback on the ontology in use is mainly provided during the Maintenance step, when users point out bugs, issues, and new requirements. Users can also report needs to be addressed in future projects. The provided feedback helps improve the ontology under development and contributes to identifying improvement opportunities in the OE process, tools, and methods used, among others. Feedback is also collected during the Requirements and Implementation steps to ensure that the developed modules correctly represent the desired knowledge.

BASF has taken the first steps toward an *end-to-end flow*. The adopted OE process organizes activities in a flow in which outputs of one activity are used as inputs for the next, involves different organizational functions, manages stakeholder needs and feedback, evaluates ontology usage, and is supported by automation CI/CD pipelines (DevOps). However, some aspects can be improved. For example, user feedback could be continuously and automatically captured, and the alignment between the ontology project and the organization's strategy, budget, and deadlines could be more explicit (e.g., through activities devoted to it). Moreover, the process still lacks automation in some activities. It relies on an integrated infrastructure comprising several tools and guidelines, and on other tools outside this infrastructure. However, *integrated tool support* still has some limitations regarding the tools' scope and the integration itself. Concerning scope, there is no tool devoted to ontology requirements activities and continuous/automatic feedback. Re-

garding integration, the integrated infrastructure includes pipelines for automatic verifications, validations, documentation, and deployment, Web-Protégé, endpoints for querying and searching, etc. Other tools (e.g., project management tools) are not part of the integrated infrastructure provided by BASF, but are used in several projects to complement the infrastructure features. As they are not part of the infrastructure, they work in isolation, hampering full integration. The lack of full tool support impacts the end-to-end flow because of the strong bond between flow and automation.

Aiming at *holistic continuous improvement*, in addition to user feedback, BASF collects feedback from other stakeholders, considering all organizational functions involved in the OE process. They are asked to participate in surveys and express their opinion about the ontologies, built solutions, OE process, and their participation in it. This helps get a holistic view of the OE process, its products, and how they are perceived in the organization. Moreover, there are frequent meetings involving several development teams when they share ideas, feedback, and so on, and join efforts to improve OE practices, tools, and guidelines.

4.3. Perceptions from the Field and Lessons Learned

In this section, we share lessons learned identified from the collected data, feedback from participants, and our experience at the organization.

Providing a comprehensive and integrated suite of tools is needed but challenging: Automation is key for enabling a workflow throughout the OE process. However, achieving a comprehensive and integrated toolset is not trivial. As seen in BASF, despite having a suite integrating several tools, organizations may need to incorporate other tools to get more comprehensive support. However, tool integration is challenging and may take time. OE involves activities that handle artifacts of different natures (e.g., text, model, code), which makes it difficult to achieve full integration. When using tools to support implementation, testing, and publication, artifacts are machine-readable, which facilitates integration. In contrast, when dealing with tools to support requirements elicitation and conceptual modeling, it is harder to integrate requirements recorded in documents and management tools with modeling tools. Hence, organizations can spend effort on integrating the tools or use some of them separately (as is currently the BASF case) and manage integration manually, considering the produced artifacts (e.g., by ensuring that the requirements recorded in a document are addressed in the ontology models and code). Considering the difficulties in achieving a fully automated end-to-end flow, organizations should choose the tools suitable for their needs (i.e., those that are most beneficial, deliver value faster, and build up from that) and integrate them as much as possible. For the ones not integrated into the automated flow, it is necessary to set rules and procedures to ensure integration through the produced artifacts. Besides achieving a satisfactory tooling suite for ontology development itself, it is also important and a current challenge to ensure that ontology development pipelines and resulting artifacts can be seamlessly integrated into information systems pipelines, providing a unified view of the workflow. The artifacts resulting from ontology development must be available for consumption by other systems to allow exchange during the steps of both processes.

Defining use cases helps align ontology development and business: Use cases are useful to clarify why the ontology is developed (e.g., support data analytics, structure

knowledge for system development) and how it will be used. Defining them helps verify if the ontology development fits the business needs. To identify use cases, some key business questions can be asked, such as "what benefits (e.g., in terms of time, effort, money) will the ontology bring to the company?", "which pains will the ontology help solve?", "will the ontology impact the core business or a business unit?" or "what if we do not develop the ontology?". Such questions are useful to guide stakeholders in identifying use cases where the ontology will deliver business value and help avoid waste and overproduction. These use cases also support the development of ontology-based systems, helping stakeholders visualize how they want to consume the ontology inside the systems and discover new system requirements. This ensures the ontology usage will bring value to the business by continuously improving the system. On the other side, the system usage can also bring new requirements to the ontology, creating a living and evolving ecosystem.

