SABiOx: the Extended Systematic Approach for Building Ontologies

Camila Zachhé de Aguiar[1], Vítor E. Silva Souza[1]

[1]*Ontology & Conceptual Modeling Research Group (NEMO) – Federal University of Espírito Santo (UFES), Brazil – Av. Fernando Ferrari, 514, Goiabeiras, Vitória, ES, 29075-910*

Abstract

In the literature, several Ontology Engineering methods have been proposed in recent years, with the purpose of guiding the ontology construction process and thus producing ontologies with better quality. However, the field still lacks widely accepted and mature methods, which present a standardization of activities, documentation, step-by-step instructions for building ontologies and sufficient details of their life cycle. In this paper, we present SABiOx: an Ontology Engineering method which aims to guide the construction of ontologies in more detail, supported by agile principles, with iterative and incremental cycles. SABiOx covers both the construction of reference and operational ontologies, defining a life cycle formed by five phases (Requirement, Setup, Capture, Design and Implementation) detailed in activities, artifacts and roles involved, and supported by Knowledge Acquisition, Documentation, Configuration Management, Evaluation, Reuse and Publication activities. We report on the application of SABiOx to ontology construction by inexperienced ontology engineers, demonstrating how the guide have contributed to understanding and clarifying ontology construction activities and producing the desired outcomes. In addition, we also compare SABiOx with other methods present in the literature.

Keywords

Ontology Engineering, Ontology Engineering Method, Ontology.

1. Introduction

Ontology construction methods have been proposed in Ontology Engineering (OE) with the aim of guiding the ontology construction process and improve the quality of the results, establishing a standard for their construction. Thus, Ontology Engineering must follow a well-defined method, dealing with practical aspects that allow communities of specialists and ontologists to reach a consensus on the domain of the ontology, in addition to keeping it alive and evolving [1]. Although several methods have been proposed in recent years, the field still lacks widely accepted and mature methods [2]. Most methods do not present standardized activities, documentation and ontology engineering techniques [3], as well as sufficient details of the ontology life cycle [4, 5] or sufficient step-by-step instructions for building the ontology [6].

In this context, we present the SABiOx method — the Extended Systematic Approach for Building Ontologies — an Ontology Engineering method proposed with the objective of guiding the construction of ontologies (of any domain) in more detail supported by agile principles, with iterative and incremental cycles and constant revisions of the produced artifacts. SABiOx covers the construction of both reference and operational ontologies, highlighting the importance of concept consensus, the adoption of a foundational ontology, the application of ontological analysis and the reuse of resources.

As the name suggests, our proposal is based on the SABiO method [7], extended with details on the ontology life cycle and its activities, and adjusted according to our experience in building core, domain and network ontologies. The SABiO is a method for building reference and operational ontologies [8] that incorporates best practices from Software Engineering (SE) and Ontology Engineering (OE). SABiO focuses on the development of domain ontologies, and explicitly recognizes the importance of using

ONTOBRAS'24: 17th Seminar on Ontology Research in Brazil, October 07–10, 2024, Vitória, ES, Brazil

*Corresponding author.

camila.z.aguiar@ufes.br (C. Z. d. Aguiar); vitor.souza@ufes.br (V. E. S. Souza)

https://nemo.inf.ufes.br/equipe/camila-de-aguiar/ (C. Z. d. Aguiar); http://www.inf.ufes.br/~vitorsouza/ (V. E. S. Souza)

0000-0001-7945-6489 (C. Z. d. Aguiar); 0000-0003-1869-5704 (V. E. S. Souza)

