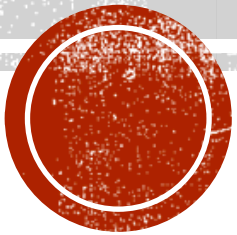
# MODULARISATION IN ONTOLOGIES



Zubeida C. Khan

zkhan@csir.co.za

Senior researcher @ CSIR

# OUTLINE

- Introduction
- Why modularise an ontology?
- History of modularisation
- What is lacking in modularisation?
- Define modularisation
- Classifying modules
- A framework for ontology modularisation
- Theories and techniques for modularisation
- Evaluation metrics
- New algorithms
- Conclusion

# INTRODUCTION

- Modularity: dividing, separating the components of a large system such that modules can be recombined.

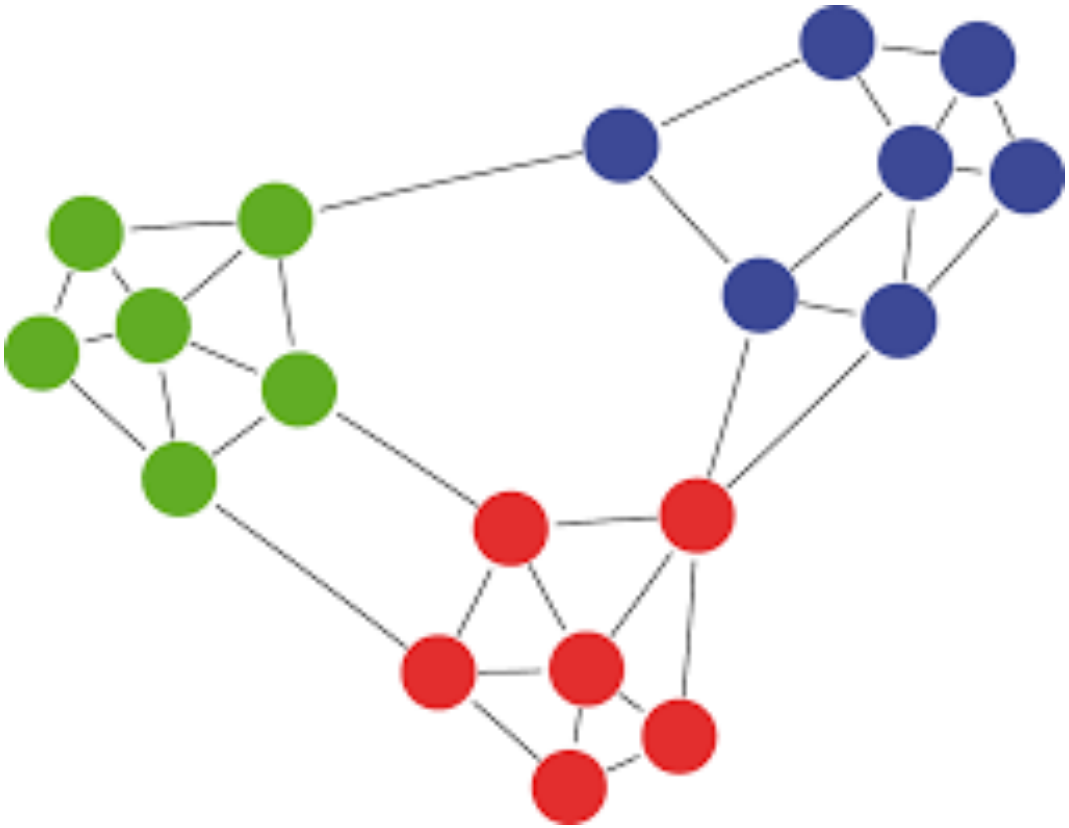
- In CS, modular programming is about separating the functionality of a program into independent, interchangeable modules for specific functions.

# INTRODUCTION

- Ontology developers have been using modularisation to deal with large and complex ontologies.
- It is used to simplify/ downsize an ontology for a particular task.
- What is modularity?
- The possibility to perceive a large knowledge repository as a set of smaller repositories or modules that together compose the entire repository [8].
- There are many different types of ontology modules, such as language expressivity modules, domain-specific modules, more/less-detailed modules, to name a few.

# WHY MODULARISE AN ONTOLOGY?

- Reuse
- Simplify
- Scalability for processing
- Scalability for management



# HISTORY OF MODULARISATION

- In 1995, researchers proposed a 'bottom-up' approach for the development of a chemical ontology [1].
- In 2003, Rector proposed normalisation towards achieving ontology modularisation [2].
- In 2004, Stuckenschmidt and Klein proposed a partitioning algorithm for ontologies based on hierarchy [4].
- In 2004, Noy proposed the traversal method whereby based on an element of the input vocabulary, relations in the ontology are traversed to gather concepts to be included in a module [5].
- In 2005, Keet proposed using abstraction to simplify ontology models by removing unnecessary details for use cases based on a set of rules [6].
- In 2005, Cuenca Grau et al. proposed a partitioning algorithm to generate ontology modules from large ontologies [3].

# HISTORY OF MODULARISATION

- In 2008, locality-based modules were proposed by Cuenca Grau et. al - given an input signature seed, entities of the ontology that reference the signature seed are preserved in the module [7].
- In 2009, Parent and Spaccapietra defined several goals of modularity: scalability for reasoning and maintenance, complexity management, understandability and reuse [8].
- In 2009, D'Aquin et. al reviewed existing tools in terms of modularity criteria e.g., local correctness, size of module, encapsulation, etc [9].
- The method of splitting an ontology into 'atoms' by atomic decomposition was proposed in 2011 by Del Vescovo [10].
- In 2012, Abbés characterised ontology modules in terms of patterns [11].
- In 2015, Khan and Keet began research on modularisation.

# WHAT IS LACKING?

- There is currently no foundation for modularity, i.e., a user has no guidance on :
  - how to initiate modularisation for a large ontology
  - which type of module to extract
  - which tool to use
  - how to determine if the module is of good quality
- Problems with  modularisation tools:
  - Some fail to partition large ontologies because they focus on preserving the logical properties of the modules while others lose some of the relational properties of the ontologies
  - Most generate views instead of module file output.

# DEFINE MODULARISATION

- *A Module M is a subset of a source ontology O, $M \subset O$, or M is an ontology existing in a set such that, when combined, make up a larger ontology. M is created for some use-case $u \in U$, number of $u \geq 1$, and is of a particular type $t \in T$, number of $t = 1$. t is classified by a set of distinguishing properties $\{p_1, ..., p_k\} \in P$, number of $p \geq 1$, and is created by using a specific modularisation technique $mt \in MT$, number of $mt = 1$, and has a set of corresponding evaluation metrics $\{em_1, ..., em_k\} \in EM$, number of $em \geq 1$, which is used to assess the quality of M.*

# DIMENSIONS FOR MODULARISATION

- Modules have several dimensions:
  - Use-cases: Purposes or goals for modularisation.
  - Type: A way of classifying a module.
  - Properties: Something that a module exhibits.
  - Techniques: Used to create a module.
  - Evaluation metrics: How to measure a module. Is it good or bad?

