Towards Federated Ontology-Driven Data Integration in Continuous Software Engineering

Paulo Sérgio dos Santos Júnior
LEDS Research, Department of Informatics, Federal Institute of Education, Science and Technology of Espírito Santo
Serra, ES, Brazil
paulo.junior@ifes.edu.br

João Paulo A. Almeida
Ontology and Conceptual Modeling Research Group (NEMO), Computer Science Department, Federal University of Espírito Santo
Vitória, ES, Brazil
jpalmeida@ieee.org

Monalessa P. Barcellos
Ontology and Conceptual Modeling Research Group (NEMO), Computer Science Department, Federal University of Espírito Santo
Vitória, ES, Brazil
monalessa@inf.ufes.br

ABSTRACT

Organizations have adopted Continuous Software Engineering (CSE) practices aiming at making software development faster, iterative, integrated, continuous, and aligned with the business. In this context, they often use different applications (e.g., project management tools, source repositories, and quality assessment tools) that store valuable data to support daily activities and decision-making. However, data items often remain spread in different applications that adopt different data and behavioral models, posing a barrier to integrated data usage. As a consequence, data-driven software development is uncommon, missing valuable opportunities for product and process improvement. In this paper, we explore an ontology network addressing CSE aspects to develop a data integration solution in which networked ontologies are the basis to build reusable and autonomous software components that work together in a system federation to provide meaningful integrated data. We achieve a comprehensive and flexible solution that can be used as a whole or partially, by extracting only the components related to the subdomains of interest.

CCS CONCEPTS

• **Software and its engineering → Software creation and management**.

KEYWORDS

Continuous Software Engineering, Data Integration, Ontology

ACM Reference Format:
Paulo Sérgio dos Santos Júnior, João Paulo A. Almeida, and Monalessa P. Barcellos. 2023. Towards Federated Ontology-Driven Data Integration in Continuous Software Engineering. In *XXXVII Brazilian Symposium on Software Engineering (SBES 2023), September 25–29, 2023, Campo Grande, Brazil*. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3613372.3613380

SBES 2023, September 25–29, 2023, Campo Grande, Brazil

ACM ISBN 979-8-4007-0787-2/23/09…$15.00
https://doi.org/10.1145/3613372.3613380

1 INTRODUCTION

The need to continuously plan, build, operate, deploy, and evaluate software has been recognized by the Software Engineering community and addressed under the banner of Continuous Software Engineering (CSE) [1, 2, 13]. CSE consists of a set of practices and tools that supports a holistic view of software development to make it faster, iterative, integrated, continuous, and aligned with the business. It understands the development process not as a sequence of discrete activities, performed by distinct and disconnected teams, but as a continuous flow, considering the entire software life cycle [13].

To perform CSE, organizations need supporting tools and often use different applications to aid different aspects of software development [13]. For example, agile management practices can be supported by project management and time-tracking applications, while continuous development and integration can be supported by integrated development environments, version control, and code quality tools. The intensive use of applications in software development creates opportunities involving the various kinds of data they store, enabling data-driven software development [4], i.e., the use of data to drive software engineering activities and decision-making.

Although applications employed throughout the software lifecycle store useful data to support data-driven software development, data items are usually distributed in heterogeneous applications, which can make it difficult to integrate them. As a consequence, using data to drive software development has been uncommon, missing valuable opportunities for informed decision-making. Particularly in agile software development, decisions have been commonly based on subjective aspects, such as previous experiences of the managers and stakeholders, intuitions, or a combination of these [31]. A recent study investigating CSE adoption in Brazilian software organizations corroborated this perception and revealed that only 17% of them use data to drive software development [19].

One of the reasons organizations fail to leverage data stored in applications is the difficulty to access, integrate, analyze, and view data handled by heterogeneous applications. In general, each application implements its own data and behavioral models and focuses on specific aspects of the software process, with little concern with sharing and integration aspects, leading thus to several conflicts [6]. Particularly in the agile development context, the challenge is to use data to support the development process in such a way that does not represent a bottleneck to process agility.

There is a need to extract useful information from data stored in applications and present it to the team effectively and proactively, without requiring extra effort from it [33].

