SABiO: Systematic Approach for Building Ontologies

Ricardo de Almeida Falbo

1Federal University of Espírito Santo, Vitória, Brazil
falbo@inf.ufes.br

Abstract. This paper presents the version 2.0 of SABiO - a Systematic Approach for Building Ontologies. SABiO focus on the development of domain ontologies, and propose also support and management processes, which are strongly linked to the development process. SABiO distinguishes between reference and operational ontologies, providing activities that apply to the development of both types of domain ontologies.

Keywords: ontology engineering, domain ontology, ontology development

1 Introduction

The role of ontologies in Conceptual Modeling and Software Engineering is increasingly being recognized as fundamental by both the research and practitioner communities. At the same time, the Ontology Engineering community also recognizes the great help Software Engineering can provide for improving practices in the ontology engineering process [1, 2].

In fact, building ontologies is not an easy task. Ontology building presents complexities that are comparable to the development of any software artifact [2]. Thus, like any complex software development activity, for building quality ontologies, we need methods and tools to support their development.

In 1997, we defined SABiO, a Systematic Approach for Building Ontologies, whose first version was published in 1998 [3]. SABiO was proposed based on Uschold and King skeletal methodology [4], adding some features to improve it, such as a graphical languages for modeling ontologies, an axiom classification, and the use of competency questions, as proposed by Gruninger and Fox [5]. SABiO was conceived for supporting the development of domain reference ontologies. By domain reference ontology we mean a domain ontology that is constructed with the sole objective of making the best possible description of the domain in reality, with regard to a certain level of granularity and viewpoint. A reference ontology is to be a special kind of conceptual model, an engineering artifact with the additional requirement of representing a model of consensus within a community. It is a solution-independent specification with the aim of making a clear and precise description of domain entities for the purposes of communication, learning and problem-solving [6].

Since then, SABiO has been used for building several domain ontologies. In 2004, we devised some improvement opportunities to evolve it to a better approach for
building ontologies [7]. Based on the feedback given by ontology engineers that have used SABiO, as well as to incorporate best practices recognized by the Ontology Engineering community, we evolved SABiO. The main differences between versions 1.0 and 2.0 of SABiO are: (i) SABiO 2.0 extends the original development process for considering the design, implementation and test of operational ontologies; (ii) SABiO 2.0 considers other support processes beyond ontology documentation, evaluation and integration. The new version suggests also support processes for knowledge acquisition, reuse, and configuration management processes; (iii) SABiO 2.0 considers also a project management process; (iv) SABiO 2.0 recognizes the importance of the use of foundational ontologies in the development of domain ontologies, and proposes the use of an ontologically well-founded language during ontology capture; (v) Regarding ontology evaluation, SABiO 2.0 addresses both ontology verification and validation.

This paper presents the new version of SABiO, and it is organized as follows. In Section 2, we present an overview of SABiO 2.0. Section 3 presents SABiO's development process, describing each one of its activities, as well as the roles involved in their accomplishment, artifacts required and produced, and techniques and guidelines for performing them. Section 4 presents SABiO's support processes. Section 5 discusses related work. Finally, in Section 6, we present our final considerations.

2 An Overview of SABiO

Figure 1 presents an overview of SABiO 2.0. As this figure shows, SABiO prescribes an ontology development process, which is the backbone of the method. SABiO development process comprises five main phases: (i) Purpose identification and requirements elicitation; (ii) ontology capture and formalization; (iii) ontology design; (iv) ontology implementation; and (v) ontology test. Support processes are performed in parallel to the development process.