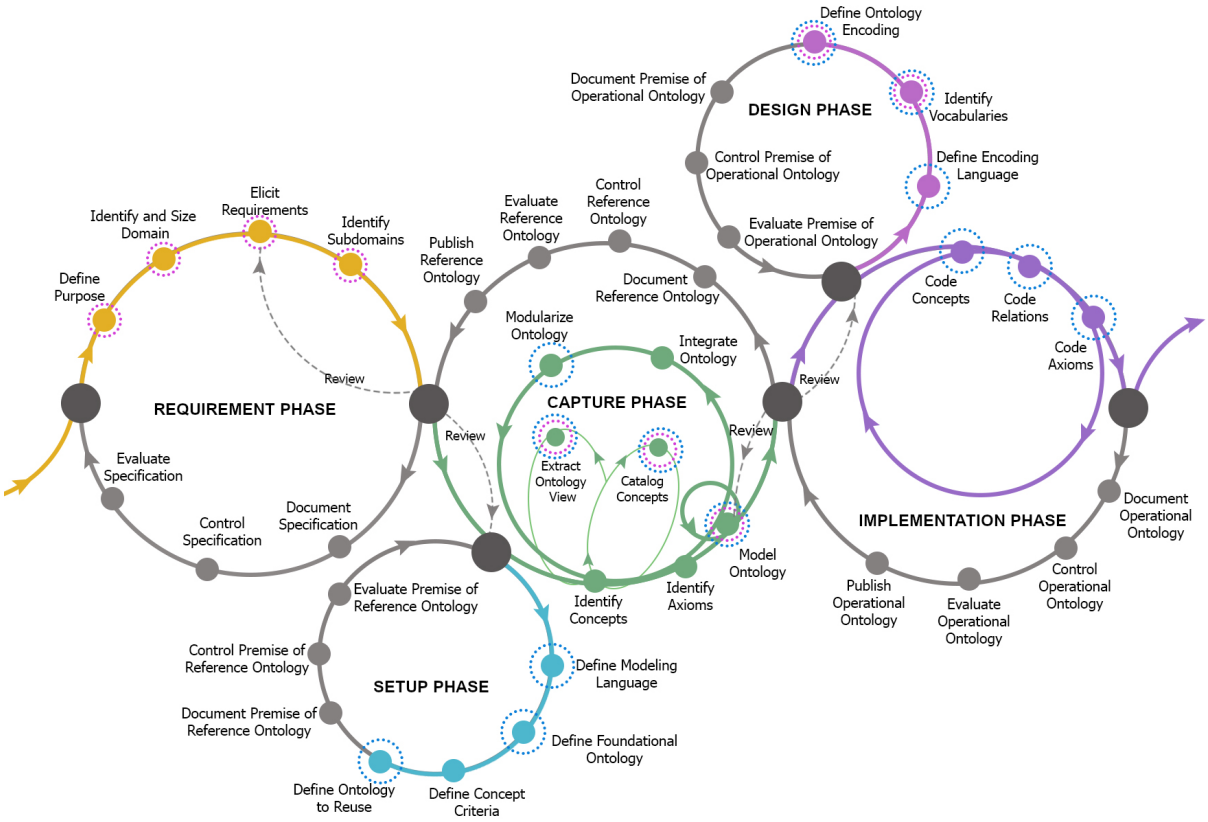

Figure 1: Phases and activities of the SABiOS method.

foundational ontologies and performing ontological analysis in the ontology development process.

The paper is organized as follows: Section 2 presents the SABiOx method, providing an overview of its proposed life cycle and describing its activities; Section 3 discusses related work on ontology engineering methods; Section 4 reports on the application of SABiOx and compares it with other methods present in the Ontology Engineering literature; finally, Section 5 concludes with final considerations.

2. SABiOx: the Extended Systematic Approach for Building Ontologies

SABiOx proposes an iterative and incremental **ontology life cycle** composed of five phases, each of which defining a **phase life cycle** composed by a set of activities. Figure 1 provides an overview of SABiOx's phases (solid line circles) and activities (dots over the circle lines). Each phase is concerned with a different aspect of the Ontology Engineering process. The Requirements phase elicits the requirements for the ontology, i.e., what it is intended to capture/represent. The Setup phase defines the baseline (e.g., modeling language, reuse of other ontologies, etc.) for building ontology models. The Capture phase identifies and models the conceptualization (i.e., concepts, relations, axioms, etc.) to meet the elicited requirements. The Design phase specifies technological features (e.g., encoding language, reuse of existing vocabularies, etc.) for the implementation of the ontology. Finally, the Implementation phase encodes the ontology in an operational language (e.g., OWL).

The phases are defined in a progressive way so that each phase uses the result of the previous phase to provide gradual enrichment in the life cycle, e.g., the specification document produced in the Requirements phase is used during Setup and Capture. Conversely, the outcome of each phase can be revised (represented by the dashed lines in Figure 1), e.g., the outcome of the Implementation phase can give rise to a revision in the Design and Capture phases. Hence, the life cycle is designed

to accommodate the iterative construction of ontologies and their evolution over time, i.e., iterating through the cycles generates new increments or changes in the ontology. For each phase of the ontology life cycle there can be several phase life cycles, executed based on agile principles (inspired by Software Engineering).

Each one of SABiOx's phases define a **phase life cycle**, consisting of **main activities**, which define the main procedures for producing the result of that phase, and **supporting activities**, which define auxiliary procedures for the main activities. In Figure 1, supporting activities are represented by gray dots/lines following the main activities (when performed in sequence) or colored dotted circles around main activities (when performed in parallel).

Supporting activities are divided in six categories: Knowledge Acquisition activities lead to the extraction of knowledge from different sources; Documentation activities record the results of other activities in the process (e.g. ontology specification document, ontology reference document, concept catalog, ontology view, conceptual model, operational ontology document and operational ontology); Configuration Management activities control the versioning and changes in the produced artifacts; Evaluation activities assess the quality of the produced results by applying verification with competency questions, validation with ontology instances and tests with queries about the competency questions over the operational ontology; Reuse activities reuse ontological and non-ontological resources (e.g. literature, ontologies, vocabularies, etc.); and Publication activities make the produced ontology available.

SABiOx does not prescribe a specific agile process to be followed. However, it defines the roles that are expected to be involved in its phases and activities. These roles could be performed by the same person in an individual effort or by different people in a collective, distributed effort in the context of an organization. The **Ontology Owner** is a person interested in building the ontology for a given purpose and responsible for defining its requirements. The **Ontology Team** is a person or group of people effectively responsible for building the ontology. Such team is composed by one or more **Domain Experts**, with knowledge/expertise on the domain of the ontology, and **Ontology Engineers**, with knowledge/expertise on ontology design and implementation.

In the following subsections, we describe each phase of SABiOx in a bit more detail, illustrating each activity with a running example: the development of OOC-O, an ontology on Object-Oriented Code, that aims to identify and represent the semantics of the entities present at compile time in object-oriented (OO) source code [9]. Due to space constraints, we only provide an overview of each phase. For the interested reader, a guide to SABiOx with fully detailed descriptions of its activities and the complete set of artifacts produced for OOC-O are available as supplementary material[1].

2.1. Requirements Phase

The Requirements phase aims to identify the purpose of the ontology within a defined domain and to discover the needs that the ontology should satisfy regarding its content and characteristics.

This phase starts with the Define Purpose activity, in which the *ontology engineer* elicits expectations from the *ontology owner* in order to answer three questions: **what** is the domain that the ontology intends to represent, **for what** the ontology will be useful, and **why** the ontology should be built. For OOC-O, the purpose was defined as: to represent the concepts of object orientation present in a source code (*what*); so that these concepts can be interpreted and identified by different programming languages in a unique way (*for what*); because each programming language defines its own syntax and semantics, resulting in source code with heterogeneity and interoperability difficulties (*why*).

