Applied Ontology 10 (2015) 1–3
IOS Press

An empirically-based framework for ontology modularization

Zubeida Casmod Khan [a,b,*] and C. Maria Keet [a]
[a] *Department of Computer Science, University of Cape Town, South Africa*
E-mail: mkeet@cs.uct.ac.za
[b] *Council for Scientific and Industrial Research, Pretoria, South Africa*
E-mail: zkhan@csir.co.za

Abstract. Modularity is being increasingly used as an approach to solve for the information overload problem in ontologies. It eases cognitive complexity for humans, and computational complexity for machines. The current literature for modularity focuses mainly on techniques, tools, and on evaluation metrics. However, ontology developers still face difficulty in selecting the correct technique for specific applications and the current tools for modularity are not sufficient. These issues stem from a lack of theory about the modularisation process. To solve this problem, several researchers propose a framework for modularity, but alas, this has not been realised, up until now. In this article, we survey the existing literature to identify and populate dimensions of modules, experimentally evaluate and characterise 189 existing modules, and create a framework for modularity based on these results. The framework guides the ontology developer throughout the modularisation process. We evaluate the framework with a use-case for the Symptom ontology.

Keywords: ontology, module, modularisation, modularity, partitioning, module extraction

1. Introduction

Modularity in ontology engineering is applied as a solution for dealing with information overload for both machines and humans, as it eases computation tasks and simplifies the understanding and interpretation of knowledge by offering smaller subsets of an ontology. In the early years, even manual reorganisation of an ontology's content ("normalisation") was proposed to make extracting modules less hard (Rector (2003)), but much has been achieved since. A rapid increase in the use of modularity for dealing with large ontologies has resulted in an abundance of approaches and tools in the field. Notable advances in the field include automated tools, such as SWOOP (Kalyanpur *et al.* (2006)) and OWL module extractor (Cuenca Grau *et al.* (2008)) and evaluation metrics to assess the quality of modules (d'Aquin *et al.* (2007, 2009); Pathak *et al.* (2009); Tartir *et al.* (2005)). While there is some research that provides information about some modularity criteria (d'Aquin *et al.* (2009); Schlicht and Stuckenschmidt (2006)), there is a lack of a foundational theory for ontology modularity; e.g., it is unclear which evaluation metrics are to be considered for different module types and what type of modules different techniques produce. d'Aquin *et al.* (2009) found that the evaluation of a modularisation depends on an application's requirements, that there is no universal modularisation, and that a formal well-defined framework for modularity is lacking. This opens up a number of issues and questions; e.g., difficulty in selecting the appropriate modularity technique, insufficient modularity tools for applications, and it is unclear which one should be applied for which scenario. For instance, we tried to modularise the Data Mining OPtimization (DMOP) ontology (Keet *et al.* (2015)) with several modularisation tools, but all modules were too large to use (Keet *et al.* (2014)), and extracting content on object properties from DOLCE with the 'copy' feature, their asserted characteristics such as transitivity were not extracted (Keet *et al.* (2013)). Also, existing techniques are not sufficient in creating compact modules (d'Aquin *et al.* (2009); Khan and Keet (2013)). Evaluation of modularisation techniques reveal that some tools fail to partition large ontologies because they focus on

*Corresponding author: Zubeida Casmod Khan, Department of Computer Science, University of Cape Town, South Africa, and Council for Scientific and Industrial Research, Pretoria, South Africa. zkhan@csir.co.za

preserving the logical properties of the modules while others lose some of the relational properties of the ontologies, and that most tools generate views instead of module file outputs (Oh *et al.* (2010); Oh and Yeom (2011)).

In praxis, it also remains largely unclear how to manage ontology modules once they are generated and start leading their own life, being merged with or imported into another ontology. For instance, one may have slimmed a highly axiomatised module into an OWL 2 EL fragment of it or extracted only two main branches of the class hierarchy and their relations: in the former case, it would not matter for one's project if the original ontology was augmented with axioms whose expressiveness is beyond OWL 2 EL, in the latter case, one may have to re-generate the module to reflect the changes made to the original one. Currently, there is no way of knowing this automatically, and modules are typically not even annotated with such type of information (unless they were created for certain experiments), let alone annotated in a structured way across modules.

These issues raise a plethora of questions not only from an engineering viewpoint to create tools, but also, still, from a conceptual and ontology engineering viewpoint. Some of the unanswered questions regarding modularity are:

Q1: What are the use-cases, techniques, types, and annotation features that exist for modules?
Q2: How do module types differ with respect to certain use-cases?
Q3: Which techniques can we use to create modules of a certain type?
Q4: Which techniques result in modules with certain annotation features?

Existing literature on modularity provides information about such aspects of modularity but an explicit and comprehensive list of the dimensions of modularity is lacking. In this paper, we wish to seek the solutions to the issues, and answers the questions, by 1) identifying the dimensions to be taken into consideration for ontology modularity, 2) populating the dimensions with criteria, which are surveyed from research on and application of modularity, 3) assess usage of modules in ontology engineering by surveying and analysing existing modules using the extracted dimensions, and 4) using the results of the two surveys, to create an ontology modularity framework by creating relations between the criteria that reveal dependencies, and hence suggestions on which dimensions fit well together and which do not.

The resultant framework of the module dimensions and dependencies can be used to steer the modularisation process, and form the basis for metadata for ontology modules, which promotes ontology reuse. In addition, the current state of ontology modularity with respect to tools has been refined, and reveals that tools are not sufficient nor maintained, resulting in that there is still a heavy reliance on manual methods.

The remainder of the paper is structured as follows. We describe the preliminaries and state of the art in Section 2. The ontology modules' dimensions are described in Section 3 and the experimental evaluation of module usage is described in Section 4. The modularity framework is described in Section 5. We discuss in Section 6 and conclude in Section 7.

2. Preliminaries

In order to identify and populate the dimensions of modularity, it is necessary to provide a clear definition for modularity and to summarise the state of the art with regard to advances in modularity.

2.1. What is modularity?

While existing literature does provide some definitions of modularity, no definition is universally accepted. It seems the case that the existing definitions are unique to the problem at hand. In order to define what an ontology module is, we consider first the main existing definitions.

Definition 1 (Parent and Spaccapietra (2009)): *In its most generic meaning, it denotes the possibility to perceive a large knowledge repository (be it an ontology or a database) as a set of modules, i.e. smaller repositories that, in some way, are parts of and compose the whole thing.*

Definition 2 (Doran *et al.* (2007)): *An ontology module is a reusable component of a larger or more complex ontology, which is self-contained but bears a definite association to other ontology modules, including the original ontology.*

Definition 3 (d'Aquin *et al.* (2007)): *We define an ontology module in a very general way as a part of an ontology: a module $\mathcal{M}_i(O)$ of an ontology O is a set of axioms, such that $\Sigma(\mathcal{M}_i(O)) \subseteq \Sigma(O)$.*

Definition 4 (Del Vescovo *et al.* (2013)): *Modules are suitably small subsets of an ontology O that behave for specific purposes like the original ontology over a given signature Σ, i.e., a set of terms (classes and properties).*

Definition 5 (Tsarkov (2012)): *A module is a subset of an ontology that captures all the knowledge the ontology contains about a given set of terms.*

In Definition 1, modules are parts and require a whole thing, but there need not be a 'whole' ontology; e.g., BioTopLite (Schulz and Boeker (2013)) and GFO-basic (Herre (2010)) are lighter versions of a larger ontology, without being part of a set of inter-related modules. Definitions 2, 3, and 4, refer to an original ontology, but not all modules have a source ontology; e.g., the myExperiment ontology (Newman *et al.* (2009)) is a decomposition of the domain into structural modules, hence there is no source ontology. Definition 5 is far too strict to hold for all types of modules that exist, and is focused on modules with the condition of logical completeness. The module definitions hold for their respective application areas but there are some gaps in the existing modularity definitions. We fill in such gaps with our own more comprehensive definition for modularity.

Definition 6 *Module: A module M is a subset of a source ontology O, $M \subset O$, 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.*

This definition aims to cover the gaps in Definitions 1-5: 1) It does not restrict modules to those that exist in a set and together compose a whole, 2) it allows that modules need not have a source ontology, and 3) it does not restrict modules to those that capture all the knowledge of an ontology over a given signature (locality modules). It also introduces modularity dimensions such as use-cases, techniques, annotation features, and evaluation criteria which have not been previously defined but are important because they guide the modularisation process.

2.2. State of the art

There are several studies on the properties and dimensions of ontology modules. Abbès *et al.* (2012) presents preliminary results on characterising modular ontologies based only on three structural criteria (size, cohesion, and coupling), leading to patterns based on ontology imports. Schlicht and Stuckenschmidt (2006) also created a set of structural criteria for ontology modules—connectedness, size, and redundancy of representation—which are said to affect efficiency, robustness,and maintainability for the application of semantics-based peer-to-peer systems. They evaluated SWOOP and PATO tools using those structural criteria and found that SWOOP favours modules with a good connectedness, i.e., reduction of communication costs, over modules with suitable size values, while with PATO, as the selected threshold value is increased, so is the size suitability of the module, while the connectedness value worsens.

Besides structural, there are also logic-based semantic notions of modularisation (Konev *et al.* (2008, 2009)), which have a main focus on module inseparability, meaning that the module and the source ontology are deemed to be inseparable if they give the same answers to any query. Loebe (2006) proposed a number of requirements for logical modules that could be used to guide the modularisation process, such

as logical correctness and completeness. However, the author acknowledges that the requirements do not hold for all applications and that specialised methods should be applied for different applications.

d'Aquin *et al.* (2009) proposed criteria and evaluation of modularity techniques and conducted an experimental evaluation using existing modularisation tools, which revealed that, at this stage, it is still unclear which evaluation techniques should be associated with which module use-case and properties. The modules that were generated with the tools were considerably different, and d'Aquin *et al.* (2009) concluded that modularity techniques are influenced by the properties of an ontology and other criteria, and that a well-defined benchmark is lacking but required for modularity.

Parent and Spaccapietra (2009) proposed the underlying assumptions and goals for modularity, e.g., maintenance anf scalability for reasoning, with the motivation that the way in which modularisation is approached depends on such goals. There is also a list of strategies that are proposed for creating modules, e.g., semantic-driven and structure-driven strategies. In a study on the foundational goals of modularity, Borgo (2011) classifies ontology modules as different types. There are several different types, e.g., isolating/developing branches of a taxonomy, collecting categories according to a domain, and isolating patterns.

3. Dimensions of ontology modularity

The state of the art section revealed that there is existing literature on the characteristics regarding modularity but most of them centre around evaluation metrics, with a strong focus on structural metrics. Other aspects concerning modularity such as rationale, methods, module types, and annotation features are lacking, at this stage. In this section, we fill in those gaps by introducing different types of modularity dimensions. The list of modularity dimensions was created after surveying, analysing, and structuring the current literature with respect to modules. We describe and populate each dimension with relevant subdimensions.