One source of difficulty for data integration is semantic heterogeneity, which can result in conflicts whenever divergent interpretations are given to the same information item, a situation that may not even be detected [33]. Neglecting these "semantic conflicts" can lead to poorly integrated solutions that fail in achieving their purposes (e.g., providing incorrect information) [21]. In the last decades, ontologies have become the predominant way to deal with these issues in semantic integration initiatives [20]. Ontologies establish a common conceptualization about the applications subject domains in order to support communication and integration.

In [25], we explored the use of an ontology on agile development with Scrum to integrate software development data spread across applications and thereby support data-driven software development. The results were promising. By using the proposed approach and the resulting integrated solution, a Brazilian public software development unit reported improved estimates and product quality. Despite that, data from applications addressing certain CSE processes, such as continuous integration and continuous deployment, was not included in the original solution. Aiming to build a more comprehensive data integration solution that covers several CSE processes, we decided to expand the scope of integrated data by means of an ontology network – i.e., a set of ontologies integrated to each other in a network – addressing CSE comprehensively. In order to give flexibility to the resulting solution, we organize solution's components corresponding to the various sub-ontologies in a system federation. This allows one to adopt the solution as a whole as well as use only those components related to the subdomains of interest.

This work introduces two contributions. The first is the *Continuum* ontology network, which is integrated into the Software Engineering Ontology Network (SEON) [22] and addresses CSE aspects. The second contribution is an ontology-driven data integration solution called *The Band*. For integrating data and properly supporting data-driven software development, it is necessary to create a coherent information system architecture in which the various software-related processes, data storages, and applications are integrated in such a way that they appear seamless from the point of view of the individual user [32]. For that, in *The Band*, we use networked ontologies as the basis to build reusable and autonomous software components and organize them in a Federated Information System (FIS) [5] architecture in which all components work together in the federation. As a result, data from different applications can be integrated and visualized to provide meaningful information to aid in software development activities and decision-making. The innovation proposed in this paper relies on combining ontology network and FIS for building reusable software components able to recover, integrate and present relevant data to enable data-driven software development.

This paper is organized as follows: Section 2 provides a brief background for the paper; Section 3 introduces *Continuum*; Section 4 presents an overview of *The Band* and discusses how it uses *Continuum* and FIS; Section 5 discusses related work; and finally, Section 6 presents our final considerations and future work.

2 BACKGROUND

Continuous Software Engineering. CSE encompasses a holistic set of continuous activities [13] for software engineering supported by a number of practices and tools. In CSE, customers are proactive, and users and other stakeholders are involved in the process, learning from usage data and feedback. Planning is continuous, and so is requirements engineering, which focuses on features, modularized architecture and design, and fast realization of changes. Agile practices are employed, including short development cycles, continuous integration of work, continuous delivery, and continuous deployment of releases. 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 [18].

Integration. Integration can be defined as the act to incorporate components into a complete set, conferring to it some expected properties. The components are combined in a way to form a new system constituting a whole and creating synergy. Integration is a complex task and when is performed at the semantic level, it is necessary to deal with the intended meaning of concepts in a data schema or operation signature, requiring to contrast and harmonize the conceptualizations underlying applications to be integrated [27]. Ontologies can be used at this level to assign semantics to information items.

Ontologies. An ontology is a formal, explicit specification of a shared conceptualization [15]. The conceptualization is an abstract and simplified view of the world which is intended to be represented for some purpose [29]. Ontologies can be organized in a three-layered architecture [26] with (i) *Foundational ontologies* modeling the very basic and general concepts and relations that make up the world (e.g., objects, events, participation); (ii) *Core ontologies* providing refinement to foundational ontologies by adding detailed concepts and relations in a specific area that still spans across various domains (e.g., covering in general organizational processes, organizational structure, legal relations); and (iii) *Domain ontologies* focusing on a particular domain (e.g., software testing, banking, inventory management, and car insurance).

Another distinction sets apart ontologies as conceptual models, called *reference ontologies*, from ontologies as computational artifacts, called *operational ontologies* [16]. The former aim at making the best possible description of a domain, regardless of its computational properties; the latter is designed with the focus on guaranteeing desirable computational properties [10].

For large and complex domains, such as Software Engineering, ontologies can be organized in an *Ontology Network* (ON), which consists of a set of ontologies (the *networked ontologies*) connected to each other through relationships in such a way to provide a comprehensive and consistent conceptualization [30]. In this work, we use SEON [22], an ON that describes various subdomains of Software Engineering and organizes its ontologies according to the aforementioned layers.

