Automatically changing modules in modular ontology development and management

Zubeida C. Khan
Department of Computer Science
University of Cape Town
Cape Town 7701
Council for Scientific and Industrial Research
Pretoria 0001
zkhan@csir.co.za

C. Maria Keet
Department of Computer Science
University of Cape Town
Cape Town 7701
mkeet@cs.uct.ac.za

ABSTRACT

Modularity has been proposed as a solution to deal with large ontologies. This requires, various module management tasks, such as swapping an outdated module for a new one or a computationally costly one for a leaner fragment. No mechanism exists to exchange an arbitrary module automatically. To solve this manual task, we modify the SUGOI algorithm into SUGOI-Gen; with SUGOI-Gen, one can swap any module within a modular system, implemented it, and wrapped a GUI around it. We carried out an experimental evaluation with six ontologies covering three different use-cases to determine whether arbitrary interchangeability is practically doable, and to what extent such changes affect the quality of the module and automated reasoning over it. The results are positive, with the success rate varying between 22-100% depending on the number of mappings between the source and target module. The evaluation also revealed that the interchangeability does indeed have an impact on a module's metrics. Regarding reasoning, when comparing an original ontology to one where a module has been swapped, the processing time is greatly improved for all except one of the swapped modules in the set.

CCS CONCEPTS

•Computing methodologies →Ontology engineering; •Information systems →Web Ontology Language (OWL); •Software and its engineering →*Designing software;*

KEYWORDS

modularity, modularisation, module, ontology metrics, ontology interchangeability, reasoning

ACM Reference format:
Zubeida C. Khan and C. Maria Keet. 2017. Automatically changing modules in modular ontology development and management. In *Proceedings of SAICSIT '17, Thaba Nchu, South Africa, September 26–28, 2017,* 10 pages.
DOI: 10.1145/3129416.3129443

1 INTRODUCTION

Ontology developers use modules to deal with cognitive difficulties due to information overload. When a subject domain is large, a set of smaller modules can be used to represent the domain instead of a large monolithic ontology. In some sets of ontology modules, foundational ontologies and top-domain ontologies are used to structure the subject domain. Foundational ontologies provide high-level categories that are common across various domains, and they have been proposed as a component to facilitate interoperability. A number of foundational ontologies have been developed e.g., DOLCE, BFO [18], GFO [7], SUMO [21], and YAMATO [19]. A top-domain ontology contains the fundamental concepts of a domain. They are commonly used in large, complex domains such as medical systems, and distributed media management [26]. There may be more than one top-domain ontology for a particular domain as well; e.g., BioTop and [1] and GFO-Bio [8] both provide knowledge about life sciences. In other scenarios, modules are used to extract a portion of the ontology for a particular use-case. For instance, the DMOP ontology is modularised by removing some expressive power to a DMOP-EL module to achieve faster automated reasoning [11].

Over the years, a considerable amount of research has been carried out to provide foundations for and to realise modularity [2, 24, 25, 30] and it has been applied for various applications, such as [1, 9, 11, 20, 23]. However, there is little support for module management, i.e., software tools for modules. For instance, consider the modular ontology Subcellular Anatomy Ontology (SAO) [17] containing BFO ontology as a foundational ontology. Consider that upon validation of the ontology the developers notice that none of the SAO domain entities exist in the bfo:occurrent branch of the ontology, i.e., SAO does not need those entities. One solution is to modularise the SAO ontology by removing the bfo:occurrent branch and all sub-entities. Another option is to swap the BFO foundational ontology in the system for a subset module that already exists, the `bfo-continuants.owl` ontology module from the ROMULUS repository [16]. Currently, there is no tool support to do this.

Such changes in modules may affect the quality of the set of ontology modules (networked ontologies), which may even be the driving force of interchanging modules. Ontology module quality has been an active topic of research and various studies have been conducted to determine how to measure the quality of a module [4, 13, 15]. There are various evaluation metrics for a module [4, 13, 15, 22, 27, 29, 31], and it has been shown that different module types have their own set of evaluation metrics with expected values that can be used to check the quality of a module [15]. However, no work has been done on addressing how to *improve* the metrics for an ontology.

Seeing that in many applications modules are used together as components, and some of the components in a modular system are top-domain, foundational, ontology design pattern, or arbitrary modules with potential alignments, this introduces new questions for ontology engineers concerning modularity. The questions we focus on here are as follows, where O_A denotes a domain module, and O_X and O_Y are some modules (be they foundational, top-domain, ontology design pattern, or arbitrary):

(1) If O_A is linked to a module O_X, is it feasible to interchange it to be linked to a different module O_Y?
(2) Does interchanging ontology O_A between O_X and O_Y have an impact on the quality of the modules in the set?
(3) Does interchanging ontology O_A between O_X and O_Y have an impact on the time taken for reasoning?

The aim of this paper is to answer these questions and determine whether interchanging certain modules across different modular systems can be achieved and impact the quality of the modules, and the time taken for reasoning. To achieve this, we modify the algorithms and implementation of SUGOI ('Software Used to Gain Ontology Interchangeability'), which was proposed for assisting with semantic interoperability by allowing one to swap between three foundational ontologies [12]. This generalised version of the algorithm that we propose here, SUGOI-Gen, allows the user to upload a domain ontology, a linked ontology module to be interchanged, and a mapping file[1]. SUGOI-Gen then swaps the linked ontology module to a target according to alignments declared in the mapping file. To the best of our knowledge, there is currently no other tool to assist with module management for the interchangeability of modules in a system. To answer the research questions, we conducted an experimental evaluation with six use-cases for ontology module interchangeability. The evaluation revealed that interchanging modules with SUGOI-Gen, in most cases, has a positive impact on the modularisation metrics of a module. This aids the user in selecting the best candidate module for a use-case. Interchanging a module was also shown to have a positive impact on the time taken for reasoning a module if the target module was smaller. When the target module is larger than the source module to be swapped, the metrics and reasoning time did not have any improvement, as one may expect. SUGOI-Gen and its supplementary materials are available at the foundational ontology library ROMULUS at http://www.thezfiles.co.za/ROMULUS/ontologyInterchange.html.

In the remainder of the paper, we first introduce scenarios for module interchangeability in Section 2. This is followed by the design and implementation for the module interchangeability in Section 3. An experimental evaluation is conducted in Section 4, followed by a discussion in Section 5. Finally, we conclude in Section 6.

[1]Such mappings can be declared manually or through tools such as Logmap [10]

2 MODULE INTERCHANGEABILITY SCENARIOS

To demonstrate ontology interchangeability for modules, we have identified four case-studies to investigate which we discuss here. We begin the section with preliminaries about modularisation and the files that are used for interchangeability. Thereafter we introduce the scenarios.

2.1 Preliminaries

Since we are investigating the impact of interchanging ontologies for modularity, we first state which definition we use for modularity. Multiple similar definitions have been proposed, and we chose an inclusive one: "A module *M* is a subset of a source ontology *O*, *M* ⊂ *O*, either by abstraction, removal or decomposition, or module M is an ontology existing in a set of modules such that, when combined, make up a larger ontology. Module *M* is created for some use-case *U*, and is of a particular type *T*. *T* is classified by a set of annotation features *P*, and is created by using a specific modularisation technique *MT*, and has a set of evaluation metrics *EM* which is used to assess the quality of module *M*." [14].