Planning needs to be rechecked with stakeholders regularly to maintain development and business alignment: Business needs can change rapidly, leading to readjustments in project priorities and budget. In ontology development, these changes may cause, for instance, domain experts to reduce their participation, ontology engineers to be reallocated, and requirements considered key to become secondary. Often, business needs are discussed internally in the business unit. Hence, it is crucial to stay close to business stakeholders to follow changes in business needs and priorities so that ontology development can be re-planned accordingly. Similarly, the use cases initially identified should be revisited in each iteration. It is common that, as ontology development progresses, stakeholders have clearer ideas of ontology usage. Therefore, use cases, requirements, and prioritization need to be revisited to enable development plans to be adjusted promptly, contributing to better resource management. Moreover, ontology development planning and prioritization must be coupled with information systems development planning and prioritization to ensure that the ontology modules being delivered are in tune with the information needed by the system at each point in time, in order to organize and streamline the deliveries.

A well-established flow between the development of reference and operational ontologies contributes to the continuous evolution of these artifacts: Translation of a reference ontology into an operational one should be as smooth and automated as possible to facilitate continuous evolution. The operational ontology should not be created from scratch by a human mirroring the knowledge from the reference ontology, which is error-prone and difficult to maintain. An environment that enables the reference ontology model to be automatically transformed into the operational ontology is necessary. This can be achieved by using tools such as Chowlk or model transformation approaches (e.g., Model-Driven Development) that translate the reference model into an operational model by following transformation rules. When this transformation is seamless, operational artifacts are produced faster and can be tested more frequently, contributing to continuous built-in quality and evolution. As the ontology acts as a pillar for developing ontology-based systems, faster and reliable delivery of operational artifacts impacts the system development. Additionally, the integration of the new ontology modules can be tested faster and more frequently, improving the quality and evolution of the information system.

Continuity requires culture change: Besides technical changes, COE involves

a cultural change of a well-established end-to-end flow and jointing responsibility to deliver high-quality and ready-to-use ontologies. Continuity requires the involvement of people from different organizational functions, and they are not mere spectators but responsible for the results. Achieving cultural change may not be easy. Starting with small changes and gradually adding new ones is a good approach. Once people experience the benefits, they become disseminators, helping to spread the culture to other members [dos Santos Júnior et al. 2022]. At BASF, the changes started in the core development activities. People perceived the benefits and became eager for the next changes, which have been implemented gradually. Furthermore, outreach activities are useful to create a collective awareness of the importance of adopting new OE strategies. In addition, commitment from the top levels of the company hierarchy is necessary to establish the new strategies. Ontology development needs to become a business value to ensure a community commitment to a long-lived ontology journey. This journey travels through the valley of information systems development as well, where the integration of ontologies must be seen by the roles with decision-making capabilities as a high-value weapon for the delivery of interoperable and sustainable systems. Ontologies ensure that the data foundations are solid to enable the systems to go through the changes necessary to evolve and continue to leverage the business capabilities as the world and market change.

Clear communication is essential: Communication problems are a recurring bottleneck in systems development. For instance, gathering feedback and reaching an agreement among different stakeholders often involves communication challenges (e.g., conflicts and expectations management). With COE, communication challenges can be accentuated because many stakeholders are involved in the OE process, from different organizational functions and with different backgrounds, viewpoints, and needs. To address communication challenges, it is necessary to establish communication strategies and inform the interested parties. The strategies must address all stakeholders and consider the entire OE process. For example, as at BASF, after identifying the stakeholders, meetings can be performed with each stakeholder or with the ones playing the same role to get their needs and understandings. After analyzing all the perspectives, ontology engineers can present the results to the stakeholders and discuss conflicts and disagreements to reach a consensus on what needs to be met.

The ontology concepts can also be evaluated by the stakeholders individually to reduce bias. The consensual concepts can be communicated, and the divergent ones can be discussed with all. Another communication challenge stems from the domain complexity, which may lead ontology engineers not to ask the right questions to the right people. SE requirements engineering techniques can help in this matter. Likewise, works such as [Norris et al. 2021], which describes techniques from behavioral sciences to engage experts in ontology development, can be applied to ease communication. After all, communication demands soft skills to understand stakeholders' profiles and behaviors and find the best way to approach them. Finally, as many people are involved in the end-to-end flow that aims at delivering valuable ontologies, it is paramount to communicate the OE process clearly to all involved parties. This involves representing the process graphically to provide an overview of it and describing its phases, activities, involved roles, inputs, outputs, and supporting tools, as has been done at BASF.