Next, in Identify and Size Domain the *ontology engineer* continues to work with the *ontology owner* in order to identify the knowledge that the ontology should cover (*horizontal dimension*), as well as the level of detail that the ontology should cover of the domain (*vertical dimension*), also specifying what is out of the scope of the ontology. For OOC-O, in the horizontal dimension the domain was defined as concepts related to object-oriented software development; whereas for the vertical dimension the domain was limited to represent source code at compile time only, not covering runtime concepts.

[1] https://drive.google.com/drive/folders/1h_M7tvSruM13DTxRXn-ZYPrCocv7j_Ue?usp=sharing

Then, in Elicit Requirements the *ontology engineer* elicits functional (FRs) and non-functional requirements (NFRs) from the *ontology owner*. FRs define what is or is not relevant to the ontology (e.g., competency questions the ontology should be able to answer) and are used in the subsequent activities to evaluate whether the ontology meets its requirements. NFRs, on the other hand, are not related to the contents of the ontology, but instead to quality aspects (e.g., usability, maintainability) or design constraints (e.g., implementation language). For OOC-O, a set of FRs, such as, "What are the main elements of an OO source code?", "Which elements make up a class?", etc; and a set of NFRs, such as, being modular, being grounded on a foundational ontology, etc were elicited.

In Identify Subdomains, the *ontology engineer* can modularize the elicited requirements in subdomains, allowing the targeted or distributed construction of the ontology in later phases. For OOC-O, three subdomains were defined: *Core* (general object-oriented concepts), *Class* (class-related concepts) and *Class Member* (concepts related to class members — attributes and methods)).

The Requirements phase follows with two supporting activities under the responsibility of the *ontology engineer*: Document Specification produces an Ontology Specification Document with the results of the previous activities and Control Specification controls the changes, versions and deliveries of the information present in the specification document, providing Configuration Management capabilities to the process. Finally, Evaluate Specification, allows the *ontology owner* to assess whether the information elicited and registered in the Ontology Specification Document meets the expectations. If changes or further elicitation is needed, another iteration of this phase life cycle ensues.

2.2. Setup Phase

The Setup phase aims to define questions that will guide the elaboration of the reference ontology, i.e., decisions that have an impact on the conceptual modeling tasks to follow. The phase starts with Define Modeling Language, in which the *ontology engineer* defines the modeling language to be used in the construction of the reference ontology (conceptual model). The choice of language directly influences the integration and reuse of the ontology under construction, since ontologies developed in different languages may require greater adaptation efforts. For OOC-O, the OntoUML language [10] was adopted.

Next, in Define Foundational Ontology the *ontology engineer* defines the foundational ontology to be adopted, which guides the modeling process of the ontology under construction and the views of the reused ontologies. To do this, the popularity, usability, and adherence of the foundational ontology for the defined domain must be considered, as well as the expertise of the ontology engineer in the chosen ontology. For OOC-O, the UFO ontology [11] was chosen.

Then, in Define Concept Criteria, the *ontology engineer* works with the *domain expert* to define the criteria that will be adopted to identify if a concept is adherent to the domain of the ontology under construction. For this, a list of objective criteria, with quantitative information, or subjective criteria, with qualitative information, is defined. These criteria can be defined for individual analysis of each knowledge source or joint analysis over all the cataloged sources. Given the depth of domain knowledge at this point, defining criteria can be challenging. Thus, as knowledge is enriched and decisions about the domain are made, one should return to this activity to update these criteria. For OOC-O, the following criteria were established: the concept refers to programming constructs used at compile-time; the concept is present in more than 50% of the programming languages analyzed; and the concept is related to the main elements of object-orientation: class, method and attribute.

In Define Ontologies to Reuse, the *ontology engineer* defines the reference ontologies to be reused, i.e., ontologies or ontology fragments that match the purpose of the ontology under construction. The reuse of ontologies is a challenge [3]. One must search for candidate ontologies in search engines, repositories, and known ontologies, analyzing, among other things, whether the candidate ontology represents appropriate concepts to anchor the ontology under construction and/or the coverage of the candidate ontology with respect to the requirements of the ontology under construction. For OOC-O, existing ontologies that deal with source code and object-orientation were searched in search engines and known ontologies. Six ontologies were found and analyzed according to purpose and adherence to the domain of OOC-O. In the end, the *Source Code Ontology* (SCO) [12] was selected for reuse.

The Setup phase follows with Documentation, Configuration Management and Evaluation activities quite similar to the ones from the previous phase.

2.3. Capture Phase

The Capture phase aims to represent in a well-founded way the domain conceptualization based on the specification elaborated in the Requirements phase and the questions defined in the Setup phase, which should be revised at each iteration in order to add, detail, and delete requirements and questions related to the domain needs. In this context, conceptualization is an intentional semantic framework that encodes the implicit rules that constrain the structure of a part of reality [13].

It starts with Identify Concepts, an activity divided in two parts. In the first, Catalog Concepts, the *ontology engineer* works with the *domain expert* and/or the *ontology owner* to identify which domain concepts are part of the purpose of the ontology. Data sources (e.g., books, papers, standards, etc.) could also be used. Concepts related to the ontology domain should be cataloged according to the criteria defined in the Define Concept Criteria activity from the Setup phase. For OOC-O, books and specification documents of five OO programming languages (Eiffel, Smalltalk, Java, C++ and Python) were defined as knowledge sources and used for capturing the concepts. In the second, Extract Ontology View, the *ontology engineer* extracts the view of the reference ontologies to be reused in the modeling of the ontology under construction. For this, a modularization strategy should be used over the original ontology, since a view of the ontology will be developed that will reflect a part or a set of concepts extracted from it. As the view is being built, ontology analysis must be applied to ensure ontology compatibility with the ontology under construction. For OOC-O, the view of the SCO ontology was elaborated in order to represent only the concepts from the object-orientation domain.