Several OWL ontology files are used in the interchangeability and we describe the terms used for each one. This terminology is illustrated with the SAO ontology in Figure 1.

- The *Source Modular Ontology* (${}^s\mathcal{M}$) that the user wants to interchange, which comprises the *Source Interchange Module* (${}^s\mathcal{M}_i$) that is the modular component of the source ontology that is to be interchanged, the *Source Domain Modules* (${}^s\mathcal{M}_d$) that is the domain modules in the ontology, and any equivalence or subsumption mappings between entities in ${}^s\mathcal{M}_i$ and ${}^s\mathcal{M}_d$.
- The *Target Modular Ontology* (${}^t\mathcal{M}$) which has been interchanged, which comprises the *Target Interchange Module* (${}^t\mathcal{M}_i$), that is the modular component of the target ontology that the user has selected to interchange to, the *Target Domain Modules* (${}^t\mathcal{M}_d$) that is the domain modules in the ontology, and any equivalence or subsumption mappings between entities in ${}^t\mathcal{M}_i$ and ${}^t\mathcal{M}_d$.
- *Mapping ontology* (${}^aO$): the mapping ontology between the ${}^s\mathcal{M}_i$ and the ${}^t\mathcal{M}_i$.
- *Domain entity*: an entity from ${}^s\mathcal{M}_d$ or ${}^t\mathcal{M}_d$.

2.2 Swapping an ontology for a leaner version of the ontology

For this scenario we consider ontology modules that contain a domain component and an arbitrary module that has a potential alignment to a 'leaner' module. That is, a particular type of fragment of an ontology, such as one represented in a language of lower expressiveness, instead of the 'comprehensive' version of the ontology. A user may want to interchange one of the modules in the set of ontologies for another to perhaps enhance user comprehension or increase performance of the automated reasoner, or of the ontology-driven information system. It is also worthwhile to investigate whether swapping one of its modules for a smaller version of the module has an impact on the modularisation metrics, and the time taken for reasoning.

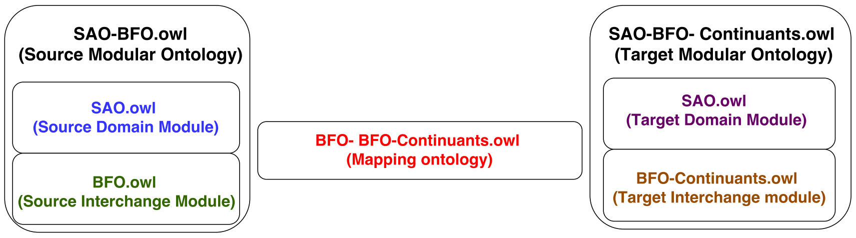


Figure 1: The terminology used for the files involved in interchangeability, using the SAO ontology as example.

A characteristic of this scenario is that the interchanged modules do have an exact overlap in content, just fewer axioms and possibly a subset of the vocabulary as well in the 'leaner' version.

Example 2.1. In the DMOP data mining ontology [11], there is the domain ontology, DMOP and the foundational ontology, DOLCE-Lite. Aside from this, there already exists an EL version of the DOLCE-Lite ontology in the ROMULUS repository [16]. We could consider interchanging the DOLCE-Lite foundational ontology module for the DOLCE-EL foundational ontology module. The basic steps of the algorithm for interchanging between DOLCE to DOLCE-Lite in the DMOP ontology are as follows:

(1) Create a new ontology file, a ${}^{t}\mathcal{M}$: `dmop-dolce-el.owl`.
(2) Copy the entire ${}^{t}\mathcal{M}_i$ to the ${}^{t}\mathcal{M}$: copy the dolce-el ontology into `dmop-dolce-el.owl`.
(3) Copy the axioms from the ${}^{s}\mathcal{M}_d$ to the ${}^{t}\mathcal{M}$: e.g., consider the axioms, axiom1: dmop:Characteristic $\sqsubseteq$ dolce:abstract-quality which exists in the ${}^{s}\mathcal{M}$ dmop-dolce. We add this axiom to the `dmop-dolce-el.owl` ${}^{t}\mathcal{M}$ and it is referred to as a 'new' axiom.
(4) Change the 'new' axioms to reference ${}^{t}\mathcal{M}_i$ entities, if mappings exist: for axiom1, there is an equivalence mapping between `dolce:abstract-quality` and `dolce-el:abstract-quality`, hence we change axiom1 dmop:Characteristic $\sqsubseteq$ dolce:abstract-quality to dmop:Characteristic $\sqsubseteq$ dolce-el:abstract-quality.
(5) If a mapping does not exist, perform on-the-fly subsumption. There are mappings for every DOLCE to DOLCE-EL entity hence we skip this step.
(6) Delete entities that exist in the ${}^{t}\mathcal{M}$ that are from the ${}^{s}\mathcal{M}_i$ but do not appear in an axiom with entities from the ${}^{t}\mathcal{M}_d$, resulting in the final ${}^{t}\mathcal{M}$, `dmop-dolce-el.owl`. Delete the dolce:abstract-quality entity from `dmop-dolce-el.owl`.

2.3 Swapping aligned ontology design patterns

For this scenario we consider ontology modules that contain a domain component and an ontology pattern module that has a potential alignment. It is worthwhile to investigate whether swapping the pattern for its aligned pattern has an impact on the modularisation metrics. For instance, consider the class vs. object property, which can also be found as issue in conceptual modelling: e.g., should 'marriage' be a class Marriage with a number of persons participating in it, or a relationship/object property married-to, i.e., to reify a relationship or not. Or the modelling approach of subsumption vs. inherence: e.g., for a library ontology, take the knowledge 'Librarian inheres in some Person', or: social objects are related to physical objects through an inheresIn object property (Librarian $\sqsubseteq \exists$inheresIn.Person), yet an ontology developer wishes to use this knowledge in a database for a library system, for which an assertion of 'Librarian isA Person' (Librarian $\sqsubseteq$ Person) is much more efficient. Hence, the instantiation of an ontology *pattern*—i.e., involving more than one element—in the library ontology has to be swapped for the simple named class subsumption. Five such types of patterns are aligned in [6], and there are surely more variants.

A characteristic of this scenario is that the ontology patterns are not necessarily of equal size and may use different language features, such as simple class subsumption vs. existential quantification. The overall effect on ontology metrics for one pattern interchange is not expected to have an impact, but it will if this were to be done throughout an ontology for each such instance.

2.4 Swapping foundational ontologies

For this scenario we consider ontology modules that contain a domain component and foundational ontology component with a potential alignment to a target foundational ontology component. Since several foundational ontologies have been developed, this poses problems if a heterogeneous system needs to access ontologies that are linked to conflicting foundational ontologies. Swapping a foundational ontology for another could be performed to solve this problem of semantic interoperability. It is also worthwhile to investigate whether swapping its foundational ontology to an aligned foundational ontology has an impact on the modularisation metrics, and the time taken for reasoning.

A characteristic of this scenario is that the source, mapping, and target ontologies cover the philosophical, high-level categories that are common across various domains.

We provide Example 2.2 for illustrating the interchangeability between foundational ontologies.

