Towards a Framework for Continuous Ontology Engineering

Jordana Sarmenghi Salamon, Monalessa Perini Barcellos

Ontology and Conceptual Modeling Research Group (NEMO), Department of Computer Science, Federal University of Espírito Santo, Vitória – ES, Brazil

Abstract
Ontologies are recognized as a successful approach to solve semantic and knowledge-related problems. However, even providing many benefits, ontologies have still not reached broad use in industry. Often, the benefits are undermined by the time and resources needed to develop the ontologies and by the difficulties to maintain and evolve existing ontologies. Moreover, traditional ontology engineering methods may fail to deal with industry needs and requirements changes. To overcome these issues, we can look at Software Engineering, where a recent field of study, Continuous Software Engineering (CSE), aims at delivering valuable software through agility, continuity, integration, and alignment to business. Although some ontology engineering approaches have been based on agile principles, the continuous nature of ontology engineering activities has still not been considered. In this paper, we introduce Continuous Ontology Engineering and propose COntE, a framework containing a set of processes connected by data and information flows to be performed in an integrated, continuous and iterative way. We also raise some research questions and make some discussions about them.

Keywords
Ontology, Ontology Engineering, Framework, Agile, Continuous

1. Introduction

An ontology is an artifact that represents knowledge by describing a certain reality with some purpose. As any artifact, ontologies have a lifecycle. They are designed, implemented, evaluated, modified, reused, etc. [1]. Ontology engineers are supported by a wide range of ontology engineering (OE) methods and tools. However, building ontologies is still a complex task even for experts [2].

Although there are several methods to aid in OE, many of them fail to provide details about the activities to be performed and techniques to be employed along with a defined lifecycle model [3]. Furthermore, there are challenges in OE that still need to be addressed. For example, even though ontologies support the development of information systems to solve interoperability and knowledge-related problems, using ontologies has not been yet a common approach among practitioners. The main reason is perhaps that it may take more time for a developer to build an ontology-driven system than a usual one since it may be necessary to develop the ontology and only then to develop the system [4]. This can be a problem for solutions that support fast-growing processes, like Industry 4.0. For companies involved in the development of solutions for Industry 4.0, time is an important asset that must be considered when planning the solution. If a company takes too long to deliver a solution, it can lose space in the market. These and other ontology applications require OE methods that help ontology engineers continuously gather and prioritize requirements from several users, keep domain experts engaged, deliver ontology modules according to time demands, respond to changing knowledge, and evolving the ontology as needed [5]. Some of these challenges are similar to those faced in Software Engineering (SE). Hence, some SE methods and practices may be useful to help build ontologies that are reliable, long lived and continually evolved.

In the last years, the SE area has proposed new methods, preconizing agile and continuous

Proceedings of the 15th Seminar on Ontology Research in Brazil (ONTOBRAS) and 6th Doctoral and Masters Consortium on Ontologies (WTDO), November 22-25, 2022.
EMAIL jssalamon@inf.ufes.br (A. 1); monalessa@inf.ufes.br (A. 2)
ORCID: 0000-0002-3209-5273 (A. 1); 0000-0002-6225-9478 (A. 2)

development. Some advantages of agile methods include better software quality, adapting to changing requirements, improved communication, process adaptability and higher customer satisfaction [6]. However, besides being agile, software development needs to be continuous. The need for continuous software planning, building, operation, evaluation, and alignment to business is addressed in Continuous Software Engineering (CSE), a recent subarea of SE that aims to establish a continuous flow between software-related activities, taking into consideration the entire software life cycle [7]. CSE involves practices and tools that aim at establishing an end-to-end flow between customer demand and the fast delivery of a product or service. The 'big picture' by which this might be achieved goes beyond agile principles and surfaces a more holistic set of continuous activities. It also requires tight integration and flow between them [7].