Next, in Identify Axioms, the *ontology engineer* works with the *domain expert* to identify the constraint and inference axioms that the conceptual model of the ontology under construction must consider. As the conceptual model is developed, one should return to this activity to ensure a more complete and unambiguous model. For OOC-O, axioms such as $\forall c_1, c_2 : Class, i : Inheritance, inheritsIn(c_1, i) \wedge inheritedFrom(c_2, i) \rightarrow subClassOf(c_1, c_2)$ were defined for the inheritance relationship, which is established when a subclass is inherited from a superclass.

Then, in Model Ontology, the *ontology engineer* continues to work with the *domain expert* in order to model concepts and relationships covered by the purpose of the ontology in a technology-independent ontology representation language. This modeling should apply both ontology analysis, in order to categorize concepts according to a foundational ontology, and conceptual ontology patterns, in order to leverage fragments of foundational ontologies. Ontology modeling can follow a top-down, bottom-up or middle-out approach [3]. The ontology concepts and relationships should be modeled using the modeling language defined in activity Define Modeling Language. For each modeled concept, you must define its category according to the underlying ontology defined in the activity Define Foundational Ontology. For each relationship established, you must define its name, direction, type and cardinality. In addition, as the model is being built, conceptual ontology standards should be applied to ensure the standard quality of the model. This activity requires decision making about the domain and its representation, and should consider the knowledge acquired from the knowledge sources.

The *ontology engineer* and the *domain expert* continue to work closely during activities Integrate Ontology and Modularize Ontology, iterating with the previous activities from this phase. In the former, they integrate both the views of the ontology under construction and the views of the reused ontologies. In the latter, they identify the modules into which the ontology can be decomposed and which can be considered separately while being interconnected with other modules [14].

As modeling is evolutionary, the model must continuously undergo adjustments, apply ontology analysis, integrate ontology views, and reorganize its modules, returning to these activities whenever necessary. For OOC-O, the conceptual model of the object-oriented domain was built, integrated with SCO and modularized in three parts, following the subdomains identified earlier.

The Capture phase also follows with Documentation, Configuration Management and Evaluation activities quite similar to the ones from the previous phases. A final supporting activity, Publish

Reference Ontology, makes the artifact that results from this phase available to its target audience.

2.4. Design Phase

The Design phase aims to define technological issues that will guide the implementation of the operational ontology, i.e., decisions that are dependent on the technological environment. The reference ontology, elaborated in the Capture phase, provides the initial information to specify the design that may be originated by different technological choices. This phase bridges the gap between the conceptual modeling of reference ontologies and their encoding in terms of an operational ontology language [8].

The phase starts with Define Encoding Language, in which the *ontology engineer* defines the representation language to be applied in the construction of the operational ontology, i.e., in the codification. The choice of language directly influences the integration and reuse of the ontology under construction. For OOC-O, the OWL language was chosen.

Next, in Identify Vocabularies, the *ontology engineer* identifies the vocabularies to be adopted in the coding of the ontology under construction, both base vocabularies to assist in the construction of the ontology and vocabularies of the reused ontologies. In this context, *vocabulary* is defined as the set of concepts and relations of a given domain, without necessarily adopting a complex formalism such as that of an ontology. For OOC-O, the vocabularies SCO, OWL, RDF(S) and XML Schema were reused.

Then, in Define Ontology Encoding, the *ontology engineer* indicates how the architecture of the operational ontology will be derived from the reference ontology, considering the defined coding language. An ontology can be defined as a set of representational primitives that model a knowledge domain, characterized as concepts/classes, relations/properties, and attributes/properties of data types [5]. In the case of the OWL language, the architecture will be based on classes and properties extracted from the conceptual model. It is also suggested that the nomenclature and the use of vocabularies to be adopted in encoding the ontology are defined. The architecture should adopt modularization for ease of understanding and reuse. In OOC-O, concepts and relations are represented as OWL classes and properties. Naming patterns were defined in order to ensure unique identifiers for each element.

As with the Setup phase, the Design phase follows with Documentation, Configuration Management and Evaluation activities.

2.5. Implementation Phase

The Implementation phase aims to implement the reference ontology produced in the Capture phase into an operational ontology, according to the design specifications elaborated in the Design phase. This phase transforms the reference ontology into a machine-interpretable ontology, making it available for use in applications, decision support, and domain reasoning.

The phase is mainly composed by activities Code Concepts, Code Relations and Code Axioms, in which the *ontology engineer* encodes, respectively, the concepts, relations and axioms of the reference ontology as elements of the formal operational ontology language, according to the assumptions defined in the operational ontology document in the Design phase. For OOC-O, the concepts, relations and axioms from the reference ontology are encoded as OWL classes, properties and constraints.

Similar to the Capture phase, the Implementation phase follows with Documentation, Configuration Management, Evaluation and Publication activities.

3. Related Works and Baseline

We found in the literature methods that adopt different principles, processes and activities for the construction of ontologies and that have already been discussed in literature reviews [2, 1, 15]. There are works aimed at (i) acquisition and reuse of knowledge, (ii) processing and transformation of information, and (iii) development of domain and application specific ontologies. In what follows, we briefly summarize methods related to SABiOx that were selected from recent publications [1, 15], focusing on those that define an ontology life cycle and/or allow collaborative/distributed work.

