Onto4ALLEditor:

a Graphic Web Ontology Editor for Information Science

Fabrício Martins Mendonça[1]

Jeanne Louise Emygdio[2]

Lucas Piazzi de Castro[3]

Eduardo Ribeiro Felipe[4]

Abstract: Of all the artificial intelligence technologies, ontologies are the most dependent on human expertise and classification skills. The methodologies for building ontologies share a stage that requires the use of an ontology editor for formalizing knowledge. For students in information science. The use of such editors is still an obstacle, since the information science curriculum does not cover the skills required for ontology development. To bridge this gap, we have developed Onto4AllEditor, a web-based graphic ontology editor, with the aim of providing resources for the creation of lightweight ontologies. Onto4AllEditor aims also to foster ontologies within Information Science.

Keywords: ontology; ontology editor; knowledge representation, modeling, library & information science, artificial intelligence.

1 INTRODUCTION

Since the 1990s, the number of initiatives for using ontologies in information system has increased dramatically. Although only massively adopted over the last 15 years, the idea behind ontologies had already been mentioned in the 1960s (MEALY, 1967). There were a significant number of initiatives in the 1980s (WAND et. al, 1999) and in 1990s. By the beginning of the 2000s, ontologies were already widely disseminated. The interest in ontologies for information systems originates in the inconsistency of practices during the early years of modeling, which has been identified as the main reason for issues that plague information systems in the 21st century (SMITH and WELY 2001).

[1] Doutor em Ciência da Informação pela UFMG. Professor na Universidade Federal de Juiz de Fora (UFJF). Email: fabricio.mendonca@ice.ufjf.br
[2] Doutora em Ciência da Informação pela UFMG. Email: jeanne.emygdio@gmail.com
[3] Graduando em Ciência da Computação pela UFJF. Developer. Email: lpiazzi@ice.ufjf.br
[4] Doutor em Ciência da Informação pela UFMG. Professor na UNIFEI Itabira. Email: erfelipee@gmail.com

Conceptual modeling formally describes aspects of the physical and social world for purposes of understanding and communication (MYLOPOULOS, 1993). Over the last 50 years, conceptual models have been motivated by the search for improvements in representation in information systems. Ontology is an artifact able to bring such improvements (GUIZZARDI, 1995).

Throughout its short history, ontologies have concentrated on technological fields like conceptual modeling and knowledge representation. However, other scientific areas have been interested in the subject such as library and information Science (LIS) (ALMEIDA, 2013) and linguistics (SCHALLEY, 2019). LIS has researched and developed methods for classification - a core subject for the area - which explains the discipline's interest in ontologies. However, as an applied social science field, LIS has been more concerned with theoretical aspects from philosophical perspective (SMITH, 2004). This theoretical emphasis has resulted in practical deficiencies in, for example, techniques and tools for constructing ontologies. In addition, the LIS curriculum, in general, does not cover subjects required to build ontologies, such a as logic and formal modeling.

In many senses, the use of editors for building ontologies is still a challenge in LIS. For example, the Protégé Editor (MUSEN, 2015) after two decades of use and having received the best evaluations form users (MALIK, 2017) (WARREN, 2013), is still a difficult resource for LIS beginners. Indeed, this is not only true within the LIS community. Other scientific areas that make use of ontologies have also reported common hindrances, as revealed in interviews with users of Protégé, Web Protégé, TopBraid, and SWOOP. The main issues highlighted are: (1) lack of a collaborative environment; (2) lack of support for viewing ontologies according to user preferences; (3) poor alternatives to debugging ontology errors; (4) lack of support for searching terms in external ontologies for the purpose of reuse; and (5) unfriendly interface for navigating the class hierarchy (VIGO et al, 2014).

Within this context, the present paper describes an ongoing research project that aims to disseminate the development of ontologies within LIS. Within the scope of the project, we have developed Onto4AllEditor, a web-based graphic ontology editor, which provides basic resources for the creation of ontologies. In addition, the editor includes resources to mitigate

the issues identified by users of the most famous tools. Onto4AllEditor was intended to foster the construction of ontologies by facilitating development by users with no experience in ontologies. Certain features for this purpose have been incorporated into the tool, for example: (1) a collaborative graphic interface; (2) association between tasks in the editor and stages of a methodology; (3) a console of warnings and modeling errors; and (4) an intuitive way to reusing top-level ontologies. In addition to these features, the development of the editor has included usability tests with a group of LIS's users with the goal of improving the editor according to the needs of LIS. Indeed, with this project we expect to expand research in ontologies by offering Onto4AllEditor as an educational resource to students and researchers of LIS.

The remainder of the paper is organized as follows: the second section - Background - provides background information, highlighting issues in building ontologies and the stages of the *OntoForInfoScience* methodology; the third section - Ontology editors: examples and issues - lists well-known ontology editors and refers to other related work; the fourth section - Onto4AllEditor: features and functionalities - presents the most important features of Onto4AllEditor; the fifth section - Testing in LIS community - outlines the software and usability tests performed, as well as their findings; and, sixth section - Final remarks - offers our final notes and proposals for future work.

2 BACKGROUND

This section presents a theoretical background, highlighting aspects that motivated the creation of Onto4AllEditor. First, we describe well-known issues in the process of ontology building, then provide a brief description of the methodology underlying the editor.

2.1 Issues in building ontologies

Despite the large number of ontologies developed over the past 20 years, the literature indicates the development errors are still the norm. The survey performed here does not cover all errors, which could total dozens, but only introduces the most common types. Comprehensive works reveal different approaches for types of errors: common patterns (RECTOR el al, 2004); anomalies or pitfalls (POVEDA-VILLALON et al, 2010)

(POVEDA-VILLALON et al, 2012); ontological anti-patterns (ROUSSEY et al, 2009) (SALES and GUIZZARDI, 2015); errors of definitions in class and relationships (CEUSTERS et al 2004) (MUNN and SMITH, 2008); overloads in relationships (GUARINO and WELTY, 2004). In addition, several proposals for error classifications have been proposed (GANGEMI et al, 2006) (POVEDA-VILLALON et al, 2010), which do not seem to be in agreement. Certain authors have compiled extensive lists of errors, but our list below is not exhaustive, covering only the most common.

Errors involving classes:

- Inaccurate and recursive definitions (MUNN and SMITH, 2008);
- Misuse of instance-class (SCHULZ et al, 2006);
- Use of synonyms instead of equivalence (POVEDA-VILLALON et al, 2010);
- Use of "miscellaneous classes" (SALES and GUIZZARDI, 2015).

Conceptual and logical errors:

- Creation of polysemia (POVEDA-VILLALON et al, 2010);
- Use-mention confusion (MUNN and SMITH, 2008);
- Use of multiple inheritance (NOY and McGUINNES, 2001);
- Misuse of logical "and" and "r" (ROUSSEY et al, 2009) ;
- Misuse of "some not" and "not some" operators (RECTOR el al, 2004);
- Use of the "Relator Mediating Overlapping" (SALES and GUIZZARDI, 2015);
- Use of "SumOfSome" (ROUSSEY et al, 2009);
- Use of "UniversalExistence" (SALES and GUIZZARDI, 2015).