In the context of systems development, communication needs to be clear, effi-

cient, and frequent among the multidisciplinary team to ensure that all those involved in the development processes are aware of the artifacts being delivered, timelines, and how the processes align and cross each other. When clear communication is not enforced or communication channels do not exist, both processes can diverge, creating solutions that are not valuable for the business and wasting resources.

Continuous experimentation and continuous feedback are not usual in OE: When developing an information system, it is common to get automatic and continuous feedback from users during application usage. In OE, in turn, collecting feedback in such a way is not usual because the ontology does not run or is not used as a software application. In OE, feedback is often captured to support product decisions (e.g., ontology enhancements) and feed new business (e.g., prioritization of new ontology projects). However, capture is not necessarily continuous or automated. Collecting feedback requires commitment from all organizational functions involved in the OE process. Getting closer to the way CSE addresses user feedback might improve feedback capture in COE. But it is still necessary to investigate this in-depth. Concerning experimentation, it can be beneficial to decide on methods and tools to be used. However, if they have already been defined by the organization, there is less opportunity to experience new ones. Even so, depending on the project or team characteristics, other methods or tools may be more suitable and should be considered. Thus, providing an environment that allows for running experiments and collecting feedback is desirable. Moreover, when addressing continuous feedback in ontology-based information systems, the ontology can be influenced by the system feedback, as a side effect of the changes made in the system due to the feedback. Furthermore, when adapting and continuously experimenting on the system, the ontology will serve as a reliable backbone to ensure that changes to the system or new features remain coherent across modules and do not hamper global consistency.

5. Related Work

Although there are several works that propose OE methods aiming at fast and iterative development, and reinforce the need for integrated tools to support the OE process (e.g., [Poveda-Villalón et al. 2022] and [Allemang et al. 2021]), only a few are concerned with the integration of OE into systems development. Besides our work, only [Cummings and Stacey 2018] and [Salamon and Barcellos 2022] advocate the connection between the development of an ontology and the application that intends to use it. Cummings and Stacey (2018) propose Lean Ontology Development (LOD), providing a set of principles to adapt agile and lean software development to ontology development. Salamon and Barcellos (2022), in turn, propose a framework composed of a set of interrelated processes that describe COE.

Different from our work, the cited works do not define fundamental principles that provide knowledge to support OE continuously and aligned with the development of ontology-based systems. Moreover, they do not discuss aspects such as the importance of an end-to-end flow connecting different organizational functions and alignment between development and business, among others. Our work focuses on the overall process, providing the conceptual foundations to understand COE and improve both the ontology development process and the connection between the ontology and the system development processes, and discussing aspects that flow from one to the other and around the organization. With this work, we complement the LOD principles

[Cummings and Stacey 2018] and provide knowledge that is not available in the COntE framework [Salamon and Barcellos 2022], which is only focused on processes.

Another distinction of our work is that we present a real case describing the application of continuous practices in projects that involve both ontology and information systems development. Additionally, we bring a set of actionable lessons learned from this case that can be implemented in organizations that want to improve their processes.

6. Final Considerations

Data, as the core resource of corporate digital transformation, is becoming increasingly important in decision-making, enabling companies to more accurately grasp market dynamics, optimize resource allocation, and improve operational efficiency, thereby gaining a competitive edge in fierce market competition [Yang 2025]. To maximize its value, it must be interconnected, and first, it is important to strengthen its foundations. For that, we can use ontologies. However, developing ontologies and integrating them into information systems that handle the data is not trivial. To ease this load, both development processes must be carried out in harmony. Thus, we need to improve the alignment between ontology and systems development. For that, in this paper, we discussed principles of COE, a novel approach to ontology engineering, its application in a large company, and lessons learned from that experience.

This paper contributes to advancing research on a recent topic. The implications for researchers include opportunities to advance the topic based on the identified gaps and challenges. For practitioners, in turn, the implications include the possibility of applying the principles, practices, and lessons discussed in this paper to implement OE practices or improve existing ones. This work is a pioneer and opens the road to discussing the need to integrate OE and system development better and provide advances in this matter.