Some OE approaches have already brought agile practices from SE to OE (e.g., [8], [9], [10], [11]). Although these proposals provide advances on OE by changing the traditional (i.e., not agile) and prescriptive perspective of ontology development to a perspective more iterative and aligned with contemporaneous needs, they do not consider the continuity nature of OE activities when they are performed iteratively (e.g., how to continuously integrate new modules into an existing ontology?). To better meet users' needs and respond to changes is not enough to engage stakeholders and develop the ontology through short cycles. It is necessary to integrate OE processes into a flow that is performed in an integrated, repeated, and continuous way so that ontologies can be, like software, collaboratively and iteratively built, as well as automatically integrated, until the aimed ontology is reached. Therefore, we argue that CSE practices can be explored in the OE context to enable a better understanding of the continuous nature of some OE activities.

In this paper, we introduce the term *Continuous Ontology Engineering* (COE) as a set of processes performed in a continuous, iterative and integrated way to develop, maintain, reuse and evolve ontologies. We also propose COntE, a Continuous Ontology Engineering framework that uses CSE as inspiration to define a workflow that brings agility, continuity, integration and business alignment into OE. Based on COntE, we discuss some research opportunities and introduce some ideas to address them. Section 2 provides the background for the paper and discusses some related works; Section 3 introduces COntE; Section 4 discusses some research questions; and Section 5 concludes the paper.

2. Background

2.1. Agile and Continuous Software Engineering

In the last years, some initiatives have emerged in the SE field aiming to speed up the software development process and improve the connection between its activities [7,12]. For example, Continuous Integration [13] seeks to eliminate discontinuities between development and delivery. In a similar approach, DevOps [14] recognizes that the integration between software development and software operation must be continuous.

The need for continuity in software development is addressed in Continuous Software Engineering (CSE), which seeks to transform discrete development practices into more iterative, flexible and continuous alternatives, keeping the goal of building and delivering quality products according to established time and costs [7]. In CSE, customers are proactive, and users and other stakeholders are involved in the process, learning from usage data and feedback. Planning is continuous, so as requirements engineering, which focuses on features, modularized architecture and design, and fast realization of changes. Agile practices are employed, including short development cycles. There is continuous integration of work, continuous delivery, and continuous deployment of releases, as well as version control of code, branching strategies, fast commit of code, code coverage and code reviews. Quality assurance involves automated tests, regular builds, pull requests, audits and run-time adaption. Knowledge is shared and continuous learning happens, capturing decisions and rationale [15]. An overview of CSE processes and practices can be found in [16], [7], [17] and [12].

2.2. Ontology and Ontology Engineering

An ontology is a formal and explicit specification of a shared conceptualization [18]. 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 its computational properties. Operational ontologies, in turn, are designed with the focus on guaranteeing desirable computational properties and, thus, are machine-readable ontologies [19].

The OE field has seen the emergence of several OE methods and approaches. Methods such as SABiO [20], NEON [21] and Methontology [22] follow a more prescriptive and traditional approach. Here, we borrow the 'traditional' term from SE, meaning that they do not follow agile principles and practices. There are also proposals, such as eXtreme Design [8], SAMOD [9], UPON Lite [10], and AMOD [11], which bring agile practices from SE to OE.

Although there are similarities between our proposal and the ones based on agile practices, these works do not consider the continuous nature of OE activities. They focus solely on agile practices (e.g., incremental development in short cycles, closer user). Our proposal goes beyond that. It advocates continuity, i.e., processes should be performed iteratively and also in an integrated and continuous way until the desired ontology is produced. This is achieved by integrating processes and supporting them with integrated tools. Often, methods consider and integrate only some activities from the ontology development process and do not address the need for a set of integrated tools to support them. Disregarding continuity hampers collaborative ontology development, which is particularly important to build large ontologies or ontology networks, where ontologies are rather big and the network is developed and evolved by different people, requiring integrated tools and automated workflow.