# USE-CASES

- U1: Maintenance
- U2: Automated reasoning
- U3: Validation
- U4: Processing
- U5: Comprehension
- U6: Collaborative efforts
- U7: Reuse

# TYPES

- Functional modules: A large ontology is modularised by dividing it into functional components/ subject domains.
  - T1: Ontology design patterns
  - T2: Subject domain modules
  - T3: Isolation branch modules
  - T4: Locality modules
  - T5: Privacy modules
- Structural modules: Those that have been partitioned based on structure/ hierarchy.
  - T6: Domain coverage modules
  - T7: Ontology matching modules
  - T8: Optimal reasoning modules

# TYPES

- Abstraction modules: Some detail is hidden to make a simpler view of the ontology.
  - T9: Axiom abstraction modules
  - T10: Entity type modules
  - T11: High-level abstraction modules
  - T12: Weighted modules
- Expressiveness modules: An ontology is modularised according to a specific ontology sub-language by removing some of its expressive power.
  - T13: Expressiveness sub-language modules
  - T14: Expressiveness feature modules

# PROPERTIES

- Properties of a module: Something that a module exhibits by itself.
  - 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
  - P8: Imports

# PROPERTIES

- Properties of a set of related modules: Thee properties that a set of modules exhibit altogether, and in relation to one another.
  - P9: Overlapping
  - P10: Mutual exclusion
  - P11: Union equivalence
  - P12: Partitioning
  - P13: Inter-module interaction
  - P14: Pre-assigned number of modules

# TECHNIQUES

- Graph theory approaches: Graph theory approaches are those that have been designed to be applied to the general problem of community detection.
  - MT1: Graph partitioning
  - MT2: Modularity maximisation
- Statistical approaches: Statistical approaches emphasise on using statistical equations to create ontology modules.
  - MT3: Hierarchical clustering
- Semantic approaches: The entities and axioms of the ontology are used for the modularisation approach.
  - MT4: Locality modularity
  - MT5: Query-based modularity

# TECHNIQUES

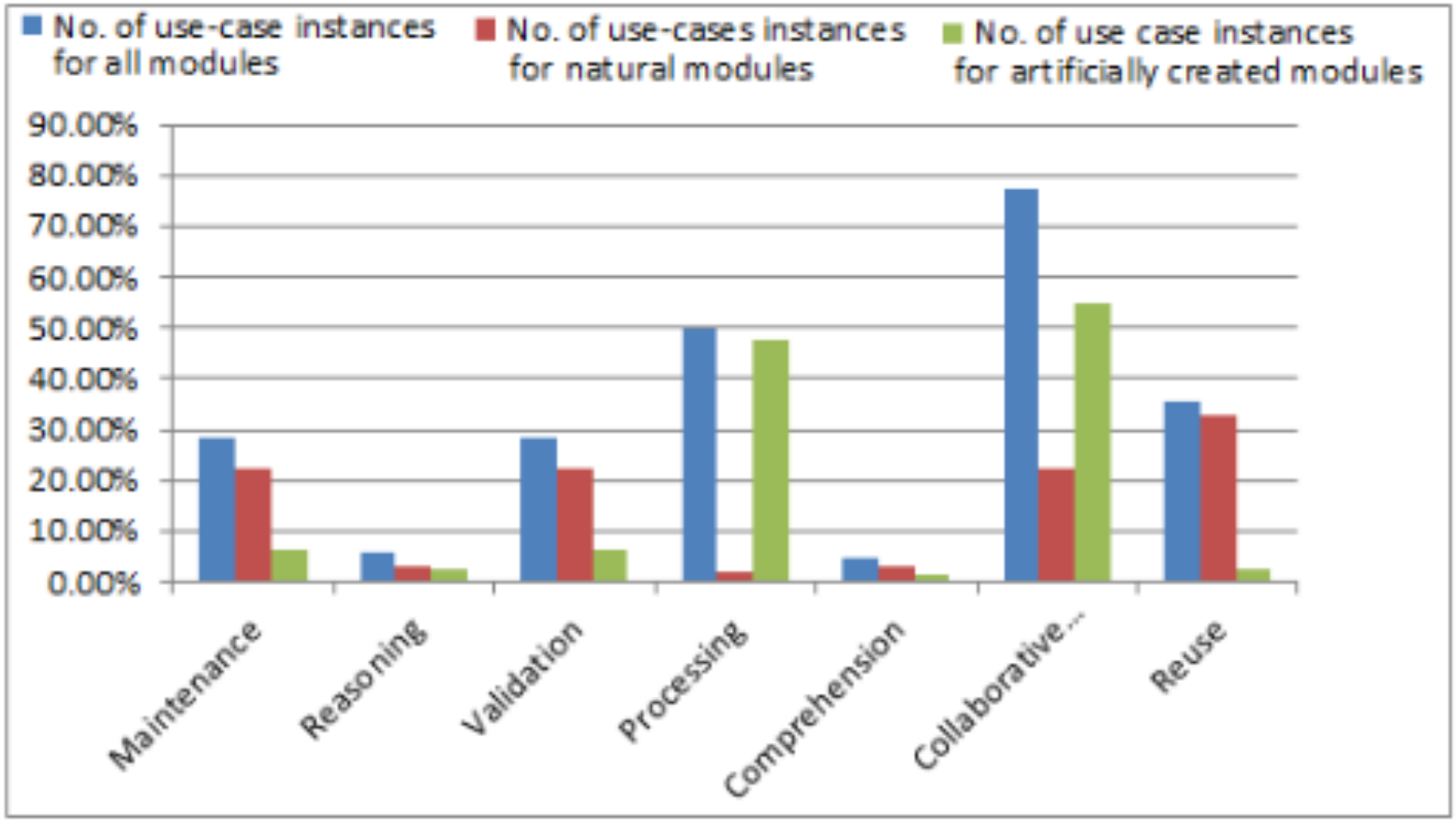
- MT6: Semantic-based abstraction
- MT7: *A priori* modularity
- MT8: Manual modularity
- MT9: Language simplification

# CLASSIFYING MODULES: AN EXPERIMENTAL EVALUATION

- How do module types differ with respect to certain use-cases?
- Which techniques can we use to create modules of a certain type?
- Which techniques result in modules with certain properties?