3.1. Use cases

Modularisation may be applied to an ontology for a number of goals, or purposes which include ontology maintenance, partial reuse, among others, mentioned in existing works (d'Aquin *et al.* (2007, 2009); Pathak *et al.* (2009); Schlicht and Stuckenschmidt (2008); Turlapati and Puligundla (2013)). We define these as use-cases for modularity placed into three orthogonal groups.

3.1.1. Ontology usage use-cases

Sometimes ontologies are difficult to use due to the fact that they are too large, or contain information that is irrelevant for the application at hand. For this, the use-cases for modularisation are centered around ontology usage.

U1: Maintenance Ontologies are constantly evolving resulting in a need for constant maintenance. Large ontologies cannot be easily maintained by one person. It is a task that is prone to error and omission. Ontology developers often face difficulty in building large ontologies due to sensemaking, searching and exploration of an ontology. Such problems are related to the loss of contextual awareness when traversing an ontology (Vigo *et al.* (2014)). Dividing an ontology into modules can assist with the maintenance of large and complex ontologies. In the case of updates in domain knowledge, not all the modules in a system need to be modified; the evolution could be localised within the relevant module(s). Maintenance also enables collaboration among a team, which is discussed as a use-case in a subsequent paragraph.

U7: Reuse For some applications, developers only require a small component of an ontology for reuse in another application. For instance, in the Subcellular Anatomy Ontology (SAO) ontology (Larson *et al.* (2007)), there are only whole entities, and no process entities. As such, the BFO-Continuant ontology of the ROMULUS repository (Khan and Keet (2013)) can be used rather than the entire BFO ontology.

3.1.2. Human factor use-cases

These use cases support human cognitive abilities by simplifying an ontology into modules.

U3: Validation Validation deals with checking an ontology for errors and whether it meets requirements. A single expert performing validation of a large ontology is not feasible, because identifying errors, such as inconsistency and redundancy, and guaranteeing that the ontology meets all the functional requirements in large ontologies is a difficult process (Vigo *et al.* (2014)). Having smaller modules would ease the burden of a large domain into smaller, simpler components for the expert.

U5: Comprehension It is confusing for humans to understand and use ontologies with thousands of terms. Keet (2007) proposes to use abstraction by removing some knowledge from the system to assist with comprehending the ontology. Since ontologies are sometimes designed and created by domain experts without expertise in logic, they use visual ontology engineering tools for development. Such tools aids with development but have drawbacks with large ontologies, thereby making it difficult to comprehend the complete ontology. Some approaches propose model exploration techniques to assist with comprehension whereby the concepts with corresponding relations of an ontology are visually generated in order to understand them (Bauer *et al.* (2009); Bergh *et al.* (2011)). For large ontologies, model exploration techniques could be problematic because of the challenge with ontology processing explained previously (U4).

Note that comprehension differs from validation. For validation, all components of the ontology are considered. However, for comprehension, simpler views omitting unnecessary components are considered.

U6: Collaborative efforts Collaborative efforts allows a team of developers to work together in creating an ontology. Modularisation enables the division of work tasks. In order to avoid conflict between different versions of the ontology by different developers, the ontology is divided into different components to allow specific people to create and modify modules without altering the entire system. For instance, the set of myExperiment ontology modules (Newman *et al.* (2009)) promotes collaboration among scientists for publishing workflows and experiment plans, in order to share them, and, on a grander scale, the OBO Foundry ontologies (Smith *et al.* (2007)).

3.1.3. Ontology tool use-cases

At times, the nature of ontologies, such as its size, or computational complexity of the language, can put a strain on ontology tools. In light of this, there are use-cases for modularity based on ontology tool performance.

U2: Reasoning Ontology reasoners perform poorly when reasoning complex and large ontologies of thousand of concepts (Vigo *et al.* (2014)); e.g., the data mining optimisation ontology (DMOP) has a classification time of approximately 10 minutes (Keet *et al.* (2014)). Reasoner performance decreases as the ontology size and number of axioms and rules increases. Consequently, reasoners will perform better with regard to efficiency if there is less knowledge to infer. In some cases, one will only be required to reason over modules that have been evolved since the last reasoning task.

U4: Processing Ontology related tools such as development, mediation, and metrics tools perform poorly when processing large ontologies (Antezana *et al.* (2009); Belloze *et al.* (2012); Paulheim (2008)). For instance, with the NCI cancer ontology by Golbeck *et al.* (2003), the BioPortal visualisation tool (Whetzel *et al.* (2011)) took several minutes to load the ontology, and using the OWL metrics tool[1] to compute its metrics took 12 minutes to process before it returned an ontology parsing error, using a machine with an Intel Core 2 Duo Processor with 4GB of RAM. These types of scalability issues are a challenge for developers when using these large ontologies. As demonstrated, the complexity of processing for large ontologies is known to be critical. Since smaller ontologies take a shorter time to open, load, and use with tools, having smaller interrelated modules instead of large and complex ontologies could possibly improve the performance of the processing tools.

[1] `http://mowl-power.cs.man.ac.uk:8080/metrics/owlmetrics`

3.2. Annotation features

A module is described by one or more annotation features. These annotation features provide information about things that occur in modules, and how they are related to each other and interact.

3.2.1. Modification annotation features

These annotation features describe the ways in which an ontology is modified in order to create a new module.

P1: Seed signature A seed signature occurs when the user specifies some input entity to base the resulting module on (Cuenca Grau *et al.* (2008); Del Vescovo (2011); Del Vescovo *et al.* (2011)). All entities related in some way to the entity chosen as seed signature are included in the module. For instance, for modularising the DOLCE ontology for a module with only wholly-present objects, the 'endurant' entity is selected as a seed signature (Khan and Keet (2013)).

P2: Information removal Information removal is when parts of the ontology are selected to be removed from the ontology, resulting in a module without all the detail of the original ontology. For information removal, an input entity need not be selected as a basis for modularisation as in the case for the seed signature characterisation above. For instance, the NCS ontology on Bantu noun classes (Chavula and Keet (2015)) reuses only part of the GOLD ontology, as it has no need for, among others, phonetic properties.

P3: Abstraction Abstraction is hiding undesirable knowledge from an ontology at different levels (Giunchiglia *et al.* (1997); Keet (2005)). It is used to provide a simplified view of the ontology. Hence, there are modules with more/less detail in the system. However, the source ontology with the original knowledge still exists in the system as a related module.

P3.1: Breadth abstraction This occurs in a module where some relational properties of entities in the module are removed in order to provide a simpler view of the structure of the ontology, therefore the 'breadth' of the ontology is reduced.

P3.2: Depth abstraction This occurs in a module where high-level classes from the original ontology exist and lower-level classes are removed, therefore the 'depth' of the ontology is reduced.

P4: Refinement Refinement occurs in ontology modules where new alternate axioms are added to the module, as a result of the modularisation process. This could be to assist with inter-module links, or when computationally-expensive ontology language features are modified resulting in new axioms. For instance, to reduce reasoning time for the DMOP ontology, the `InverseObjectProperties` axiom was removed and replaced with the OWL `ObjectInverseOf` axiom (Keet *et al.* (2014)).

3.2.2. Relational properties

These properties describes how a module is related to other modules.

P5: Stand-alone This describes a module that has no external links or imports with other ontologies, and can exist on its own. It is self-contained and can be modified without having dependencies on other modules. For instance, the BioTopLite module (Schulz and Boeker (2013)), a top-domain level ontology for the life sciences domain, does not contain any inter-module relations with other ontologies nor does it have any import statements linking other ontologies to it.

P6: Source ontology A source ontology describes cases where there is an original ontology which has been modularised in some way resulting in the module. For instance, the DMOP-profile-EL module has the source ontology DMOP (Keet *et al.* (2014)).

P7: Proper subset This describes a module that contains a subset of entities that are contained in another source ontology. The module has fewer entities than the source ontology. For instance, the FMA_subset ontology module omits all relationships other than `is_a`, `part_of`, and `has_part` and thus has fewer entities than the original FMA ontology (Rosse and Mejino (2003)).

P8: Imports This describes a module that contains other ontology components, by using the `owl:import` statement declared for importing another ontology. For instance, the Spatial Ontology module (Hois *et al.* (2009)) from the set of architectural design modules that imports the DOLCE ontology (Masolo *et al.* (2003)).

3.2.3. Set annotation features

These annotation features describe the interaction of a set of inter-related modules.

P9: Overlapping Overlapping in modules refer to cases where entities in an ontology system can be found in more than one module of the system (Parent and Spaccapietra (2009)). These modules partially cover the same concepts. In overlapping modules, entities in different modules may have dependencies on one another. Thus changes to one module may have an effect on others.

P10: Mutual exclusion Mutual exclusion, or disjointness in modules, refers to the case where entities in a system of ontology modules are found in exactly one of the modules; i.e., modules have no entities in common (Parent and Spaccapietra (2009)), with the exception of `owl:Thing` and `owl:TopObjectProperty` in case of OWL ontologies. While this is easier in maintenance as it avoids duplications, it is difficult to create due to those axioms that relate entities.

P11: Union Equivalence Union equivalence occurs when the union of a set of modules is semantically equivalent when compared to the original ontology.

P12: Partitioning Partitioning occurs in large ontology whereby it is structurally divided into a set of independent modules, thereby allowing concurrent reuse in distributed systems (d'Aquin *et al.* (2009)). Independence is meant that the modules cover sufficiently different knowledge of the domain; e.g., representing the anatomical knowledge about the limbs and about the eyes of an animal.

P13: Inter-module interaction This describes modules that have links to other modules to relate entities in a similar way to their existence in the original ontology to ensure that the knowledge is preserved. Inter-module interaction among modules exist if there are either bridge ontologies in the set to link together modules, or linking languages are explicitly used within the modules. For instance, in the EDAM bioinformatics ontology (Ison *et al.* (2013)), the object property `is_format_of` has as domain the class `Format` and range `Data`. When it was partitioned with SWOOP for this study, the `Data` class was present in the first partition while `Format` and `is_format_of` were present in the third partition, having used ε-connections to create interaction among these entities that existed in different modules.

P14: Pre-assigned number of modules This occurs when the number of modules to be created or generated in a system is known prior to development. For instance, the modularisation tool requires one to state the number of to-be-generated modules upfront, or an ontology is to be divided into a number of modules based on the developers that will collaboratively create and maintain the ontology.