As discussed, COE evolves agile development by preconizing an end-to-end flow that goes beyond development activities and considers a more comprehensive view of the OE processes, involving several organizational functions. By doing that, COE includes agile practices plus other practices (e.g., DevOps) and concerns. This perspective is particularly necessary for industry settings that adopt continuous practices in the development of ontology-based systems. COE is a recent topic and further research is necessary to address aspects such as tools integration, culture change (e.g., shared responsibility, focus beyond the development activities), and soft skills development.

As for the limitations of this work, we must highlight its generalization capability. The information shared in the paper is case-based (i.e., we reported the experience of a single organization). A recognized limitation in this context is the ability to generalize from the case-specific to different cases. Wieringa and Daneva (2015) argue that in such cases generalization can be established for similar cases and, although are not universal, they are useful in practice. Another limitation refers to the participation of two authors who work at the company. On one hand, this helps get a deeper understanding of how the company works and can facilitate collecting information from other employees. On the other hand, bias cannot be eliminated. Therefore, new studies are necessary to produce robust evidence on the use of COE and its influence on information systems development.

In this sense, as future work, we plan to study other organizations. It should also be noted that there are still no well-established practical guidelines on how to carry out

COE. The COntE framework proposed by Salamon and Barcellos (2022) provides an overview of COE processes, but does not detail them. Therefore, we intend to describe guidelines and develop a diagnostic instrument to support organizations in implementing COE practices. As future steps at BASF, we plan to address gaps and improvement opportunities to help the company evolve in the COE journey and improve the integration between ontology and system development.

Acknowledgments

This research is supported by the Coordination for the Improvement of Higher Education Personnel - CAPES Brazil (Finance Code 001), the Espírito Santo Research and Innovation Support Foundation - FAPES (Processes 2023-5L1FC, 2022-NGKM5, 2021-GL60J, and T.O. 1022/2022), and the National Council for Scientific and Technological Development - CNPq (Process 304787/2025-6).

References

Abdelaziz, A., Darwish, N. R., and Hefny, H. A. (2017). Towards a machine learning model for predicting failure of agile software projects. *Int. Journal of Computer Applications*, 168(6):22–26.

Abdelghany, A. S., Darwish, N. R., and Hefni, H. A. (2019). An agile methodology for ontology development. *Int. Journal of Intelligent Engineering and Systems*, 12(2):170–181.

Allemang, D., Garbacz, P., Gradzki, P., Kendall, E., and Trypuz, R. (2021). An infrastructure for collaborative ontology development. In *Formal Ontology in Information Systems*, volume 344 of *Frontiers in Artificial Intelligence and Applications*, pages 112–126. IOS Press.

Alobaid, A., Garijo, D., Poveda-Villalón, M., Santana-Perez, I., Fernández-Izquierdo, A., and Corcho, O. (2019). Automating ontology engineering support activities with ontoology. *Journal of Web Semantics*, 57:100472.

Barcellos, M. P. (2020). Towards a framework for continuous software engineering. In *Proc. of the XXXIV Brazilian Symposium on Software Engineering*, pages 626–631.

Baskerville, R. L. (1997). Distinguishing action research from participative case studies. *Journal of systems and information technology*, 1(1):24–43.

Blomqvist, E., Hammar, K., and Presutti, V. (2016). Engineering ontologies with patterns-the extreme design methodology. *Ontology Engineering with Ontology Design Patterns*, 25:23–50.

Chávez-Feria, S., García-Castro, R., and Poveda-Villalón, M. (2022). Chowlk: from uml-based ontology conceptualizations to owl. In *European Semantic Web Conference*, pages 338–352. Springer.

Copeland, M., Brown, A., Parkinson, H. E., Stevens, R., and Malone, J. (2012). The swo project: A case study for applying agile ontology engineering methods for community driven ontologies. *Proc. of the 3rd Int. Conf. on Biomedical Ontology (ICBO 2012)*.

Cummings, J. and Stacey, D. (2018). Lean ontology development: An ontology development paradigm based on continuous innovation. In *Proc. of the 10th Int. Joint Confer-*

ence on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2018) - Volume 2: KEOD, pages 365–372. SciTePress.

Davis, E. (1998). Naive physics perplex. *AI magazine*, 19(4):51–79.

Debois, P. et al. (2011). Devops: A software revolution in the making. *Journal of Information Technology Management*, 24(8):3–39.

dos Santos Júnior, P. S., Barcellos, M. P., and Calhau, R. F. (2022). First step climbing the stairway to heaven model-results from a case study in industry. *Journal of Software Engineering Research and Development*, 10:1–5.