Certain standards and practices can minimize errors. Two standards are: (1) the ontology design patterns (ODPs), which incorporate successful templates (GANGEMI and PRESUTTI, 2009); and (2) inductive modeling rules which derive rules from ODPs (GUIZZARDI et al, 2011). Best practices include:

- The use of clear and concise definitions; the use of primitive and defined classes (RECTOR el al, 2004);

- the inclusion of genuine ontological relationships; non-use of multiple inheritance (MUNN and SMITH, 2008);

- the inclusion of inverse relationships; the definition of all textual properties of classes and relations; avoid recursive definitions (ERDMANN and WATERFELD, 2012);

- the use of inductive rules derived from three types of ODPs: phase pattern, subtype patterns, role modeling pattern (GUIZZARDI et al, 2011).

Despite all these efforts, complications still plague the process of building ontologies.

2.2 Overview of the methodology for building ontologies

Methodologies for ontology building have arising since the 1990s, including Methontology (GOMÉZ-PÉREZ and FIGUEROA, 2009), Method 101(NOY and McGUINNES, 2001); NeOn (FIGUEROA, 2008) and Up for ONtology (UPON) (DE NICOLA et al, 2009). These methodologies, although well established, maintain errors (see second section, Background) and as a result, one can find low-quality ontologies on the Internet (VIGO et al, 2014).

The Onto4AllEditor project was conceives to address known difficulties and needs within the LIS area. The first initiative was the OntoForInfoScience methodology (MENDONÇA, 2015), because we noticed that difficulties have originated in the fact that most methodologies do not detail their steps (MENDONÇA and ALMEIDA, 2016). In this sense, OntoForInfoScience sought to make the ontological development cycle accessible, by explaining technical, logical, and philosophical terms that, in general, are not part of LIS curriculum.

OntoForInfoScience fosters the reuse of ontologies suggesting that ontologies develop from a top-level, in addition to taking advantage of existent domain ontologies. For this reason, OntoForInfoScience is based on the best aspects of other methodologies, reusing steps of the well-known: Methontology, NeOn, and Method 101. OntoForInfoScience prescribes a set of eight methodological steps: specification, knowledge acquisition, conceptualization, ontological grounds, formalization, evaluation, documentation, and availability. With the exception of the knowledge acquisition step, all others can be performed within the

Onto4AllEditor. With this centralization, we seek to facilitate ontological modeling and guide the LIS professional in the process.

3 ONTOLOGY EDITORS: EXAMPLES AND ISSUES

As mentioned, we believe that some ontology building issues can be attributed to an editor's limitations. This section presents well-known editors. First, we list editors available on the Internet and then provide comments and evaluation on their use.

3.1 Ontology editors: examples and issues

The following is a non-exhaustive list of editors cited in three updated references (W3C, 2020) (IOF, 2020) (BRAUN et al, 2009) - (Table 1).

Table 1. Alphabetical list of ontology editors

Editor	Features	Reference
CmapTools	This conceptual mapping editor permits the representation of nodes and conceptual maps in a graphic environment.	https://cmap.ihmc.us/cmaptools/
Eddy for Graphol	This editor uses the Graphol graphic language, developing graphs composed of nodes and relationships.	https://www.obdasystems.com/eddy
Graffo	This open-source editor is able to present classes, properties and restrictions in OWL.	https://essepuntato.it/graffoo/
Hozo	This tool includes an editor, a server and an ontology manager in a distributed environment, allowing collaboration.	http://www.hozo.jp/
ICOM	This tool allows manipulation of multiple ontologies through a graphic interface with support for class diagrams and reasoners.	https://www.inf.unibz.it/~franconi/icom/
Menthor	This open-source editor allows integration with OntoUML,	https://ontouml.org/ontouml/tooling/

	stereotypes, and exportation to other tools.	
NeOn Toolkit	This tool includes a methodology for developing large-scale ontologies in a distributed environment.	http://neon-project.org/
NORMA	This is an editor for object-role modeling and ontologies with support for automatic reasoning.	Not available
OBO-Edit	This open-source editor was coded in Java and created to support the OBO ontological format.	http://oboedit.org/
OLED	This editor uses the OntoUML language, based on UFO, offering validation, simulation, and pattern detection.	https://ontouml.org/ontouml/tooling/
OWLGrEd	This editor has a graphic interface to edit and view ontologies and supports extensions with plugins for UML modeling.	http://owlgred.lumii.lv/
Protégé	This open-source editor has an interface for creating and editing ontologies, as well as a query processor.	https://protege.stanford.edu/
SWOOP	This open-source editor has OWL support and a web-based architecture; it includes automatic reasoning support.	http://www.mindswap.org/
TopBraid Composer	This commercial IDE for RDF modeling in a graphic interface is useful for inferences, mapping and queries.	https://www.topquadrant.com/products/topbraid-composer/
yEd Graph Editor	A commercial editor that allows multiple tasks with graphs, semantic networks, and OWL.	https://www.yworks.com/products/yed
WebProtégé	A collaborative web version of Protégé that supports OWL and OBO editions; it is able to import and export formats.	https://protege.stanford.edu/

Fonte: elaborado pelo autor (2021)

3.2 About using ontology editors

We conducted a survey with the aim of exploring user comments on ontology editors. Our search returned roughly two dozen works over the past ten years, with comments about the use and application of editors, although such works do not include all editors mentioned in the overview list (the prior section). The more relevant comments are summarized in the remainder of this section.

Quantitative research with 65 professionals working on ontologies identified, among other issues, the most commonly used editors (WARREN, 2013). Protégé was first, with 50% of respondents using the tool; TopBraid Composer, a commercial editor, came in second with 14%; the NeOn Toolkit[29], CmapTools (CAÑAS et al, 2004) and SWOOP (KALYANPUR et al, 2006) appeared with 6%, 5% and 4%, respectively. Within the scope of editors focused on Biomedicine, the OBO Editor (DAY-RICHTER et al, 2007) and Neurolex (FAHIM et al, 2012) appeared with 3% and 2%, respectively.

Some of the research seek to measure the usability aspects of editors, for example, by presenting participants with a sequence of tasks in more than one editor for comparison (MUSEN, 2015). The time spent with tasks in Protégé was about a half the time spent with others, such as TopBraid, NeOn, and SWOOP. In the usability test, TopBraid Composer achieved the best results, followed by Protégé, SWOOP, NeOn Toolkit, and CmapTools. Despite achieving the best score in usability, TopBraid was not considered the best editor, a position occupied by Protégé, according to the positive feedback from the participants. The NeOn Toolkit appeared in the third position, SWOOP in the fourth, and IHMC Cmap Tools was considered the worst among the evaluated tools.