There are some obvious combinations and exclusions of features for a module, such as that P11 implies P6, P8 implies P9, a module cannot have both features P9 and P10, P5 exclude P9-P15, P7 excludes P10, and P2 and P3 do not go with P10, P11, and P13. Conversely, P7, P9, and P14 may apply to a single module, as may P2, P7, and P12.

3.3. Types

Ontology modules can be classified into different types, based on how modularisation of the ontology occurs. Besides using and refining Borgo's module subtypes, mentioned in Section 2, we have identified new subtypes: locality, privacy, axiom abstraction, type abstraction, high-level abstraction, weighted abstraction, expressiveness sub-language, and expressiveness feature modules which are further described in the following sections.

3.3.1. Functional modules

For these type of modules, the users identify the functional components within an ontology to be separately modularised, which assists with selective re-use of an ontology.

T1: Ontology design pattern modules An ontology is to be modularised by identifying a part of the ontology that can be reused as a best practice for recurring ontology issues; hence, one can isolate a new ontology design pattern (ODP) (Gangemi and Presutti (2009)) for general reuse. For instance, the Set ODP (Ciccarese and Peroni (2014)) can be reused for any domain instead of starting from scratch, and there are several content ODPs. However, not all ODPs are considered modules; e.g., the Lexico-Syntactic ODPs are not.

T2: Subject domain modules A large domain must be subdivided according to the subdomains present in the ontology. For instance, the set of architectural design modules (Hois *et al.* (2009)) such as Spatial ontology, Building construction, among others.

T3: Isolation branch modules A subset of entities from an ontology is extracted. However, entities with weak dependencies to the signature are not to be included in the module. For instance, to isolate the 'endurant' branch of DOLCE (Masolo *et al.* (2003)), the `dolce:physical-endurant` entity is a direct subclass of `dolce:endurant` to include but not the `dolce:perdurant` because it is linked to `dolce:endurant` in terms of participation.

T4: Locality modules A subset of entities from an ontology is extracted. However, all entities that are dependent on the subset are included in the module. For the example in T3, this means that the `dolce:perdurant` entity is to be included in the module, along with others that are related to `dolce:perdurant`.

T5: Privacy modules Some information must be hidden or removed from an ontology so that modules can be kept private from each other.

3.3.2. Structural modules

Structural modules are those ontologies that have been partitioned into modules based on structure. The focus of the modularisation is on the syntax of the ontology graph. Each module is to be separate from one another; and ideally to have disjoint modules so that the union of all modules is equivalent to the original ontology.

T6: Domain coverage modules There is a large ontology, and developers wish to facilitate ontology maintenance by dividing ontologies structurally, without considering the semantics of the ontology. Hence, the modules are divided by their graphical structure and placement of entities in the taxonomy such that similar size modules could be generated. If the ontology modules are to be maintained collaboratively among a team with a specific number of ontology developers, the number of modules to be created could be specified and the structure of the ontology is exploited to create modules. For instance, the Foundational Model of Anatomy Ontology (Rosse and Mejino (2003)) contains contains over 100 000 entities describing the exhaustive biomedical informatics domain. This could be modularised structurally for ease of use.

T7: Ontology matching modules An ontology must be modularised to assist with ontology matching by partitioning it into disjoint modules so that there is no repetition of entities when matching occurs. Most matching techniques implement structural or string-matching techniques, hence the semantics of the original ontology need not necessarily be preserved. For instance, the Common Anatomy Reference Ontology (CARO) (Haendel *et al.* (2008)) aims at aligning existing anatomy ontologies. To assist with aligning it to domain ontologies, CARO could be partitioned to smaller modules.

T8: Optimal reasoning modules A large ontology is to be divided into smaller modules to assist with overall reasoning over the ontology and to ensure that reasoners do not malfunction. The DMOP ontology contains over 758 entities and over 4000 axioms and takes almost 10 minutes for the reasoner to perform classification; it would be less time-consuming to maintain and extend if localised reasoning in a module would be possible. This differs from creating modules that are of a less-expressive ontology language which is discussed later.

3.3.3. Abstraction modules

For abstraction modules, some detail must be hidden from the ontology to create a simpler view of the ontology with less detail.

T9: Axiom abstraction modules This is a module having fewer axioms with object properties relating classes, thereby decreasing the horizontal structure of the ontology. For instance, to create taxonomies for classification purposes from ontologies.

T10: Entity type modules This is a module where a certain type of entity is removed from the ontology, e.g., data properties or object properties. For instance, removing the application-specific instance data (individuals) from an ontology to reuse in another application.

T11: High-level abstraction modules This is a module where only higher-level classes of the ontology are required, thereby decreasing the vertical structure of the ontology. For instance, the DMOP-branch-Toplevel module (Keet *et al.* (2014)) containing only the high-level entities of DMOP.

T12: Weighted modules The developer decides on entities in the ontology that are more important than others. For instance, using abstraction rules to assign higher weighting to entities that are deemed more important than others (Campbell *et al.* (1996); Keet (2005)).

3.3.4. Expressiveness modules

For expressiveness modules, an ontology is modularised according to an ontology sub-language by removing some expressiveness power.

T13: Expressiveness sub-language modules These modules contains limited language features that are captured in a sub-language of a core ontology language.For instance, the OpenGalen (Rector *et al.* (2003)) module in OWL 2 EL was created to test the lightweight ELK (Kazakov *et al.* (2012)) reasoner for EL ontologies.

T14: Expressiveness feature modules These modules contains limited language features that are not necessarily defined by any sub language and consider modelling alternatives to preserve the meaning of the ontology. For instance, the DMOP-WithoutInverseRoles modules was created by removing the OWL `InverseObjectProperties` language feature and replaced with the OWL `ObjectInverseOf` axiom (Keet *et al.* (2014)).

Understanding the type of a particular module is of interest towards creating a foundation for ontology modularity.

3.4. Techniques

We identify the techniques used by approaches to create modules, and classify them into categories. Such techniques are not only restricted to the ontology field, but also more broadly, such as from graph theory and statistical approaches.

3.4.1. Graph theory techniques

These approaches are designed to solve the general problem of community detection. In graphs, communities are clusters of nodes that are fairly independent of each other with links between them.

MT1: Graph partitioning Graph partitioning is the problem of dividing a graph into partitions with the condition that vertices are not shared across different partitions, and the number of partitions is known. For ontologies, graph partitioning algorithms would be most useful in cases where structural division of the ontology modules is a driving force for it would generate modules of equal size.

There are several examples of the application of graph partitioning to ontology modularisation (Ahmed *et al.* (2015); Amato *et al.* (2015); Kalyanpur *et al.* (2006); Schlicht and Stuckenschmidt (2008)). In the PATO partitioning tool (Schlicht and Stuckenschmidt (2008)), graph partitioning is performed by using maximal line islands (Batagelj (2003)) to compute partitions. A maximal line island checks that for a set of nodes, the strength of the connection between the nodes inside the set is higher than the strength of any

connection those outside the set. Unlike traditional graph partitioning, in PATO, the number of partitions to be created is unknown and automatically generated (Schlicht and Stuckenschmidt (2008)).

MT2: Modularity maximisation Modularity maximisation methods (Newman (2004)) aim at optimising the connection between nodes in graphs by using a modularity function Q. Q measures the concentration of edges within modules compared with random distribution of links between all nodes regardless of modules. For ontologies, this means that irrespective of the location of the concepts in the hierarchy, modules will be created based on concepts that have strong axiomatic relations with others.

3.4.2. Statistical techniques

This approach uses statistical equations to create ontology modules. To do so, the entities in the ontology are converted to data and statistical methods and functions are applied onto the data with the aim of creating modules.

MT3: Hierarchical clustering Hierarchical clustering is used to group together data, when little is known about it, such as the number of partitions. It is aimed at building a hierarchy of clusters, by an agglomerative or divisive strategy. An agglomerative strategy is one in which each data point is placed in separate clusters, and clusters are merged based on a distance function between data points in clusters while divisive strategy is one in which data is divided recursively as one moves down the hierarchy.

There is an application of using hierarchical clustering for ontology modularisation (Garcia *et al.* (2012)), where two hierarchical clustering algorithms obtained similar results when compared to other graph theory approaches. However, other approaches had better results, at least for the case of modularising a version of the pizza ontology because their modules grouped together vegetarian, non-vegetarian, and general pizza entities while the modules of the hierarchical approach did not.

3.4.3. Semantic techniques

For these approaches, the entities and axioms of the ontology are used to drive the modularity approach. The common aspect in each of these approaches is that it is user driven, i.e., a user provides initial information about entities to drive the modularisation process.

MT4: Locality-based modularity This approach is used to generate modules based on a signature with the condition that conservative extension holds for the given module. Conservative extension (Cuenca Grau *et al.* (2008)) means that every axiom's meaning from the original ontology is preserved in the module and is greatly influenced by the atomic structure of the ontology. For instance, the biological ontologies from the BioPortal repository (Whetzel *et al.* (2011)) have been proven to modularise well using locality methods (Del Vescovo *et al.* (2011)), thanks to them having on average 2 axioms per atom. On the other hand, to generate a locality module of endurant entities (whole objects) from the DOLCE ontology (Masolo *et al.* (2003)), many perdurant entities (entities unfolding in time) would also be contained in the module, because there exists an axiom $\texttt{endurant} \sqsubseteq \exists\ \texttt{participant-in perdurant}$,in the DOLCE ontology. Therefore the module would not only contain endurant entities because the conservative extension of the original ontology is guaranteed. The atoms in the DOLCE ontology are large and therefore not suitable for locality approaches, as shown by Khan and Keet (2013).

MT5: Query-based modularity These approaches require that the user initially creates a query with a language such as SPARQL and a module is automatically created based on that query. Noy and Musen (2009) use queries to create ontology views by selecting an input entity and traversing the ontology to select other relevant entities to be included in the module until a particular depth is reached. Similarly, given an input entity, the KMI tool (d'Aquin *et al.* (2006)) recursively inspects the ontology to include the other relevant elements found in the definition of the entity.

MT6: Semantic-based abstraction Abstraction, the principle of simplifying complex models by removing some unnecessary details, is applied to ontologies to create simpler modules. Semantic-based abstraction is an approach whereby the semantics of the model is analysed using a set of pre-defined rules to determine key entities, where the key entities are deemed more important than others (Campbell *et al.* (1996); Keet (2005)). For instance, for the Blood and Bacteriocins ORM models (Keet (2005)), mandatory

Table 1
Modularisation techniques implemented by each tool.

Modularity Tool	**Modularisation Technique**
SWOOP	MT1: Graph partitioning
TaxoPart	MT1: Graph partitioning
OWL module extractor	MT4: Locality-based
Protégé copy/move/delete axioms	MT4: Locality-based
PATO	MT1: Graph partitioning
PROMPT	MT5: Query-based

roles are weighted with 10 points while single-role set constraint are weighted with 5 points. Similarly, this could be applied to ontologies by designing a set of weighted rules to guide the ontological abstraction process.