Methontology [16] is a proposal for ontology engineering based on the software development process, including its rigor. The method follows three categories: *management*, to define what, how and how many resources will be dedicated and their quality; *development*, to specify requirements, model knowledge, transform into a formal semi-computable model, implement in computational language and provide maintenance; and *support*, which includes knowledge acquisition, assessment, integration, documentation and configuration management activities that are carried out in conjunction with development activities. The life cycle is iterative, based on prototypes and reuse of existing ontologies.

UPON [17] is based on the Unified Process (UP) and supported by the Unified Modeling Language (UML), with an iterative and incremental nature. The method includes cycles, phases, iterations and workflows. The phases are defined as *inception*, to capture requirements and perform some conceptual analysis; *elaboration*, to identify the fundamental concepts; *construction*, to design and implement; and *transition*, to test and release ontologies. For each phase, iterations are executed, which follow the workflows requirements, analysis, design, implementation and test.

OTKM [18] is a proposal for the introduction and maintenance of ontologies based on knowledge management applications with a focus on Knowledge Processes and Meta Knowledge Processes. The method follows the phases: *feasibility study*, to identify problems and opportunities; *kickoff*, to gather requirements; *refinement*, to model and formalize the ontology; *evaluation*, to validate technical and user issues; and *application/evolution*, to put the ontology to use and evolve it throughout the time.

101 Method [19] is a proposal for the iterative development of operational ontologies that supports design decisions, guiding the activities of determining the domain of the ontology with competency questions, considering the reuse of existing ontologies, enumerating important terms in the ontology, defining classes and hierarchies, class properties, constraints and instances. The life cycle of the method is not precisely defined, but it presents the construction process and the sequence of its activities.

HCOME [20] is a proposal for the development and evaluation of living ontologies in the context of communities of knowledge workers. The method defines the life cycle formed by the *specification phase*, to define the object, scope and requirements; *conceptualization phase*, for the development and maintenance of the ontology through knowledge acquisition; and *exploration phase*, to apply and evaluate the ontology. Activities are performed iteratively until a consensus is reached.

DILIGENT [21] is a proposal for the construction of reference ontologies in a distributed and collaborative way, based on the evolution of prototypes. To mediate conflicts and differences in the representation of consensual knowledge about the domain, the method adopts an argumentation framework. The method follows the steps *build*, *local adaptation* (the built ontology is distributed so that its users perform specific local changes), *analysis*, *revise* (the shared ontology is revised and updated), and *local update* (update the local ontology with the new version of the shared ontology).

NeON [22] is a proposal for building ontological networks with reuse of ontological and non-ontological resources, including nine scenarios, glossary of processes and activities, life cycles and methodological guidance. The scenarios can be combined in different ways to build the ontology, the *building ontological networks from scratch* scenario being mandatory and integrating all other scenarios. The activities *knowledge acquisition*, *documentation*, *configuration management*, *evaluation* and *assessment* must be carried out throughout the development of the ontology network.

SAMOD [23] is a proposal inspired by Test-Driven Development to develop ontologies with small steps from an interactive workflow to the creation of well-developed and documented models from domain descriptions. The method follows interactive steps consisting of defining a new test case, merging the current model with the model developed in the previous step and refactoring the current model executing testing for all test cases.

SABiO [7] is the baseline of the SABiOx method, consisting of five main phases: purpose identification and requirements elicitation; ontology capture and formalization; operational ontology design; operational ontology implementation; and testing. Furthermore, these phases are supported by well-known activities in the Software Engineering field, such as knowledge acquisition, reuse, configuration management, evaluation, and documentation.

SABiOx was compared with the above methods as part of our efforts to evaluate our proposal, presented next.

4. Evaluation

We evaluated the proposal of SABiOx with two efforts: a comparison with the related work presented in Section 3 plus our baseline, SABiO [7], and a questionnaire submitted to inexperienced Ontology Engineers that have applied SABiOx.

4.1. Comparison with Related Work and Baseline

For the comparison with related work, we followed criteria adapted from protocols established in other studies [2, 15]. Table 1 shows the comparison of the methods according to the defined criteria.

Eleven criteria and their respective characteristics (abbreviated in parentheses) were defined, namely: **C1** Type of ontology: reference (REF), operational (OPE), not defined (NO); **C2** Collaborative construction: is (YES) or is not (NO) addressed; **C3** Reusability of existing ontologies: is (YES) or is not (NO) supported; **C4** Interoperability: is (YES) or is not (NO) addressed, by providing some strategy (e.g., sharing the same standards or high level concepts) that promotes interoperability in the ontologies that are built; **C5** Degree of application dependency: application dependent based on application knowledge base (DEP), application semi-independent based on possible usage scenarios (SEM), and application independent, no assumptions are made about the intended uses of the ontology (IND); **C6** Life cycle: sequential (SEQ), traditional cycle that performs activities sequentially so that the next activity begins after the previous one is completed and all terms are identified at the beginning of the process; incremental (INC), partial specification of requirements leading to the development of pieces of the ontology at each increment; prototype (PRO), specification and ontology are built during the process and according to the needs; iterative (ITE), each iteration advances the ontology development process, but does not necessarily get an ontology increment at each iteration; and not defined (NO), no clearly proposed life cycle; **C7** Approach to identify concepts: bottom-up (BOT), top-down (TOP), middle-out (MID) or not defined (NO); **C8** Possible data sources for identifying ontology concepts: domain experts (EXP), ontological resources (ONT) and data-driven (DAT); **C9** Evaluation strategies for the produced artifacts: verification with competency question or domain experts (VER), validation with instantiation of concepts (VAL) and test with queries on the ontology (TES); **C10** Expected roles in life cycle activities: ontology engineers (OEN), knowledge engineers or knowledge workers (KEN), domain experts (DEX), user (USE), ontology owner (OOW), ontology designer (ODE), ontology programmer (OPR), ontology tester (OTE) or not defined (NO); **C11** Method details: process (PRO), phase or stage (PHA), activity or workflow (ACT), output (OUT), input (INP), example (EXA), tool (TOO) and not defined (NO).