An interview with 30 ontologists discussed best editor for beginners (SIRICHAROEN, 2018). Protégé was indicated as the most suitable and, among the others evaluated, only TopBraid Composer had more than one mention. The results showed that, while about 70% of ontologists recommended Protégé, only 10% indicated TopBraid Composer. The other tools in the survey were: NeOn Toolkit, SWOOP, and OntoStudio. One relevant aspect mentioned

by those who chose Protégé was the pizza tutorial available on the Internet (see the discussion below). Other reasons included the large user community and the fact that it is free and open source.

From our surveys of ontology editors, we identified requirements to be included in our ontological modeling editor. Through this experience, we have included the following features in Onto4AllEditor: it is free software, it contains a dynamic and intuitive interface available on the Internet, it provides immediate feedback on modeling errors, it facilitates the reuse of top-level ontologies, and it guides users through the steps of a methodology. All of these features were implemented to facilitate the manipulations of lightweight ontologies by LIS professionals and domain experts. We detail these features and functionalities in the next section.

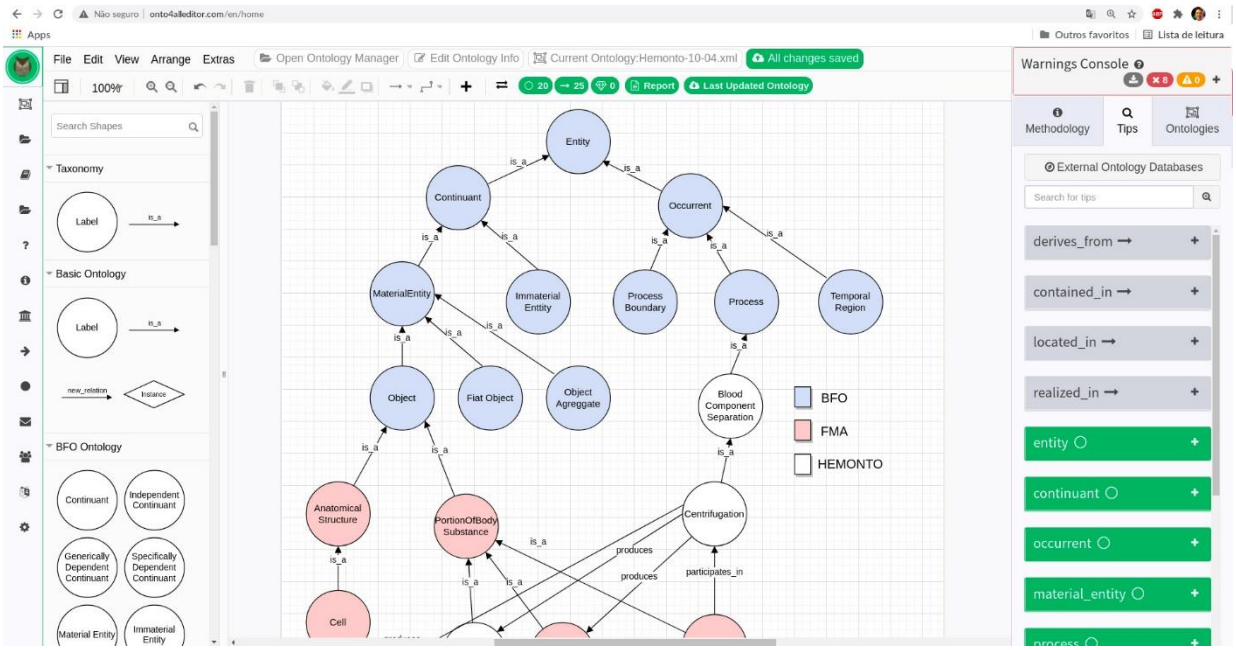
4 ONTO4ALLEDITOR: FEATURES AND FUNCTIONALITIES

This section describes Onto4AllEditor explaining its features and functionalities, and the steps required to using it to create ontologies. One should note that the Onto4AllEditor was preceded in its main features by pioneering initiatives such as the OntoUML Lightweight Editor (OLED) (GUERSON et al, 2015) and CROWD (BRAUN et al, 2020).

Although Onto4AllEditor has precedents, it shows the distinctive feature of the graphic interface and the warning for modeling errors in real time. The strategy was to implement a warning console, which is typical in programming interface development environments (IDE). In addition to error warnings, like those detected by compilers in IDEs, Onto4AllEditor also alerts users about methodological errors, which are not mandatory to correct. The warnings implemented to date are presented later in this section.

Neither modeling languages such as UML, nor logical languages such as OWL with its reasoning ability, are subject of study in most LIS programs. Onto4AllEditor is an alternative to access the complex subject of ontologies in addition to disseminating it. As a lightweight tool, it can be used by newcomers from other areas. The central feature is graphic modeling, which facilitates the building of ontologies in an intuitive and responsive interface (Figure 1).

Figure 1. Graphic modeling interface for ontologies in Onto4AllEditor



Fonte: elaborado pelo autor (2021)

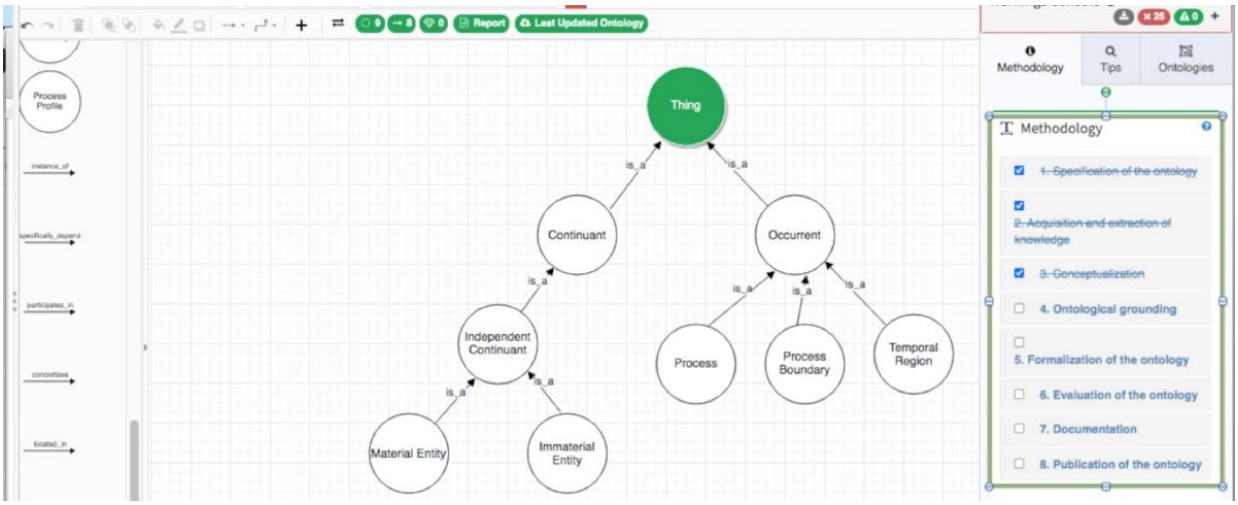
Figure 1 depicts a fragment of HEMONTO, an ontology of hemotherapy. In the upper tab of the menu, the graphic interface is activated by the insertion of classes, relations and properties. Colors and legends are used to indicate relevant information, for example, classes imported from other ontologies or OWL primitives. HEMONTO (Figure 1) imports classes from Basic Formal Ontology (BFO) and Foundational Model Anatomy (FMA).