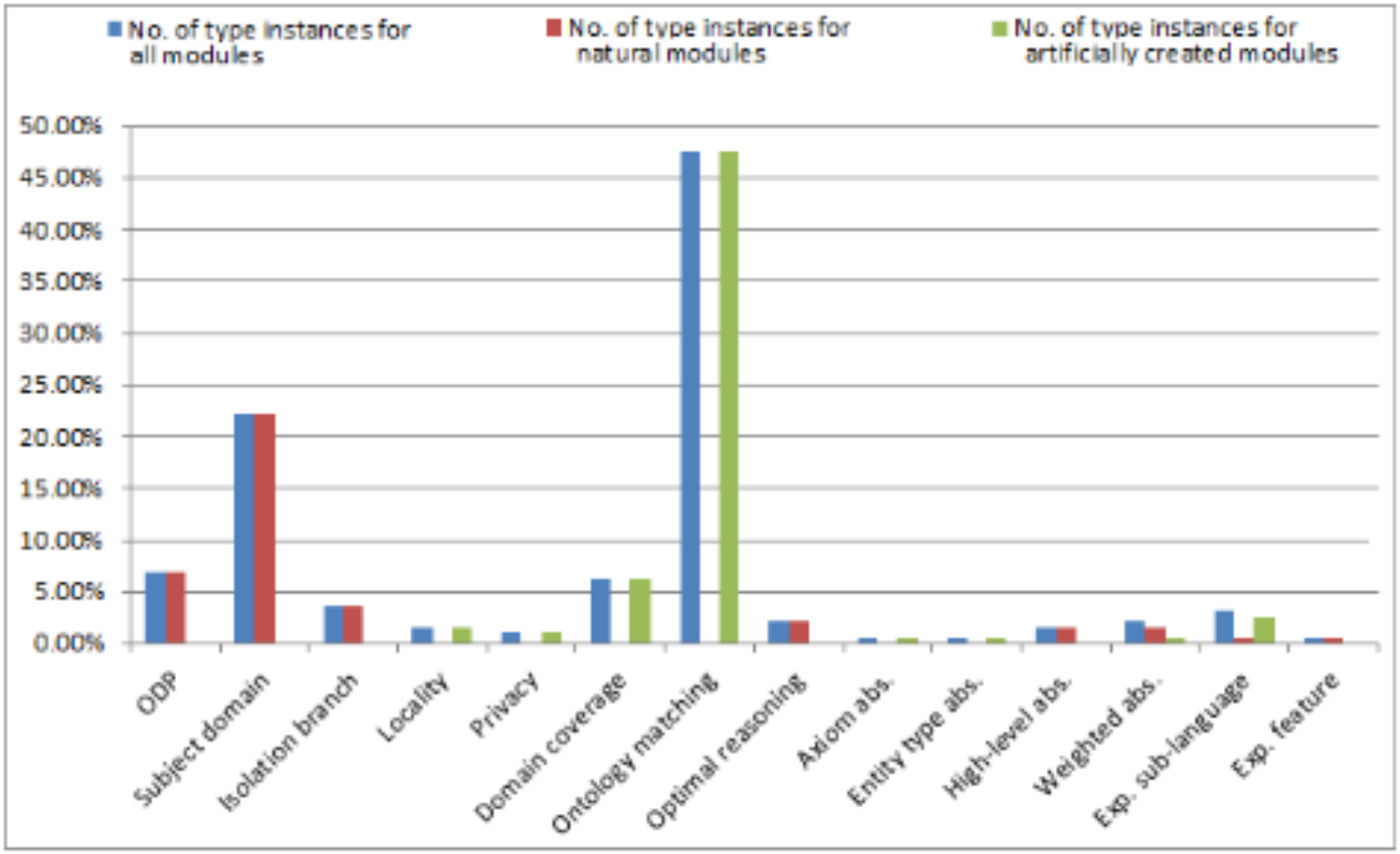
- 1. Collect ontology modules
- 2. Classify each ontology module according to its use-cases, techniques, properties, and types
- 3. Conduct a statistical analysis to determine the frequency of dimensions