Federated Information Systems. A FIS is a set of distinct and autonomous information system components that participate in a federation. Participants in the first place operate independently, but have possibly given up some autonomy to participate in a federation [5] (e.g., adhering to federation-wide data conventions).

3 *CONTINUUM* ONTOLOGY NETWORK

Continuum is an ON that aims at representing the conceptualization related to the processes involved in CSE. In the Software Engineering big picture, CSE appears as a (large) subdomain involving other subdomains. Thus, we developed *Continuum* as a subnetwork of SEON [22]. Therefore, we reused some elements of SEON, such as its architecture, integration mechanisms, and some key pre-existing networked ontologies.

Figure 1 shows the architecture of the current version of *Continuum*. *Continuum* follows the same three-layer architecture used in SEON. Thus, UFO (the *Unified Foundational Ontology*) [17] grounds core ontologies that, in turn, are used to define domain-specific ontologies. Domain-specific concepts can also be directly grounded in UFO. In Figure 1, each circle (network's node) represents an ontology. Ontologies above the double dotted red line are the ones from SEON that are directly used in *Continuum*, namely: Enterprise Ontology (EO) [12], Software Process Ontology (SPO) [3, 23], System and Software Ontology (SysSWO) [8], Reference Software Requirements Ontology (RSRO) [11], Reference Ontology on Software Testing (ROoST)[28], Configuration Management Process Ontology (CMPO) [7], Quality Assurance Process Ontology (QAPO) [23], and Reference Ontology of Software Defects, Errors, and Failures (OSDEF) [9]). A dotted circle represents an ontology under development. Arrows denote the dependency relationships between networked ontologies, indicating that concepts from the target ontology are reused by the source ontology.

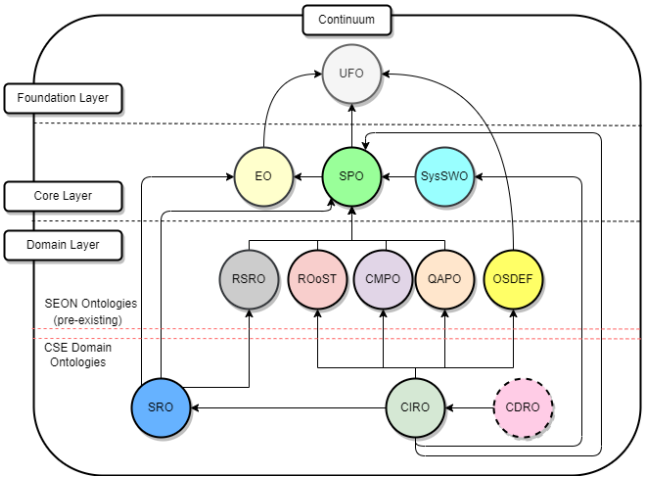

Figure 1: *Continuum*'s architecture.

The current version of *Continuum* is composed of the *Scrum Reference Ontology* (SRO) [25], which addresses aspects (e.g., processes, roles, and artifacts) related to agile software development with Scrum and the *Continuous Integration Reference Ontology* (CIRO), which regards practices and other concepts related to continuous integration (e.g., building, testing, and inspection processes). The *Continuous Deployment Reference Ontology* (CDRO), which concerns aspects related to the continuous delivery and deployment domain, is under development. Figure 2 presents a fragment of *Continuum*, including concepts from SRO and CIRO.

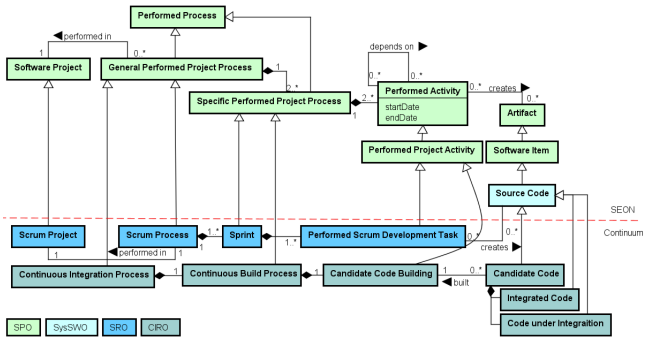

Figure 2: Fragment of *Continuum*

SRO and CIRO use concepts from SEON ontologies to describe concepts about Scrum and Continuous Integration. By integrating several subdomains, *Continuum* identifies the connections between different CSE processes. For example, a *Source Code* is created in an activity of the *Scrum Process* (*Performed Scrum Development Task*) (SRO) and integrated into the software in an activity of the *Continuous Integration Process* (CIRO).