The reuse of ontologies is strongly recommended (GRENON and SMTH, 2004). Reuse is implemented in both OntoForInfoScience (step 4) and in Onto4AllEditor. In the editor, the administrator can register classes, relationships and properties of top-level ontologies, so that other users can reuse these resources in other domain ontologies. One example is the relationships “contained_in” and “derives_from”, which are reused from BFO and illustrated in Figure 1. There is also the possibility of searching for classes and properties recorded in the editor database, facilitating reuse.

The editor is able to perform seven out of eight steps of OntoForInfoScience (see Figures 2 and 3). In order to guide the ontologist, the methodological steps are displayed in a dialog box with an indicator of progression (Figure 2). When accessing a specific step, the editor opens another dialog box (Figure 3) with a description of the step and the tasks to be

performed. Figure 3 shows the explanation of the first step (specification of the ontology) and how to carry it out from a template. The second stage of the methodology involves acquiring domain knowledge from other sources, and which is not performed by Onto4AllEditor.

Figure 2. Stages of the OntoForInfoScience methodology



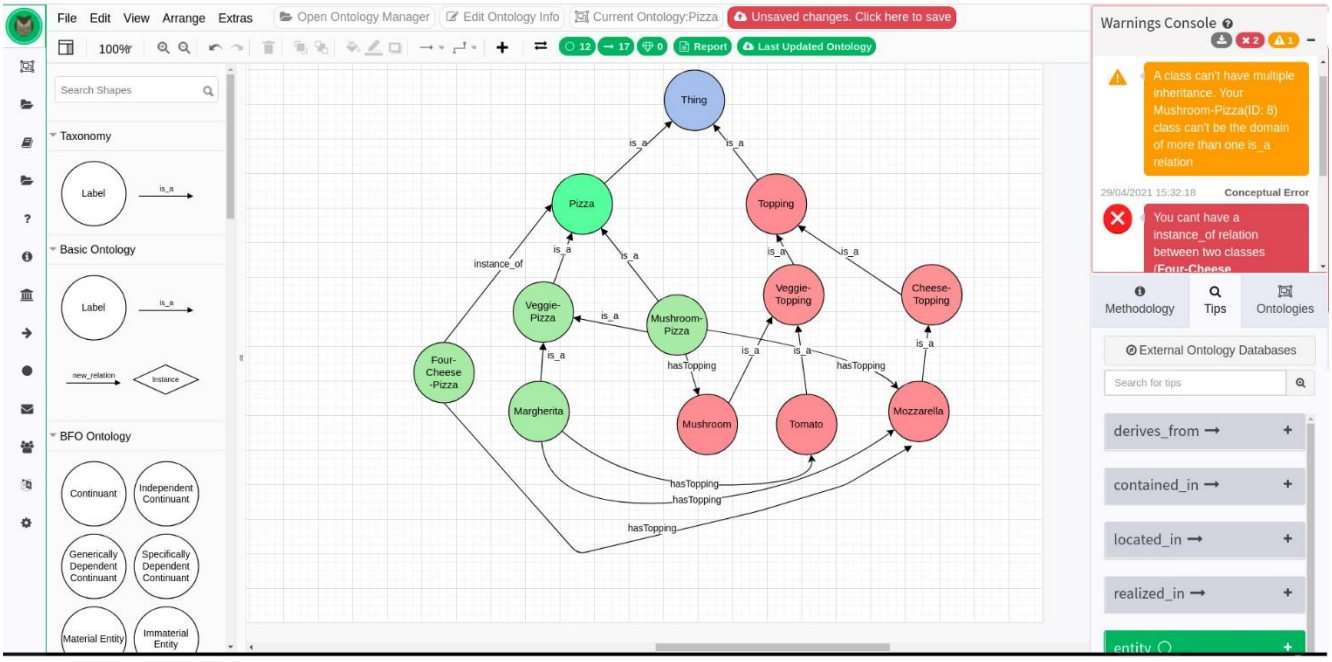
Fonte: elaborado pelo autor (2021)

Step 6 (validation) was implemented in a warning console (Figure 3), which displays alerts about modeling actions to be avoided, in addition to accounting for the number of classes, relationships, and instances already entered. The warnings we have implemented to date are: (1) duplicate classes; (2) incorrect use of the “instance_of” relationship; (3) multiple inheritance; (4) circularity; (5) misuse of inverse relationships; (6) lack of annotations and metadata; and (7) errors in the use of the graphic environment.

Figure 3 illustrates the warnings on the console for the example ontology of pizzas, in which the user receives a pair of warnings (highlighted): (1) multiple inheritance in the Mushroom-Pizza class, which is, wrongly, simultaneously a subclass of Pizza and Veggie-Pizza; and (2) incorrect use of the “instance_of” relationship between the Four-Cheese-Pizza and Pizza classes, since this relationship must connect an instance to a class. In both cases, the console dynamically detects these issues, recording and displaying them along with the error time,

the respective class or relationship involved, and a warning identifier. The warning report can be exported from the console to a text file.

Figure 3. Warning console



Fonte: elaborado pelo autor (2021)

Step 7 (documentation) is performed with Onto4AllEditor throughout the development process. To this end, users can access a repository to manage their own ontologies. The repository displays the last ten edited ontologies as well as the five favorites. Finally, step 8 (availability), consists of exporting the ontology into one of the three formats available in the editor: image (extension svg), semi-structured text (extension xml), and logic (extension owl). If ontology is exported to logic, it can be handled in other editors.

Finally, we describe technologies involved in the editor's implementation. Onto4AllEditor uses the Laravel framework, version 6.x, using Model View Controller architecture, and more than one programming language: PHP (back-end), HTML, CSS, and JavaScript (front-end) with the package Laravel-AdminLTE, as well as MySQL database. Two specific JavaScript libraries were adopted: Mx Graph, modified to allow the creation of the interactive diagramming component; and the jQuery API, employed in the dynamic features of the

editor, such as the alert. The management and version control of the source code is done through GitHub. Onto4AllEditor is free to use and available at https://onto4alleditor.com/en.

5 TESTING IN LIS COMMUNITY

As we mentioned, one of the reasons for creating Onto4AllEditor was to foster the building and use of ontologies in LIS. With this purpose in mind, the development of the editor considered the needs of the LIS community by performing specific software and usability tests. In this section, we provide some of the results of recent tests of three levels, as follows:

Unit test, which separately verifies the functionality of testable software elements. It is not conducted by the developers themselves, but is done by accessing the source code and by debugging tools;

Integration test, which verifies the interaction between the components. This kind of test must take place continuously and can follow top-down, bottom-up or architecture-based strategies for producing an incremental scenario of component integration or subsystems identified as functional elements;

System test, which evaluates the behavior of the entire system. This kind of test is suitable for the evaluation of non-functional requirements such as security, performance, accuracy, interfaces to other applications, hardware devices, and operating environments.