# RESULTS



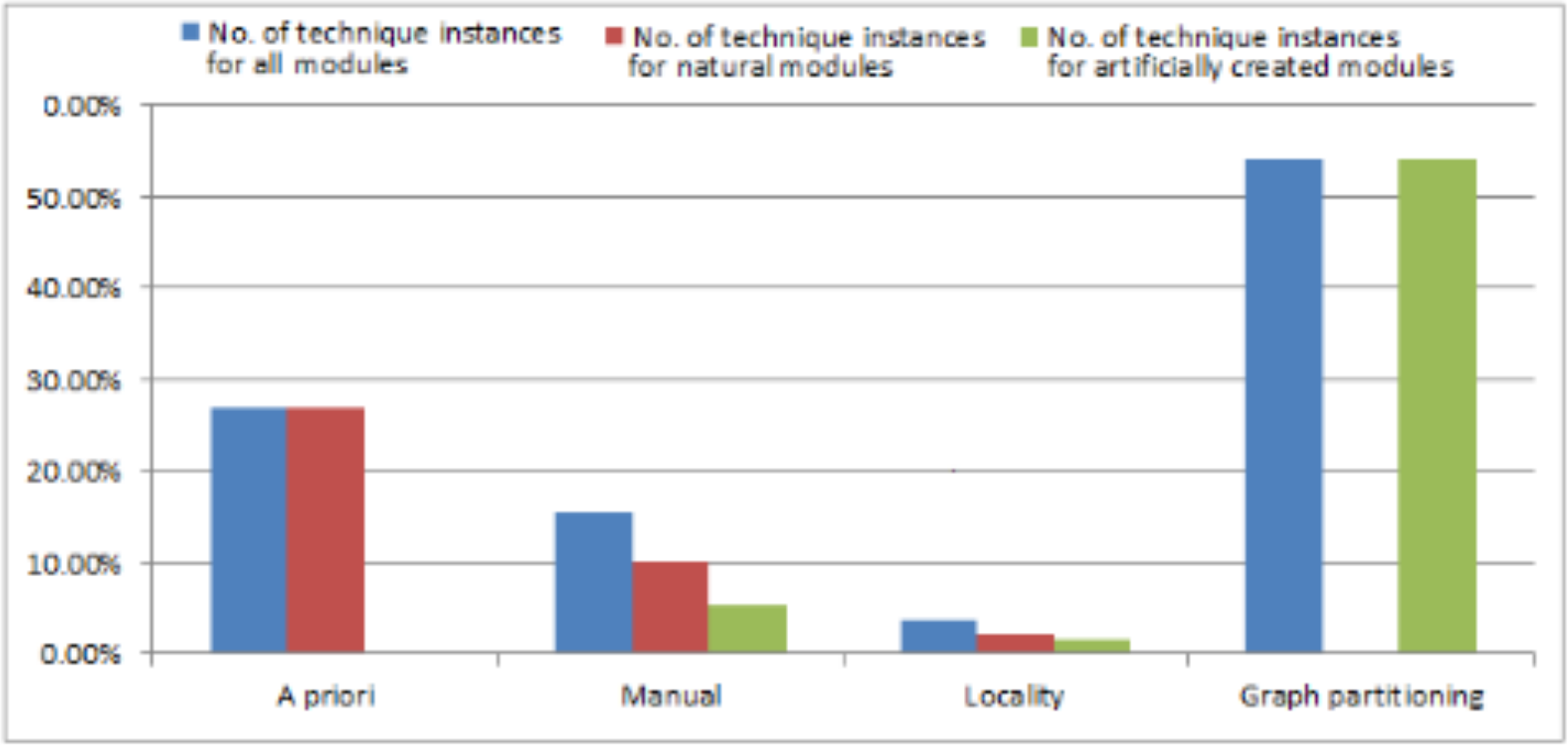

The frequency of each use-case for the set of 189 modules.

# RESULTS



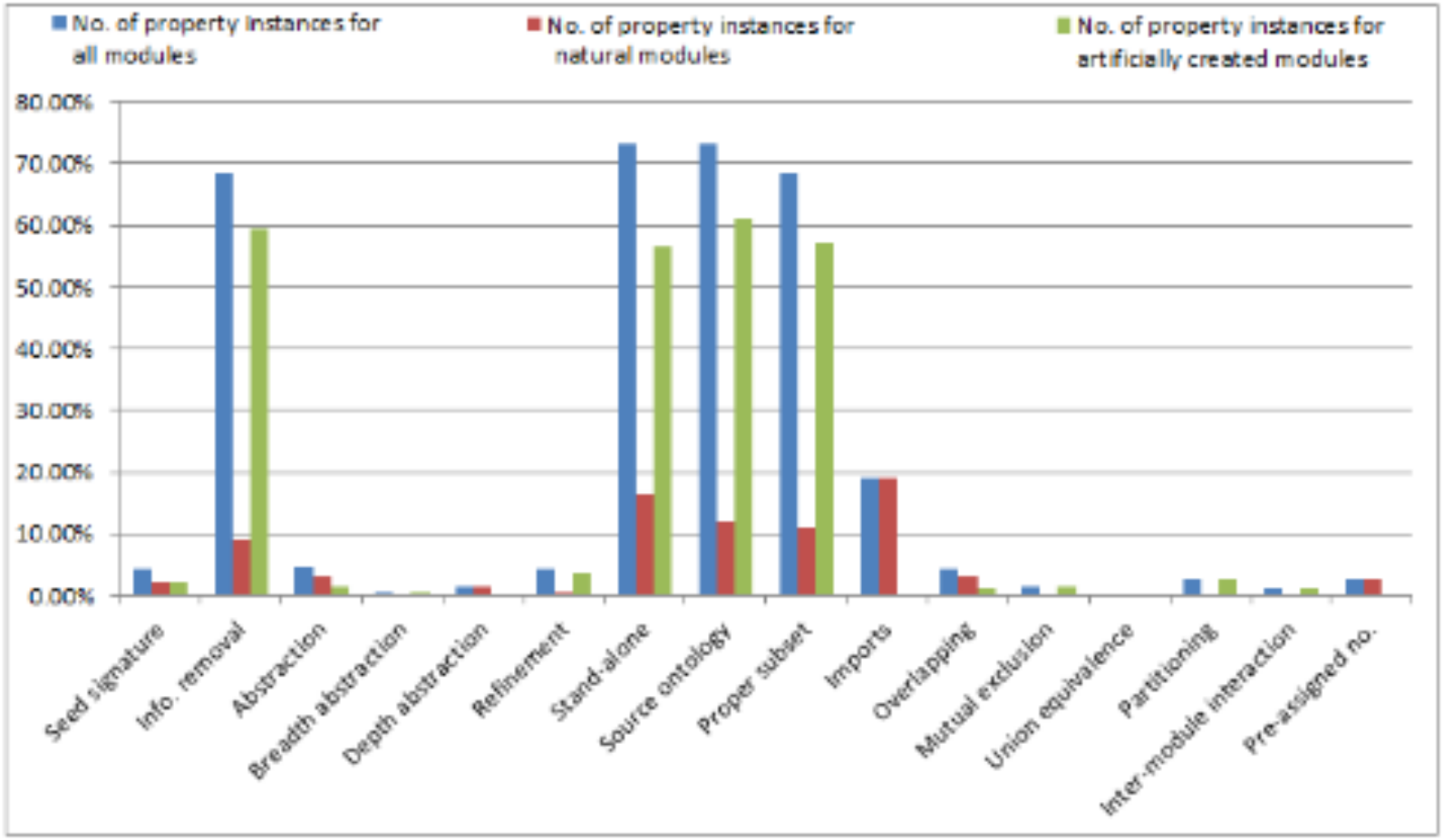

The frequency of each type for the set of 189 modules; exp. = expressiveness, abs. = abstraction.