4 THE BAND

The Band[1] is a data integration software solution based on SEON and FIS architectures. The networked ontologies used in *The Band* indicate the subdomains supported by applications that can be integrated. For example, by using SRO and CIRO, *The Band* allows integrating data from applications addressing agile development and continuous integration aspects. By using QAPO, data concerning software quality can also be integrated. The networked ontologies function as reference ontologies for data integration.

The Band works as a FIS. Each networked ontology is transformed into an *Ontology-Based Service* (OBS) that is a system of *The Band* federation and has its own repository, called *Ontology-Based Data Repository* (OBDR). Therefore, each OBS captures, stores, and shares data related to the domain portion addressed by the referred networked ontology.

Figure 3 illustrates the correspondence between ontology network (on the left-hand side) and system federation (on the right-hand side) in *The Band*. Dotted rectangles specify the abstraction layers (Foundational, Core, and Domain) [22]. A solid arrow indicates that the source ontology reuses concepts from the target ontology, with the consequence that the respective OBSs (on the right-hand side of the figure) need to exchange data related to that concept (dotted arrows). This general realization pattern is employed throughout *Continuum*. For example, the *Source Code* concept (from SysSwO) is reused by SRO and CIRO. Hence, the OBSs referring to these ontologies must communicate with each other to share data related to *Source Code*.

Since each OBS is built from a networked ontology, it embodies the ontology's concepts, axioms and relationships. An OBS is an autonomous system that has the capability of sharing data and

[1] *The Band*: the name alludes to the analogy that each ontology-based service of the architecture is a *musician* that plays an *instrument* (ON concepts, relations, and rules) and the services together are responsible for creating *music* (information) from *musical notes* (data application) to engage an *audience* (organization).

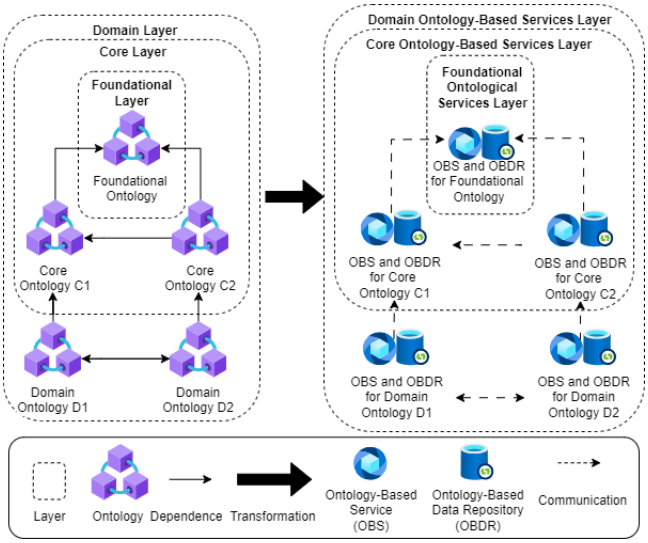

Figure 3: Transformation of ON into FIS.

functionality with other OBSs or with a client. To be autonomous, each OBS should manage all data related to the domain portion addressed by the respective networked ontology. Therefore, each OBS considers all concepts and relationships of the networked ontology, even if the concepts are from different source ontologies (i.e., reused concepts). For example, if the *Source Code* concept (from SysSwO) is reused by SRO, it needs to be addressed by the OBS referring to SRO. Similarly, if the *Source Code* concept (from SysSwO) is reused by CIRO, it needs to be addressed by the OBS referring to CIRO. As a result, the same concept is handled by more than one service. On one hand, this requires communication to ensure data consistency (e.g., if the OBSs referring to SRO and CIRO are used to integrate application data, it is necessary to ensure that data related to *Source Code* is consistent in both SRO and CIRO OBDRs). On the other hand, this allows one to use only the OBSs directly related to the domain portion involved in the integration scenario. For example, if the integration scenario regards agile development, one can use the OBS referring to SRO and does not need to use the OBS referring to SysSwO because the SysSwO concepts relevant to the Scrum context (e.g., *Source Code* as a *Software Item* produced in a *Performed Scrum Development Task*) are also included in SRO.