MT7: A priori modularity An *a priori* modularity method (Thakker *et al.* (2011)) is one in which the modular structure of the domain is decided and the modules are created at the onset.

MT8: Manual modularity For manual modularity, the ontology developer decides which entities and axioms should be removed from an ontology, and manually creates a 'custom' module based on this. Unlike the *a priori* modularity method, here the modules are not created at the onset of development, but created later on, based on some existing ontology. For instance, for the DMOP-WithoutInverseProperties module, some language features of the ontology were manually removed to improve the reasoning (Keet *et al.* (2014)).

The existing modularisation tools are displayed in Table 1 alongside their underlying modularisation techniques. For the techniques not listed, we could not find tools.

3.5. Evaluation metrics

It is necessary to evaluate the resultant modules of the ontology modularisation techniques, in terms of the quality of the generated module and the features of the tool. Existing studies (d'Aquin *et al.* (2007, 2009); Schlicht and Stuckenschmidt (2006)) mention some techniques such as size, logical correctness, cohesion, etc. Identifying and populating evaluation metrics with sub-dimensions is intended for future work as it also requires an application component with which to quantitatively measure ontology modules.

The heterogeneous dimensions that we have identified, discussed and populated demonstrates that ontology modularity is not a straight-forward, solitary concept but rather a methodological approach with specific conditions resulting in different ontology modules. The dimensions described in this section form the answer to our first research question regarding modularity, "1) What are the use-cases, techniques, types, and annotation features that exist for modules?". The next step is to categorise a set of real modules using these dimensions towards creating dependencies between the dimensions.

4. Usage of modules by ontology developers: a quantitative assessment

The purpose of the experimental evaluation is to classify existing modules available on the web with the modularity dimensions, and to determine the dependencies between them towards creating a framework for modularity that will be able to assist ontology developers in working with ontology modules.

4.1. Materials and methods

The method for the experiment is as follows:

1. Collect ontology modules from ontology repositories and existing literature on modules.
2. Classify each ontology module according to the proposed module dimensions.
3. Conduct a statistical analysis to determine the frequency of dimensions occurring in each module.

The materials used for the experiment were as follows: Protégé v4.3 (Gennari *et al.* (2003)), SWOOP v2.3 (Kalyanpur *et al.* (2006)), OWL Module Extractor (Cuenca Grau *et al.* (2008)), TaxoPart (Hamdi *et al.* (2009)), and a set of ontology modules. The sample size was 189 ontology modules of varying domains, such as architectural, data mining, biological, chemical, linguistic, among others. Of these 189 modules, 146 belonged to 11 sets of inter-related modules. A set of inter-related modules is when a large subject domain is represented by a set of modules rather than a large ontology. For instance, one of the 11 sets is the 10 modules of the myExperiment (Newman *et al.* (2009)) ontology. All the test files used for this evaluation can be downloaded from `www.thezfiles.co.za/Modules/testfiles.zip`.

4.2. Results and Discussion

Modules that were found on the web include those of type T1, T2, T3, T8, T11, T12, T13, and T14. We could not find any modules of type T4, T5, T6, T7, T9, and T10, so we generated them by using tools (SWOOP, OWL Module Extractor, and TaxoPart) or manually. 74 of the modules were publicly available and the rest were generated for this study.

Thereafter, each of the modules were classified according to the characteristics: its type, use-case(s), annotation feature(s), and technique.

4.2.1. Analysis by dimension

A statistical analysis was conducted to determine the frequency of each characteristic, which we describe here.

Frequency of use-case The frequency of each use-case among the set of ontology modules is shown in Figure 1. The dominant use-case or purpose among the modules was U6 (collaborative efforts) which accounted for over 70% of the use-cases. Modules of U6 in the sample set include the myExperiment (Newman *et al.* (2009)) and Gist (McComb (2010)) ontologies. Indeed, ontology modularisation has been motivated by the need for collaboration among multiple ontology developers in a number of publications (Pathak *et al.* (2009); Cuenca Grau *et al.* (2007); Thakker *et al.* (2011)). One of the success factors of the SNOMED ontology project (Lee *et al.* (2013)) is collaboration which is unsurprising because it is large, containing over 300 000 entities and thus requires a team of experts for development.

Thereafter, U4 (processing) followed with 49.74% of the modules. There were many such modules in this set because the Spatial (Dahdul *et al.* (2014)) and Common Anatomy Reference Ontology (CARO) (Haendel *et al.* (2008)) ontologies were automatically split with a specialised ontology alignment partitioning tool, TaxoPart. This resulted in a large number of modules containing, in most cases, fewer than 5 entities, that would allow for easy processing for use with automatic alignment tools.

Next, U7 (reuse), U1 (maintenance), and U3 (validation) use-cases account for 35.45%, 28.57%, and 28.57% of the modules, respectively. Modules motivated by all three of these use-cases include the data mining OntoDM (Panov *et al.* (2008)), myExperiment (Newman *et al.* (2009)), and OntoSpace (Bateman *et al.* (2003)) ontology modules whereby a large domain was divided according to subject domains.

U2 (reasoning) and U5 (comprehension) were the least popular use-cases, with 5.82% and 4.76% of the set, respectively. For reasoning, there was some split domain DMOP ontology modules motivated for divide-and-conquer reasoning as well as a less-expressive EL language module for DMOP (Keet *et al.* (2014)). For comprehension, there were lighter versions of ontologies with less knowledge, such as Biotoplite based on Biotop (Schulz and Boeker (2013)), and GFO-Basic based on GFO (Herre (2010)).

From the set of modules, all the use-cases are exhibited for both the natural modules and the artificially created modules for the study.

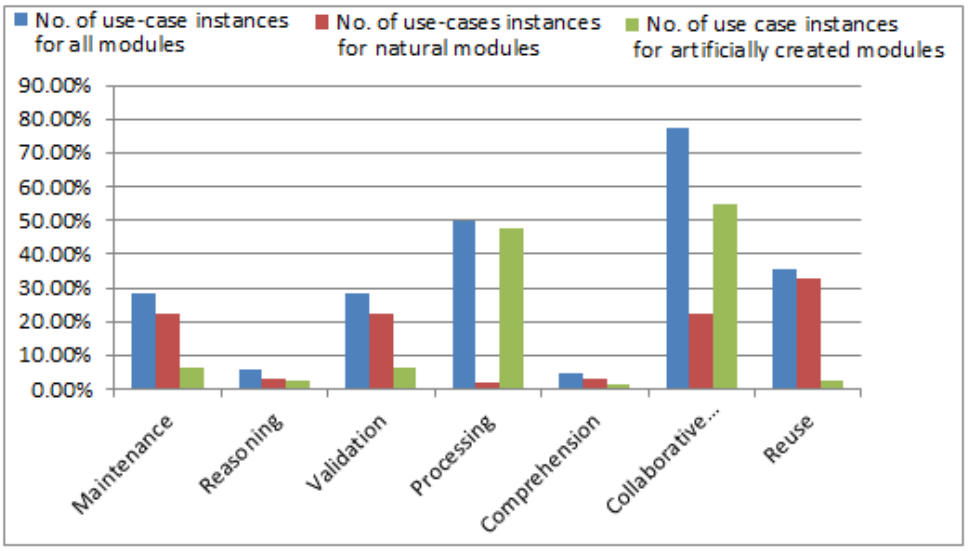

Fig. 1. The frequency of each use-case for the set of 189 modules.

Frequency of type Figure 2 shows the frequency of each type for the set of 189 modules is skewed toward module type T7 (ontology matching modules). From the 189 modules, almost half of them were ontology matching modules; this is because the TaxoPart tool generated a large number of ontology matching modules for the two source ontologies, CARO and Spatial ontologies, where each generated module contained less than 5 entities in most cases. Second, there were a considerable number of T2 modules (subject domain modules), 22.22% (n= 42 modules); these were freely available in ontology repositories.

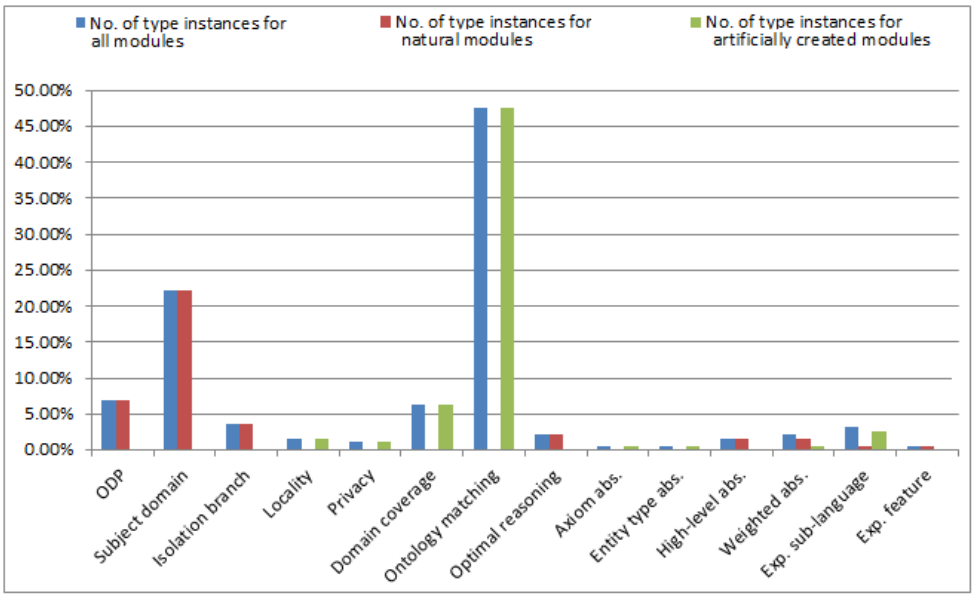

Fig. 2. The frequency of each type for the set of 189 modules; exp. = expressiveness, abs. = abstraction.

For the remaining types of modules in the set, there were very few of each type, ranging from 6.88% to 0.53%. These module types were difficult to find in existing repositories, and in cases where publications described such modules, alas the test files were not referenced, or not available at the given URLs. Many of these modules were generated for this experiment.

The natural modules were T1 (Ontology design pattern), T2 (Subject domain), T3 (Isolation branch), T8 (Optimal reasoning), T11 (High level abstraction), T12 (Weighted abstraction), T13 (Expressiveness sub-

language), and T14 (Expressiveness feature). The module types that could not be found naturally, hence generated for this study, were types T4 (Locality), T5 (Privacy), T6 (Domain coverage), T7 (Ontology matching), T9 (Axiom abstraction), and T10 (Entity type abstraction).