The purposes that guide the creation of test cases are variable and also divided between two major audiences: software developers and software users. Given our goal, we focused on testing for an audience of LIS students. Within this context, we chose certain dimensions to perform the tests47:

Acceptance: to evaluate whether the system presents a desired behavior with respect to users' requirements to come up with an acceptance criterion.

Interface: to reach correctness in the interaction between software components ,enabling the control information and exchange of data.

Usability: to assess the learning curve to use the software, in addition to the support documentation and the systems ability to respond to user errors.

From a methodological point of view, we articulated the essential parameters for the tests as follows: (1) levels of unit tests and integration; (2) reliability, configuration and compliance purposes from the perspective of developers; and (3) purposes of acceptance, interface, usability, and human-machine interaction from the perspective of users. These three parameters with the three aforementioned dimensions satisfy a scenario in which programmers, working on the evolution and customization of an open-source framework, are not ontologists themselves.

We defined the tests to be performed, on the taxonomy and basic ontology functionalities, within a period of 15 days for each test, preceded by a meeting between the professors and the team of volunteers. The tests were carried out according to plans previously prepared by a volunteer with training in software engineering. They were divided into two parts: the test plan for taxonomy and the test plan for basic ontology.

5.1 Test plan for taxonomy functionality

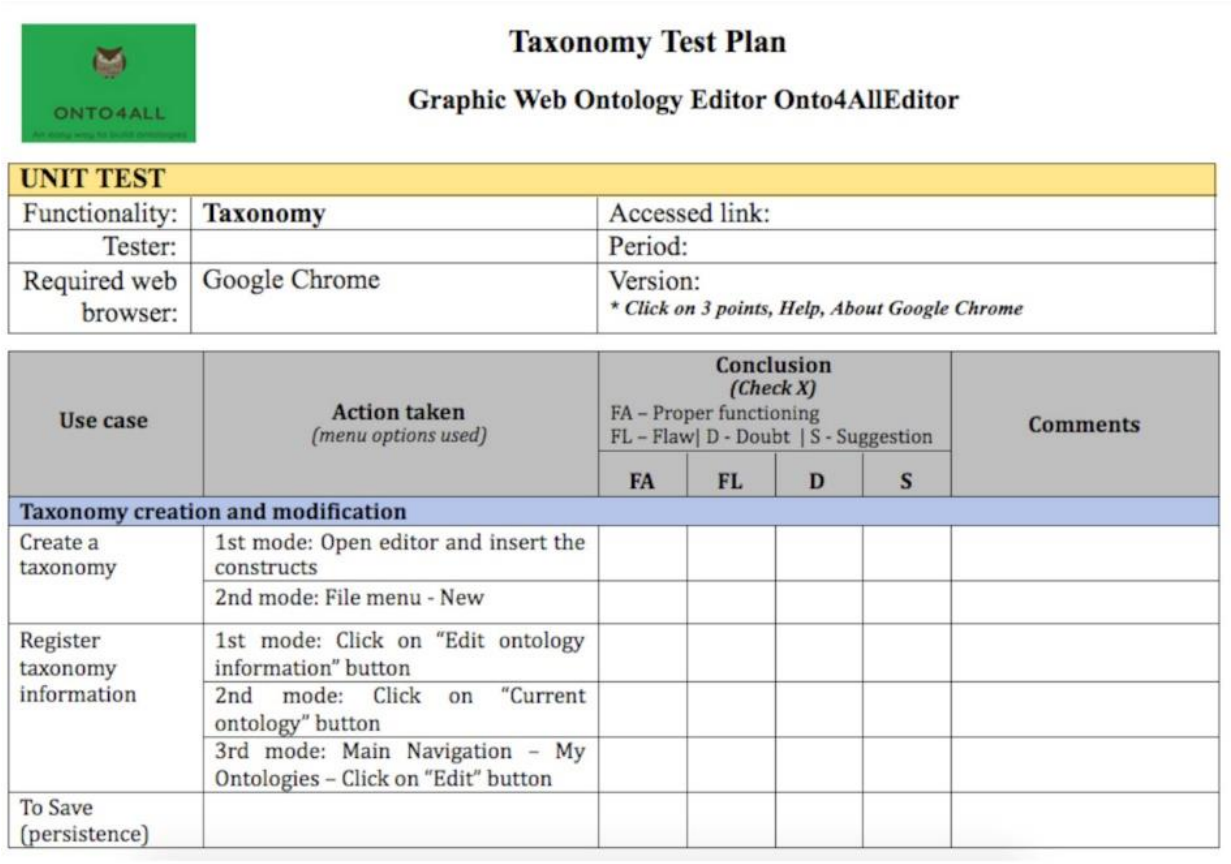
The taxonomy functionality serves the basic purpose of building the ontology hierarchical structure in the form of an inverted tree, based on is-a relations. The test plan designed for this functionality comprised five test cases (CT):

- Creation and modification of the taxonomy;
- Registration and modification of annotations on the taxonomy categories;
- Use of OWL constructs;
- Operation of the warning console;
- Operation of the buttons for metrics.

Since the editor's graphic interface enables several ways of accessing to each resource, the volunteers were asked to record the action taken, to evaluate the resource operation and to

register additional observations. A fragment of the Taxonomy Test Plan is depicted in Figure 4.

Figure 4. Fragment of the Taxonomy Test Plan



Taxonomy Test Plan

Graphic Web Ontology Editor Onto4AllEditor

UNIT TEST		
Functionality:	**Taxonomy**	Accessed link:
Tester:		Period:
Required web browser:	Google Chrome	Version: ** Click on 3 points, Help, About Google Chrome*

Use case	**Action taken** *(menu options used)*	**Conclusion** *(Check X)* FA – Proper functioning FL – Flaw| D - Doubt | S - Suggestion				**Comments**
		FA	**FL**	**D**	**S**	
Taxonomy creation and modification						
Create a taxonomy	1st mode: Open editor and insert the constructs					
	2nd mode: File menu - New					
Register taxonomy information	1st mode: Click on "Edit ontology information" button					
	2nd mode: Click on "Current ontology" button					
	3rd mode: Main Navigation – My Ontologies – Click on "Edit" button					
To Save (persistence)						

Fonte: elaborado pelo autor (2021)

The issues addressed in each use case are summarized in Table 2, in addition to a classification regarding levels and purposes established according to the following codes:

- Level: "U"- unitary; "I"- integration;
- Purpose D (developers): "D1"- Reliability; "D2- Configuration; "D3"- Compliance;
- Purpose C (users): "C1"- Acceptance; "C2"- Interface; "C3"- Usability.
- The browser (N) used for testing and its version

Table 2. Taxonomy test cases

CT	Questions for analysis	Level		Purpose			Purpose		
		U	I	D1	D2	D3	C1	C2	C3
1	Taxonomy creation and modification; Registration and modification of information about the scope of the taxonomy; Persistence.	x		x		x		x	
2	Registration and modification of annotation on the categories; Persistence.	x		x		x			x
3	Insertion; Movement; Exclusion; Relationship.	x				x	x	x	x
4	Coherence of messages; Clarity; Ease in locating errors.		x	x		x	x	x	x
5	Totalization of constructs; Download; Quick access.	x	x	x	x	x	x	x	x
N	Browser and version used in the test.				x				
L	Language.				x				

Fonte: elaborado pelo autor (2021)

5.2 Test plan for the basic ontology functionality

The basic ontology functionality permits modification of a previously constructed taxonomy by adding nontaxonomic relations. The test plan designed for this functionality comprised seven test cases (CT):

1. Insertion of non-taxonomic relationships;
2. Modification of classes and ontological relationships;
3. Use of OWL constructs;
4. Operation of the warning console;
5. Operation of the metric buttons;
6. Exporting and importing files created on the Onto4all editor
7. Importing files created in other ontology editors.