SABiO focuses on the development of domain ontologies. Two main types of domain ontologies can be developed using SABiO: reference domain ontologies and operational domain ontologies. As aforementioned, a reference domain ontology is a special kind of conceptual model, aiming to make a clear and precise description of the domain elements for the purposes of communication, learning and problem solving. An operational ontology, in turn, is an implementation version of a reference ontology. Operational ontologies are designed with the focus on guaranteeing some desirable computational properties [6]. Thus, before implementing an operational ontology, a design phase should be accomplished taking non-functional requirements as well as the ontology implementation platform into account. If someone is interested in building a reference domain ontology, then she shall accomplish only the first three activities of the development process. If someone wants to build an operational domain ontology, then she has to perform the entire development process.
It is important to highlight that although Figure 1 suggests a somewhat sequential workflow, SABiO does not prescribe any specific life cycle model. Thus, life cycle models such as Waterfall, Incremental and Spiral can be adopted. However, we strongly recommend incremental and iterative development, and thus life cycle models such as the Incremental Model or the one underlying the Rational Unified Process (RUP) are preferred. For details about these and other life cycle models, see [8].

In the next section we discuss in details the five phases of the SABiO development process. In Section 4, we focus on the support processes.

3 SABiO’s Development Process

As aforementioned, the SABiO development process comprises five main phases. Each phase is composed of activities, which are to be performed by workers playing certain roles. The main roles considered in SABiO are: (i) domain expert, who is a specialist in the ontology domain, and provides the knowledge that is to be modeled and implemented in the domain ontology [9]; (ii) ontology user, representing everyone who intends to use the ontology for a given purpose [9]; (iii) ontology engineer, who is responsible for the reference ontology, and thus is responsible for the initial phases of the ontology development process; (iv) ontology designer, who is responsible for the design of an operational ontology; (v) ontology programmer, who is responsible for implementing an operational ontology; (vi) ontology tester, who is re-
sponsible for testing an operational ontology. It is worthwhile to point out that a given worker can play several roles in a given project. For instance, generally the roles of ontology designer and ontology programmer are played by the same worker, since both roles require great knowledge regarding the ontology implementation platform.

3.1 Ontology Purpose Identification and Requirements Elicitation

The first phase in SABiO's development process is Purpose Identification and Requirements Elicitation. As Fig. 2 shows, this phase comprises three activities that occur in an iterative way, all of them involving the participation of the ontology engineer, domain experts and potential ontology users.

Initially, we need to identify the ontology purpose and its intended uses. Once defined the ontology purpose, we should elicit its requirements. These should take the intended uses into account, and can be stated as competency questions: the questions that the ontology should be able to answer [5]. By establishing the ontology competence, we reach an effective way to determine what is relevant to the ontology and what is not, i.e. we define its scope. Moreover, we give a justification for the ontology, and we provide a way for its evaluation.

If the domain of interest is complex, we need to modularize the ontology. This allows us to deal with the inherent complexity of the domain, as well as to make room to an incremental development, eventually by different groups. SABiO suggests decomposing the ontology into sub-ontologies. To do so, it is necessary to perform an initial knowledge acquisition in this phase. UML package diagrams should be developed for graphically showing the sub-ontologies and the dependencies between them.

We should highlight that these three activities must be performed in an iterative way. Based on the purpose initially established for the ontology and the intended uses elicited with potential users of the ontology, competency questions are outlined. By defining these questions and briefly acquiring knowledge from the experts in the field to identify the inherent complexity of the domain, the ontology engineer may identify sub-ontologies and allocate the competency questions to the sub-ontologies identified. The growing understanding of the domain can lead to a better understanding of the purpose of the ontology and the identification of new intended uses, starting a new cycle.
3.2 Ontology Capture and Formalization

This is the most important phase in the ontology development process. The goal is to capture the domain conceptualization based on the ontology competence. The relevant concepts and relations should be identified and organized [4]. A graphical model is a key instrument for supporting communication, meaning negotiation and consensus establishment with domain experts. This graphical model should be accompanied by a dictionary of terms. In the current version of SABiO, we advocate in favor of using OntoUML [10] as modeling language. OntoUML is a UML class diagram profile that incorporates important foundational distinctions made by the Unified Foundational Ontology (UFO) [10].