By creating OBSs based on networked ontologies, it is possible to observe the relationships among them and identify which data needs to be exchanged among different OBSs. By organizing OBSs in a FIS, relevant FIS criteria can be considered to contribute to defining the solution architecture. For example, by applying the transparency criterion[5], *The Band* must allow a client to search data without knowing where it is stored and using a query language based on a common conceptualization (i.e., concepts from the ON); while an OBS must have the capability of sharing and exchanging data with other OBSs. By applying the autonomy criterion [5], OBSs must be able to work independently of other OBSs and handle all necessary data to deal with the domain of interest.

4.1 *The Band* Architecture

An overview of *The Band* architecture is presented in Figure 4. *The Band* contains five main components:

(i) *Application Software Artifacts* (ASAs): are software items (e.g., a code library) that enable the communication of applications with the federation. Each ASA is application-specific and is developed to communicate with an application based on its data model to retrieve and send data to it.

(ii) *ETL components* use ASAs to extract data from an application's database, transform, and load it in OBDRs. They are split into two components: the *Extract Component* and the *Transform/Load Component*. The former extracts data from an application and publishes it in a message queue. The latter consumes data from a message queue, applies transformations based on semantic mappings, and stores data in an OBDR. This way, it is possible for several OBSs to consume data from a single queue and store transformed data in distinct OBDRs. The semantic mappings use networked ontologies as a bridge between the applications and identify which elements of the different applications are equivalent according to the ON conceptualization.

(iii) *Ontology-Based Services* (OBS): As explained in the previous section, OBSs are services corresponding to a networked ontology (e.g., the OBS referring to SRO handles data regarding sprints, development tasks, developers, user stories, among others). Each OBS has an OBDR which is based on the networked ontology conceptual model and, thus, it represents the concepts of that ontology. It can be implemented in different ways, such as a relational database or a graph database. The OBS uses its OBDR to store and share data with other OBS.

(iv) *Data Publishing Components*: enable (a) the extract components (*ETL components*) to share data with the OBSs when data is extracted from an application, and (b) the OBSs to share and exchange data with each other OBSs when changes happen in data stored in the related OBDRs (i.e., when data is inserted, deleted, or updated in the OBDR). When data stored in an OBDR is changed, the data publishing component uses a queue and automatically propagates the changes in all OBDRs that contain the changed data. This action is needed to keep data consistency in all OBDRs and allow each OBS (plus its OBDR) to be autonomous in the FIS.

(v) *Data Access Components*: provide interfaces to data retrieving (e.g., dashboards, Rest and GraphQL APIs, or data repository command) and visualization (e.g., dashboard).

The components are organized in a four-layer architecture. From bottom to top, the first layer is ***Application Integration Layer***, which contains ASAs to communicate with applications and *ETL components* (extract components) that extract data from the applications and send it to the ***Internal Data Communication Layer***. It also contains *Data Publishing Components* that support the propagation of data changes to different OBDRs of the ***Federated Ontology-Based Service Layer*** (via ***Internal Data Communication Layer***), keeping data consistent.

The ***Internal Data Communication Layer*** contains *Transform/load Components* that use *ASAs*, and *Data Publishing* components from the **Application Integration Layer** to load application data into OBDRs and support data sharing among different OBSs contained in the ***Federated Ontological Service Layer***to keep data consistent.

The ***Federated Ontological Service Layer*** receives retrieve commands from the ***Federated Data Access Layer***, which contains *Data Access Components* and provides interfaces to data retrieving and visualization, and sends data to be presented at that layer. When data loaded in an OBDR is changed, the ***Federated Ontology-Based Service Layer*** communicates with the ***Internal Data Communication Layer*** to propagate the changes in all OBDRs containing the changed data and, thus, ensure data consistency.

The ***Federated Data Access Layer*** contains *Data Access Components* that provide interfaces to data retrieving (e.g., APIs) and visualization (e.g., dashboard), via requests to the ***Federated Ontology-Based Service Layer***. At this layer, a client can use GraphQL API and OBDR Command *Data Access Components* to manipulate data (i.e. create, update, and delete) or retrieve data from the OBDR using HTTP Protocol or SQL commands, respectively. Dashboard allows a client to visualize integrated data using charts and tables. It uses the OBDR commands (e.g., SQL queries) to retrieve data from one or more OBDRs.