To compare SABiOx with related work and baseline in a more qualitative way, we mapped their activities to common *development* — *specification*, *conceptualization*, *formalization*, *implementation* and *maintenance/evolution* — and *support* — *knowledge acquisition*, *evaluation*, *integration*, *documentation*, *reuse* and *configuration management* — activities identified in the literature [24, 25].

Table 2 summarizes the *development* activities of the analyzed methods, except for *maintenance/evolution*, which is addressed only by OTKM as the *Application/Evolution* aims to keep the ontology up to date. As noted in the table, the *specification* activity is not defined in DILIGENT and the *implementation* activity is not clear in HCOME and DILIGENT. Further, *support* activities are rarely addressed by methods (except the *evaluation* activity) and, when addressed, do not present detailed information.

The *Evaluation* activity is treated in Methontology in the *Evaluation* phase with verification of the ontology correctness with the specification and validation of its purpose; UPON has the *Test* phase with coverage of the ontology in relation to scenarios and competency questions; OTKM has the *Evaluation* phase with technical properties of the ontology and correspondence to user needs; 101 Method has the *Step 7* with the creation of ontology instances; HCOME has the *Exploration* phase with stakeholder assessment of the ontology; NeON has the *Evaluation* activity with technical validation of requirements; SAMOD has the *Step 1* with model, data, and SPARQL query tests; SABiO has the *Evaluation* process and the *Testing* phase with technical review to verify if requirements are met (answers to competency questions) and validate by instantiation and execution of test cases (queries in the operational ontology).

On the other hand, *Knowledge Acquisition* is only superficially treated in Methontology and SABiO,

Table 1
Comparison of methodologies based on the established criterion.

Method	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11
Methontology	REF OPE	NO	YES	NO	IND	PRO	NO	EXP	VER VAL	NO	PRO PHA ACT OUT EXA
UPON	REF OPE	YES	YES	YES	SEM	INT INC	TOP BOT MID	EXP ONT DAT	VER VAL	OEN KEN DEX	PHA ACT
OTKM	NO	YES	YES	NO	DEP	ITE	TOP BOT MID	EXP DAT	VER	OEN DEX	PHA ACT OUT EXA
101 Method	OPE	NO	YES	NO	IND	ITE	TOP BOT MID	EXP	VAL	NO	PRO PHA ACT
HCOME	OPE	YES	YES	NO	SEM	ITE	NO	EXP ONT DAT	VER	KEN	PHA ACT TOO
DILIGENT	REF OPE	YES	YES	NO	NO	ITE	NO	EXP	VER	OEN KEN DEX USE	PRO PHA ACT INP OUT EXA TOO
NeON	REF OPE	YES	YES	YES	SEM	ITE INC SEQ	TOP BOT MID	EXP ONT DAT	VER VAL	DEX USE OEN	PHA ACT INP OUT EXA
SAMOD	REF OPE	YES	YES	NO	SEM	ITE	MID	EXP	VAL TES	DEX OEN	PHA
SABiO	REF OPE	YES	YES	YES	IND	NO	NO	EXP DAT	VER VAL TES	DEX USE OEN ODE OPR OTE	PRO PHA ACT
SABiOx	REF OPE	YES	YES	YES	IND	ITE	TOP BOT MID	EXP ONT DAT	VER VAL TES	OOW DEX OEN	PRO PHA ACT INP OUT EXA

being mentioned in OTKM, HCOME and NeON; *Documentation* is only superficially treated in Methontology, NeON and SABiO, being mentioned in OTKM, HCOME and SAMOD; *Reuse* is treated in details in NeON and superficially in SABiO, being mentioned by the other methods; *Integration* is only superficially treated in Methontology and OTKM, being mentioned by NeON; and *Configuration Management* is only superficially mentioned in NeON and SABiO.

The scenario presented in this section demonstrates how current Ontology Engineering methods detail their phases and activities and highlights the motivation to extend the SABiO method to provide more information and clarity in the ontology development process, since gaps in the understanding and execution of the method can impact the adoption of the methodology and the quality of the artifacts produced. SABiOx presents complementary characteristics in relation to other methods, mainly with respect to the proposal of: (i) guiding the development of the reference and operational ontology, (ii) adopting a small set of roles (ontology owner, domain expert and ontology engineer), (iii) defining a clear life cycle to follow, (iv) demonstrating the application of the method by example, and (v) providing a detailed guide to carry out all activities proposed by the method (development and support activities).

Table 2
Mapping of methods based on their development activities.