As Fig. 3 shows, this phase begins with the conceptual modeling. During conceptual modeling, initially the ontology engineer should identify the main concepts and relations in the domain. Concepts should be properly organized in taxonomies. Since SABiO professes the use of OntoUML, the ontology engineer should classify each concept according to the types defined in OntoUML (kind, subkind, phase, role, category, rolemixin etc.). Moreover, OntoUML constraints for relating these types should be respected. Ontological patterns and modeling rules inherent to OntoUML, such as the ones for modeling subkinds, phases, roles [11] and rolemixins [10], should be applied in order to achieve consistent conceptual models.

The choice of the terms to be used to make reference to the concepts and relations in the ontology should be carefully made, avoiding terms with cloudy interpretation. Concepts that can be described using other concepts should clearly refer to them. Primitive concepts, i.e. those that cannot be defined in terms of other concepts in the ontology, should be defined using natural language and examples, taking the due care to avoid ambiguities and inconsistencies. Term definitions for concepts and relations should be registered in the Dictionary of Terms.

The ontology conceptualization capture is strongly supported by knowledge acquisition activities. Knowledge can be elicited with domain experts, as well as from sources of consolidated knowledge, such as books, international standards, and reference models. Knowledge elicited from these sources can be incorporated to the ontology by applying techniques for transforming non-ontological resources into ontology fragments [12, 13]. In special, ontological analysis is an important technique for building ontologies from non-ontological resources. For an example of such case, see [14]. Existing ontologies, ontology conceptual patterns [15] and foundational ontolo-
gies are also important resources to be reused. When reusing fragments of existing ontologies or ontology conceptual patterns, it may be useful to apply techniques for ontology merge, reengineering or mapping [12].

Concepts, relations and properties are the basis of an ontology, but they can be not enough to capture the domain conceptualization. Constraints must also be taken into account. Thus, axioms specifying constraints and definitions of derived concepts in an ontology should be specified. Initially, we do not need to write down formal axioms, rather the axioms should be written in natural language, simply reflecting inferences and constraints on the universe of discourse.

The axioms in an ontology can present two different forms and purposes: derivation axioms and consolidation axioms. Derivation axioms are those that allow new information to be derived from the previously existing knowledge. They are typically presented as inference rules. Consolidation axioms, on the other hand, typically define constraints for establishing a relation or for defining an object as an instance of a concept, aiming at guaranteeing consistency.

The process of defining axioms should be guided by the competency questions. The axioms in the ontology must be necessary and sufficient to express the competency questions and to characterize their solutions. If the ontology elements (concepts, relations, properties and axioms) are not enough for this purpose, then additional concepts, relations, properties or axioms must be added to the ontology. In this sense, the ontology capture is an iterative process, strongly linked with the evaluation process.

In this phase, ontology quality criteria should be observed, chiefly: (i) clarity, concerning the meaning of the defined terms; (ii) coherence, mainly between the textual definitions, examples, conceptual models and axioms; (iii) minimal ontological commitment, to allow the parties committed to the ontology to be free to specialize and instantiate the ontology as needed; (iv) competence, in order to verify if the ontology is capable to answer its competency questions.

In order to avoid ambiguity, axioms should be written in a formal language. In this language, in contrast to the natural language, we have signs that are unambiguous and formulations that are exact and, therefore, clarity and correctness can be evaluated with greater easiness and accuracy. Thus, axioms should also be written in a formal language. When it is not necessary any special commitment with a specific formalism that proves itself to be adequate to the ontology in development, the first order logic (FOL) tends to be an adequate formalism, since it embeds few ontological commitments. Thus, in a general way, SABiO recommends the use of FOL to formalize the axioms in the reference ontology.

It is important to stress that formal axioms are not able to substitute their descriptions in natural language; rather, they are to be used to support these descriptions or to be added to them, working as a way where some ideas are checked in relation to completeness and, perhaps, coherence. In fact, each representation plays a specific role.

When FOL is the formalism adopted, we need to map the ontology elements in constants, functions and predicates. After that, it is possible to write the formal axioms. For allowing traceability in relation to the conceptual models, we recommend that the names of the predicates are the same of the corresponding elements in the conceptual models.
Finally, it is worthwhile to point out that, many times, during the ontology capture and formalization phase, new competency questions for the ontology arise. Thus, iteration with the previous phase (Purpose Identification and Requirements Elicitation) is very common.