Frequency of technique For the frequency of techniques among modules, as displayed in Figure 3, it is apparent that MT1 (graph partitioning) is the most popular of the techniques, with over half, or 54% of the modules. This is because graph partitioning techniques were used by the TaxoPart tool for the large portion of Spatial and CARO ontology matching modules, discussed in the previous section. Furthermore, graph partitioning approaches was applied in the SWOOP partitioning algorithm for splitting up the large domain modules for the Amino Acid (Stevens and Lord (2012)), Edam bioinformatics (Ison *et al.* (2013)), and MEO Metagenome and Microbes Environmental[2] domain ontologies.

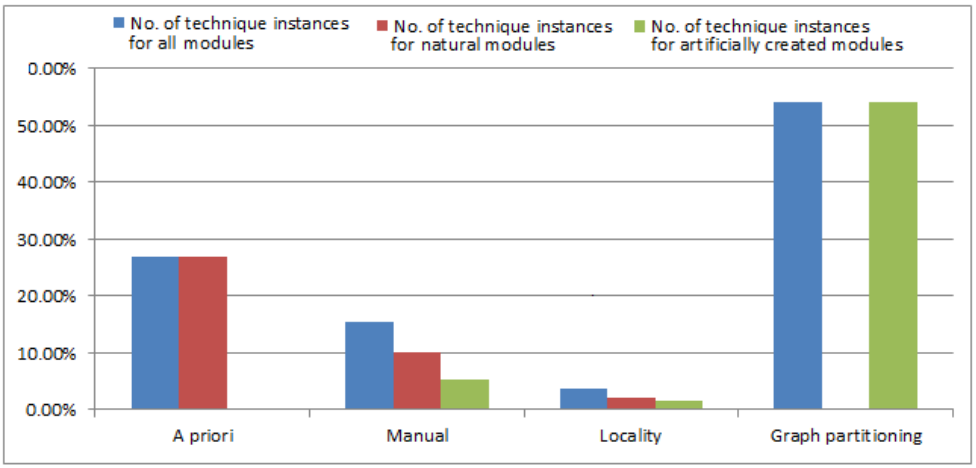

Fig. 3. The frequency of each technique for the set of 189 modules.

Next, MT7 (*a priori*) modularity techniques were used for 27% of the modules. These sets of modules include the set of aforementioned Architectural, Gist, and OntoDM modules. MT8 (manual methods), accounted for 15% of the modules, including, the Biotoplite ontology, and the Set ontology design pattern (Ciccarese and Peroni (2014)). Lastly, MT4 (locality-based) techniques accounted for the smallest number of modules, 4%. These included a module with the seed signature seizure_types, based on the Epilepsy ontology (Sahoo *et al.* (2014)). Indeed, the locality-based modularity technique and principles have been discussed in a number of existing works (Cuenca Grau *et al.* (2008); Del Vescovo (2011); Del Vescovo *et al.* (2013); Sattler *et al.* (2009)) but evidence of such modules in applications is scarce.

The techniques that were used for the natural modules include MT8 manual, MT4 locality, and MT7 *a priori* techniques. The techniques that were used for generating the artificial modules for the study were MT1 graph partitioning, MT4 locality, and MT8 manual techniques.

Frequency of annotation feature For the frequency of annotation features among modules, as displayed in Figure 4, annotation feature P5 (stand alone) and P6 (source ontology) is exhibited in most of the modules (73.02%). Indeed a large number of modules in the set contain no links or imports to other modules and are thus stand-alone, and most modules in the set are based on an original source ontology.

Annotation feature P2 (information removal) is present in 68.25% of the modules, meaning that some detail is removed resulting in a smaller module with less knowledge. Annotation feature P7 (proper subset) is also present in 68.25% of the modules. The remaining annotation features, P1, P3, P4, P5, P6, and P10, are present in only a few of the modules ranging from 19.04% to 0.53%. P3.1 (breadth abstraction) is only exhibited in 1 module, in the FGA_taxonomy module, that was generated for this study for

[2] http://mdb.bio.titech.ac.jp/meo

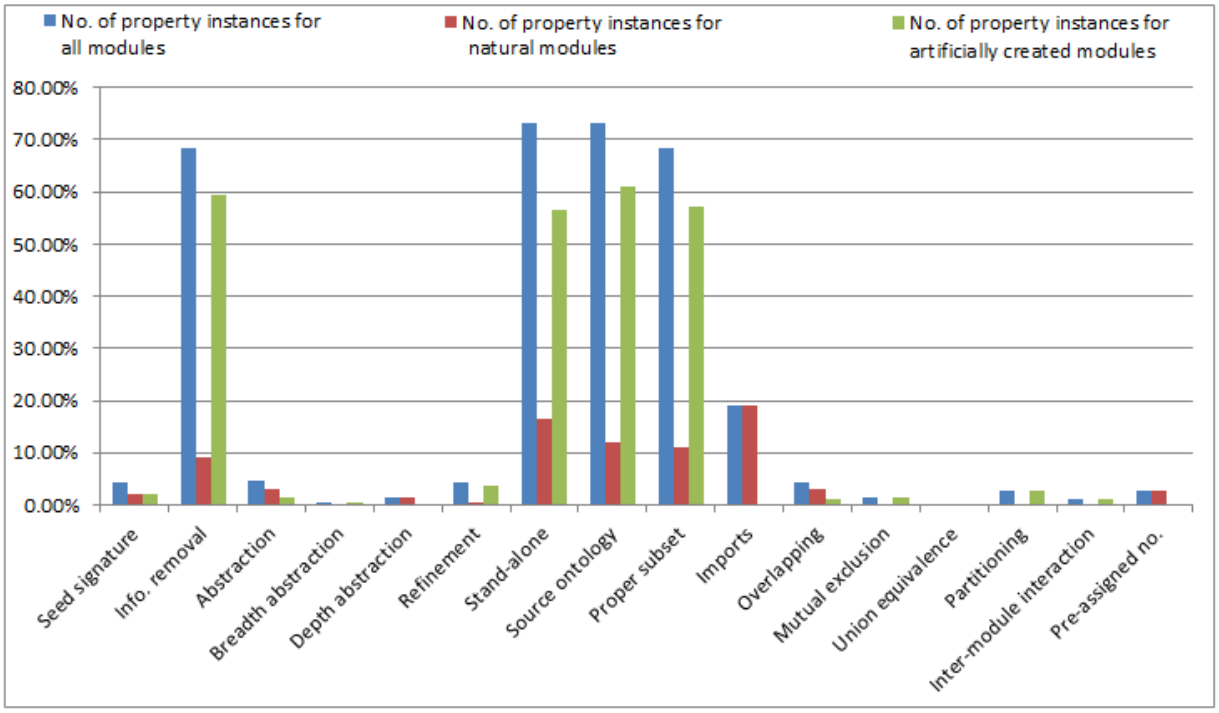

Fig. 4. The frequency of each annotation feature among modules.

creating a bare taxonomy from the original Fungal Gross Anatomy (FGA) ontology[3]. For the refinement annotation feature, P4, its low presence (4.23%) in the set is no surprise because refinement is concerned with adding more detail to a module, thus refining it, and going against the basic definition of modularity in which a module is a smaller subset of a source ontology. Refinement existed in the DMOP-WithoutInverseProperties module because some OWL language features of the ontology was removed to improve the reasoning but new axioms were added to preserve the semantics of the ontology. Refinement also occurred in most of the modules that were created by partitioning using SWOOP because new axioms were introduced to enable linking among them.

There were 11 sets of inter-related modules. 72.73% of them, exhibit P9 (overlapping), whereby entities in a set overlap exist in more than 1 module of a set. The overlapping annotation feature in a module ensures the knowledge preservation within a set of modules, but poses other challenges such as module maintenance and consistency. Annotation feature P12 (partitioning) was also a prevalent annotation feature among sets of inter-related modules, consisting of 45.45%. Since most of the modules were generated using graph partitioning techniques (recall the CARO and Spatial alignment modules), this is no surprise. Annotation feature P14 (pre-assigned number of modules), was present in 45.45% of the modules in the set, whereby the number of modules to be created for the system is known. Such modules include the set of Gist modules, and the set of myExperiment modules. Annotation feature P10 (mutual exclusion) was present in 27.27% of the module set, whereby entities were not shared across modules; the MEO, CARO, and Spatial ontology module sets exhibited this annotation feature. Annotation feature P13 (inter-module interaction) is exhibited in 18.18% of the ontology module sets; it is present in the set of Amino acid modules (Stevens and Lord (2012)), and the set of EDAM bioinformatics modules, thanks to the ε-connections links generated by the SWOOP partitioning tool to allow interaction.

Annotation feature P11 (union equivalence) was not present in any of the sets of modules as existing tools fail to ensure such an annotation feature. When the module sets were merged to check for the union equivalence, there were two reasons for the lack of union equivalence. Firstly, there were extra axioms added to the modules using ε-connections for inter-module interaction; thus, for some sets, the union of

[3] http://www.yeastgenome.org/fungi/fungal_anatomy_ontology/

the modules was larger than the original ontology. Secondly, SWOOP did not preserve the annotation axioms of the original ontology; thus, for some sets, the union of the modules was smaller than the original ontology.

For the natural modules, all the annotation features except P10 (Mutual exclusion), P11 (Union equivalence), P12 (Inter-module interaction), and P13 (Pre-assigned number of modules) exist. For the artificially created modules, all the annotation features except P3.2 (depth abstraction), P8 (imports), and P11 (Union equivalence) exist.

5. Toward a framework for modularity

Given the insights obtained with the literature review in Section 3 to elucidate modularity dimensions and annotation features, and with the assessment of actual usage of ontology modules and modularisation by ontology developers (Section 4), we go one step further with this survey by elucidating observed dependencies between the annotation features. This leads to a basic framework for modularity, of which the high-level view is shown in Figure 5.

Fig. 5. A high-level view of the framework for modularity.

The dependencies are used to refine and answer the earlier proposed questions:

2. Given that we wish to create an ontology module with a certain purpose or *use-case* in mind, which modularity *type* of module could this result in? (How do module types differ with respect to certain use-cases?)
3. If we wish to create a module of a certain *type*, which is the best *technique* to use? (Which techniques can we use to create modules of a certain type?)
4. By using a particular *technique*, which *annotation features* will the resultant module exhibit? (Which techniques result in modules with certain annotation features?)

The dimensions of the framework are related as follows. A module's *use-case* results in modules of a certain *type*. A module of a certain *type* is created by a modularisation *technique*. Modularisation *techniques* result in modules with certain *annotation features*.