Example 2.2. The SEGO ontology [5] is about sensing geographical occurrences, and it is linked to the DOLCE foundational ontology. Let us assume that we wish to integrate the SEGO ontology to the Infectious Disease Ontology (IDO) [3] to gain information about the geographical occurrences of diseases. The IDO ontology, however, is linked to a different foundational ontology than the SEGO ontology. IDO is linked to the BFO foundational ontology. These conflicting foundational ontologies prevent such interoperability. In order to solve this problem, we could consider using SUGOI-Gen to interchange the SEGO ontology from DOLCE to BFO. As an alternative, we could interchange the IDO ontology from BFO to DOLCE. The interchange is subject to the availability of mappings between DOLCE and BFO foundational ontologies. The basic steps of the algorithm for interchanging between DOLCE to GFO as a foundational ontology are as follows, using the SEGO ontology as an example:

(1) Create a new ontology file, a ${}^t\mathcal{M}$: sego-gfo.owl.
(2) Copy the entire ${}^t\mathcal{M}_i$ to the ${}^t\mathcal{M}$: copy the GFO ontology into sego-gfo-bio.owl.
(3) Copy the axioms from the ${}^s\mathcal{M}_d$ to the ${}^t\mathcal{M}$: e.g., consider the axioms, axiom1: sego:sensor $\sqsubseteq$ dolce:physical-object and axiom2: sego:geo-process $\sqsubseteq$ dolce:stative which exist in the ${}^s\mathcal{M}$ sego. We add these axioms to the sego-gfo.owl ${}^t\mathcal{M}$ and they are referred to as 'new' axioms.
(4) Change the 'new' axioms to reference ${}^t\mathcal{M}_i$ entities, if mappings exist: for axiom1, there is an equivalence mapping between dolce:physical-object and gfo:Material_object, hence we change axiom1 sego:sensor $\sqsubseteq$ dolce:physical-object to sego:sensor $\sqsubseteq$ gfo:Material_object. For axiom2, there is no equivalence mapping between dolce:stative and GFO entities; we skip this step.
(5) If a mapping does not exist, perform on-the-fly subsumption: For axiom2, dolce:stative has a superclass dolce:perdurant and the mapping ontology has dolce:perdurant $\equiv$ gfo:Occurrent, so dolce:stative $\sqsubseteq$ gfo:Occurrent is added to sego-gfo.owl.
(6) Delete entities that exist in the ${}^t\mathcal{M}$ that are from the ${}^s\mathcal{M}_i$ but do not appear in an axiom with entities from the ${}^t\mathcal{M}_d$, resulting in the final ${}^t\mathcal{M}$, sego-gfo.owl. Delete the dolce:physical-object entity from seo-gfo.owl.

The position of the sego:geo-process entity is shown in Figure 2 for DOLCE as the ${}^s\mathcal{M}$ and GFO as the ${}^t\mathcal{M}$.

2.5 Swapping top-domain ontologies

For this scenario we consider ontology modules that contain a domain component and a source top-domain component with a potential alignment to a target top-domain component. A user may want to interchange the top-domain component for another to assist with ontology integration and possibly improve ontology metrics and the time taken for reasoning

While it may seem the same as the previous case for foundational ontologies, top-domain ontologies may be linked to a foundational ontology, so the use case includes a 'middle out', needing to address links both above and below it in the taxonomy.

Example 2.3. BioTop [1] and GFO-Bio [8] are both top-domain biological ontologies and CELDA [28] is an ontology for complex cells. CELDA imports the BioTop ontology as a top-domain ontology. Suppose there is a set of bio-medical ontologies that use GFO-Bio as a top-domain ontology and we wish to use CELDA in an application together with the set of bio-medical entities. It is difficult to achieve seamless integration because CELDA has BioTop as a top-domain component. Hence, we consider interchanging CELDA's top-domain ontology from BioTop to GFO-Bio. The basic steps of the algorithm for interchanging between BioTop to GFO-Bio as a top-domain ontology as an example are as follows, and Figure 3 illustrates the example:

(1) Create a new ontology file, a ${}^t\mathcal{M}$: CELDA-gfo-bio.owl.

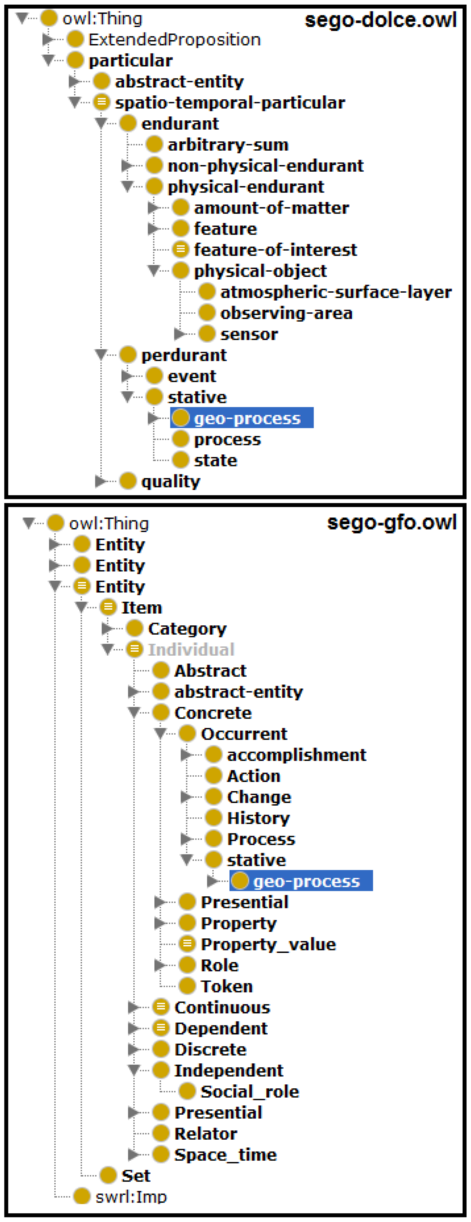


Figure 2: The position of the geo:process entity in the ${}^s\mathcal{M}$ and ${}^t\mathcal{M}$ ontologies for the SEGO example.

(2) Copy the entire ${}^t\mathcal{M}_i$ to the ${}^t\mathcal{M}$: copy the GFO-Bio ontology into CELDA-gfo-bio.owl.
(3) Copy the axioms from the ${}^s\mathcal{M}_d$ to the ${}^t\mathcal{M}$: e.g., consider the axioms, axiom1: celda:compound $\sqsubseteq$ biotop:MaterialObject and axiom2: celda:GO_008150 $\sqsubseteq$ biotop:bio_molecular_process which exist in the ${}^s\mathcal{M}$ CELDA. We add these axioms to the CELDA-gfo-bio.owl ${}^t\mathcal{M}$ and they are referred to as 'new' axioms.