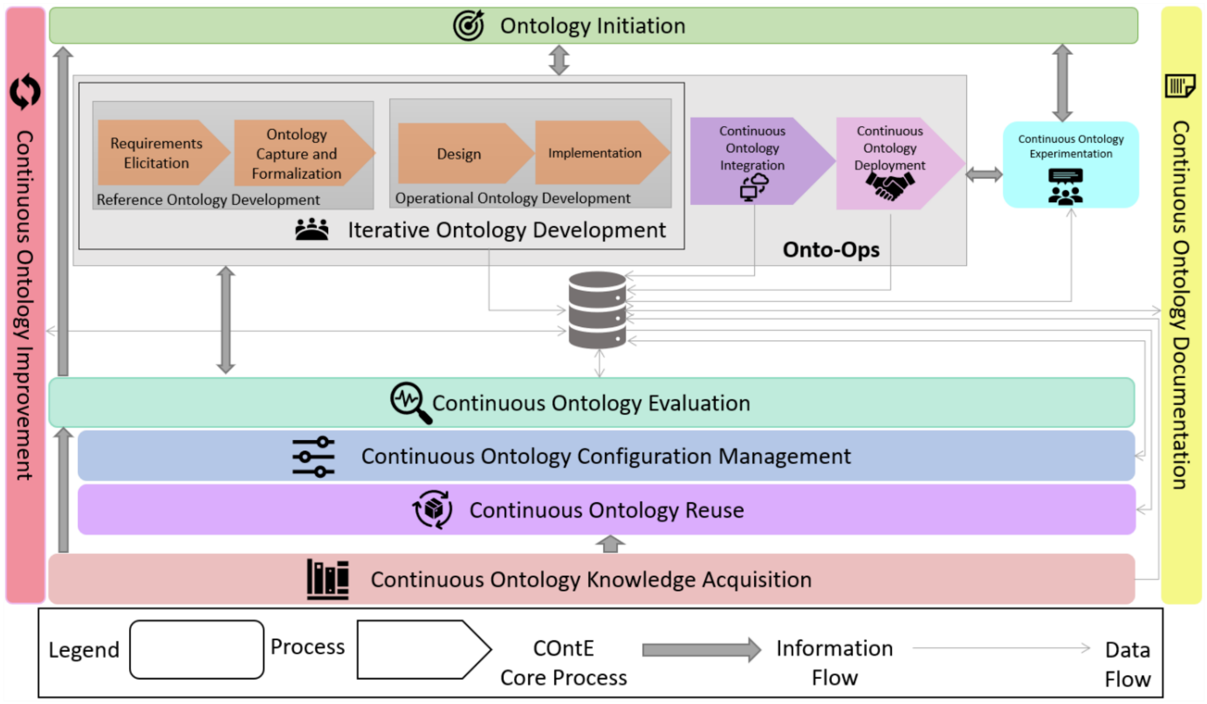
3. A Framework for Continuous Ontology Engineering

Aiming to characterize and provide knowledge about COE, we have worked on the Continuous Ontology Engineering Framework (COntE), which represents a set of processes to be performed in the COE context and the main relations between them. Its purpose is to provide an overview of COE and serve as a basis for future research. It was inspired mainly by [12]. The processes were defined taking processes suggested in SABiO [20] into account. We chose SABiO because it is based on well-known methods from literature and has been used for decades to develop several ontologies. Moreover, before defining COntE, we analyzed 15 OE methods based on agile practices (e.g., eXtreme Design [8]) and also some traditional ones (e.g., NEON [21]) and we noticed that the processes defined in SABiO cover the main activities addressed by them.

Figure 1 shows an overview of COntE, which considers that processes can be performed simultaneously and gradually. In the figure, processes containing development activities that are at the core of COE are represented by pentagons; rectangles with rounded edges represent other processes; thin arrows represent data flow from/to the repository and wide arrows represent information flow between processes. Data flows mean that the process produces and stores data in the repository, or uses data stored in it. Information flows mean that processes exchange information, which does not necessarily involve data from the repository. For visualization reasons, information flows between core processes are omitted in the figure. Processes represented as vertical rectangles are related to all the other processes, while other processes are represented as horizontal rectangles meaning that they are related to the Onto-Ops and Experimentation processes.

In a nutshell, **Ontology Initiation** is the first step to ontology development. In this process, the goals, purpose and intended uses of the ontology are defined. This helps shape the teams that will handle the ontology development and understand the ontology scope, providing an initial development plan. At the core of COE are the **Iterative Ontology Development** practices, which are based on agile SE and include, among others, the notion of small and empowered teams, backlog, short time-box, closer and active user and daily standup meetings. The idea is that the ontology will be built in an iterative and incremental manner, using whatever development method the team choses and delivering small modules at a time. The team is composed of ontology engineers, domain experts and users. All of them participate actively, with different responsibilities. Domain experts provide knowledge about the domain to be modeled, ontology engineers are in charge of ontology development, and users help understand the ontology intended use and evaluate if it was satisfied. The development is divided into **Reference Ontology Development**, which ends when the conceptual model is built, and **Operational**

Ontology Development, which ends when the ontology implementation is done.