# RESULTS




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

# RESULTS



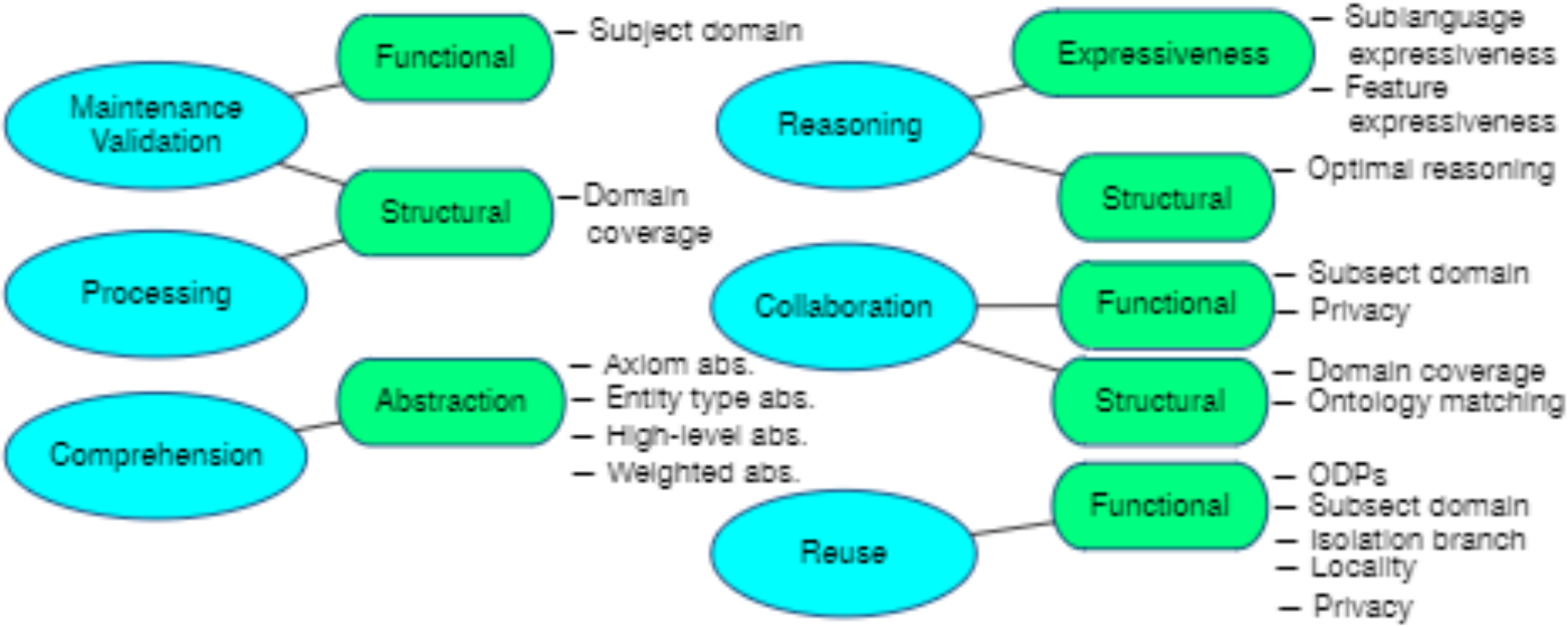

The frequency of each property among modules.

# FRAMEWORK FOR MODULARISATION

- A module's *use-case* results in modules of a certain *type*. (How do module *types* differ with respect to certain *use-cases*?)
- A module of a certain *type* is created by a modularisation *technique*. (Which *techniques* can we use to create modules of a certain *type*?)
- Modularisation *techniques* result in modules with certain *properties*. (Which *techniques* result in modules with certain *properties*?)

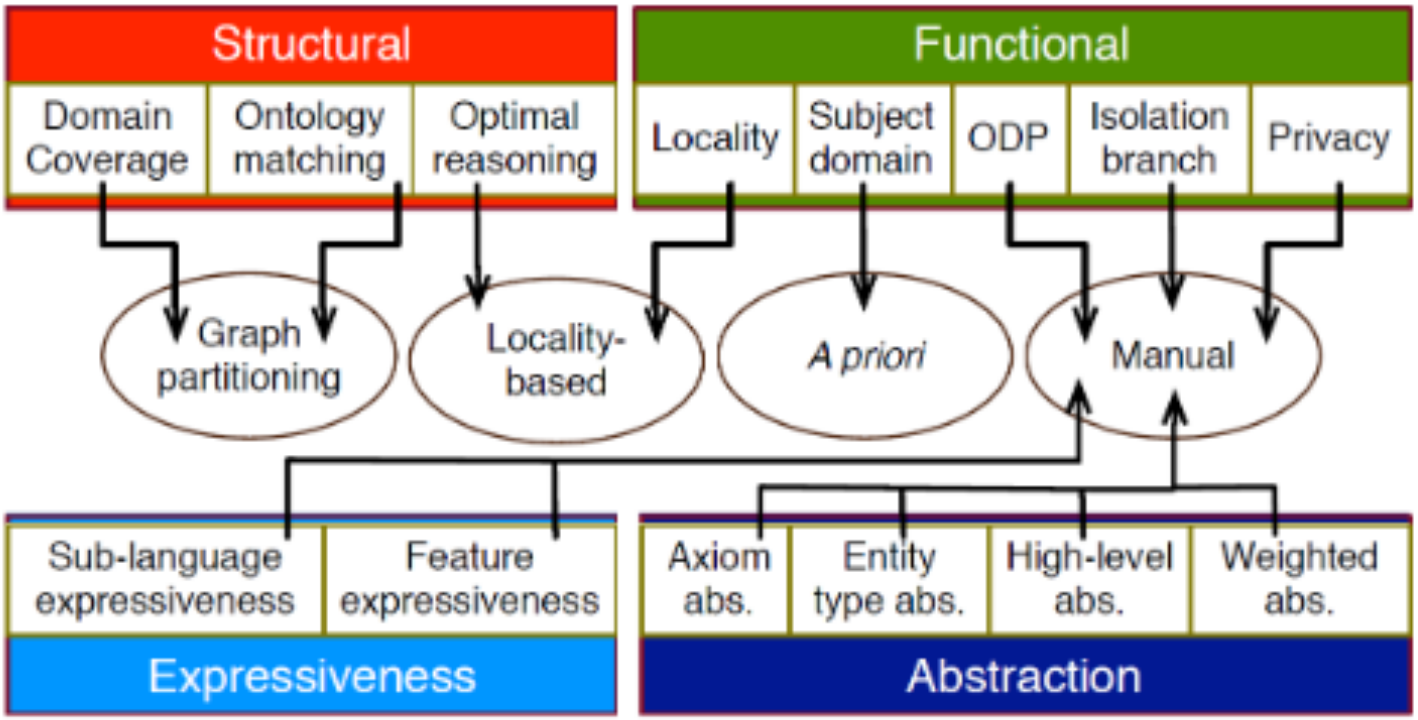

A high-level view of the framework for modularity.