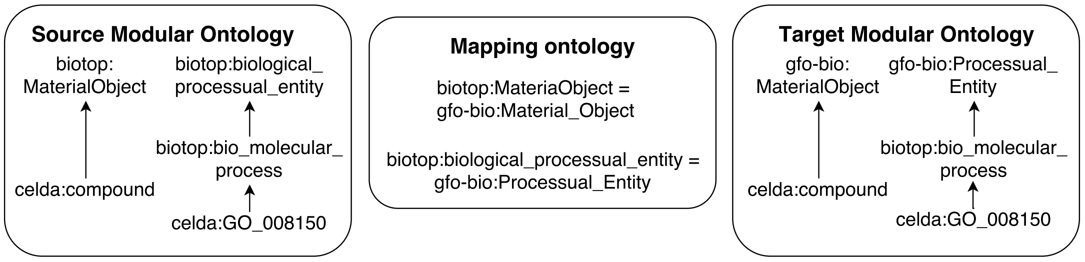

Source Modular Ontology
biotop: MaterialObject
biotop:biological_ processual_entity
biotop:bio_molecular_ process
celda:compound
celda:GO_008150
Mapping ontology
biotop:MateriaObject = gfo-bio:Material_Object
biotop:biological_processual_entity = gfo-bio:Processual_Entity
Target Modular Ontology
gfo-bio: MaterialObject
gfo-bio:Processual_ Entity
biotop:bio_molecular_ process
celda:compound
celda:GO_008150

Figure 3: Examples of interchanging the celda:compound and celda:GO_008150 domain entities from ${}^s\mathcal{M}_i$ BioTop to ${}^t\mathcal{M}_i$ GFO-Bio with SUGOI-Gen, using equivalence and subsumption mappings.

(4) Change the 'new' axioms to reference ${}^t\mathcal{M}_i$ entities, if mappings exist: for axiom1, there is an equivalence mapping between `biotop:MaterialObject` and `gfo-bio:Material_object`, hence we change axiom1 celda:compound $\sqsubseteq$ biotop:Material Obj- ect to celda:compound $\sqsubseteq$ gfo-bio:Material_ Object. For axiom2, there is no equivalence mapping between biotop:bio_ molecular_process and GFO entities; we skip this step.
(5) If a mapping does not exist, perform on-the-fly subsumption: For axiom2, biotop:bio_molecular_process has a superclass biotop:biological_processual_entity and the mapping ontology has biotop:biological_processual_entity $\equiv$ gfo-bio:Biological_process, so biotop:bio_molecular_process $\sqsubseteq$ gfo-bio:Biological_ process is added to `CELDA-gfo-bio.owl`.
(6) Delete entities that exist in the ${}^t\mathcal{M}$ that are from the ${}^s\mathcal{M}_i$ but do not appear in an axiom with entities from the ${}^t\mathcal{M}_d$, resulting in the final ${}^t\mathcal{M}$, `dmop-gfo.owl`. Delete the biotop:Material Object entity from `CELDA-gfo-bio.owl`.

3 MODULE INTERCHANGEABILITY DESIGN AND IMPLEMENTATION

This section introduces the algorithms and the implementation with user interface.

3.1 SUGOI-Gen Algorithm

The paper that introduced SUGOI [12] describes only the main steps of the algorithm for foundational ontology interchangeability, but not the algorithm itself, and works with three, fixed, foundational ontologies only. Here, we provide a full algorithm, which also has been modified to allow the user to provide the ${}^s\mathcal{M}_i$, ${}^t\mathcal{M}_i$, and ${}^a O$ to perform general module interchangeability. Algorithms 1-2 show the precise procedure for changing a ${}^s\mathcal{M}_i$ to a ${}^t\mathcal{M}_i$ resulting in a new ${}^t\mathcal{M}$. The algorithm is illustrated with practical examples in the following section.

3.2 Generalising the SUGOI interchangeability tool

To the best of our knowledge, the only existing tool capable of 'swapping' `.owl` files is SUGOI [12]. However, SUGOI is restricted to swapping three foundational ontologies only: DOLCE, BFO, and GFO, and it thus cannot be used for either interchanging other foundational ontologies or any other types of aligned modules in general, i.e., it cannot satisfy three of the four scenarios. To solve

Algorithm 1: SUGOI ontology interchangeability algorithm

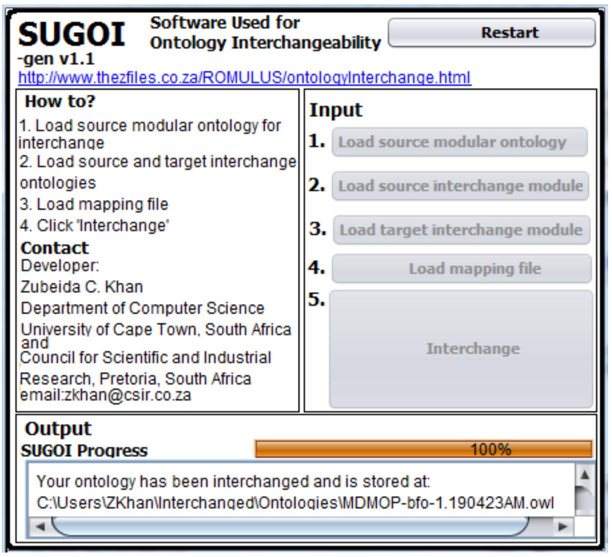
```
input  : ${}^s\mathcal{M}$, ${}^s\mathcal{M}_i$, ${}^t\mathcal{M}_i$, ${}^a O$
output : ${}^t\mathcal{M}$
/* Steps 1-2. Create an ontology ${}^t\mathcal{M}$. Copy axioms
   from the ${}^t\mathcal{M}_i$ to the ${}^t\mathcal{M}$                    */
${}^t\mathcal{M} \leftarrow {}^t\mathcal{M}_i$;
/* Step 3. Copy domain axioms to the ${}^t\mathcal{M}$              */
foreach entity in ${}^s\mathcal{M}$ do
    if entity not in ${}^s\mathcal{M}_i$ then
        currentAxiom ← get current axiom;
        add currentAxiom to ${}^t\mathcal{M}$;
    end
end
/* Step 4. Map domain entities to the ${}^t\mathcal{M}_i$           */
foreach entity in ${}^t\mathcal{M}$ do
    /* if entity is a domain entity                    */
    if entity not in sourceFoundationalOntology and entity
      not in ${}^t\mathcal{M}_i$ then
        currentAxiom ← get current axiom;
        entitySet ← get entities in signature of currentAxiom;
        foreach signatureEntity in entitySet do
            if signatureEntity in ${}^a O$ then
                eSignatureEntity ← get equivalent entity of
                  signatureEntity;
                currentAxiom ← replace signatureEntity
                  with eSignatureEntity in currentAxiom;
            end
        end
    end
end
```

this, we have implemented the algorithms presented in the previous section and modified also the GUI so as to perform interchangeability with any ontology, including allowing the user to upload their own interchange module and mapping files.

Figure 4 is a screenshot of the online desktop version of SUGOI-Gen. The jar file of the online desktop version, and a screencast is available at the aforementioned URL.