Method	Specification	Conceptualization	Formalization	Implementation
Methontology	Specification phase with requirement and competency question	Conceptualization phase with glossary of terms, hierarchy, instances of concepts and conceptual model	Conceptualization phase with conceptual model	Implementation phase with ontology codified in a formal language
UPON	Requirement workflow with use case and competency question	Analysis workflow with resource reuse, conceptual model and glossary of terms	Design workflow with categorization, refinement and organization of concepts, taxonomy and conceptual model	Implementation workflow with ontology codified in a formal language
OTKM	Kickoff phase with specification and ontology reuse	Kickoff phase with semi-formal ontology description	Kickoff phase with organization of concepts hierarchically and grouped	Refinement phase with formalization of ontology into taxonomy and appropriate relationships
101 Method	Step 1 with definition of the domain, scope and competency questions	Step 3 with list of important domain terms	Step 4 with definition of classes and hierarchical taxonomy	Step 5 and Step 6 with definition of properties, data type and cardinality
HCOME	Specification phase with requirements specification	Conceptualization phase with ontology reuse	Conceptualization phase with ontology modeling, integration and merging	Not defined
DILIGENT	Not defined	Local Adaptation phase with local ontology changes	Build phase with building the shared ontology and Revision phase with consensus changes in local ontologies replicated in the shared ontology	Not defined
NeON	Requirements Specification activity with definition of purpose, scope, FRs* (competency questions), NFRs, intended uses, end-users, implementation language and glossary of terms	Conceptualization activity without further details beyond organizing and structuring the knowledge into a meaningful model	Formalization activity without further details beyond transforming the meaningful model into a semi-computable model	Implementation activity without further details beyond implementing the model in an ontology language
SAMOD	Step 1 with description of motivation scenarios and competency questions	Step 1 with definition of the glossary of terms	Step 1 with conceptual modeling in graphical representation	Step 1 with converting the model to OWL
SABiO	Purpose and Requirements Elicitation phase with definition of purpose, intended use, FRs (competency questions), NFRs and ontology modularization	Capture and Formalization phase with reuse of ontological and non-deontological resources	Capture and Formalization phase with conceptual modeling in graphical representation, dictionary of terms, ontological analysis and definition of axioms	Design and Implementation phases with specification of the design, architecture and environment of the ontology and implementation in operational language
SABiOx	Requirements phase with definition of purpose, domain dimension, FRs (competency questions), NFRs and subdomains	Capture phase with identification of concepts, reuse of ontological and non-ontological resources, catalog of concepts and vision of reused ontologies	Capture phase with conceptual modeling of a graphical representation, axioms, integration and modularization	Design and Implementation phases with identification of vocabularies, definition of structure and codification of concepts, relationships and axioms in operational language

* FR: Functional Requirement, NFR: Non-Functional Requirement

4.2. Feedback from Ontology Engineers

SABiOx is the result of our experience in developing domain and core ontologies over the years and has been maturing as new ontologies were developed using the method, until we reached the current version presented in this article. Thus, although a recent proposal, the SABiOx method has already been used in the development of the core ontology SCO [12] and three domain ontologies, OOC-O [9], ORM-O [26] and DepIn-O [27], all of them performed by inexperienced ontology engineers from our research group in academic settings. To get feedback from users of SABiOx, we asked three of the aforementioned ontology engineers to answer a questionnaire (the author of OOC-O did not participate), whose results are discussed as follows.

Regarding their profile, the questionnaire asked about (i) Level of education: 1 in undergraduate studies and 2 in (post)graduate studies; (ii) Degree: 1 in Computer Engineering and 2 in Computer

Science; (iii) Experience in ontology development: 1 had already developed an ontology and 2 had not developed any ontology before applying SABiOx; (iv) Experience in using an ontology engineering method: 1 had used SABiO method before; (v) Challenges using other methods: again, the same subject reported difficulty in understanding the method and how it should be applied.

Regarding the application of the SABiOx method, we observed that (i) Applied phases: all users conducted the Requirement, Setup and Capture phases and one user also performed Design and Implementation; (ii) Ontology type: all users developed a domain ontology; (iii) Challenges using SABiOx: users reported difficulty in applying the activities of the configuration management process with respect to document and ontology version control and difficulty in interpreting how to perform some more complex activities such as ontology modeling and concept capture; (iv) Contribution using SABiOx: all users indicated that the method contributed to the development of the ontology, reporting that the method ensured robustness to the complete ontology construction process, from conception to implementation, as well as enriched the documentation prepared for the ontology, guided the acquisition of knowledge about the ontology domain and helped to understand the expected results of the activities by providing illustrative examples.

Thus, the information obtained from the questionnaire and from our observation in the application of the SABiOx method provide us some evidence of the contribution of the method to Ontology Engineering, particularly for inexperienced ontology engineers, by guiding the development process and adopting iterative and incremental cycles focused on the quality of the artifacts produced. In this sense, we observed that the cycles allow for in-depth discussion of the domain, concepts and modeling, contributing to its incremental maturation, that the involvement of domain experts is crucial for the quality of the ontology, that the use of a foundational ontology offers ontological patterns that contribute to a well-founded ontology and its interoperability, that the reuse of ontologies offers important subsidies to advance in development more quickly and clearly, and that a guide with detailed information and examples contributes to an ontology of quality and for the inclusion of inexperienced engineers in the area. Of course, such evaluation is still preliminary and further efforts in applying SABiOx are necessary for the method to gain the much desired maturity.

5. Final Considerations

This article presented SABiOx, a method for Ontology Engineering that extends SABiO [7] by providing more details on the ontology life cycle and more clearly guide the ontology construction process with iterative and incremental cycles. Comparison with other methods proposed in the literature and feedback from early users of SABiOx indicate that our proposal is able to fill a gap in the field, contributing with a more opinionated process for developing ontologies, particularly suitable for inexperienced ontology engineers. SABiOx presents in detail (but without excess) the activities that compose the phases of the process, describing what must be done, indicating techniques and strategies for carrying them out, as well as input/output artifacts and examples. In addition to the main activities normally addressed by the methods, SABiOx also describes support activities that are not described (and sometimes not even mentioned) by other methods, which might lead ontology engineers to doubts and uncertainties.