# DEPENDENCIES BETWEEN



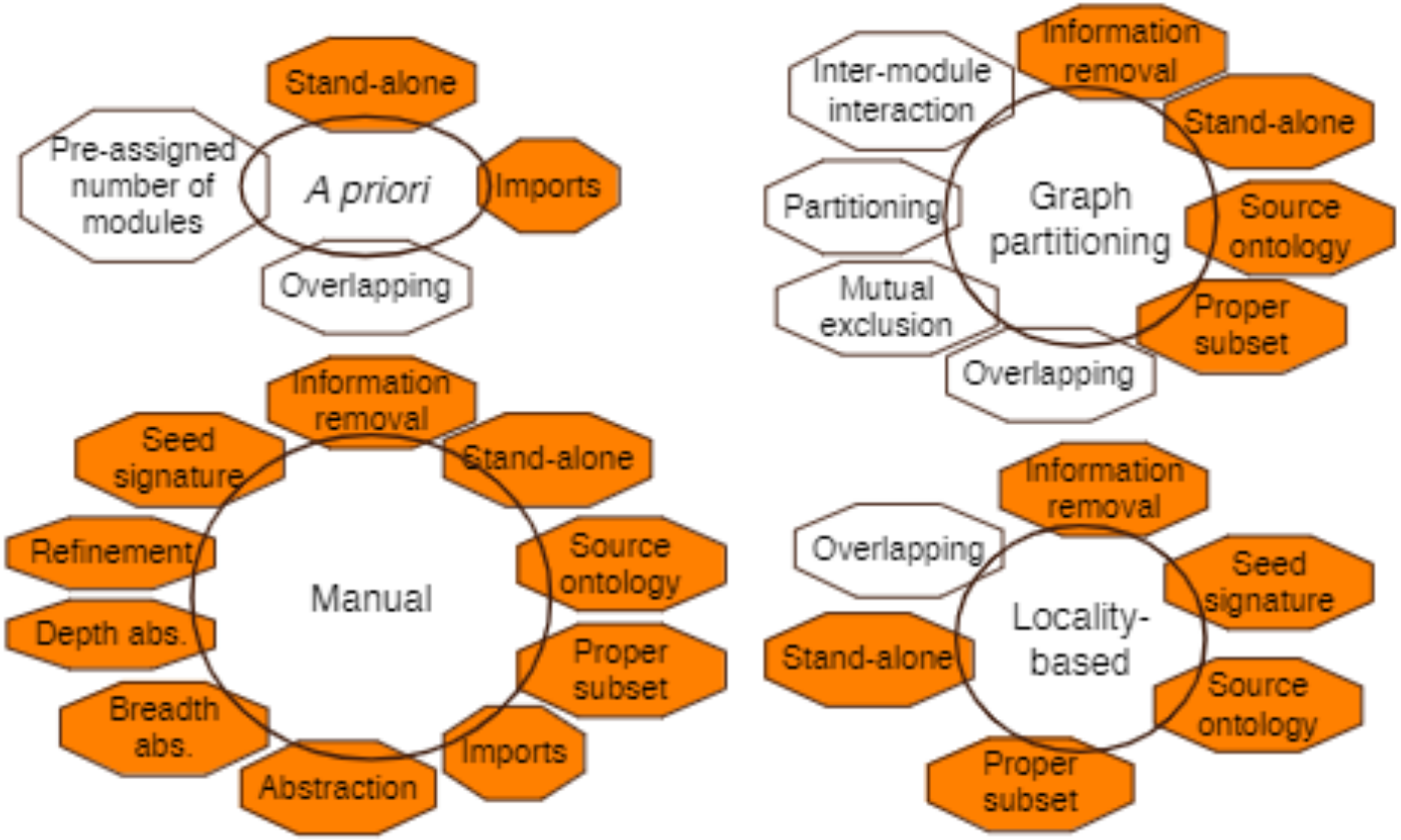

The dependencies between use-cases and module types; abs = abstraction, ODP = ontology design pattern.

# DEPENDENCIES BETWEEN TYPE AND TECHNIQUE




The dependencies between module types and techniques; abs = abstraction, ODP = ontology design pattern.

# DEPENDENCIES BETWEEN TECHNIQUE AND PROPERTY



# THEORIES AND TECHNIQUES FOR MODULARISATION

- Problems with modules: insufficient information about how to assess/ evaluate.
- Problems with tools:
- Too strict on logical properties, completeness and correctness to allow for the creation of smaller modules.
- Mainly focus locality-based, graph partitioning, and language-based techniques.

# EVALUATION METRICS

A summary of the set of evaluation metrics with their expected value range and values that are considered good. For the good values, we use a 4-point scale of small (0-0.25), medium (0.25-0.5), moderate (0.51-0.75), and large (0.75-1), and true/false values.

| **Evaluation metric** | **Value range** | **Value type** | **Good value** |
|---|---|---|---|
| EM1: Size | $i \geq 0$ | integer | - |
| EM2: Relative size | $1 \geq i \geq 0$ | decimal | small to medium |
| EM3: Appropriateness | $1 \geq i \geq 0$ | decimal | large |
| EM4: Atomic size | $1 \geq i \geq 0$ | decimal | - |
| EM5: Intra-module distance | $i \geq 0$ | decimal | - |
| EM6: Relative intra-module distance | $i \geq 0$ | integer | - |
| EM7: Cohesion | $1 \geq i \geq 0$ | decimal | small |
| EM8: Correctness | true or false | boolean | true |
| EM9: Completness | true or false | boolean | true |
| EM10: Inter-module distance | $i \geq 0$ | decimal | - |
| EM11: Coupling | $i \geq 0$ | decimal | small |
| EM12: Redundancy | $1 \geq i \geq 0$ | decimal | small to medium |
| EM13: Encapsulation | $1 \geq i \geq 0$ | decimal | large |
| EM14: Independence | true or false | boolean | true |
| EM15: Attribute richness | $i \geq 0$ | decimal | - |
| EM16: Inheritance richness | $i \geq 0$ | decimal | - |

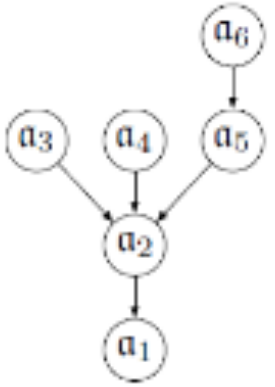
# EVALUATION METRICS

- EM4: Atomic size: An atom is a group of axioms within an ontology that have dependencies between each other [10].
- We define the atomic size as the average size of a group of inter-dependent axioms in a module.
- We formulate an equation to measure the atomic size of a module by using the number of atoms and number of axioms present in the module.
- Atomic size(M) = $|Axiom|/|Atom|$

# EVALUATION METRICS

- Consider the example in the screenshot of an atomic decomposition [10]. The number of atoms in the example is 6 and there are 7 axioms in total. The atomic size is hence $7/6 = 1.17$.
- This tells us that there is an average of 1.17 axioms per atom for the example.