Algorithm 2: SUGOI ontology interchangeability algorithm-part 2

```
/* Step 5. Perform on-the-fly subsumption, if a
   domain entity is not linked to a $^t\mathcal{M}_i$ */
foreach entity in $^t\mathcal{M}$ do
    if entity not in $^t\mathcal{M}_i$ then
        if entity has no superclasses in $^t\mathcal{M}$ then
            ancestorSet ← get ancestor entities of entity from
              sourceFoundationalOntology;
            mappableSet ← empty set;
            foreach ancestorEntity in ancestorSet do
                if ancestorEntity exists in mapOntology then
                    add ancestorEntity to mappableSet;
                end
            end
            /* get lowest level entity */
            selectedEntity ← get lowest level entity from
              mappableSet;
            mappedSelectedEntity ← get entity equivalent to
              selectedEntity from mapOntology;
            newAxiom ← create axiom stating that entity is a
              subclass of mappedSelectedEntity;
            add newAxiom to $^t\mathcal{M}$;
        end
    end
end
/* Step 6. Delete source foundational ontology
   entities that are not referenced by the domain
   entities */
foreach entity in $^t\mathcal{M}$ do
    if entity in $^s\mathcal{M}_i$ then
        entitySet ← get referencing entities of entity;
        foreach referencedEntity in entitySet do
            /* if referencedEntity is a domain entity
               */
            if referencedEntity not in $^s\mathcal{M}_i$ and
              referencedEntity not in $^t\mathcal{M}_i$ then
                checker ← true;
            end
        end
        /* if entity is not referenced by any domain
           entities */
        if checker == false then
            if entity in $^t\mathcal{M}_i$ then
                remove entity from $^t\mathcal{M}$;
            end
        end
    end
end
save $^t\mathcal{M}$;
generate log file with metrics;
```


SUGOI Software Used for Ontology Interchangeability -gen v1.1

Restart

http://www.thezfiles.co.za/ROMULUS/ontologyInterchange.html

How to?
1. Load source modular ontology for interchange
2. Load source and target interchange ontologies
3. Load mapping file
4. Click 'Interchange'

Contact
Developer:
Zubeida C. Khan
Department of Computer Science
University of Cape Town, South Africa
and
Council for Scientific and Industrial Research, Pretoria, South Africa
email:zkhan@csir.co.za

Input
1. Load source modular ontology
2. Load source interchange module
3. Load target interchange module
4. Load mapping file
5. Interchange

Output
SUGOI Progress 100%

Your ontology has been interchanged and is stored at:
C:\Users\ZKhan\Interchanged\Ontologies\MDMOP-bfo-1.190423AM.owl

Figure 4: The interface of SUGOI-Gen.

4 EXPERIMENTAL EVALUATION

The first purpose of the experimental evaluation is to investigate the interchangeability of modules for the identified scenarios to determine how well the algorithm for interchangeability performs. For this, we assess whether SUGOI-Gen can successfully interchange a $^s\mathcal{M}$ to a $^t\mathcal{M}$ and determine the amount of the ontology that will be effectively interchanged, which refers to those entities within the $^t\mathcal{M}_i$ that have been mapped with equivalence relations, thereby not required to use parts of the $^s\mathcal{M}_i$ in the $^t\mathcal{M}$. Second, we assess the source and target modules to determine the effects that interchangeability may have on a modular ontology in terms of the module's metrics and reasoning processing.

4.1 Materials and methods

The assessment has been designed using three of the four scenarios for module interchangeability: swapping an ontology for a leaner version, swapping foundational ontologies, and swapping top-domain ontologies.

(1) Collect module sets from existing works for each of the three scenarios.
(2) Create a mapping ontology file for each module set.
(3) Interchange the *Source Modular Ontology* ($^s\mathcal{M}$) to a *Target Modular Ontology* ($^t\mathcal{M}$).
(4) Analyse the *raw interchangeability* of each $^t\mathcal{M}$, i.e., a measure to determine the amount of the $^t\mathcal{M}$ that has been correctly interchanged using equivalence mappings thereby not referring to the $^s\mathcal{M}_i$ entities. This measure was introduced in early experiments with SUGOI [12] and is calculated from the $^t\mathcal{M}$ as follows: Let *GT, good target linking axioms*, represent the sum of axioms that link domain ontology entities and $^t\mathcal{M}_i$ entities in the $^t\mathcal{M}$. Let *BT, bad target linking axioms*, represent the sum of axioms that link domain ontology entities and $^s\mathcal{M}_i$ entities in the

${}^{t}\mathcal{M}$; the raw interchangeability is calculated as follows:

$$Raw\ interchangeability = \frac{|GT|}{|GT\ +\ BT|} \times 100 \tag{1}$$

(5) Compare the time taken for reasoning for the ${}^{s}\mathcal{M}$ against the ${}^{t}\mathcal{M}$.
(6) Run the TOMM metrics tool for the ${}^{s}\mathcal{M}$ and the ${}^{t}\mathcal{M}$ module sets.
(7) Analyse and compare the metrics for the ${}^{s}\mathcal{M}$ and ${}^{t}\mathcal{M}$ module sets.

The materials consist of six OWL ${}^{s}\mathcal{M}$, two from each scenarios, covering domains about animals, space and time, anatomy, cells, etc., mapping files created for each ${}^{s}\mathcal{M}_i$ to ${}^{t}\mathcal{M}_i$ alignment created with Logmap [10] and manually, the SUGOI-Gen interchangeability tool, and TOMM metrics tool [15] as it generates evaluation metrics to measure the quality of a module. The mapping files that were used for the foundational ontology modules were manually created for previous experiments [12], and for the rest of the modules we used Logmap to create them. A mapping file for the alignment between BioTop and GFO-Bio is shown in Figure 5. All the test files used for this experimental evaluation can be downloaded from http://www.thezfiles.co.za/ROMULUS/ontologyInterchange.html.

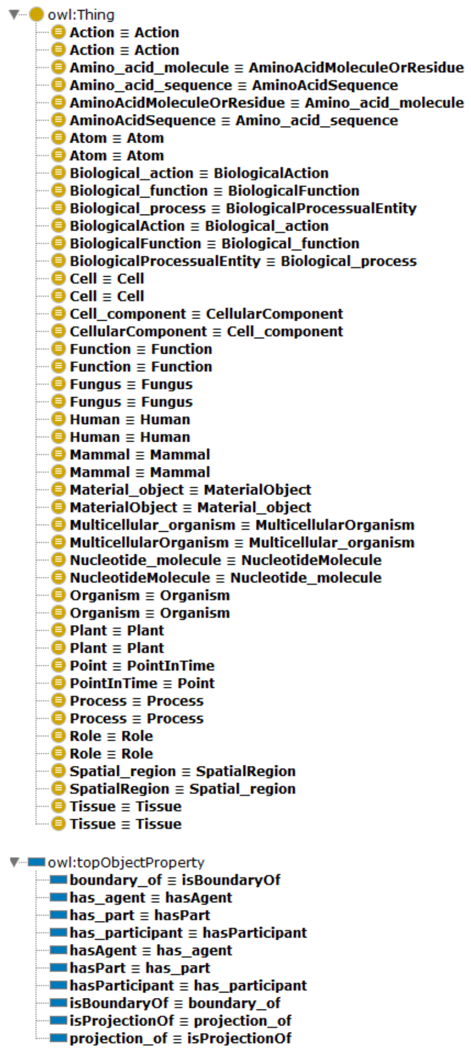
Multiple metrics have been proposed for ontology modularisation, with the most recent, and longest, list in [15]. For module interchange, not all ontology metrics are relevant, however; e.g., completeness, coupling, and redundancy are relevant for the modularisation *process* rather than being a property of the module *itself*. This reduces the relevant metrics to the following ones:

Size: Size refers to the number of entities in a module.
Atomic size: The average size of a group of inter-dependent axioms in a module.
Intra-module distance: The sum of number of relations in the shortest path from one entity to the other, for every entity in the module.
Cohesion: The extent to which entities in a module are related to each other.
Attribute richness: The average number of attributes per class in a module.
Inheritance richness: The average number of subclasses per class in a module.