Again, volunteers were asked to register the actions taken to access the resource, to evaluate the operation and to register important observations. A fragment of the basic ontology test plan is depicted in Figure 5.

Figure 5. Fragment of the Basic Ontology Test Plan

Use case	Action taken *(menu options used)*	Conclusion *(Check X)* FA – Proper functioning FL – Flaw| D - Doubt S - Suggestion				Comments
		FA	FL	D	S	
Insertion of ontological relations *** Use constructs ONLY from the BASIC ONTOLOGY area**						
Open the taxonomy created in the taxonomy test						
Enter the first type to a class	Click on available construct in construct bar					
Enter the second type to the same class as before	Click on the first instance inserted with the right button and then, choose the "duplicate" option					
Enter a third type for a class other than the previous class.	Click on an instance inserted in the ontology with the right button and then on the option "copy". Place the mouse cursor in a free area of the ontology, right-click and then click on the "paste here" option.					
Create at least 3 non-taxonomic relations between classes. Consider 3 types of relations: • Direct • Reverse						

Fonte: elaborado pelo autor (2021)

The issues addressed in each use case are summarized in Table 3, in addition to establishing a classification according to both levels and purposes using the following codes:

- Level: “U”- unitary; “I”- integration;
- Purpose D (developers): “D1”- Reliability; “D2”- Configuration; “D3”- Compliance;
- Purpose C (consumers): “C1”- Acceptance; “C2”- Interface; “C3”- Usability.

- The browser (N) used for testing and its version.

Table 3. Basic ontology test cases

CT	Questions for analysis	Level		Purpose			Purpose		
		U	I	D1	D2	D3	C1	C2	C3
1	Insertion of types for existing categories; Insertion of non-taxonomic relations; Definition of inverse and equivalent relationships; Persistence.	x	x	x		x		x	
2	Registration of annotations on the types and relationships created; Changes in the presentation of types and classes; Definition of disjunctions; Persistence.	x	x	x		x	x	x	x
3	Insertion; Movement; Exclusion; Relationship.	x				x	x	x	x
4	Coherence of messages; Clarity; Ease in locating errors.		x	x		x	x	x	x
5	Totalization of constructs; Download; Quick access.		x	x	x	x	x	x	x
6	Exporting files in image, owl and xml extensions; Importing of previously exported files; Persistence.	x	x	x	x	x	x	x	x
7	Importing files in .owl and .xml formats; Persistence.	x		x		x	x		
N	Browser and version used in the test.				x				
L	Language.				x				

Fonte: elaborado pelo autor (2021)

5.3 Test Results

From January to March 2021, five student volunteers in graduate and undergraduate programs in LIS, geographically diverse, performed the test plans designed to evaluate the taxonomy and basic ontology functionalities. They included two PhD candidates in information science, one Master's candidate in information science, and two undergraduate students in library science. All of them had had an introductory ontology discipline course, each at their respective level. Accordingly, they had knowledge about ontologies ranging from basic to advanced.

For the taxonomy tests, using Google Chrome (88.0.4324.104, official version, 64-bit), four volunteers proceed the test using the Onto4All interface in Portuguese and one in English. The performance during the taxonomy tests identified three functional flaws:

1. When using constructs, the editor allows the creation of anonymous relationships between classes, starting from a pre-existing class (CT3);
2. An issue in the "last saved ontology" access button, which causes a runtime error (CT5);
3. The recognition of the "owl:Thing" construct even when it was written in lower case (CT1), contrary to OWL specification.

In addition, the tests identified an integration failure between the "Open ontology manager" and "Edit ontology information" commands caused by the information synchronization during the execution of the "Unsaved changes" command (CT5). The volunteers also suggested eleven improvements, mainly regarding resources to guide LIS users, for example, to exhibit explanatory notes during the use of annotations and properties (CT2).

During the basic ontology tests, the volunteers identified two functional flaws:

1. When importing ontologies, the editor does not recognize the OWL file format (CT7);
2. The name of the construct in bold carries HTML tags for the construct label (CT2).

The volunteers also suggested five general improvements and identified five demands for integration tests to make the editor easier to use.

6 FINAL REMARKS

The present paper introduced the ontology graphic web editor — Onto4AllEditor — as an alternative for ontological modeling oriented toward no specialists. The editor is based on OntoForInfoScience, a methodology for the ontological development cycle and aimed at

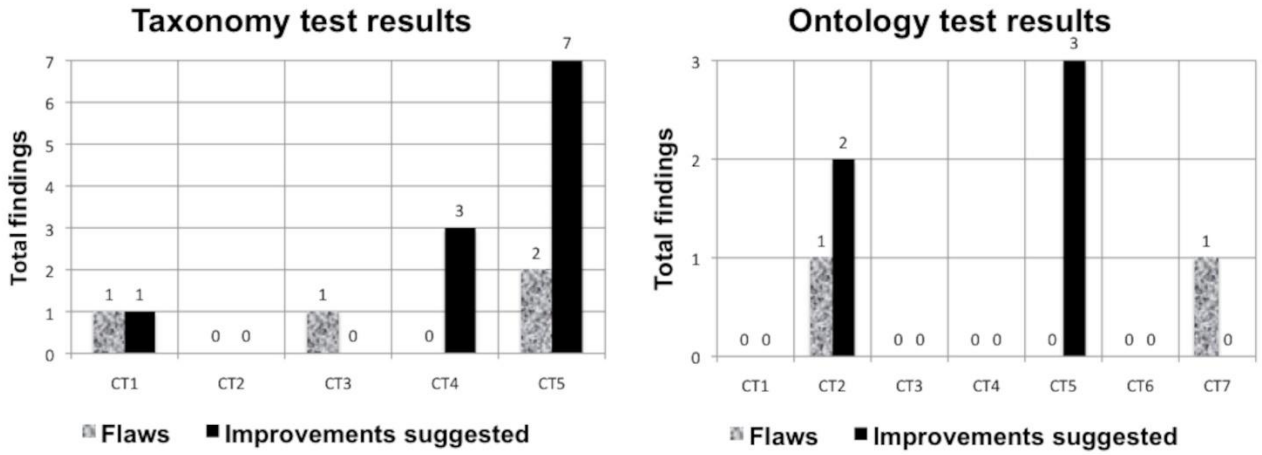
less experienced developers. Both the methodology and the editor are part of an ongoing project that aims to promote the development of ontologies and thus strengthen research in LIS.