### 3.3 Ontology Design

Once a reference ontology is produced, many times we want to get an operational version to be used by computer applications. In order to achieve this operational version, we need to design and implement it in a particular operational ontology language. In the design phase, the conceptual specification of the reference ontology should be transformed into a design specification by taking into account a number of issues ranging from architectural issues and non-functional requirements, to target a particular implementation environment. The same reference ontology can potentially be used to produce a number of (even radically) different designs [6].

Fig. 4 shows the activities that comprise the ontology design phase. First, the ontology engineer must work together with the ontology designer to elicit non-functional requirements for the operational ontology, and to define the platform on which it will be implemented. Unlike reference ontologies, operational ontologies are not focused on representation adequacy, but are designed with the focus on guaranteeing desirable computational properties. The design phase, thus, is necessary to bridge the gap between the conceptual modeling of reference ontologies and the coding of them in terms of a specific operational ontology language [6] (such as, for instance, OWL and RDFS, but other DL-based languages [16], Datalog-based languages [17], relational databases [18], etc.). Issues that should be addressed in the design phase include: determining how to deal with the differences in expressivity of the languages that are used in each of these phases; or how to produce lightweight specifications that maximize specific non-functional requirements, such as reasoning performance.

![Ontology Design Diagram](image)

Fig. 4. The Ontology Capture and Formalization Phase.

Once defined the implementation platform, the ontology designer must revisit the ontology modularization defined in the beginning of the project. Now, she has to take the non-functional requirements and the characteristics of the implementation plat-
form into account. As an output, the ontology designer produces the ontology architecture. Ontology architectural patterns [15] can be reused in this activity.

Finally, during detailed design, the ontology designer has to address the problems related to the lower expressivity of the operational languages when compared to the models expressed in OntoUML and the axioms formalized in FOL (outputs of the ontology capture and formalization phase). Generally, heavyweight ontologies must give rise to lightweight ontologies. Ontology design patterns and ontology programming patterns (idioms) [15] are very useful during detailed design.

3.4 Ontology Implementation

The implementation phase regards implementing the ontology in the chosen operational language. In fact, there is not a clear frontier between the detailed design and the ontology implementation. Thus, the solution of problems related to the implementation platform, and the application of ontology design patterns and ontology idioms can occur in both phases.

![Fig. 5. The Ontology Implementation Phase.](image)

3.5 Ontology Test

In SABiO, ontology test refers to dynamic verification and validation of the behavior of the operational ontology on a finite set of test cases, against the expected behavior regarding the competency questions. In this sense, SABiO's testing phase is driven by the competency questions. A test case comprises an implementation of a competency question as a query in the chosen operational language, plus instantiation data from the fragment of the ontology being tested and the expected result (based on the considered instantiation).

As Fig. 6 shows, ontology test in SABiO comprises three activities. Initially, test cases are run in the context of a sub-ontology. As sub-ontologies are integrated, ontology integration testing is performed. In this activity, the same test cases are re-run, but now considering the sub-ontologies already integrated. Finally, ontology testing is performed. In this activity, the test cases are again run in the context of the full ontology. During ontology testing, new test cases can be defined for testing non-functional requirements. Thus, ontology testing may include, among others, recovery and stress testing for web ontologies, and performance testing for checking inference performance. Validation testing can also be performed by using the operational ontology in...
actual software applications, according to the intended uses originally proposed to the ontology. Validation testing should be performed by ontology users.

![Diagram of the Ontology Test Phase]

Fig. 6. The Ontology Test Phase.

4 SABiO’s Support Processes

As shown in Fig. 1, SABiO considers five main support processes: knowledge acquisition, documentation, configuration management, evaluation and reuse. These processes span the whole development process (or considerable parts of it).

4.1 Knowledge Acquisition Process

Knowledge acquisition occurs mainly in the initial phases of the ontology development process. Conventional methods and techniques for knowledge acquisition and for requirements elicitation applies, mainly those devoted to collaborative knowledge acquisition, such as brainstorming, and argumentation [19].