For the complete definition for each metric, together with its equation, refer to [15].

4.2 Results

All of the modules were successfully interchanged with SUGOI-Gen. To determine how the effective swapping the various modules were, we first look at the interchangeability values that were calculated with the SUGOI-Gen tool, displayed in Table 1. For the first set of modules, for interchanging foundational ontologies, the raw interchangeability values were between 25-28%; about a quarter of the ${}^{s}\mathcal{M}$ was successfully interchanged completely. This is because there are only 15 mappings that exist between DOLCE and GFO, and 7 between DOLCE and BFO, in the respective mapping files [12]. The remaining 75% of the foundational ontology modules that were not successfully interchanged are represented by the *BT* values in the raw interchangeability formula. These are the


```
owl:Thing
  Action ≡ Action
  Action ≡ Action
  Amino_acid_molecule ≡ AminoAcidMoleculeOrResidue
  Amino_acid_sequence ≡ AminoAcidSequence
  AminoAcidMoleculeOrResidue ≡ Amino_acid_molecule
  AminoAcidSequence ≡ Amino_acid_sequence
  Atom ≡ Atom
  Atom ≡ Atom
  Biological_action ≡ BiologicalAction
  Biological_function ≡ BiologicalFunction
  Biological_process ≡ BiologicalProcessualEntity
  BiologicalAction ≡ Biological_action
  BiologicalFunction ≡ Biological_function
  BiologicalProcessualEntity ≡ Biological_process
  Cell ≡ Cell
  Cell ≡ Cell
  Cell_component ≡ CellularComponent
  CellularComponent ≡ Cell_component
  Function ≡ Function
  Function ≡ Function
  Fungus ≡ Fungus
  Fungus ≡ Fungus
  Human ≡ Human
  Human ≡ Human
  Mammal ≡ Mammal
  Mammal ≡ Mammal
  Material_object ≡ MaterialObject
  MaterialObject ≡ Material_object
  Multicellular_organism ≡ MulticellularOrganism
  MulticellularOrganism ≡ Multicellular_organism
  Nucleotide_molecule ≡ NucleotideMolecule
  NucleotideMolecule ≡ Nucleotide_molecule
  Organism ≡ Organism
  Organism ≡ Organism
  Plant ≡ Plant
  Plant ≡ Plant
  Point ≡ PointInTime
  PointInTime ≡ Point
  Process ≡ Process
  Process ≡ Process
  Role ≡ Role
  Role ≡ Role
  Spatial_region ≡ SpatialRegion
  SpatialRegion ≡ Spatial_region
  Tissue ≡ Tissue
  Tissue ≡ Tissue

owl:topObjectProperty
  boundary_of ≡ isBoundaryOf
  has_agent ≡ hasAgent
  has_part ≡ hasPart
  has_participant ≡ hasParticipant
  hasAgent ≡ has_agent
  hasPart ≡ has_part
  hasParticipant ≡ has_participant
  isBoundaryOf ≡ boundary_of
  isProjectionOf ≡ projection_of
  projection_of ≡ isProjectionOf
```

Figure 5: The mapping file showing alignments for classes and object properties between BioTop and GFO-Bio.

axioms that link domain ontology entities to ${}^{s}\mathcal{M}_i$ entities in the ${}^{t}\mathcal{M}$. For instance, the axiom ontoderm:DermDiseaseType ⊑ dolce:abstract-quality containing an entity from the DOLCE ${}^{s}\mathcal{M}$ exists in the ontoderm-bfo module.

For the next set of modules, for interchanging a module for a subset module, both ontologies had a raw interchangeability of 100%; the ontologies were able to be completely interchanged because there were mappings between the ${}^{s}\mathcal{M}_i$ and ${}^{t}\mathcal{M}_i$ for all entities. DOLCE and DOLCE-EL have the same entities, and BFO and BFO-Continuant also have the same entities so they were all mapped. Lastly, for the set of modules for interchanging top-domain modules, they had a raw interchangeability of between 22-24%. There were 28 mappings available between the top-domain ontologies, BioTop and GFO-Bio but the ${}^{s}\mathcal{M}_d$ to ${}^{s}\mathcal{M}_i$ links were high, 311 and 399 for the umlssn and dco top-domain ontologies, causing the

Table 1: Comparison of the $^s\mathcal{M}$ and $^t\mathcal{M}$ and the raw interchangeability; inter. = interchangeability, R-time (s). = reasoning time in seconds.

Source and target ontologies	$^s\mathcal{M}_d$ to $^s\mathcal{M}_i$ links	Domain entities	Raw inter.	R-time (s)
Interchange a foundational ontology for another				
naive_animal-dolce	43	438		75.0
naive_animal-gfo		452	25.58%	146.0
ontoderm-dolce	14	301		19.0
ontoderm-bfo		308	28.57%	0.5
Interchange a module for a subset module				
sceneOntology-dolce	18	246		0.9
sceneOntology-dolce -el		246	100.00%	0.5
sao-bfo	66	809		0.7
sao-bfo-continuant		809	100.00%	0.4
Interchange a top-domain ontology for another				
umlssn-biotop	311	714		300.0
umlssn-gfo-bio		863	22.18%	0.3
dco-biotop	399	1446		1832.0
dco-gfo-bio		1650	24.06%	0.2

lower raw interchangeability values. Thus, for all ontologies in the set, some interchangeability can be achieved using SUGOI-Gen.

Comparing the number of domain entities in the $^s\mathcal{M}$ and $^t\mathcal{M}$, we note that there are extra domain entities in the $^t\mathcal{M}$. This is because a number of $^s\mathcal{M}_i$ entities have been added to the $^t\mathcal{M}$ when on-the-fly subsumption occurs (recall the biotop:bio_molecular_process entity that is added to the CELDA $^t\mathcal{M}$ in Figure 3). Next, we analyse the reasoning for the modules. All of the interchanged modules except the naive_animal have an improved time taken for reasoning after interchange. In some cases, the improvement is considerable such as the case of umlssn-biotop where the interchanged ontology has a reasoning time of just 0.3 seconds compared to the source ontology that has a reasoning time of 5 minutes. Another case is the dco interchanged ontology with a reasoning time of 0.2 seconds compared to the source ontology that has a reasoning time of 30.5 minutes.

We now inspect the ontologies to assess whether interchangeability has an effect on the modularity metrics by comparing the $^s\mathcal{M}$ metrics to the $^t\mathcal{M}$ metrics. The metrics are presented in Table 2.

For modularity, since the definition that we use from the literature states that 'A *Module M* is a subset of a source ontology O, $M \subset O$, either by abstraction, removal or decomposition...', smaller metrics indicate a favourable module. When interchanging a foundational ontology for another foundational ontology, as in the naive_animal and ontoderm ontologies, this could cause an increase or decrease in the metrics for the module. For the naive_animal ontology, all the numerical metrics had increased except the atomic size and the attribute richness. Thus the size of the module and other metrics such as the distance between entities had grown which is not favourable for modularity. This is because the GFO foundational ontology is larger than the DOLCE foundational ontology. For the ontoderm ontology, when interchanging from DOLCE to BFO, conversely most of the numerical metrics had decreased, except the intra-module distance and cohesion; thus the metrics had improved for modularity.