Ontology Initiation
Continuous Ontology Improvement
Continuous Ontology Documentation
Requirements Elicitation
Ontology Capture and Formalization
Reference Ontology Development
Design
Implementation
Operational Ontology Development
Iterative Ontology Development
Continuous Ontology Integration
Continuous Ontology Deployment
Onto-Ops
Continuous Ontology Experimentation
Continuous Ontology Evaluation
Continuous Ontology Configuration Management
Continuous Ontology Reuse
Continuous Ontology Knowledge Acquisition
Legend
Process
COntE Core Process
Information Flow
Data Flow

Figure 1. COntE Overview

Once iterative development has been adopted, there is the need to integrate and test the built module into the broader context of the ontology. This leads to the adoption of **Continuous Ontology Integration**, which involves, for the reference ontology, integration of the module into the ontology architecture (the ontology modules that were developed must be integrated in a way that reflects the defined ontology modularization) alongside with tests to ensure that the model being integrated does not cause errors and unwanted constraints to the ontology and, for the operational ontology, automated build and automatic tests in a test environment. To integrate the new module to the ontology, semantic mappings between the current version of the ontology and the new module must be carried out and integration operations must be applied to derivate the new integrated model.

After development and integration, the new version of the ontology is delivered to the users. This is addressed by the **Continuous Ontology Deployment** process, where the ontology is deployed at users at the end of iterative cycles, and after passing the continuous integration testing activities. Ontology deployment depends on the ontology uses. E.g., if the ontology to be delivered is the conceptual model of a reference ontology, then the new version must be deployed in the repository where the ontology versions are stored, and the users must be notified. On the other hand, if the ontology to be delivered is an operational ontology, the new version must also be deployed in the context of the system where it is used, and how to do that depends on APIs or systems and if it can be done automatically or not. These three processes (Iterative Development, Integration and Deployment) working together embody the DevOps approach in Software Engineering, here named as *Onto-Ops*. Its purpose is to connect ontology development and operation, when the ontology is ready to be used and is delivered to its users.

As the ontology is iteratively deployed, it is possible to perform **Continuous Ontology Experimentation**, which consists in conducting experiments to evaluate ontology use and provide continuous feedback to the core processes by considering user data and feedback. Information provided by continuous experimentation is useful to identify problems, improvement opportunities and also to adjust or create plans according to the modules to be developed or adjusted. Continuous experimentation can also be used to evaluate the methods, tools and practices adopted in the ontology development. Thus, it can support the choice of the ones more suitable for developing the ontology.

Over the core processes, other processes are continuously performed to ensure that the delivered ontology meets the user needs. **Continuous Ontology Evaluation** concerns quality assurance activities, mainly verification and validation, which are continuously performed aiming at ontology quality. Verification aims to ensure that the ontology is being built correctly. Validation, in turn, aims to ensure that the right ontology is being built (i.e., the ontology fulfills its specific intended purpose) [20]. Each built module should be evaluated through verification and validation. For example, the conceptual model of the reference ontology developed in the iteration must be submitted to verification and validation before developing its operational version.

Continuous Ontology Configuration Management oversees maintaining a baseline with an approved and in-use version of the ontology, maintaining a trail of ontology versions during the development process as well as managing the changes in all artifacts produced during the process, including documents generated in the **Continuous Ontology Documentation** process, which is responsible for defining the templates for the ontology specification documents and storing the filled versions of these documents for each developed module. Keeping an effective configuration management is essential to CEO. Since the ontology is developed iteratively, there will be many versions of it, and it is crucial to identify each of them as well as the differences between versions.