Answers to these questions are mentioned in following diagrams. For instance, regarding question 2, if we wish to create an ontology for the use-case of U5 comprehension, this could result in a T9-T12 abstraction type module (see Figure 6). Thereafter, for question 3, if the module type is either one of T9-T12 abstraction, the technique for modularisation is MT8 (manual methods); see Figure 7. Lastly, for question 4, when MT8 manual methods are used, the resulting modules exhibit the following annotation features: P1 (seed signature), P2 (information removal), P3 (abstraction), P3.1 (breadth abstraction), P3.2 (depth abstraction), P4 (refinement), P5 (stand-alone), P6 (source ontology), P7 (proper subset), or P8 (imports) (see Figure 8).

We will illustrate the framework's usage in the following example.

Example 1 Let us assume that we wish to reuse the Symptom ontology (Baclawski *et al.* (2004)), a domain ontology about symptoms and signs of diseases.

Use-case Create a module about symptoms on the skin for reuse in a domain ontology about dermatology.

Type Check which type of modules result from reuse: all subtypes of functional modules. Since we need only skin symptoms, we consider an isolation branch, or locality module. We wish to preserve all entities with dependencies to the skin entities, so we create a locality module.

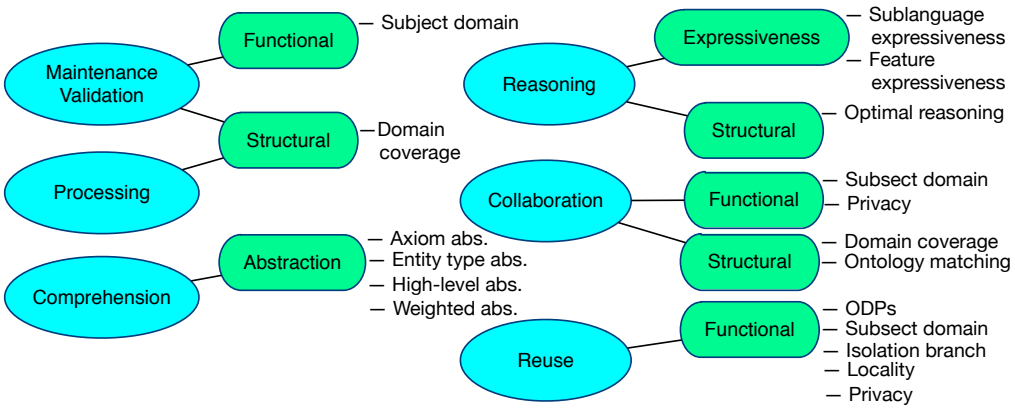

Fig. 6. The dependencies between use-cases and module types.

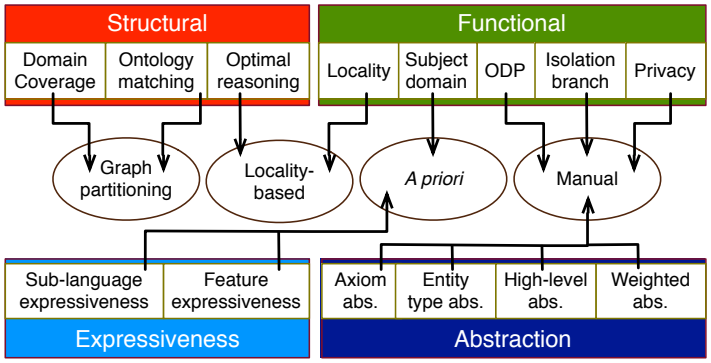

Fig. 7. The dependencies between module types and techniques.

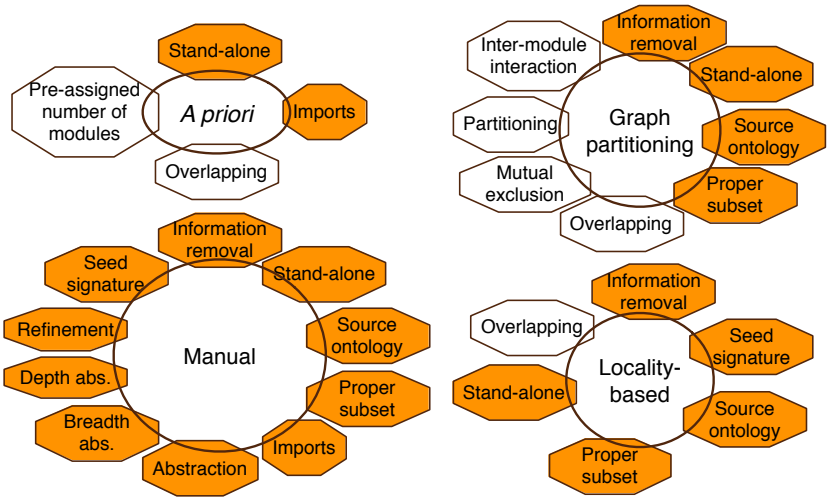

Fig. 8. The dependencies between techniques and annotation features. The shaded hexagons represent the modification and relational annotation features, the unshaded represent the set annotation features.

Technique For the locality module, a locality-based technique is selected, using the OWL Module extractor tool to extract a module containing knowledge about the skin symptoms, with a seed signature `skin and integumentary tissue symptom`.

Annotation feature Modules created with locality-based techniques could have these annotation features: information removal, seed signature, source ontology, proper subset, stand-alone, and overlapping. The generated module exhibits all these annotation features, except overlapping (since it is not a set).

Three other use cases are described in (Khan and Keet (2015)), notably verifying the QUDT modules (Hodgson and Keller (2011)), which were not part of the 'test set' of the previous section, a scenario of reusing a small section of the FMA (Rosse and Mejino (2003)) for an ontology about infections, and how OpenGalen (Rector *et al.* (2003)) may be modularised in the light of tractability of reasoning over it. These use cases, like the one just described, all start with the use case, and the rest follows from that in a step-wise fashion using the links between use case, type, technique, and annotation feature. With more software support for the actual modularisation and easy software-based guidance alike a recommender system based on module requirements, a more comprehensive evaluation can be undertaken.

6. Discussion

The modularity dimensions led to the creation of the ontology modularity framework which can be used to answer the earlier proposed questions thanks to the dependency relations among modularity dimensions. The framework can be used to solve the first issue concerning modularity: difficulty in modularity technique selection. This is solved by referring to the framework to check which technique results from the use-case of the module. The problem of insufficient modularity tools still exists, but the framework has refined it. In particular, there is no tool to implement modularity maximisation, semantic-based abstraction, and hierarchical clustering techniques.

For the tools that are available, they are not sufficient. They are hardly maintained and sometimes not usable. We had hoped to generate modules from partitioning large ontologies. However, the SWOOP partitioning tool could not be applied for large ontologies such as the FMA ontology (Rosse and Mejino (2003)) as it could not open it, despite manually changing the java heap space parameters. We had also hoped use PROMPT traversal views with Protégé (Noy and Musen (2009)) for query-based modularity, but it malfunctioned and returned a null pointer exception. OWL module extractor was considered for extracting DMOP modules. It extracts modules by using an input set of terms as a signature while ensuring the logical completeness of the module. This means that for every axiom of the original ontology, the meaning of the axiom is preserved in the module. Due to dependencies between entities in the DMOP ontology and the logical completeness constraint, OWL module extractor generated too large a module to use to improve reasoning.

There is a heavy reliance on using manual methods for module creation. For 9 out of the 14 module types, manual methods were used for module creation. The implementation of tool-based methods as a technique for some of the abstraction and expressiveness type modules is within reach, given the recent advancements in ontology API libraries such as the OWL API (Horridge and Bechhofer (2011)).

The example of the application of the framework to the Symptom ontology module extraction demonstrate that the framework is promising for guiding the modularisation process, as were other case studies (Khan and Keet (2015)). The framework provided guidance in classifying the module according to its type, which technique to use for modularity, and in addition, which annotation features the module should exhibit. While the framework currently does not define module types in terms of a set of distinguishing annotation properties, we hope that with more usage and testing with a larger set of real-world modules, this will emerge.

The annotation features can be used for ontology annotation towards improved metadata. Metadata promotes ontology discovery and reuse, and repositories such as BioPortal (Whetzel *et al.* (2011)), Ontohub

(Mossakowski *et al.* (2014)), and ROMULUS (Khan and Keet (2013)) use metadata models. There is limited metadata concerning modular ontologies (Khan and Keet (2013)), which now can be refined and improved further. If a module is not annotated with some annotation features, it will indeed be difficult to figure out its annotation features, but, in theory at least, it may be possible to determine them when either the source ontology or the other modules in the set are known.

Given the amount of publicly available modules that were found for the study (n=74), it appears that many modules were created specifically for this study and would thus affect the results of the framework. However, a large amount of the generated modules, 78.3%, (n=90) of the modules were created as M7: Ontology matching modules, each with fewer than 5 entities, from just 2 source ontologies using the TaxoPart tool. It appears that the nature of ontology matching modules is to have many tiny modules to promote processing for the ontology matching tools. For the rest of the generated modules, there was 10.4%, (n=12) generated with SWOOP, 8.7%, (n=10) manually created, and 2.6%, (n=3) generated with OWL Module extractor. They, in hindsight after modules analysis, seem to be creating modules more in line with publicly available ontology modules. A further issue is the quality of the modules that do exist. There is a dearth of guidance on evaluation metrics and characterisation of what constitutes a 'good' module, and whether such an analysis can be automated. Here, we have assumed that a module created and used is what the developer wanted.

Considering the results of the literature survey, the experimental evaluation, and taking into account its limitations, we now return to the answers for the questions posed in the introduction.

Q1: What are the use-cases, techniques, types, and annotation features that exist for modules?
The use-cases, techniques, types, and annotation features have been identified and categorised in Section 3. In short, the use-cases identified are: maintenance, reasoning, validation, processing, comprehension, collaborative efforts, and reuse. The techniques identified are: graph partitioning, modularity maximisation, hierarchical clustering, locality-based modularity, query-based modularity, semantic-based abstraction, *a priori* modularity, and manual modularity. The types are: ODPs, subject domain-based, isolation branch, locality, privacy, domain coverage, ontology matching, optimal reasoning, axiom abstraction, entity type, high-level abstraction, weighted, expressiveness sub-language, and expressiveness feature modules. Finally, the list of annotation features: seed signature, information removal, abstraction (breadth and depth), refinement, stand-alone, source ontology, proper subset, imports, overlapping, mutual exclusion, union equivalence, partitioning, inter-module interaction, and pre-assigned number of modules.

Q2: How do module types differ with respect to certain use-cases?
The manner in which a module use-case affects the the type of module that will be created is shown by the dependencies between the use-cases and types in Figure 6.

Q3: Which techniques can we use to create modules of a certain type?
The manner in which a module type affects the technique that should be used is shown by the dependencies between the type and technique in Figure 7.