Domain experts are the main source for knowledge acquisition. Without the involvement of them, the ontology project can be impaired. Other important sources of knowledge are consolidated bibliographic material, such as classical books and international standards, reference models, system models, existing ontologies, and ontology patterns. Thus, the knowledge acquisition process is strong related to the reuse process, as well as other ontology-related activities not explicitly shown as a support process in SABiO, such as ontology merge, ontology reengineering and so on.

4.2 Reuse Process

Along the development process, there are many opportunities for reusing conceptualizations already established for the domain in hands, including ontological resources and non-ontological resources. Regarding ontological resources, we envision four main types of resources: existing domain ontologies, core ontologies, foundational ontologies, and ontology patterns.

Existing ontologies for the domain can be reused, totally or partially. Especially in the case of partially reusing existing domain ontology, techniques of ontology merge, mapping and eventually reengineering apply [12]. Reuse of core ontologies (ontolo-
gies that provide a precise definition of structural knowledge in a specific field that spans across different application domains in this field [20]) are made mainly by means of specialization. Concepts and relations of the core ontology are extended by means of subtype relations, in order to capture the more specific conceptualization regarding the domain in hands. New concepts, relations, properties and axioms can then be introduced in the domain ontology. Foundational ontologies can also be re-used. In this case, reuse can be done by means of specializations (as in the case of core ontologies), but also by analogy. In reuse by analogy, foundational concepts and relations are not explicitly extended in the domain ontology, but implicitly used for deriving the structure of a portion of the domain ontology. In this sense, reuse by analogy is strongly related to the structuring of the concepts and relations in a domain ontology. A third way of reusing foundational ontologies is by using its foundations to analyze fragments of the domain ontology (ontological analysis). The use of OntoUML as modeling language for developing reference domain ontologies, and OntoClean [21] are examples of ways of performing ontological analysis.

Reuse can also be achieved by means of ontology patterns (OPs). In [15], Falbo et al. revisit Gangemi and Presutti's [22] classification for OPs, and propose a new one that is closely related to patterns in Software Engineering. According to Falbo et al. [15], there are four main types of OPs: conceptual OPs, architectural OPs, design OPs, and programming OPs (idioms).

Ontology Conceptual Patterns are fragments of either foundational ontologies (Foundational OPs) or domain reference ontologies (Domain-related OPs). They are to be used during the ontology conceptual modeling activity, and focus only on conceptual aspects, without any concern with the technology or language to be used for deriving an operational ontology. Ontology Conceptual Patterns are analogous to Analysis Patterns in Software Engineering. Foundational OPs (FOPs) are reusable fragments of foundational ontologies. Since foundational ontologies span across many fields and model the very basic and general concepts and relations that make up the world [23], FOPs can be applied in any domain. They are reused by analogy between the pattern and the problem in hand. An example of a FOP is the rolemixin pattern, which addresses the problem of specifying roles with multiple disjoint allowed types. This pattern was extracted from the ontology of substantial universals of the Unified Foundational Ontology (UFO) [10]. Domain-related OPs (DROPs) are reusable fragments extracted from reference domain ontologies. In contrast with FOPs, DROPs are reused by extension, i.e. concepts and relations of the pattern are specialized when the pattern is reused.

Ontology Architectural Patterns are patterns that describe how to arrange an ontology (generally a large one) in terms of sub-ontologies or ontology modules, as well as patterns that deal with the modular architecture of an ontology network, where the involved ontologies play the role of modules [22]. These patterns can be used both during the purpose identification and requirements elicitation phase (sub-ontologies identification), and in the ontology design phase (ontology architectural design activity). Since modularity is recognized as an important quality characteristic of good ontologies, we advocate for their use since the first stages of ontology development, for splitting the ontology into smaller parts, allowing tackling the problems one at a
time. When applied in the architectural design activity, the purpose is to reorganize the ontology modules for addressing technological aspects, in special by taking non-functional requirements into account [15].