Continuous Ontology Knowledge Acquisition identifies and catalogues knowledge sources that can be (re-)used to gather information about the domain the ontology should cover. This process is continuously performed as the ontology is in constant evolution. Thus, for each module, it is necessary to acquire knowledge of the domain portion to be addressed. The **Continuous Ontology Reuse** process is concerned with promoting reuse during the ontology development. In this context, catalogued knowledge and other artifacts produced during the ontology development can be reused in the next iterations. Moreover, knowledge and artifacts (e.g., ontology patterns, ontology models, domain description) produced by other people can also be reused, contributing to speed up the development process. Finally, **Continuous Ontology Improvement** regards continuously identifying improvements both in the ontology and in the development process, taking into consideration the defined goals and helping adjust them when necessary. The improvements identified during an iteration can be accommodated in that iteration (when it is a critical improvement) or addressed in next iterations.

Concerning the relations among the processes, **Ontology Initiation** provides the ontology goals, purpose, and intended uses to Onto-Ops, so that development activities are carried out accordingly. On the other hand, the Onto-Ops processes provide information (e.g., completed modules, team productivity, new requirements) to revise and align the initial plans to accommodate new requirements and needs. The relation between these processes is analogous to the BizDev approach in CSE [7], which integrates management and development processes to ensure that the project plans are defined and reviewed to enable the delivery of valuable software. Here, the processes are integrated to ensure the delivery of a valuable ontology. **Continuous Ontology Experimentation** uses information from the core processes to evaluate ontology use and provides these processes with results from the experiments, which may change existing requirements or lead to new ones. The results may also cause changes in the used tools or performed practices. It also provides information to **Ontology Initiation** to adjust and guide plans and decisions for the next iterations. **Ontology Initiation**, in turn, provides **Continuous Ontology Experimentation** with the ontology goals, purpose, and intended uses, which must be considered in the experiments. **Continuous Ontology Evaluation** uses information from the core processes (e.g., competency questions, produced artifacts,) and inform them about evaluation results. This information is also relevant to adjust plans considering the actions necessary to correct non-conformities. **Continuous Ontology Knowledge Acquisition** captures and catalogues knowledge from sources such as documents, repositories, models, domain experts and provide reusable knowledge (e.g., ontology patterns, design solutions) to **Continuous Ontology Reuse**. Besides, it provides domain information that can be used for verification and validation in **Continuous Ontology Evaluation**. Lastly, **Continuous Ontology Improvement** and **Continuous Ontology Documentation** have information flows with all the other processes. For example, **Continuous Ontology Evaluation** informs verification and validation results to **Continuous Ontology Improvement**, helping identify improvements that can be made in the ontology and the processes. Information about improvements, in turn, is relevant to the establishment of new quality standards and evaluation criteria. **Continuous Ontology Documentation** provides the core processes with the templates to specify the ontology. These templates are also used in verification and validation (**Continuous Ontology Evaluation**).

As for data flows, in summary, the core processes store conceptual models, operational ontologies, results from automatic tests and deployment. Continuous Ontology Experimentation records user data and experiments results. Similarly, Continuous Ontology Evaluation can record verification and validation results. Continuous Ontology Knowledge Acquisition stores catalogued knowledge. Continuous Ontology Configuration Management records versions and changes control. Continuous Ontology Documentation stores documents produced during the other processes execution. Finally, Continuous Ontology Improvement records improved artifacts. To achieve an effective integration and continuity, data must be shared during processes execution.

4. Research Questions and Opportunities

We have used COntE as starting point to identify research questions that can be explored in future research. Our plan is to advance on the research and add the produced results to the framework, turning it into a body of knowledge and a more robust support to ontology engineering. Next, we briefly discuss three research questions.

(RQ1) How to achieve continuity in OE? One of the challenges of performing COE is to effectively implement continuity. COE can only be achieved if there is a continuous flow among the processes. Thus, COE relies on the use of several tools that must be integrated forming a COE environment. This brings two other challenges: how to select the tools and how to integrate them into a COE environment. We have established some initial selection criteria (required technology to use the tool, potential to integration to others tools, ontology engineer experience, supported processes/activities, ease of use, user support, among others) and have investigated some OE tools (e.g., OLED [23], ROBOT [24]) and also SE tools - which are in a more advanced state of development and evolution and are already used by a large community of users (e.g., Visual Paradigm, MagicDraw, Astah, Github, GithLab, Trello) that could be combined to support COE. We are also identifying OE activities not addressed by the tools to decide how to support them (e.g., by implementing new features in the COE environment). We also started to investigate how to integrate the tools. Our goal is to reach a set of tools working together and supporting some of the COE processes. After that, new tools can be gradually added, improving the COE environment and increasing its comprehensiveness.