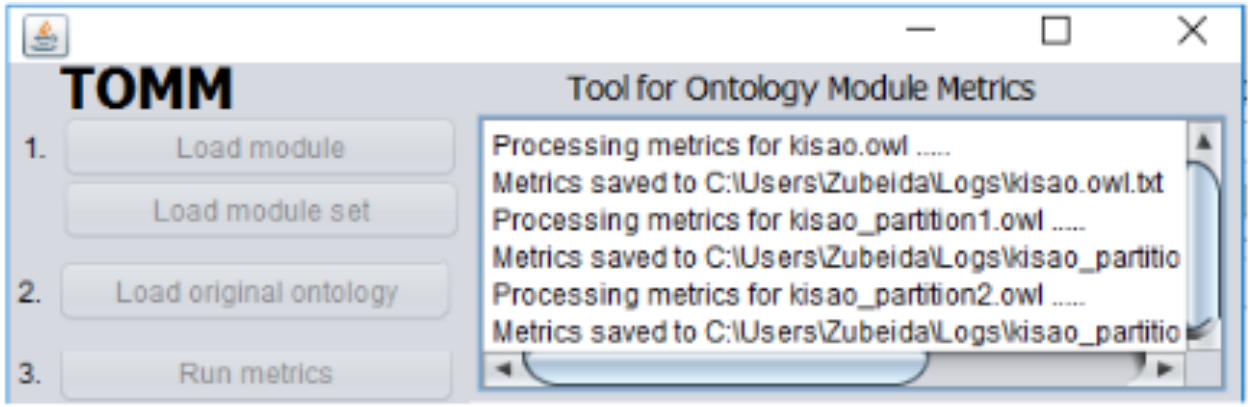
$\alpha_1 =$ 'Animal $\sqsubseteq (= 1\text{hasGender}.\top)$',
$\alpha_2 =$ 'Animal $\sqsubseteq (\geq 1\text{hasHabitat}.\top)$',
$\alpha_3 =$ 'Person $\sqsubseteq$ Animal',
$\alpha_4 =$ 'Vegan $\equiv$ Person $\sqcap \forall$eats.(Vegetable $\sqcup$ Mushroom)',
$\alpha_5 =$ 'TeeTotaller $\equiv$ Person $\sqcap \forall$drinks.NonAlcoholicThing',
$\alpha_6 =$ 'Student $\sqsubseteq$ Person $\sqcap \exists$hasHabitat.University',
$\alpha_7 =$ 'GraduateStudent $\equiv$ Student $\sqcap \exists$hasDegree.{BA, BS}'




Here the $\bot$-atoms in the AD contain the following axioms respectively: $\mathfrak{a}_1 = \{\alpha_1, \alpha_2\}$, $\mathfrak{a}_2 = \{\alpha_3\}$, $\mathfrak{a}_3 = \{\alpha_4\}$, $\mathfrak{a}_4 = \{\alpha_5\}$, $\mathfrak{a}_5 = \{\alpha_6\}$, $\mathfrak{a}_6 = \{\alpha_7\}$.

# TOMM METRICS TOOL

- It is unclear which metrics should be used to evaluate which modules? (types)
- To uncover this, we have created the Tool for Ontology Modularity Metrics (TOMM).
- TOMM is programmed with all the equations to calculate each metric.
- Stand-alone, java file.




The interface of TOMM.

# CLASSIFYING QUALITY OF MODULES: EXPERIMENTAL EVALUATION

- Determine which metrics can be used to evaluate which module types
- How to tell if a module is of good quality.

- 1. Collect a set of modules
- 2. Run the TOMM tool for each module
- 3. Conduct an analysis

# RESULTS

Averages for the structural metrics of the set of modules.

| Type | No. of modules | Size | Relative Size | No. of axioms | Appropr-iateness | Atomic size | Intra module distance | Relative intra-module distance | Cohesion |
|---|---|---|---|---|---|---|---|---|---|
| **T1** | 13 | 11.08 | 0.10 | 410.00 | 0.38 | 5.35 | 17.00 | 20.69 | 0.04 |
| **T2** | 42 | 125.62 | - | 409.19 | 0.64 | 5.18 | 16080.50 | - | 0.03 |
| **T3** | 7 | 85.43 | 0.79 | 367.86 | 0.24 | 6.31 | 8595.00 | 0.99 | 0.09 |
| **T4** | 3 | 29.00 | 0.34 | 261.67 | 0.47 | 10.10 | 853.33 | 63.65 | 0.09 |
| **T5** | 2 | 33.50 | 0.30 | 168.50 | 0.61 | 7.20 | 714.00 | 1.04 | 0.11 |
| **T6** | 10 | 417.20 | 0.21 | 922.5 | 0.49 | 3.17 | 504773.90 | 0.03 | 0.13 |
| **T7** | 90 | 2.26 | 0.02 | 14.02 | 0.009 | 1.33 | 0.97 | 2.48 | 0.15 |
| **T8** | 4 | 844.75 | 0.60 | 2166.75 | - | 3.77 | 163319.00 | 1.03 | 0.01 |
| **T9** | 1 | 94.00 | 1.00 | 884.00 | - | 2.89 | 12322.00 | - | 0.07 |
| **T10** | 1 | 103.00 | 0.56 | 257.00 | 0.99 | 4.21 | 23596.00 | 1.04 | 0.07 |
| **T11** | 3 | 279.67 | 0.51 | 715.67 | 0.89 | 3.72 | 1767.67 | 0.88 | 0.01 |
| **T12** | 3 | 158.00 | 0.41 | 582.00 | 0.02 | 5.84 | 23304.30 | 1.93 | 0.03 |
| **T13** | 6 | 305.50 | 1.00 | 1019.17 | 0.46 | 4.35 | 233449.00 | 1.00 | 0.02 |
| **T14** | 1 | 1360.00 | 0.97 | 4369.00 | - | 5.57 | 1396298.00 | 1.00 | 0.02 |