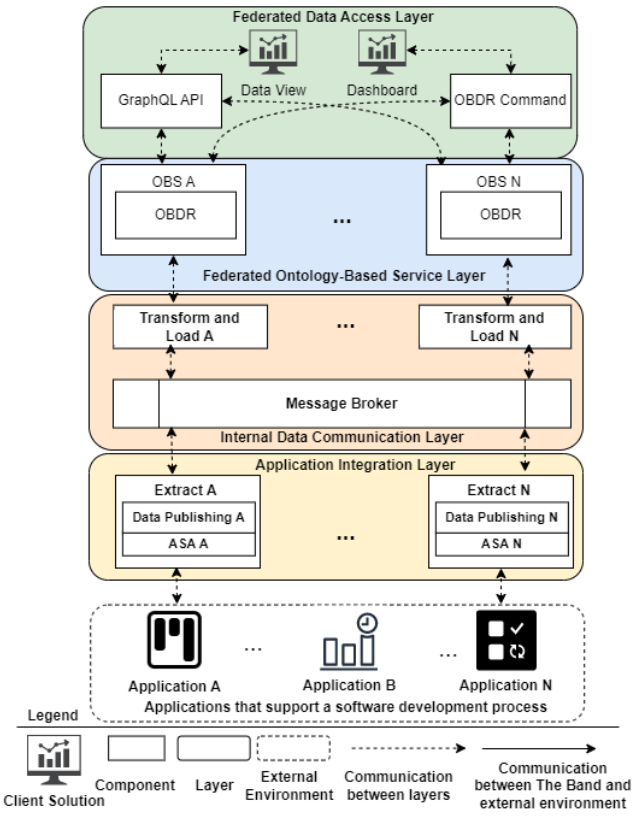

Figure 4: *The Band* architecture overview.

4.2 Implementing *The Band*

We have implemented an instance of *The Band* aiming at integrating data from Microsoft Azure DevOps[2] (an application that supports project management), Gitlab[3] (a source repository and application that supports CI and CD), Sonar[4] (an application that addresses software quality aspects), and Clockify[5] (a time-tracking application).

The Band is a data integration solution that looks like an "iceberg" because only 10% of it (*Data Access Components*) is above the water while 90% (ASAs, OBSs, OBDRs, Extract and Transform/Load components) is under the water. In other words, a client only sees the data access components (e.g., Dashboard and Data View) while the other components supporting the data access components are not noticed by the client. Next, we present information about the technologies we used to implement *The Band*.

The dashboards were developed using Metabase[6] and Dremio[7]. Dremio allows creating a data lakehouse with the OBDRs, and, thus defining commands to search OBDRs and retrieve data from them. Metabase allows using Dremio's SQL commands to create dashboards. The data views were developed using Budibase[8], a No-Code development platform that allows creating applications to visualize data stored in a data source.

The OBSs were developed using Java and Spring Boot framework[9] to create REST and GraphQL webservices, while the OBDRs were implemented using the relational database PostgreSQL.

Finally, the ASAs, Extract, Transform/Load components were developed using Python, Apache Beam[10], and Spring Cloud Stream, respectively. Apache Beam and Spring Cloud Stream allow extracting data from different data sources, applying transformations (e.g., join, filter, combine, union, and split) on data, and loading transformed data on OBDR using Python and Java, respectively. Examples of the implemented components are available in [24].

5 RELATED WORK

There are some works that propose the use of ontologies to support data integration (e.g.,[6, 14]). However, none of them addresses CSE or combines ON and FIS in an overall solution. In this work, we take a step further by exploring ON and FIS architectures to provide an integration solution made of reusable components that can work both as a FIS that covers CSE as a whole and as individual autonomous systems devoted to specific CSE aspects.