For the next use case, interchanging a module for a subset module, since a subset module is smaller, it should result in improved metrics for modularity. For the SceneOntology, after the interchange, the atomic size, number of axioms, attribute richness, and inheritance richness has decreased. However, the intra-module distance has increased, meaning that the entities moved further apart due to the removal of some of the relations between classes. A larger intra-module distance might promote tool processing since the ontology module would have less relations, be less-expressive and have a smaller number of axioms while a smaller intra-module distance means that the classes are closer together and easier to traverse and this might promote human understanding. For the next use-case, interchanging a top-domain ontology for another top-domain ontology, there is a decrease in all the metrics except the inheritance richness value meaning that the target ontology has a higher number of subclasses per class than the source ontology. Hence interchanging top-domain modules for the set in question is indeed favourable.

5 DISCUSSION

Thanks to SUGOI-Gen and the experimental evaluation using a set of modules, we now know that interchangeability with modules is indeed possible. The success of the interchangeability depends on the number of mappings that are available between the source and target modules. For the set of modules used in our experiment, the success of interchangeability ranged from 22% to a 100%. The foundational ontology and top-domain ontology modules had a lower raw interchangeability because only some of the entities could be mapped. For the subset modules, all their entities were mappable resulting in the 100% raw interchangeability.

An ontology developer can also gain insight on how a certain foundational, top-domain, or leaner modules could impact the metrics of a module, and which module might be better to use, depending on the developer's desired metrics. For instance, a developer may prefer to have a module with a larger intra-module distance such as the case with the sceneOntology-dolce-el module to promote tool processing at the expense of human comprehension, or vice versa. SUGOI-Gen could also be used to swap modules within a system for other ones to improve processing times for reasoning tools. For the set of modules in our experiment, all the modules that were interchanged except one had an improved time for reasoning. Comparing the original and interchanged ontologies to check which OWL constructs are used to represent the same domain knowledge is an interesting avenue of research for future work to determine the trade-off between expressive power and reasoning between various foundational, top-domain, ontology design pattern, or arbitrary modules.

We now return to the questions regarding interchangeability posed in the introduction. Regarding question 1: it is feasible to interchange a module within an ontology for another module. However, the success of the interchangeability depends on the number of mappings that are available between the source and target modules. Regarding question 2: interchangeability does have an impact

Table 2: The metrics for the ${}^{s}\mathcal{M}$ and ${}^{t}\mathcal{M}$ ontologies.

Source and target ontologies	Size	Atomic size	Number of axioms	Intra-module distance	Cohesion	Attribute richness	Inheritance richness
Interchange a foundational ontology for another							
naive_animal-dolce	545	6.82	3085	186 156	0.017	2.62	3.10
naive_animal-gfo	598	6.26	3104	292 913	0.02	2.37	3.19
ontoderm-dolce	408	5.14	1470	313 977	0.04	0.76	3.16
ontoderm-bfo	347	4.42	1217	345 117	0.06	0.51	3.31
Interchange a module for a subset module							
sceneOntology-dolce	353	5.57	1298	150 432	0.04	1.09	4.53
sceneOntology-dolce-el	353	5.13	1221	151 118	0.04	0.89	4.34
sao-bfo	848	7.82	8037	2 108 127	0.04	0.45	2.90
sao-bfo-continuant	830	7.86	7931	1 990 462	0.04	0.46	2.90
Interchange a top-domain ontology for another							
umlssn-biotop	1119	7.56	5795	4 664 881	0.04	2.97	4.00
umlssn-gfo-bio	1105	7.10	5360	3 262 657	0.04	2.57	4.39
dco-biotop	1851	4.52	8925	297 579	0.005	1.42	3.21
dco-gfo-bio	1802	4.11	8470	383 592	0.006	1.09	3.13

on the metrics, depending on whether the module that has been interchanged is smaller or larger (DOLCE vs. GFO), more-expressive or less- expressive (DOLCE vs. DOLCE-EL), etc., this could impact certain module metrics, and the ontology developer may consider interchangeability to assess which module is the best fit for an application. Lastly regarding question 3: the interchangeability has a significant impact on the time taken for reasoning. A comparison of the reasoning times for the source and target modules reveals that, for the set of modules used in this experiment, the reasoning times were greatly improved.

6 CONCLUSION

In this paper we investigated ontology interchangeability on modules and the impact on their metrics. We presented the design and implementation of of SUGOI-Gen, which can be used to interchange a module for another module within a system, covering the four principal scenarios of module interchange: leaner versions of an ontology, foundational and top-domain ontologies, and the outline of ontology pattern swapping, provided that the user uploads the source and mapping files. An experimental evaluation of three different use-cases using six ontologies reveal that some interchangeability is possible, and that the success of interchangeability depends on the mappings that are available. The investigation revealed that an ontology developer can gain insight on how a certain foundational, top-domain, or subset modules could have an effect on the metrics, and the processing times for reasoning for modules.

For future work, we intend to evaluate SUGOI-Gen with ontology developers to determine whether interchangeability can assist also with other ontology development problems, such as ontology comprehension. It may also be interesting to investigate further the trade-off between expressive power and language features usage and automated reasoning performance when interchanging the various foundational, top-domain, ontology design pattern, or arbitrary modules in a modularised ontologies and ontology networks.

REFERENCES

[1] Elena Beisswanger, Stefan Schulz, Holger Stenzhorn, and Udo Hahn. 2008. BioTop: An upper domain ontology for the life sciences – A description of its current structure, contents and interfaces to OBO ontologies. *Applied Ontology* 3, 4 (2008), 205–212.

[2] Stefano Borgo. 2011. Goals of Modularity: A Voice from the Foundational Viewpoint. In *The Fifth International Workshop on Modular Ontologies (WOMO'2011) (Frontiers in Artificial Intelligence and Applications)*, Oliver Kutz and Thomas Schneider (Eds.), Vol. 230. IOS Press, 1–6. Ljubljana, Slovenia, August.

[3] Lindsay Grey Cowell and Barry Smith. 2010. *Infectious Disease Ontology*. Springer New York, 373–395.

[4] Mathieu d'Aquin, Anne Schlicht, Heiner Stuckenschmidt, and Marta Sabou. 2009. Criteria and Evaluation for Ontology Modularization Techniques. In *Modular Ontologies: Concepts, Theories and Techniques for Knowledge Modularization*. Lecture Notes in Computer Science, Vol. 5445. Springer, 67–89.

[5] Anusuriya Devaraju, Werner Kuhn, and Chris S. Renschler. 2015. A formal model to infer geographic events from sensor observations. *International Journal of Geographical Information Science* 29, 1 (2015), 1–27. DOI: http://dx.doi.org/10.1080/13658816.2014.933480

[6] Pablo R. Fillottrani and C. Maria Keet. 2017. Patterns for Heterogeneous TBox Mappings to Bridge Different Modelling Decisions. In *Proc. of ESWC'17 (LNCS)*, E. Blomqvist and others (Eds.), Vol. 10249. Springer, 371–386. 30 May - 1 June 2017, Portoroz, Slovenia.

[7] Heinrich Herre. 2010. General Formal Ontology (GFO): A Foundational Ontology for Conceptual Modelling. In *Theory and Applications of Ontology: Computer Applications*. Springer, Heidelberg, Chapter 14, 297–345.