Q4: Which techniques result in modules with certain annotation features?
The manner in which the module technique affects the annotation features that it exhibits is shown by the dependencies between the module technique and annotation features in Figure 8.

Overall, this is, to the best of our knowledge, the, thus far, most comprehensive list of aspects of ontology modules and a first insight into the dependencies between all those dimensions and criteria.

7. Conclusion

We have identified issues and questions concerning modularity. To address them, we identified and populated dimensions concerning modularity which was used in an experimental evaluation with a set of 189 ontology modules resulting in dependencies among the modularity dimensions. The classification of the modules using the dimensions led to the creation of a framework for ontology modularity which can be used to solve the developer's issue concerning modularity technique selection, to refine the issue

concerning insufficient tools for modularisation, and to systematically guide the entire modularisation process.

Several open issues to address have been noted in Section 6, such as tooling support. We are currently filling the evaluation metrics gap, and will look into surveying the developers of the modules used in the test set.

References

Abbès, S. B., Scheuermann, A., Meilender, T., and d'Aquin, M. (2012). Characterizing Modular Ontologies. In *The 6th International Workshop on Modular Ontologies (WoMO '12)*, volume 875 of *CEUR-WS*. 24 July, Graz, Austria.

Ahmed, S. S., Malki, M., and Benslimane, S. M. (2015). Ontology partitioning: Clustering based approach. *International Journal of Information Technology and Computer Science*, **7**(6), 1–11.

Amato, F., De Santo, A., Moscato, V., Persia, F., Picariello, A., and Poccia, S. (2015). Partitioning of ontologies driven by a structure-based approach. In *9th International Conference on Semantic Computing (ICSC'15)*, pages 320–323. IEEE. Anaheim, California, USA, February 7-9 2015.

Antezana, E. *et al.* (2009). The cell cycle ontology: an application ontology for the representation and integrated analysis of the cell cycle process. *Genome Biology*, **10**(5), R58.

Baclawski, K., Matheus, C. J., Kokar, M. M., Letkowski, J., and Kogut, P. A. (2004). Towards a symptom ontology for semantic web applications. In *The Third International Semantic Web Conference, (ISWC'14)*, volume 3298 of *LNCS*, pages 650–667. Springer. Hiroshima, Japan, November 7-11.

Batagelj, V. (2003). Analysis of large networks - islands. Dagstuhl seminar 03361: Algorithmic Aspects of Large and Complex Networks. Dagstuhl, August 31 - September 5.

Bateman, J. A., Fischer, K., Moratz, R., Farrar, S., and Tenbrink, T. (2003). Project I1-OntoSpace: Ontologies for Spatial Communication. In *DiaBruck, 7th Workshop on the Semantics and Pragmatics of Dialogue*, pages 163–164. 4th-6th September,Wallerfangen, Germany.

Bauer, J., Sattler, U., and Parsia, B. (2009). Explaining by example: Model exploration for ontology comprehension. In *The 22nd International Workshop on Description Logics (DL'09)*, volume 477 of *CEUR-WS*. Oxford, UK, July 27-30, 2009.

Belloze, K. T., Monteiro, D. I. S. B., Lima, T. F., Jr., F. P. S., and Cavalcanti, M. C. R. (2012). An evaluation of annotation tools for biomedical texts. In *The Joint V Seminar on Ontology Research in Brazil and VII International Workshop on Metamodels, Ontologies and Semantic Technologies*, volume 938 of *CEUR-WS*. Recife, Brazil, September 19-21.

Bergh, J., Gerber, A., Meyer, T., and van Zijl, L. (2011). Path analysis for ontology comprehension. In *The Seventh Australasian Ontology Workshop (AOW'11)*, CRPIT. ACS. 5 December, Perth Australia.

Borgo, S. (2011). Goals of modularity: A voice from the foundational viewpoint. In *The Fifth International Workshop on Modular Ontologies (WOMO'2011)*, volume 230 of *Frontiers in Artificial Intelligence and Applications*, pages 1–6. IOS Press. Ljubljana, Slovenia, August.

Campbell, L. J., Halpin, T. A., and Proper, H. A. (1996). Conceptual schemas with abstractions: Making flat conceptual schemas more comprehensible. *Data Knowledge Engineering*, **20**(1), 39–85.

Chavula, C. and Keet, C. M. (2015). An orchestration framework for linguistic task ontologies. In *Proceedings of the 9th Metadata and Semantics Research Conference (MTSR'15)*, CCIS, page in print. Springer. 9-11 Sept., 2015, Manchester, UK.

Ciccarese, P. and Peroni, S. (2014). The collections ontology: Creating and handling collections in OWL 2 DL frameworks. *Semantic Web Journal*, **5**(6), 515–529.

Cuenca Grau, B., Horrocks, I., Kazakov, Y., and Sattler, U. (2007). A logical framework for modularity of ontologies. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI'07)*, pages 298–303. Hyderabad, India, January 6-12, 2007.

Cuenca Grau, B., Horrocks, I., Kazakov, Y., and Sattler, U. (2008). Modular Reuse of Ontologies: Theory and Practice. *Journal of Artificial Intelligence Research*, **31**, 273–318.

Dahdul, W. M., Cui, H., Mabee, P. M., Mungall, C. J., Osumi-Sutherland, D., Walls, R., and Haendel, M. (2014). Nose to tail, roots to shoots: spatial descriptors for phenotypic diversity in the biological spatial ontology. *Journal of Biomedical Semantics*, **5**, 34.

d'Aquin, M., Sabou, M., and Motta, E. (2006). Modularization: a Key for the Dynamic Selection of Relevant Knowledge Components. In *1st International Workshop on Modular Ontologies (WoMO'06)*, volume 232 of *CEUR-WS*. Nov 5, Athens, Georgia, USA.

d'Aquin, M., Schlicht, A., Stuckenschmidt, H., and Sabou, M. (2007). Ontology modularization for knowledge selection: Experiments and evaluations. In *The 18th International Conference on Database and Expert Systems Applications (DEXA'07)*, volume 4653 of *LNCS*, pages 874–883. Springer. Regensburg, Germany, September 3-7, 2007.

d'Aquin, M., Schlicht, A., Stuckenschmidt, H., and Sabou, M. (2009). Criteria and evaluation for ontology modularization techniques. In *Modular Ontologies: Concepts, Theories and Techniques for Knowledge Modularization*, volume 5445 of *LNCS*, pages 67–89. Springer.

Del Vescovo, C. (2011). The modular structure of an ontology: Atomic decomposition towards applications. In *Proceedings of the 24th International Workshop on Description Logics (DL'11)*, volume 745 of *CEUR-WS*. Barcelona, Spain, July 13-16.

Del Vescovo, C., Gessler, D., Klinov, P., Parsia, B., Sattler, U., Schneider, T., and Winget, A. (2011). Decomposition and Modular Structure of BioPortal Ontologies. In *10th International Conference on The International Semantic Web Conference (ISWC'10)*, volume 7031 of *LNCS*, pages 130–145. Springer. October 23-27, Bonn, Germany.

Del Vescovo, C., Klinov, P., Parsia, B., Sattler, U., Schneider, T., and Tsarkov, D. (2013). Empirical study of logic-based modules: Cheap is cheerful. In *Proceedings of the 26th International Workshop on Description Logics (DL'13)*, volume 1014 of *CEUR-WS*, pages 144–155. Ulm, Germany, July 23 - 26.

Doran, P., Tamma, V. A. M., and Iannone, L. (2007). Ontology module extraction for ontology reuse: an ontology engineering perspective. In *Proceedings of the Sixteenth ACM Conference on Information and Knowledge Management (CIKM '07)*, pages 61–70. ACM. Lisbon, Portugal, November 6-10.

Gangemi, A. and Presutti, V. (2009). Ontology design patterns. In *Handbook on Ontologies*, pages 221–243. Springer Verlag.

Garcia, A. C., Tiveron, L., Justel, C., and Cavalcanti, M. C. (2012). Applying graph partitioning techniques to modularize large ontologies. In *Joint V Seminar on Ontology Research in Brazil and VII International Workshop on Metamodels, Ontologies and Semantic Technologies*, volume 938 of *CEUR-WS*, pages 72–83. Recife, Brazil, September 19-21.

Gennari, J. H., Musen, M. A., Fergerson, R. W., Grosso, W. E., Crubézy, M., Eriksson, H., Noy, N. F., and Tu, S. W. (2003). The evolution of Protégé: an environment for knowledge-based systems development. *International Journal of Human-Computer Studies*, **58**(1), 89–123.

Giunchiglia, F., Villafiorita, A., and Walsh, T. (1997). Theories of abstraction. *AI Communication*, **10**(3,4), 167–176.

Golbeck, J., Fragoso, G., Hartel, F. W., Hendler, J. A., Oberthaler, J., and Parsia, B. (2003). The national cancer institute's thesaurus and ontology. *Journal of Web Semantics*, **1**(1), 75–80.

Haendel, M. A., Neuhaus, F., Osumi-Sutherland, D., Mabee, P. M., Mejino Jr, J. L., Mungall, C. J., and Smith, B. (2008). CARO-The common anatomy reference ontology. In *Anatomy Ontologies for Bioinformatics*, volume 6 of *Computational Biology*, pages 327–349. Springer.

Hamdi, F., Safar, B., Reynaud, C., and Zargayouna, H. (2009). Alignment-based partitioning of large-scale ontologies. In *Advances in Knowledge Discovery and Management*, volume 292 of *Studies in Computational Intelligence*, pages 251–269. Springer. Strasbourg, France, January 28-30 2009.

Herre, H. (2010). General Formal Ontology (GFO): A foundational ontology for conceptual modelling. In *Theory and applications of ontology: computer applications*, pages 297–345. Springer.

Hodgson, R. and Keller, P. J. (2011). QUDT-quantities, units, dimensions and data types in OWL and XML. *Online (September 2011)* `http://www.qudt.org`.

Hois, J., Bhatt, M., and Kutz, O. (2009). Modular ontologies for architectural design. In *The International workshop on Formal Ontologies Meet Industry (FOMI'09)*. September 2, 2009, Vicenza, Italy.

Horridge, M. and Bechhofer, S. (2011). The OWL API: A java api for OWL ontologies. *Semantic Web Journal*, **2**(1), 11–21.

Ison, J. C., Kalas, M., Jonassen, I., Bolser, D. M., Uludag, M., McWilliam, H., Malone, J., Lopez, R., Pettifer, S., and Rice, P. M. (2013). EDAM: an ontology of bioinformatics operations, types of data and identifiers, topics and formats. *Bioinformatics*, **29**(10), 1325–1332.

Kalyanpur, A., Parsia, B., Sirin, E., Cuenca Grau, B., and Hendler, J. A. (2006). Swoop: A Web Ontology Editing Browser. *Journal of Web Semantics*, **4**(2), 144–153.