Future work includes more experiments applying SABiOx, including practitioners in industry, and the continuous evolution of the method, refining its process and providing tools to support it.

References

[1] K. I. Kotis, G. A. Vouros, D. Spiliotopoulos, Ontology engineering methodologies for the evolution of living and reused ontologies: status, trends, findings and recommendations, The Knowledge Engineering Review 35 (2020) e4.

[2] R. Iqbal, M. A. A. Murad, A. Mustapha, N. M. Sharef, et al., An analysis of ontology engineering methodologies: A literature review, Research journal of applied sciences, engineering and technology 6 (2013) 2993–3000.

[3] M. Fernández-López, A. Gómez-Pérez, Overview and analysis of methodologies for building ontologies, The knowledge engineering review 17 (2002) 129–156.
[4] A. S. Abdelghany, N. R. Darwish, H. A. Hefni, An agile methodology for ontology development, International Journal of Intelligent Engineering and Systems 12 (2019) 170–181.
[5] T. Slimani, A Study Investigating Knowledge-based Engineering Methodologies Analysis, International Journal of Computers and Applications 128 (2015) 67–91.
[6] F. M. Mendonça, A. L. Soares, Construindo ontologias com a metodologia ontoforinfoscience: uma abordagem detalhada das atividades do desenvolvimento ontológico, Ciência da Informação (????).
[7] R. de Almeida Falbo, Sabio: Systematic approach for building ontologies., in: Onto. Com/odise@ Fois, 2014.
[8] G. Guizzardi, On ontology, ontologies, conceptualizations, modeling languages, and (meta)models, in: Databases and Information Systems IV – Selected Papers from the 7$^{\text{th}}$ International Baltic Conference, DB&IS 2006, 2007, pp. 18–39.
[9] C. Z. d. Aguiar, R. d. A. Falbo, V. E. S. Souza, OOC-O: A Reference Ontology on Object-Oriented Code, in: Proc. of the 38th International Conference on Conceptual Modeling, Springer, 2019.
[10] G. Guizzardi, C. M. Fonseca, A. B. Benevides, J. P. A. Almeida, D. Porello, T. P. Sales, Endurant Types in Ontology-Driven Conceptual Modeling: Towards OntoUML 2.0, in: Proc. of the 37th International Conference on Conceptual Modeling (ER 2018), Springer, 2018, pp. 136–150.
[11] G. Guizzardi, A. Botti Benevides, C. M. Fonseca, D. Porello, J. P. A. Almeida, T. Prince Sales, UFO: Unified Foundational Ontology, Applied Ontology 17 (2022) 167–210.
[12] C. Z. d. Aguiar, F. Zanetti, V. E. S. Souza, Source code interoperability based on ontology, in: Proceedings of the XVII Brazilian Symposium on Information Systems, 2021, pp. 1–8.
[13] M. Uschold, M. King, Towards a methodology for building ontologies, Citeseer, 1995.
[14] M. d'Aquin, Modularizing ontologies, Springer, 2012, pp. 213–233.
[15] A. Sattar, E. S. M. Surin, M. N. Ahmad, M. Ahmad, A. K. Mahmood, Comparative analysis of methodologies for domain ontology development: A systematic review, International Journal of Advanced Computer Science and Applications 11 (2020).
[16] M. Fernández-López, METHONTOLOGY: From Ontological Art Towards Ontological Engineering, AAAI Technical Report (1997) 33–40. doi:10.1109/AXMEDIS.2007.19.
[17] A. Nicola, M. Missikoff, R. Navigli, A proposal for a unified process for ontology building: UPON, in: International Conference on Database and Expert Systems Applications, volume 3588, Springer International Publishing, 2005, pp. 655–664. doi:10.1007/11546924_64.
[18] Y. Sure, S. Staab, R. Studer, On-to-knowledge methodology otkm, Handbook on ontologies (2004).
[19] N. F. Noy, D. L. McGuinness, et al., Ontology development 101: A guide to creating your first ontology, 2001.
[20] K. Kotis, G. A. Vouros, Human-centered ontology engineering: The hcome methodology, Knowledge and Information Systems 10 (2006) 109–131.
[21] H. S. Pinto, C. Tempich, S. Staab, Ontology Engineering and Evolution in a Distributed World Using DILIGENT, Springer, 2009, pp. 153–176.
[22] M. C. Suárez-Figueroa, A. Gómez-Pérez, E. Motta, A. Gangemi, Ontology Engineering in a Networked World, Springer, 2012.
[23] S. Peroni, A simplified agile methodology for ontology development, in: OWL: Experiences and Directions–Reasoner Evaluation: 13th International Workshop, OWLED 2016, and 5th International Workshop, ORE 2016, Bologna, Italy, November 20, 2016, Springer, 2017, pp. 55–69.
[24] O. Corcho, M. Fernández-López, A. Gómez-Pérez, Methodologies, tools and languages for building ontologies. where is their meeting point?, Data & knowledge engineering 46 (2003) 41–64.
[25] Y. Sure, C. Tempich, D. Vrandecic, Ontology engineering methodologies, Semantic Web Technologies: Trends and Research in Ontology-based Systems (2006) 171–190.
[26] F. L. Zanetti, C. Z. de Aguiar, V. E. S. Souza, Representacao ontologica de frameworks de mapeamento objeto/relacional, in: Proc. of the 12th Seminar on Ontology Research in Brazil, 2019.
[27] C. Guterres, C. Z. de Aguiar, V. E. Souza, Depin-o: An ontology on dependency injection software frameworks, 2023.