Ontology Design Patterns (ODPs) are to be used during the ontology detailed design activity. There are two main types of ODPs [15]: logical and reasoning ODPs. Reasoning ODPs addresses specific design problems related to improving reasoning with ontologies (and qualities related to reasoning, such as computational tractability, decidability and reasoning performance). Logical ODPs, in turn, regards problems related to the expressivity of the formalism to be used in ontology implementation. They help to solve design problems that appear when the primitives of the implementation language do not directly support certain logical constructs. Logical ODPs are extremely important for ontology design, since most languages for coding operational ontologies are not focused on representation adequacy, but are designed with the focus on guaranteeing desirable computational properties [15]. We should highlight, however, that many patterns that address reasoning and logical problems are, in fact, Ontology Idioms (or Ontology Programming Patterns), since they describe how to solve problems related to reasoning or to the expressivity of a specific language (e.g. OWL) [15]. Ontology idioms can be reused both during ontology detail design and ontology implementation activities.

Finally, during ontology test, test cases can be reused.

### 4.3 Documentation Process

All the results of the ontology development process must be documented, including purposes, intended uses, competency questions, conceptual models, textual descriptions of concepts and relations, informal and formal axioms, non-functional requirements, ontology architecture, design solutions, test cases and test results. Even results from some of the other support processes, such as evaluation, must be documented. So, documentation is a process that has to occur in parallel with the others.

In order to ensure uniformity in the ontology projects, it is useful to define the basic set of documents to be produced in all projects. Moreover, templates for the main documents must also be defined. SABIO suggests three main documents to be produced as the documentation of an ontology project. As a result of the first two phases of the development process, a Reference Ontology Specification should be produced. The template for this document presents the following structure:

1. Introduction
2. Purpose Description
3. Domain Description
4. Reference Ontology
   4.1. Ontology Modularization
   4.2. Sub-Ontology-1
      - Competency Questions
      - Conceptual Model
      - Informal and Formal Axioms
      - Competency Questions Verification Table
5. Dictionary of Terms

The Operational Ontology Design Specification documents the aspects related to the design phase, including: information related to the implementation platform, non-functional requirements, ontology architecture, and main design decisions.

For documenting the operational ontology (i.e., the source code implementing the ontology), SABiO suggests that the organization defines naming conventions and rules for commenting the resulting code. Finally, for documenting the ontology test phase, a test document shall be produced, including test cases and test results.

Due to the highly collaborative nature of ontology projects, we suggest the use of wikis for documentation, especially for documenting reference ontologies.

4.4 Configuration Management Process

The main documents proposed by SABiO, as well as the source code of the operational ontologies must have their configuration managed. Thus, once approved, they must be submitted to the Configuration Management, where they will be controlled at least concerning changes, versions, and delivery.

4.5 Evaluation Process

SABiO’s evaluation process comprises two main perspectives: ontology verification and ontology validation. Paraphrasing the definitions of software verification and validation, in the field of ontology we can say that the concerns of ontology verification and validation (V&V) are the following:

- **Ontology Verification**: aims to ensure that the ontology is being built correctly, in the sense that the output artifacts of an activity meet the specifications imposed on them in previous activities.
- **Ontology Validation**: aims to ensure that the right ontology is being built, that is, the ontology fulfills its specific intended purpose.