(RQ2) How to handle continuous evolution regarding both reference and operational ontologies? Ontology evolution is a central issue in COE because the whole idea of COE is based on continuously evolve the ontology until it meets all the expected requirements, achieves its purpose, and satisfies its intended uses. Ontology evolution has several challenges. Some of them are related to configuration management. Another challenge regards the evolution of conceptual models. In the first iterations, probably it will not be hard to create the ontology model and increase it by integrating new modules produced in new iterations. However, as the ontology grows, managing the integrated model may be laborious and complex. Thus, it is necessary to investigate how to ensure a proper versioning of the ontology and related artifacts, as well as to establish mechanisms to support ontology evolution (e.g., by defining guidelines and automated solutions to manage complex models).

(RQ3) How to achieve continuous experimentation in OE? Similar to CSE, in COE it is necessary to collect feedback from users and also experiment different methods, tools and practices to select the more adequate to develop a particular ontology. Thus, new research can be dedicated to investigating effective communication channels and empirical methods to obtain feedback from ontology users. Moreover, it is necessary to provide automated support to collect user feedback.

5. Final Considerations

In this paper, we introduce the term Continuous Ontology Engineering and propose COntE, a COE framework. The proposal discussed in this paper is a work in progress. Our purpose with this paper is to introduce the general idea of COE and enlighten the road ahead. We believe that COE subject is a long-term research. We raised some research questions, but others can also be explored. Moreover, other processes can be added to the framework (e.g., to address continuous ontology use).

As future work we plan to detail COntE processes (e.g., by defining activities, artifacts, etc.) and to set an integrated suite of tools to support the core processes. This way, the framework generates a body of knowledge that can be used by ontology engineers as a guide to ontology development in a continuous approach. In addition, since we bring new processes to the table (e.g., Continuous Ontology Deployment), understanding each process individually is a research opportunity in itself.

References

[1] A. Gangemi and V. Presutti, “Ontology Design Patterns”, *Handb. Ontol.*, pp. 221–243, 2009.

[2] O. Noppens and T. Liebig, “Ontology patterns and beyond towards a universal pattern language,” *Work. Ontol. Patterns*, pp. 179–186, 2009,

[3] R. Iqbal, M. A. A. Murad, A. Mustapha, and N. M. Sharef, “An analysis of ontology engineering methodologies: A literature review,” *Res. J. Appl. Sci. Eng. Technol.*, vol. 6, no. 16, pp. 2993–3000, 2013, doi: 10.19026/rjaset.6.3684.

[4] B. Yildiz and S. Miksch, “Ontology-Driven Information Systems: Challenges and Requirements,” *Int. Conf. Semant. Web Digit. Libr.*, pp. 35–44, 2007.

[5] M. Copeland, A. Brown, H. Parkinson, R. Stevens, and J. Malone, “The SWO Project: A case study of applying Agile Ontology Engineering methods in community driven ontologies,” *3rd Int. Conf. Biomed. Ontol. (ICBO 2012)*, vol. 897, no. January, 2012.

[6] A. Abdelaziz, N. Ramadan, and H. Ahmed, “Towards a Machine Learning Model for Predicting Failure of Agile Software Projects,” *Int. J. Comput. Appl.*, vol. 168, pp. 20–26, 2017, doi: 10.5120/ijca2017914466.

[7] B. Fitzgerald and K. J. Stol, “Continuous software engineering: A roadmap and agenda,” *J. Syst. Softw.*, vol. 123, pp. 176–189, 2017, doi: 10.1016/j.jss.2015.06.063.

[8] E. Blomqvist, K. Hammar, and V. Presutti, “Engineering Ontologies with Patterns - The eXtreme Design Methodology,” *Ontol. Eng. with Ontol. Des. Patterns*, pp. 23–50, 2016.