Onto4AllEditor makes use of a graphic interface and other resources that ensure the correct generation of formal content in OWL. Some editors have a high license cost, but Onto4AllEditor is an open-source free software. It is not useful directly compare Onto4AllEditor to Protégé, since the latter is a more comprehensive, stable, and mature software, a level it has reached in almost thirty years of existence. However, these are tools in completely different stages, created in distinct contexts. For the reasons presented and principally because of its emphasis on LIS, we believe that Onto4AllEditor is a worthwhile initiative.

The editor has been tested by professors, undergraduate, and graduate students from two universities in disciplines that involve ontologies. Onto4AllEditor has also been tested experimentally for the development of industrial-based ontologies in the power supply industry. Figure 3 presents the results of the taxonomy and basic ontology tests including the flaws observed, the improvements proposed by test case, and the consolidation of the results along with the purposes categorized by type of audience.

Within the set of flaws identified in the scope of taxonomies test, two notably impact the purposes for the editor, because they do not allow users to ensure that their work persisted. This occurs because the messages do not reflect the persistence that does take place. Among the two flaws identified in the scope of ontologies test, only one compromises the purposes for the editor, by not allowing the importation of ontologies in the .owl file extension. As a result of the tests, four new releases have been uploaded to the editor's repository on GitHub. The set of sixteen suggested improvements and five demands for integration tests contributed to our planned purposes.

Figure 6. Test results vs. purpose per audience



Test results vs. purpose per audience

Use case	Total flaws	Total improvements	Impacts for developers			Impacts for consumers		
			Reliability	Configuration	Compliance	Acceptance	Interface	Usability
CT1	1	1	X	-	X	-	X	-
CT2	1	2	X	-	X	-	-	X
CT3	1	0	-	-	X	X	X	X
CT4	0	3	X	-	X	X	X	X
CT5	2	10	X	X	X	X	X	X
CT7	1	0	X	-	X	X	-	-
Totalization	**6**	**16**						

Fonte: elaborado pelo autor (2021)

We conclude by indicating three most important limitations of the Onto4AllEditor at the current stage: (1) the low level of building ontologies in LIS, which hinders the tests; (2) our small group of volunteers for testing; and (3) the axiom builder is still in the beta stage. Beyond solving these main limitations, for the future we plan to develop an alternative for extracting terms from documents using natural language processing techniques and the syntactic alignment of ontologies developed within the editor. Learning from experience with Protégé, future features will be incorporated into the tool as plug-ins. We also plan to include built-in classes and ontological relationships defined in the BFO framework, so that Onto4AllEditor's users can benefit from these resources created in pioneer initiatives.

REFERENCES

ALMEIDA, M.B. Revisiting ontologies: A necessary clarification. *Journal of the American Society for Information Science and Technology*, [s. 1], v. 64, n. 8, p. 1682-93, 2013.

BOURQUE, P. and FAIRLEY, R. E. (2014). "Guide to the software engineering body of knowledge". Version 3.0, (IEEE Computer Society, 2014).

BRAUN G., ESTEVEZ E. and FILLOTRANNI, P. A Reference Architecture for Ontology Engineering Web Environments. *Journal of Computer Science & Technology*, vol. 17, no. 2, pp. 1–3, 2018.

BRAUN, G., GIMENEZ, C., CECCHI L. and FILOTTRANI, P. R. (2020). Crowd: A Visual Tool for Involving Stakeholders into Ontology Engineering Tasks. KI - Künstliche Intelligenz, v.34, n.1, (2020), 365-71. DOI: https://doi.org/10.1007/s13218-020-00657-8.

CAÑAS, A. J., HILL, G., CARFF, R., SURI, LOTT, J., GÓMEZ, G., ESKRIDGE, T. C., ARROYO, M., CARVAJAL, R. CMapTools: a Knowledge Modeling and Sharing Environment. In *First Int. Conference on Concept Mapping*, Spain, 2004.

CEUSTERS, W.; SMITH, B., KUMAR, A. and DAHEN, C. (2004) "Mistakes in medical ontologies: where do they come from and how can they be detected?" Stud. Health Technol. Inform. v.102, (2004):145–64.

DAY-RICHTER J., HARRIS M. A., HAENDEL M. The Gene Ontology OBO-Edit Working Group, Suzanna Lewis, OBO-Edit—an ontology editor for biologists. In *Bioinformatics*, Volume 23, Issue 16, 15 August 2007, Pages 2198–2200, 2007.

DE NICOLA, A.; MISSIKOFF, M.; e NAVIGLI, R. A software engineering approach to ontology building. *Information Systems* 34, p. 258–275, 2009.

ERDMANN, M. and WATERFELD W. "Overview of the Neon Toolkit". Ontology Engineering in Networked World, (2012), 281-301.

FERNÁNDEZ, M. et al. Building a chemical ontology using methontology and the ontology design environment. *Intelligent Systems*, v. 14, n. 1, p. 37-46, Jan./Feb.1999.

FERNÁNDEZ, M.; CORCHO, O. Methodologies and methods for building ontologies. In: FERNÁNDEZ, M.; CORCHO, O. *Ontological engineering*. London: Springer, 2004. p. 107-153.

GANGEMI, A. CATENNACCI, C., CIARAMITA, M. and LEHMANN, J. (2006) "Modelling Ontology Evaluation and Validation", in The Semantic Web: Research and Applications. ESWC 2006, Lecture Notes in Computer Science, v. 4011, eds. Sure Y., Domingue J. (Springer, Berlin, Heidelberg: 2006).

GANGEMI, A. and PRESSUTI, V. (2009) "Ontology design patterns", in Handbook on ontologies, (Berlin, Heidelberg: Springer, 2009), 221-43.

GÓMEZ-PÉREZ, A.; FERNÁNDEZ, M.; VICENTE, A. J. (1996). Towards a method to conceptualize domain ontologies. In: *European Conference on Artificial Intelligence – ECAI*, 1996, Budapest, Hungary.

GOMEZ-PEREZ, A. and SUAREZ-FIGUEROA, M. C. (2009). NeOn Methodology for Building Ontology Networks: a Scenario-based Methodology, 2009, accessed April 07, 2020, http://oa.upm.es/5475/1/INVE_MEM_2009_64399.pdf.

GUARINO, N. Formal Ontology in Information Systems (FOIS), Trento, Itália, 1998. Disponível em: http://citeseer.ist.psu.edu/guarino98formal.html. Acesso em: 2 jul. 2020.

GUARINO, N. and WELTY C. A. (2004). "An Overview of OntoClean", in Handbook on Ontologies. Int. Handbooks on Information Systems, eds. Steffen Staab and Rudi Studer (Berlin: Springer:2004).