Both verification and validation must begin early in the ontology development process. Moreover, we should highlight that, although shown as an activity of the software development process, ontology test is, in fact, an evaluation activity. Ontology Test is an activity performed for evaluating ontology quality, and for improving it, by identifying defects and problems. Ontology testing consists of the dynamic V&V (i.e., running code) of the behavior of an operational ontology on a finite set of test cases, against the expected behavior. On the other hand, several other static V&V activities (those not involving running the code) must be performed during the ontology development process. Those activities are performed by means of technical review. In the context of ontology engineering, the purpose of a technical review is to evaluate an intermediary work product of the ontology development process to determine its suitability for its intended use. The results should confirm (or not) that the work product meets the specifications, and adheres to standards.
Concerning ontology verification, the focus is on two main aspects: (i) Are the ontology quality criteria (competency, clarity, coherence, consistency, minimal ontological commitment, etc.) being met? (ii) Are the established standards (e.g., document templates) and processes being correctly applied (i.e. is there conformance)? Regarding the quality criteria, one stands out: competency. As aforementioned, SABiO ontology test phase is competency driven. However, this is not enough. During the ontology capture and formalization phase, the reference ontology should also be verified whether it meets the requirements posed as competency questions. This can be done by means of technical reviews, applying expert judgment. A table indicating which ontology elements (concepts, relations, properties and axioms) are able to answer each competency question should be built. The purpose of this table goes further verification. It can also be used as a traceability tool, supporting change management (see Configuration Management process). The competency questions play an essential role in the verification of the ontology completeness, especially when considering its axioms. In a particular domain, we can identify a great number of concepts, relations and properties, and to write down several axioms. Therefore, we have to pay attention not to include in the ontology more elements than the necessary. In this context we ought to have in mind the principle of minimal ontological commitment. The set of ontology elements must be necessary and sufficient to express the competency questions and to characterize their solutions and nothing else [5]. Elements in the ontology that do not contribute to answer its competency questions must be excluded.

Concerning ontology validation, the participation of domain experts and ontology users is essential. Ontology users have to evaluate whether the ontology is adequate for their intended uses. For validating the reference ontology with domain experts, the use of a graphical notation is very important, since generally they are not able to read formal specifications. Besides expert judgment, another relatively easy way to validate a reference ontology is by means of instantiation. The reference ontology should be able to represent real world situations. Thus, an instantiation table should be produced from real world situations, showing that the ontology is truly capturing the domain conceptualization.

5 Related Work

There are many ontology methods proposed in the literature. However, as pointed by Corcho et al. [24], none of the existing approaches is fully mature if compared to software engineering methodologies. Although we clearly have advanced in the last years, as shown by the findings of the survey performed by Simperl et al. in 2009 [25], there are still room for improvements, and software engineering play an important role in this scenario. Among the most popular and complete ontology engineering methodologies, the following stand out: METHONTOLOGY [24], and The NeOn Methodology for Ontology Engineering [12].

One striking feature of SABiO when compared to other ontology engineering methods is the recognition that both reference domain ontologies and operational ontologies are useful in themselves. SABiO supports the development of both types of domain ontologies. If someone is interested in building a reference domain ontology, then the ontology project shall accomplish only the first three activities of the devel-
opment process. If someone wants to build an operational domain ontology, then the ontology project has to perform the entire development process. This distinction leads to the perception of the importance of a design phase (as largely recognized in Software Engineering) in the ontology development process. Moreover, pattern-oriented reuse in SABiO is also guided by this distinction. Other OE methods do not prescribe a design phase in this sense.

6 Conclusions

This paper presented the current version of SABiO, an ontology engineering method. The striking features of SABiO when compared to other ontology engineering methods are: (i) The recognition that both reference domain ontologies and operational ontologies are useful in themselves. SABiO supports the development of both types of domain ontologies. This distinction leads to the perception of the importance of a design phase (as largely recognized in Software Engineering) in the ontology development process. Moreover, pattern-oriented reuse in SABiO is also guided by this distinction. Other ontology engineering methods do not prescribe a design phase in the sense used in Software Engineering. (ii) SABiO recognizes the importance of the use of foundational ontologies in the development of domain ontologies, and proposes the use of OntoUML during ontology capture, as well as ontological analysis techniques. (iii) SABiO explicitly considers both verification and validation during ontology evaluation.

Acknowledgments. This research is funded by the Brazilian Research Funding Agencies FAPES (PRONEX # 52272362/11) and CNPq (#311578/2011-0).

References

18. Silberndag, D., Reasoning with Ontologies in Databases: including optimization strategies, evaluation and example SQL code, VDM Verlag, 2011.
24. Corcho, O.; Fernández-López, M.; Gómez-Pérez, A. Methodologies, tools and languages for building ontologies. Where is their meeting point? Data & knowledge engineering 46 (1), 2003, pp. 41-64.