Some Mining Software Repository (MSR) efforts that aim at providing data to support decision-making in software development are related to ours (e.g., [**?**]). Differently from our work, these works were not concerned with semantic aspects or creating reusable software components. Neglecting semantic aspects can lead to conflicts whenever the same information item is given divergent interpretations [33]. Our work proposes the use of networked ontologies to assign semantics to data and structure services and repositories in the integration solution. The created OBSs and OBDRs are used to integrate applications. They work as systems of a FIS, making it easier to add new applications or change the ones

[2]https://azure.microsoft.com/en-us/products/devops

[3]https://about.gitlab.com/
[4]https://sonarcloud.io/
[5]https://clockify.me/
[6]https://www.metabase.com/
[7]https://www.dremio.com/
[8]https://budibase.com/
[9]https://spring.io/
[10]https://beam.apache.org/

that were integrated to others addressing similar scope (e.g., when the organization changes one application for another). Once semantics is assigned to applications' information items (e.g., class and attributes), it is possible to change an application data repository for another (e.g., from Microsoft Azure DevOps to Jira[11]). The cited works, in turn, provide solutions considering the data structure of the used repositories, which makes it difficult to reuse them with different repositories. By using networked ontologies, our work not only supports the integration solution but also helps to understand the domain of interest.

In conclusion, we note that none of them use ON, has an architecture based in FIS, produce ontology-based reusable software components, and address CSE aspects, as our proposal does.

6 FINAL CONSIDERATIONS

This work presented *The Band*, which combines ON and FIS to provide a federated ontology-driven data integration solution. *The Band* contains ontology-based components organized in a system federation. In this sense, the federated system can be used as a whole (i.e., covering all the subdomains represented in the networked ontologies) or one can select only the components related to the domains of interest. For example, if one wants to integrate data from applications supporting agile development management, the components referring to CIRO would not be necessary. Moreover, if an organization using *The Band* decides to change one of the used applications, data from the new application can be integrated into *The Band*, and data from the previous application is not lost. Thus, the organization can keep historical data even if an organization changes the applications, and data from the new application can be integrated with data from the previous one. Furthermore, as the ON evolves, *The Band* can be extended to cover other subdomains and, thus, increase the range of tools that can be integrated.

Currently, we are running some tests with *The Band* and developing the Continuous Deployment Ontology (CDO) to extend *Continuum* . As future work, we plan to carry out a case study in a software organization to evaluate *The Band* in a practical setting. Finally, it is worth noticing that the proposal of using ON and FIS raised in this work is not limited to the CSE (sub)domain. The proposed solution can be an inspiration for other researchers to explore ON and FIS in other domains.

ACKNOWLEDGMENTS

This research is funded in part by CNPq (313687/2020-0) and FAPES (281/2021, 1022/2022 and 2023-5L1FC).