[8] Robert Hoehndorf, Frank Loebe, Roberto Poli, Heinrich Herre, and Janet Kelso. 2008. GFO-Bio: A biological core ontology. *Applied Ontology* 3, 4 (2008), 219–227.

[9] Krzysztof Janowicz and Michael Compton. 2010. The Stimulus-Sensor-Observation Ontology Design Pattern and its Integration into the Semantic Sensor Network Ontology. In *3rd International Workshop on Semantic Sensor Networks (CEUR Workshop Proceedings)*, Vol. 668. CEUR-WS.org, 64–78. November 7, Shanghai, China.

[10] Ernesto Jiménez-Ruiz, Bernardo Cuenca Grau, and Yujiao Zhou. 2011. LogMap 2.0: towards logic-based, scalable and interactive ontology matching. In *The 4th International Workshop on Semantic Web Applications and Tools for the Life Sciences, SWAT4LS 2011*. ACM, 45–46. London, United Kingdom, December 07-09, 2011.

[11] C. Maria Keet, Claudia d'Amato, Zubeida Casmod Khan, and Agnieszka Lawrynowicz. 2014. Exploring Reasoning with the DMOP Ontology. In *3rd Workshop on Ontology Reasoner Evaluation (ORE'14) (CEUR Workshop Proceedings)*. CEUR-WS.org, 64–70. July 1, Vienna, Austria.

[12] Zubeida Khan and C. Maria Keet. 2014. Feasibility of Automated Foundational Ontology Interchangeability. In *19th International Conference on Knowledge Engineering and Knowledge Management (EKAW'14) (LNAI)*, Vol. 8876. Springer, 225–237. 24 - 28 November 2014, Linköping, Sweden.

[13] Zubeida Casmod Khan. 2016. Evaluation Metrics in Ontology Modules. In *29th International Workshop on Description Logics (DL'16) (CEUR Workshop Proceedings)*, Vol. 1577. CEUR-WS.org. 22-25 April 2016, Cape Town, South Africa.

[14] Zubeida Casmod Khan and C. Maria Keet. 2015. An empirically-based framework for ontology modularisation. *Applied Ontology* 10, 3-4 (2015), 171–195.

[15] Zubeida Casmod Khan and C. Maria Keet. 2016. Dependencies between modularity metrics towards improved modules. In *20th International Conference on Knowledge Engineering and Knowledge Management (EKAW'16) (Lecture Notes in Artificial Intelligence LNAI)*. Springer, 19–23. 19-23 November 2016, Bologna, Italy.

[16] Zubeida Casmod Khan and C. Maria Keet. 2016. ROMULUS: The Repository of Ontologies for MULtiple USes Populated with Mediated Foundational Ontologies. *Journal of Data Semantics* 5, 1 (2016), 19–36.

[17] Stephen D. Larson, Lisa Fong, Amarnath Gupta, Christopher Condit, William J. Bug, and Maryann E. Martone. 2007. A formal ontology of subcellular neuroanatomy. *Frontiers in Neuroinformatics* 2007 (2007), 1–12. DOI: http://dx.doi.org/10.3389/neuro.11.003.2007

[18] C. Masolo, S. Borgo, A. Gangemi, N. Guarino, and A. Oltramari. 2003. Ontology Library. WonderWeb Deliverable D18 (ver. 1.0, 31-12-2003).. (2003). http://wonderweb.semanticweb.org.

[19] R. Mizoguchi. 2010. YAMATO: Yet Another More Advanced Top-level Ontology. In *Proceedings of the Sixth Australasian Ontology Workshop (Conferences in Research and Practice in Information)*. CRPIT, 1–16. Sydney : ACS.

[20] David Newman, Sean Bechhofer, and David De Roure. 2009. myExperiment: An ontology for e-Research. In *Proceedings of the Workshop on Semantic Web Applications in Scientific Discourse (SWASD 2009) (CEUR Workshop Proceedings)*, Vol. 523. CEUR-WS.org. Washington DC, USA, October 26.

[21] I. Niles and A. Pease. 2001. Towards a Standard Upper Ontology. In *Proceedings of the 2nd International Conference on Formal Ontology in Information Systems (FOIS-2001)*, Chris Welty and Barry Smith (Eds.). ACM, 2–9. Ogunquit, Maine, October 17-19, 2001.

[22] Sunju Oh, Heon Young Yeom, and Joongho Ahn. 2011. Cohesion and coupling metrics for ontology modules. *Information Technology and Management* 12, 2 (2011), 81–96.

[23] Pance Panov, Saso Dzeroski, and Larisa N. Soldatova. 2008. OntoDM: An Ontology of Data Mining. In *Workshops Proceedings of the 8th IEEE International Conference on Data Mining (ICDM 2008)*. IEEE Computer Society, 752–760. December 15-19, Pisa, Italy.

[24] Christine Parent and Stefano Spaccapietra. 2009. An Overview of Modularity. In *Modular Ontologies*, Heiner Stuckenschmidt, Christine Parent, and Stefano Spaccapietra (Eds.). Lecture Notes in Computer Science, Vol. 5445. Springer Berlin Heidelberg, 5–23.

[25] Ana Armas Romero, Mark Kaminski, Bernardo Cuenca Grau, and Ian Horrocks. 2016. Module Extraction in Expressive Ontology Languages via Datalog Reasoning. *Journal of Artificial Intelligence Research (JAIR)* 55 (2016), 499–564.

[26] Ansgar Scherp, Carsten Saathoff, Thomas Franz, and Steffen Staab. 2011. Designing core ontologies. *Applied Ontology* 6, 3 (2011), 177–221. DOI: http://dx.doi.org/10.3233/AO-2011-0096

[27] Anne Schlicht and Heiner Stuckenschmidt. 2006. Towards Structural Criteria for Ontology Modularization. In *Proceedings of the 1st International Workshop on Modular Ontologies, (WoMO'06) (CEUR Workshop Proceedings)*, Vol. 232. CEUR-WS.org, 85–97. ISWC'06 November 5, 2006, Athens, Georgia, USA.

[28] Stefanie Seltmann, Harald Stachelscheid, Alexander Damaschun, Ludger Jansen, Fritz Lekschas, Jean-Fred Fontaine, Throng-Nghia Nguyen-Dobinsky, Ulf Leser, and Andreas Kurtz. 2013. CELDA - an ontology for the comprehensive representation of cells in complex systems. *BMC Bioinformatics* 14 (2013), 228. DOI: http://dx.doi.org/10.1186/1471-2105-14-228

[29] Samir Tartir, I Budak Arpinar, Michael Moore, Amit P Sheth, and Boanerges Aleman-Meza. 2005. OntoQA: Metric-based ontology quality analysis. *IEEE Workshop on Knowledge Acquisition from Distributed, Autonomous, Semantically Heterogeneous Data and Knowledge Sources* 9 (2005), 45–53.

[30] Chiara Del Vescovo. 2011. The Modular Structure of an Ontology: Atomic Decomposition towards Applications. In *Proceedings of the 24th International Workshop on Description Logics (DL 2011) (CEUR Workshop Proceedings)*, Vol. 745. CEUR-WS.org. Barcelona, Spain, July 13-16.

[31] Haining Yao, Anthony M Orme, and Letha Etzkorn. 2005. Cohesion metrics for ontology design and application. *Journal of Computer science* 1, 1 (2005), 107.