Kazakov, Y., Krötzsch, M., and Simancik, F. (2012). ELK reasoner: Architecture and evaluation. In *1st International Workshop on OWL Reasoner Evaluation (ORE-2012)*, volume 858 of *CEUR-WS*. Manchester, UK, July 1st.

Keet, C. M. (2005). Using abstractions to facilitate management of large ORM models and ontologies. In *OTM Workshops*, volume 3762 of *LNCS*, pages 603–612. Agia Napa, Cyprus, Oct 31 - Nov 4.

Keet, C. M. (2007). Enhancing comprehension of ontologies and conceptual models through abstractions. In *10th Congress of the Italian Association for Artificial Intelligence (AI*IA 2007)*, volume 4733 of *LNCS*, pages 813–821. Springer. Rome, Italy, September 10-13.

Keet, C. M., Lawrynowicz, A., d'Amato, C., and Hilario, M. (2013). Modeling issues and choices in the Data Mining OPtimisation Ontology. In *8th Workshop on OWL: Experiences and Directions (OWLED'13)*, volume 1080 of *CEUR-WS*. 26-27 May 2013, Montpellier, France.

Keet, C. M., d'Amato, C., Khan, Z., and Lawrynowicz, A. (2014). Exploring reasoning with the DMOP ontology. In *3rd Workshop on Ontology Reasoner Evaluation (ORE'14)*, CEUR-WS, pages 64–70. July 1, Vienna, Austria.

Keet, C. M., Lawrynowicz, A., d'Amato, C., Kalousis, A., Nguyen, P., Palma, R., Stevens, R., and Hilario, M. (2015). The data mining optimization ontology. *Web Semantics: Science, Services and Agents on the World Wide Web*, **32**, 43–53.

Khan, Z. and Keet, C. M. (2013). The foundational ontology library ROMULUS. In *The 3rd International Conference on Model & Data Engineering (MEDI'13)*, volume 8216 of *LNCS*, pages 200–211. Springer. September 25-27, 2013, Amantea, Calabria, Italy.

Khan, Z. C. and Keet, C. M. (2015). Toward a framework for ontology modularity. In *Proceedings of the Annual Conference of the South African Institute of Computer Scientists and Information Technologists (SAICSIT'15)*, page in print. ACM Conference Proceedings. 28-30 September 2015, Stellenbosch, South Africa.

Konev, B., Lutz, C., Walther, D., and Wolter, F. (2008). Semantic modularity and module extraction in description logics. In *The 18th European Conference on Artificial Intelligence (ECAI'08)*, volume 178 of *Frontiers in Artificial Intelligence and Applications*, pages 55–59. IOS Press. Patras, Greece, July 21-25, 2008.

Konev, B., Lutz, C., Walther, D., and Wolter, F. (2009). Formal Properties of Modularisation. In *Modular Ontologies: Concepts, Theories and Techniques for Knowledge Modularization*, volume 5445 of *LNCS*, pages 25–66. Springer.

Larson, S. D., Fong, L. L., Gupta, A., Condit, C., Bug, W. J., and Martone, M. E. (2007). A Formal Ontology of Subcellular Neuroanatomy. *Frontiers in Neuroinformatics*, **1**, 3.

Lee, D., Cornet, R., Lau, F., and de Keizer, N. (2013). A survey of SNOMED CT implementations. *Journal of Biomedical Informatics*, **46**(1), 87 – 96.

Loebe, F. (2006). Requirements for logical modules. In *The 1st International Workshop on Modular Ontologies (WoMO'06)*, volume 232 of *CEUR-WS*. November 5, 2006, Athens, Georgia, USA.

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

McComb, D. (2010). Gist: The minimalist upper ontology. Semantic Technology Conference. June 21-25 2010, San Francisco, CA.

Mossakowski, T., Kutz, O., and Codescu, M. (2014). Ontohub: A semantic repository for heterogeneous ontologies. In *The Theory Day in Computer Science Satellite workshop (DACS-2014)*. University of Bucharest, September 15-16, 2014.

Newman, D., Bechhofer, S., and Roure, D. D. (2009). myExperiment: An ontology for e-Research. In *The Workshop on Semantic Web Applications in Scientific Discourse (SWASD 2009)*, volume 523 of *CEUR-WS*. Washington DC, USA, October 26.

Newman, M. E. (2004). Fast algorithm for detecting community structure in networks. *Physical review E*, **69**(6), 066133.

Noy, N. F. and Musen, M. A. (2009). Traversing ontologies to extract views. In *Modular Ontologies: Concepts, Theories and Techniques for Knowledge Modularization*, volume 5445 of *LNCS*, pages 245–260. Springer.

Oh, S. and Yeom, H. Y. (2011). Evaluation criteria ontology modularization tools. In *The IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology - Workshops (WI-IAT '11)*, pages 365–368. IEEE Computer Society. Lyon, France, August 22-27, 2011.

Oh, S., Yeom, H. Y., and Ahn, J. (2010). Evaluating ontology modularization approaches. In *The 8th International Conference on Frontiers of Information Technology*, page 6. ACM. December 21 -23, 2010, Islamabad, Pakistan.

Panov, P., Dzeroski, S., and Soldatova, L. N. (2008). OntoDM: An ontology of data mining. In *The 8th IEEE International Conference on Data Mining (ICDM 2008)*, pages 752–760. IEEE Computer Society. December 15-19, Pisa, Italy.

Parent, C. and Spaccapietra, S. (2009). An Overview of Modularity. In *Modular Ontologies*, volume 5445 of *LNCS*, pages 5–23. Springer Berlin Heidelberg.

Pathak, J., Johnson, T. M., and Chute, C. G. (2009). Survey of modular ontology techniques and their applications in the biomedical domain. *Integrated Computer-Aided Engineering*, **16**(3), 225–242.

Paulheim, H. (2008). On applying matching tools to large-scale ontologies. In *The 3rd International Workshop on Ontology Matching (OM'08)*, volume 431 of *CEUR-WS*. Karlsruhe, Germany, October 26.

Rector, A. (2003). Modularisation of domain ontologies implemented in description logics and related formalisms including owl. In *Second International Conference on Knowledge Capture (K-CAP03)*, pages 121–129.

Rector, A. L., Rogers, J., Zanstra, P. E., and van der Haring, E. J. (2003). OpenGALEN: Open source medical terminology and tools. In *American Medical Informatics Association Annual Symposium (AMIA'03)*. AMIA. Washington, DC, USA, November 8-12.

Rosse, C. and Mejino, Jr., J. L. V. (2003). A reference ontology for biomedical informatics: the Foundational Model of Anatomy. *Journal of Biomedical Informatics*, **36**(6), 478–500.

Sahoo, S. S., Lhatoo, S. D., Gupta, D. K., Cui, L., Zhao, M., Jayapandian, C. P., Bozorgi, A., and Zhang, G. (2014). Epilepsy and seizure ontology: towards an epilepsy informatics infrastructure for clinical research and patient care. *Journal of American Medical Informatics Association*, **21**(1), 82–89.

Sattler, U., Schneider, T., and Zakharyaschev, M. (2009). Which kind of module should I extract? In *The 22nd International Workshop on Description Logics (DL'09)*, volume 477 of *CEUR-WS*. Oxford, UK, July 27-30, 2009.

Schlicht, A. and Stuckenschmidt, H. (2006). Towards structural criteria for ontology modularization. In *1st International Workshop on Modular Ontologies, (WoMO'06)*, volume 232 of *CEUR-WS*. ISWC'06 November 5, Athens, Georgia, USA.

Schlicht, A. and Stuckenschmidt, H. (2008). A flexible partitioning tool for large ontologies. In *International Conference on Web Intelligence (WI'08)*, pages 482–488. IEEE Computer Society. 9-12 December, Sydney, NSW, Australia.

Schulz, S. and Boeker, M. (2013). BioTopLite: An upper level ontology for the life sciences. evolution, design and application. In *Informatik 2013, 43. Jahrestagung der Gesellschaft für Informatik e.V. (GI), Informatik angepasst an Mensch, Organisation und Umwelt*, volume 220 of *LNI*, pages 1889–1899. GI. 16-20. September 2013, Koblenz.

Smith, B., Ashburner, M., Rosse, C., Bard, J., Bug, W., Ceusters, W., Goldberg, L., Eilbeck, K., Ireland, A., Mungall, C., OBI Consortium, T., Leontis, N., Rocca-Serra, A., Ruttenberg, A., Sansone, S.-A., Shah, M., Whetzel, P., and Lewis, S. (2007). The OBO Foundry: Coordinated evolution of ontologies to support biomedical data integration. *Nature Biotechnology*, **25**(11), 1251–1255.

Stevens, R. and Lord, P. (2012). Semantic publishing of knowledge about amino acids. In *The 2nd Workshop on Semantic Publishing*, volume 903 of *CEUR-WS*, pages 45–48. Hersonissos, Crete, Greece, May 28th.

Tartir, S., Arpinar, I. B., Moore, M., Sheth, A. P., and Aleman-Meza, B. (2005). OntoQA: Metric-based ontology quality analysis. *IEEE Workshop on Knowledge Acquisition from Distributed, Autonomous, Semantically Heterogeneous Data and Knowledge Sources*, **9**.

Thakker, D., Dimitrova, V., Lau, L., Denaux, R., Karanasios, S., and Yang-Turner, F. (2011). A priori ontology modularisation in ill-defined domains. In *Proceedings the 7th International Conference on Semantic System (I-SEMANTICS'11)*, ACM International Conference Proceeding Series, pages 167–170. ACM. Graz, Austria, September 7-9, 2011.

Tsarkov, D. (2012). Improved Algorithms for Module Extraction and Atomic Decomposition. In *The International Workshop on Description Logics (DL '12)*, volume 846 of *CEUR-WS*. Rome, Italy, June 7-10.

Turlapati, V. K. C. and Puligundla, S. K. (2013). Efficient module extraction for large ontologies. In *4th International Conference on Knowledge Engineering and the Semantic Web (KESW'13)*, volume 394 of *CCIS*, pages 162–176. Springer Berlin Heidelberg.

Vigo, M., Jay, C., and Stevens, R. (2014). Design insights for the next wave ontology authoring tools. In *Conference on Human Factors in Computing Systems (CHI'14)*, pages 1555–1558. ACM. Toronto, ON, Canada, April 26 - May 01, 2014.

Whetzel, P. L., Noy, N. F., Shah, N. H., Alexander, P. R., Nyulas, C., Tudorache, T., and Musen, M. A. (2011). BioPortal: enhanced functionality via new web services from the national center for biomedical ontology to access and use ontologies in software applications. *Nucleic Acids Research*, **39**(Web-Server-Issue).