GUERSON, J.; SALES, T. P.; GUIZZARDI, G., et al. (2015). "OntoUML Lightweight Editor: A Model-Based Environment to Build, Evaluate and Implement Reference Ontologies," In: *IEEE 19th International Enterprise Distributed Object Computing Workshop*, pp. 144-147, 2015.

GUIZZARDI, G. (2005). "Ontological Foundations For Structural Conceptual Models" (PhD Thesis, Centre for Telematics and Information Technology - University of Twente, 2005).

GUIZZARDI, G.; GRAÇAS, A. P. and GUIZZARDI, R. (2011). "Design patterns and inductive modeling rules to support the construction of ontologically well-founded conceptual models in OntoUML", in International Conference on Advanced Information Systems Engineering, (Berlin, Heidelberg: Spring:2011), 402-13.

GRENON, P. and SMITH, B. (2004) "SNAP and SPAN: Towards Dynamic Spatial Ontology". Spatial Cognition & Computation, v.4, n.1, (2004): 69-103.

GRUBER, T. R. (1992) "What is an Ontology?" Disponível em: http://www.ksl.stanford.edu/kst/what-is-an-ontology.html. Acesso: 06 de dezembro 2021.

IMAM, F. T., LARSON, S. D.; BANDROWSKI, A. et al. (2012) "Development and use of Ontologies Inside the Neuroscience Information Framework: A Practical Approach", 2012, accessed April 28, 2021, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3381282/#s1.

INDUSTRIAL ONTOLOGIES FOUNDRY (IOF). *Helpful materials on ontologies*, 2020. Disponível em: https://www.industrialontologies.org/. Acesso em: 20 março 2020.

KALYANPUR, A., PARSIA B., SIRIN E., GRAU B. C.; HENDLER, J. Swoop: A web ontology editing browser. *Web Semantics: Science, Services and Agents on the World Wide Web*, [s. 1], n. 4.2, pp. 144-153, 2006.

MALIK, Z. H. (2017). "Usability Evaluation of Ontology Engineering Tools", In: *Computing Conference* 2017, London, UK.

MEALY, G. H. (1967). Another Look at Data. Proc. of the Joint Computer Conf., pp. 525-534.

MENDONÇA, F. M. OntoForInfoScience: metodologia para construção de ontologias pelos cientistas da informação – uma aplicação prática no desenvolvimento da ontologia sobre componentes do sangue humano (Hemonto). Tese de Doutorado. Universidade Federal de Minas Gerais (UFMG), Belo Horizonte, Brasil, 2015.

MENDONÇA, F. M.; ALMEIDA, M. B. "OntoForInfoScience: a detailed methodology for construction of ontologies and its application in the blood domain". In *Brazilian Journal of Information Science*, v. 10, p. 1., 2016.

MICHAEL, E.; WATERFELD, W. (2012) Overview of the Neon Toolkit. *Ontology Engineering in a Networked World*, [s. 1], pp. 281-301, 2012.

MUNN K. and SMITH B. (2008) "Applied Ontology: An Introduction", (Heusenstamm, Germany: Ontos Verlag, 2008).

MUSEN, M.A. (2015) The Protégé project: A look back and a look forward. *AI Matters. Association of Computing Machinery SIG-AI*, [s. 1], v. 1, n. 4, 2015.

MYLOPOULOS. J. (1992). "Conceptual Modelling and Telo" in Conceptual Modelling, Databases and CASE: An Integrated View of Information Systems Development, eds. Pericles Loucopoulos and Roberto Zicari (New York: Wiley, 1992).

NOY, N. F.; McGUINNESS, D. L. (2001) Ontology Development 101: A Guide to Creating Your First Ontology. *Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report* SMI-2001-0880, California, USA, 2001.

POVEDA-VILLALON, M., SUAREZ-FIGUEROA, M. C. and GOMEZ-PEREZ, A. (2012) "Validating ontologies with oops!" in International conference on knowledge engineering and knowledge management (Springer, Berlin, Heidelberg: 2012), 267-81.

RECTOR, A.; DRUMMOND, N., HORRIDGE M.; ROGERS, J., et al. (2004) "Owl pizzas: Practical experience of teaching owl-dl: Common errors and common patterns" in Proceedings of EKAW (Springer: 2004), 63–81.

ROUSSEY, C. CORCHO, O. and BLAZQUEZ, L. M. (2009). “A catalogue of OWL ontology antipatterns” in Proceedings of the fifth international conference on knowledge capture (ACM: 2009), 205-6.

SALES, T. P and GUIZZARDI, G. (2015) “Ontological anti-patterns: Empirically uncovered error-prone structures in ontology-driven conceptual models”. Data & Knowledge Engineering, v.99, (2015): 72-104.

SCHALLEY, A.C. (2019). Ontologies and ontological methods in linguistics. In: *Lang Linguist Compass*, 2019;13: e12356. DOI: https://doi.org/10.1111/lnc3.12356.

SCHULZ, S., KUMAR, A. and BITTNER, T. (2006) “Biomedical ontologies: what part-of is and isn’t”. Journal of Biomedical Informatics, v. 39, 2006: 350-61.

SIRICHAROEN, W. V. (2018) Ontology Editors Approach for Ontology Engineering. *International Conference on Control, Automation and Robotics*, [s. 1.], 2018.

SMITH, B. (2004) Ontology and Informations Systems, 2004. Disponível em: http://www.ontology.buffalo.edu/ontology. Acesso em: 25 jan. 2016.

SMITH, B.; WELTY, C. (2001). “FOIS introduction - Ontology: Towards a New Synthesis” in Proceedings of the international conference on Formal Ontology in Information Systems - FOIS ’01(New York, USA: ACM Press, 2001), 3-9.

SUÁREZ-FIGUEROA, M. C. (2008). NeOn D 5.4.1. NeOn Methodology for Building Contextualized Ontology Networks. NeOn project. Available in: http://www.neon-project.org. Acesso em: 06 de Abril de 2020.

Top-Quadrant (2020). “TopBraid Composer”. Disponível em: https://www.topquadrant.com/products/topbraid-composer/ . Acesso em: 6 de Abril de 2020.

VIGO, M.; BAIL, S.; JAY, C. STEVENS, R. (2014). Overcoming the pitfalls of ontology authoring: Strategies and implications for tool design. *International Journal of Human-Computer Studies*, [s. 1.], v. 72, n. 12, p. 835-845, 2014.

W3C 2020. Ontology editors. Available in: https://www.w3.org/wiki/Ontology_editors. Acesso em: 06 de Abril de 2020.

WAND, Y.; STOREY, V. C.; WEBER, R. (1999). An ontological analysis of the relationship construct in conceptual modeling. *ACM Transactions on Database Systems*, New York, v. 24, n. 4, p. 494-528, 1999.

WARREN, P. (2013). Ontology Users' Survey – Summary of Results. *The Knowledge Media Institute (KMi)*, [s. 1.], 2013.