[9] S. Peroni, “A Simplified Agile Methodology for Ontology Development,” in *OWL: Experiences and Directions – Reasoner Evaluation. OWLED ORE 2016. Lecture Notes in Computer Science()*, M. Dragoni, M. Poveda-Villalón, and E. Jimenez-Ruiz, Eds. Springer, Cham, 2017, pp. 55–69.

[10] A. De Nicola and M. Missikoff, “A lightweight methodology for rapid ontology engineering,” *Commun. ACM*, vol. 59, no. 3, pp. 79–86, 2016, doi: 10.1145/2818359.

[11] A. S. Abdelghany, N. R. Darwish, and H. A. Hefni, “An agile methodology for ontology development,” *Int. J. Intell. Eng. Syst.*, vol. 12, no. 2, pp. 170–181, 2019, doi: 10.22266/IJIES2019.0430.17.

[12] M. P. Barcellos, “Towards a Framework for Continuous Software Engineering,” in *Proceedings of the 34th Brazilian Symposium on Software Engineering*, Oct. 2020, pp. 626–631, doi: 10.1145/3422392.3422469.

[13] K. Beck, *Extreme Programming Explained: Embrace Change*. Addison-Wesley, 2000.

[14] P. Debois, “Devops: a software revolution in the making?,” *Cut. IT J.*, vol. 24, no. 8, 2011.

[15] J. O. Johanssen, A. Kleebaum, B. Paech, and B. Bruegge, “Continuous software engineering and its support by usage and decision knowledge: An interview study with practitioners,” *J. Softw. Evol. Process*, vol. 31, no. 5, pp. 1–25, 2019, doi: 10.1002/smr.2169.

[16] H. H. Olsson, H. Alahyari, and J. Bosch, “Climbing the ‘Stairway to heaven’ - A mulitiple-case study exploring barriers in the transition from agile development towards continuous deployment of software,” *Proc. - 38th EUROMICRO Conf. Softw. Eng. Adv. Appl. SEAA 2012*, pp. 392–399, 2012, doi: 10.1109/SEAA.2012.54.

[17] J. O. Johanssen, A. Kleebaum, B. Paech, and B. Bruegge, “Practitioners’ eye on continuous software engineering,” in *Proceedings of the 2018 International Conference on Software and System Process*, May 2018, pp. 41–50, doi: 10.1145/3202710.3203150.

[18] R. Studer, V. R. Benjamins, and D. Fensel, “Knowledge Engineering: Principles and methods,” *Data Knowl. Eng.*, vol. 25, no. 1–2, 1998, doi: 10.1016/S0169-023X(97)00056-6.

[19] G. Guizzardi, “On Ontology, ontologies, Conceptualizations, Modeling Languages, and (Meta)Models,” *Databases Inf. Syst. IV*, vol. 155, pp. 18–39, 2007, doi: 10.1017/CBO9781107415324.004.

[20] R. D. A. Falbo, “SABiO: Systematic approach for building ontologies,” in *1st Joint Workshop ONTO.COM / ODISE on Ontologies in Conceptual Modeling and Information Systems Engineering*, 2014, vol. 1301.

[21] M. C. Suárez-Figueroa, A. Gómez-Pérez, and M. Fernández-López, “The neon methodology for ontology engineering,” in *Ontology Engineering in a Networked World*, 2012.

[22] M. Fernandez, A. Gómez-Pérez, and N. Juristo, “Methontology: from ontological art towards ontological engineering,” in *Proceedings of the AAAI97 Spring Symposium Series on Ontological Engineering*, 1997, pp. 33–40.

[23] J. Guerson, T. P. Sales, G. Guizzardi, and J. P. A. Almeida, “OntoUML lightweight editor: A model-based environment to build, evaluate and implement reference ontologies,” *Proc. 2015 IEEE 19th Int. Enterp. Distrib. Object Comput. Conf. Work. Demonstr. EDOCW 2015*, no. October, pp. 144–147, 2015, doi: 10.1109/EDOCW.2015.17.

[24] R. C. Jackson, J. P. Balhoff, E. Douglass, N. L. Harris, C. J. Mungall, and J. A. Overton, “ROBOT: A Tool for Automating Ontology Workflows,” *BMC Bioinformatics*, vol. 20, no. 1, pp. 1–10, 2019, doi: 10.1186/s12859-019-3002-3.