REFERENCES

[1] Monalessa P. Barcellos. 2020. Towards a Framework for Continuous Software Engineering. In *XXXIV SBES* (Natal, Brazil) *(SBES '20)*. 626–631.
[2] Jan Bosch. 2014. *Continuous Software Engineering: An Introduction.* Springer. 3–13 pages.
[3] Ana Bringuente, Ricardo Falbo, and Giancarlo Guizzardi. 2011. Using a Foundational Ontology for Reengineering a Software Process Ontology. *Journal of Information and Data Management* 2, 3 (2011), 511–511.
[4] Erik Brynjolfsson et al. 2011. Strength in Numbers: How Does Data-Driven Decision making Affect Firm Performance? *O&M: Decision-Making in Organizations Journal* (2011).
[5] Susanne Busse et al. 1999. Federated Information Systems: Concepts, Terminology and Architectures. *Forschungsberichte des Fachbereichs Informatik* 99, 9 (1999), 1–38.
[6] Rodrigo F. Calhau and Ricardo de A. Falbo. 2010. An Ontology-based Approach for Semantic Integration. In *14th EDOC*. IEEE Computer Society, 111–120.
[7] Rodrigo F. Calhau and Ricardo de A. Falbo. 2012. A Configuration Management Task Ontology for Semantic Integration. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing*. 348–353.
[8] Bruno B. Duarte et al. 2018. Ontological foundations for software requirements with a focus on requirements at runtime. *Applied Ontology (online)* (2018).
[9] Bruno B. Duarte et al. 2018. Towards an Ontology of Software Defects, Errors and Failures. In *Conceptual Modeling: 37th International Conference, ER 2018, Xi'an, China, October 22–25*. 349–362.
[10] Ricardo de Almeida Falbo. 2014. SABiO: Systematic Approach for Building Ontologies. (2014).
[11] Ricardo de A. Falbo and Julio C. Nardi. 2008. Evolving a Software Requirements Ontology. In *XXXIV Conferencia Latinoamericana de Informática, Santa Fe, Argentina*. 300–309.
[12] Ricardo de Almeida Falbo, Fabiano Borges Ruy, Giancarlo Guizzardi, Monalessa Perini Barcellos, and João Paulo A. Almeida. 2014. Towards an Enterprise Ontology Pattern Language. *SAC '14*, 323–330.
[13] Fitzgerald, Brian and Stol, Klaas-Jan. 2017. Continuous software engineering: A roadmap and agenda. *Journal of Systems and Software* 123, 176–189.
[14] Vinicius S. Fonseca, Monalessa P. Barcellos, and Ricardo de A. Falbo. 2017. An ontology-based approach for integrating tools supporting the software measurement process. *In Science of Computer Programming* 135, 20–44.
[15] Thomas R. Gruber. 1993. A translation approach to portable ontology specifications. *Knowledge Acquisition* 5, 2 (1993), 199–220.
[16] Giancarlo Guizzardi. 2007. Conceptualizations, Modeling Languages, and (Meta) Models. In *DB&IS'2006*, Vol. 155. IOS Press, 18.
[17] Giancarlo Guizzardi et al. 2022. UFO: Unified foundational ontology. *Applied Ontology* 17, 1 (2022), 167–210.
[18] Jan Ole Johanssen et al. 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 (2019), e2169.
[19] Paulo S. Santos Júnior, Monalessa P. Barcellos, Fabiano B. Ruy, and Moises S. Omêna. 2022. Flying over Brazilian Organizations with Zeppelin: A Preliminary Panoramic Picture of Continuous Software Engineering. In *Proceedings of the XXXVI Brazilian Symposium on Software Engineering (SBES '22)*. 279–288.
[20] Julio C. Nardi, Ricardo de A. Falbo, and João Paulo A. Almeida. 2013. Foundational Ontologies for Semantic Integration in EAI: A Systematic Literature Review. In *12th IFIP*. Springer, 238–249.
[21] Stanislav Pokraev. 2009. *Model-driven semantic integration of service-oriented applications.* Ph.D. Dissertation. University of Twente.
[22] Fabiano R., Ricardo de A. Falbo, Monalessa P. Barcellos, Simone D. Costa, and Giancarlo G. 2016. SEON: A Software Engineering Ontology Network. In *Knowledge Engineering and Knowledge Management: 20th International Conference, EKAW 2016, Bologna, Italy, November 19-23, 2016, Proceedings 20*. 527–542.
[23] Fabiano Borges Ruy. 2017. *Software Engineering Standards Harmonization: An Ontology-based Approach.* Ph.D. Dissertation. UFES.
[24] Paulo Sérgio S. Júnior, Monalessa P. Barcellos, and João Paulo A. Almeida. 2023. The Band's Code. https://gitlab.com/immigrant-data-driven-development/site.
[25] Paulo S. Santos Júnior, Monalessa P. Barcellos, Ricardo de A. Falbo, and João Paulo A. Almeida. 2021. From a Scrum Reference Ontology to the Integration of Applications for Data-Driven Software Development. *Information and Software Technology* 136 (2021), 106570.
[26] Ansgar Scherp et al. 2011. Designing Core Ontologies. *Applied Ontology* (2011), 177–221.
[27] S.Izza. 2009. Integration of industrial information systems: from syntactic to semantic integration approaches. *Enterprise Information Systems* (2009), 1–57.
[28] Erica F. de Souza, Ricardo de A. Falbo, and Nandaudi L. Vijaykumar. 2017. ROoST: Reference Ontology on Software Testing. *Applied Ontology* (2017), 59–90.
[29] Steffen Staab and Rudi Studer. 2010. *Handbook on ontologies.* Springer Science & Business Media.
[30] Mari Carmen Suárez-Figueroa et al. 2012. *Ontology Engineering in a Networked World.* Springer.
[31] Richard B. Svensson et al. 2019. The Unfulfilled Potential of Data-Driven Decision Making in Agile Software Development. In *Agile Processes in Software Engineering and Extreme Programming*. 69–85.
[32] François Vernadat. 2007. Interoperable Enterprise Systems: Principles, Concepts, and Methods. *Annual Reviews in Control* 31 (2007), 137–145.
[33] H Wache et al. 2001. Ontology-Based Information Integration: A Survey of Existing Approaches. *International Journal on Artificial Intelligence* 47 (2001), 108–117.

[11] www.atlassian.com/software/jira