Earley, S. (2020). *The AI-powered enterprise: Harness the power of ontologies to make your business smarter, faster, and more profitable*. LifeTree Media.

El Kadiri, S. and Kiritsis, D. (2015). Ontologies in the context of product lifecycle management: state of the art literature review. *Int. Journal of Production Research*, 53(18):5657–5668, Taylor & Francis.

Falbo, R. d. A. (2014). Sabio: Systematic approach for building ontologies. In *1st Joint Workshop ONTO.COM / ODISE on Ontologies in Conceptual Modeling and Information Systems Engineering (FOIS 2014)*.

Fernández-López, M., Gómez-Pérez, A., and Juristo, N. (1997). Methontology: from ontological art towards ontological engineering. In *Papers from the 1997 AAAI Spring Symposium*.

Fitzgerald, B. and Stol, K.-J. (2017). Continuous software engineering: A roadmap and agenda. *Journal of Systems and Software*, 123:176–189.

Fonseca, C. M., Sales, T. P., Viola, V., da Fonseca, L. B. R., Guizzardi, G., and Almeida, J. P. A. (2021). Ontology-driven conceptual modeling as a service. In *Joint Ontology Workshops, JOWO 2021*.

Gangemi, A. and Presutti, V. (2009). Ontology design patterns. In *Handbook on ontologies*, pages 221–243. Springer, Berlin, Heidelberg.

Garijo, D. (2017). Widoco: a wizard for documenting ontologies. In *The Semantic Web–ISWC 2017, LNCS*, volume 10588, pages 94–102. Springer.

Guizzardi, G. (2005). *Ontological foundations for structural conceptual models*. Phd thesis, Telematica Instituut Fundamental Research, University of Twente, The Netherlands. ISBN 90-75176-81-3.

Guizzardi, G. (2007). On ontology, ontologies, conceptualizations, modeling languages, and (meta)models. In *Proc. of the 2007 conference on Databases and Information Systems IV: Selected Papers from the Seventh International Baltic Conference DBIS'2006*, pages 18–39. IOS Press.

Guizzardi, G., Fonseca, C., Almeida, J., Sales, T., Benevides, A., and Porello, D. (2021). Types and Taxonomic Structures in Conceptual Modeling: A Novel Ontological Theory and Engineering Support. *Data & Knowledge Engineering*, 134:101891.

Hagedorn, T., Bone, M., Kruse, B., Grosse, I., and Blackburn, M. (2020). Knowledge representation with ontologies and semantic web technologies to promote augmented and artificial intelligence in systems engineering. *Insight*, 23(1):15–20.

Halilaj, L., Petersen, N., Grangel-González, I., Lange, C., Auer, S., Coskun, G., and Lohmann, S. (2016). Vocol: An integrated environment to support version-controlled vocabulary development. In *Knowledge Engineering and Knowledge Management. EKAW 2016. LNCS*, volume 10024, pages 303–319. Springer.

Heath, T. (2011). Linked data: Evolving the web into a global data space. *Morgan & Claypool*.

Jackson, R. C., Balhoff, J. P., Douglass, E., Harris, N. L., Mungall, C. J., and Overton, J. A. (2019). Robot: a tool for automating ontology workflows. *BMC bioinformatics*, 20:1–10.

Jaskó, S., Skrop, A., Holczinger, T., Chován, T., and Abonyi, J. (2020). Development of manufacturing execution systems in accordance with industry 4.0 requirements: A review of standard-and ontology-based methodologies and tools. *Computers in industry*, 123:103300.

Johanssen, J. O., Kleebaum, A., Paech, B., and Bruegge, B. (2019). Continuous software engineering and its support by usage and decision knowledge: An interview study with practitioners. *Journal of software: evolution and process*, 31(5).

Kumar, S. (2023). Data silos a roadblock for AIOps. *arXiv preprint arXiv:2312.10039*.

Matentzoglu, N., Goutte-Gattat, D., Tan, S. Z. K., Balhoff, J. P., Carbon, S., Caron, A. R., Duncan, W. D., Flack, J. E., Haendel, M., Harris, N. L., et al. (2022). Ontology development kit: a toolkit for building, maintaining and standardizing biomedical ontologies. *Database*, 2022:baac087.

Mealy, G. H. (1967). Another look at data. In *Proc. of the November 14-16, 1967, fall joint computer conference*, pages 525–534.

Musen, M. (2015). The protégé project: A look back and a look forward. *AI Matters*, 1(4).

Noppens, O. and Liebig, T. (2009). Ontology patterns and beyond: towards a universal pattern language. In *Proc. of the 2009 Int. Conf. on Ontology Patterns - Volume 516*, pages 179–186.

Norris, E., Hastings, J., Marques, M. M., Mutlu, A. N. F., Zink, S., and Michie, S. (2021). Why and how to engage expert stakeholders in ontology development: insights from social and behavioural sciences. *Journal of Biomedical Semantics*, 12.

Ohno, T. (1988). *Toyota Production System: Beyond Large-Scale Production*. Productivity Press.

Osumi-Sutherland, D., Courtot, M., Balhoff, J. P., and Mungall, C. (2017). Dead simple owl design patterns. *Journal of biomedical semantics*, 8.

Paschke, A. and Schäfermeier, R. (2018). OntoMaven: Maven-based ontology development and management of distributed ontology repositories. In *Synergies Between Knowledge Engineering and Software Engineering*, volume 626 of *Advances in Intelligent Systems and Computing*, pages 251–273. Springer.

Peroni, S. (2017). A simplified agile methodology for ontology development. In *OWL: Experiences and Directions – Reasoner Evaluation, LNCS*, volume 10161, pages 55–69. Springer.

Petersen, K. (2011). Is lean agile and agile lean?: a comparison between two software development paradigms. In *Modern software engineering concepts and practices: Advanced approaches*, pages 19–46. IGI Global.

Poveda-Villalón, M., Fernández-Izquierdo, A., Fernández-López, M., and García-Castro, R. (2022). Lot: An industrial oriented ontology engineering framework. *Engineering Applications of Artificial Intelligence*, 111:104755.

Ragavan, S. K. V., Khamis, A. M., Fiorini, S. R., Carbonera, J. L., Alarcos, A. O., Habib, M. K., Gonçalves, P. J. S., Li, H., and Olszewska, J. I. (2019). Ontologies for industry 4.0. *Knowl. Eng. Rev.*, 34:e17.

Reginato, C. C., Salamon, J. S., Nogueira, G. G., Barcellos, M. P., Souza, V. E. S., Monteiro, M. E., and Guizzardi, R. (2022). A goal-oriented framework for ontology reuse. *Applied Ontology*, 17(3):365–399.

Ries, E. (2011). The lean startup. *New York: Crown Business*, 27:2016–2020.

Salamon, J. S. and Barcellos, M. P. (2022). Towards a framework for continuous ontology engineering. In *XV Seminar on Ontology Research in Brazil (ONTOBRAS 2022)*, pages 158–165.

Salkin, C., Oner, M., Ustundag, A., and Cevikcan, E. (2018). A conceptual framework for industry 4.0. *Industry 4.0: managing the digital transformation*, pages 3–23.

Sandkuhl, K., Shilov, N., and Smirnov, A. (2020). Facilitating digital transformation: success factors and multi-aspect ontologies. *Int. Journal of Integrated Supply Management*, 13(4):376–393.

Spoladore, D. and Pessot, E. (2022). An evaluation of agile ontology engineering methodologies for the digital transformation of companies. *Computers in Industry*, 140:103690.

Studer, R., Benjamins, V. R., and Fensel, D. (1998). Knowledge engineering: Principles and methods. *Data & knowledge engineering*, 25(1-2):161–197.

Suárez-Figueroa, M. C., Gómez-Pérez, A., and Fernández-López, M. (2011). The neon methodology for ontology engineering. In *Ontology engineering in a networked world*, pages 9–34. Springer.

Taher, A., Vahdatikhaki, F., and Hammad, A. (2022). Formalizing knowledge representation in earthwork operations through development of domain ontology. *Engineering, Construction and Architectural Management*, 29(6):2382–2414.

Wieringa, R. and Daneva, M. (2015). Six strategies for generalizing software engineering theories. *Science of computer programming*, 101:136–152.

Womack, J. P. and Jones, D. T. (1997). Lean thinking—banish waste and create wealth in your corporation. *Journal of the Operational Research Society*, 48(11):1148–1148.

Yang, X. (2025). The role of data-driven decision-making in corporate digital transformation. *Studies in Social Science & Humanities*, 4(3):18–25.

Yildiz, B. and Miksch, S. (2007). Ontology-driven information systems: Challenges and requirements. In *Int. Conf. on Semantic Web and Digital Libraries. Indian Statistical Institute Platinum Jubilee Conference Series*, pages 35–44.