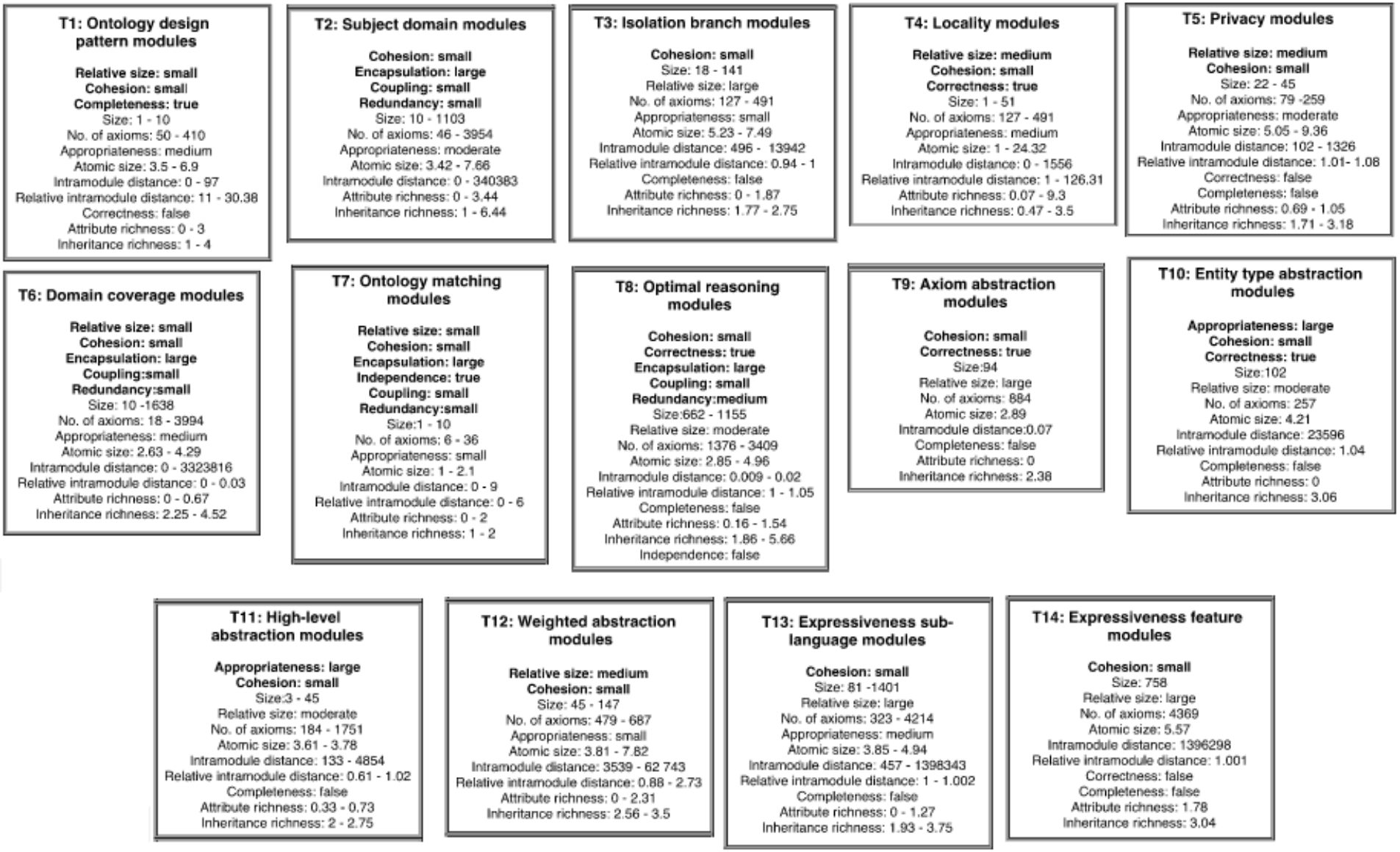
# RESULTS

Average, median, and boolean values for the logical, richness, information hiding, and relational criteria.

| | **Logical criteria** | | | | **Richness criteria** | | | |
|---|---|---|---|---|---|---|---|---|
| **Type** | **Correctness** | | **Completeness** | | **Attribute richness** | | **Inheritance richness** | |
| | **True** | **False** | **True** | **False** | **Average** | **Median** | **Average** | **Median** |
| **T1** | 0% | 100% | 100% | 0% | 0.83 | 0.67 | 1.48 | 1.00 |
| **T2** | - | - | - | - | 1.45 | 1.27 | 2.37 | 1.94 |
| **T3** | 29% | 71% | 0% | 100% | 0.84 | 0.92 | 2.30 | 2.42 |
| **T4** | 100% | 0% | 33% | 67% | 3.61 | 1.47 | 1.79 | 1.40 |
| **T5** | 0% | 100% | 0% | 100% | 0.87 | 0.87 | 2.45 | 2.45 |
| **T6** | 60% | 40% | 60% | 40% | 0.10 | 0.00 | 54.32 | 4.32 |
| **T7** | 52% | 48% | 1% | 99% | 0.05 | 0.00 | 1.19 | 1.00 |
| **T8** | 100% | 0% | 0% | 100% | 0.71 | 0.56 | 3.15 | 2.55 |
| **T9** | 100% | 0% | 0% | 100% | 0.00 | 0.00 | 2.38 | 2.38 |
| **T10** | 100% | 0% | 0% | 100% | 0.00 | 0.00 | 3.06 | 3.06 |
| **T11** | 33% | 67% | 0% | 100% | 0.58 | 0.67 | 2.44 | 2.57 |
| **T12** | 33% | 67% | 33% | 67% | 1.05 | 0.84 | 2.89 | 2.59 |
| **T13** | 83% | 17% | 0% | 100% | 0.73 | 0.76 | 2.72 | 2.49 |
| **T14** | 0% | 100% | 0% | 100% | 1.78 | 1.78 | 3.04 | 3.04 |

| | **Information hiding criteria** | | | | **Relational criteria** | | | |
|---|---|---|---|---|---|---|---|---|
| **Type** | **Encapsulation** | | **Independence** | | **Coupling** | | **Redundancy** | |
| | **Average** | **Median** | **True** | **False** | **Average** | **Median** | **Average** | **Median** |
| **T2** | 0.95 | 0.95 | 9% | 91% | 0.00 | 0.00 | 0.14 | 0.12 |
| **T6** | 0.99 | 0.99 | 70% | 30% | 0.0000256 | 0.00015 | 0.00065 | 0.00065 |
| **T7** | 1.00 | 1.00 | 100% | 0% | 0.00 | 0.00 | 0.00 | 0.00 |
| **T8** | 0.47 | 0.46 | 0% | 100% | 0.00 | 0.00 | 0.50 | 0.50 |

# DEPENDENCIES BETWEEN TYPES & METRICS



# NEW ALGORITHMS

- The current algorithms are lacking for creating certain types of modules.
- We present 5 new algorithms
  - Axiom abstraction
  - Vocabulary abstraction
  - High-level abstraction
  - Weighted abstraction
  - Feature expressiveness
- Implemented in NOMSA (Novel Ontology Modularisation SoftwAre).
- Allows users to upload an ontology file, select an algorithm, and input parameters and automatically generate a module.

# WEIGHTED ABSTRACTION ALGORITHM

- Some axioms are removed according to an absolute/relative threshold value.
- Consider the following axioms in a toy Burger ontology.

$$\begin{aligned}
\text{BeefPatty} &\sqsubseteq \text{Patty} && (1)\\
\text{Beefburger} &\equiv \text{HamBurger} && (2)\\
\text{Beefburger} &\sqsubseteq \text{Burger} && (3)\\
\text{Cheapburger} &\sqsubseteq \leq 1\ \text{hasFilling.Filling} && (4)\\
\text{Cheapburger} &\sqsubseteq \text{Burger} && (5)\\
\text{Cheese} &\sqsubseteq \text{Filling} && (6)\\
\text{Chef} &\sqsubseteq \text{Person} && (7)\\
\text{Customer} &\sqsubseteq \text{Person} && (8)\\
\text{HamBurger} &\equiv \text{Beefburger} && (9)\\
\text{HamBurger} &\sqsubseteq \text{Burger} && (10)\\
\text{HealthyBurger} &\sqsubseteq \forall \text{hasFilling.}(\text{Lettuce} \sqcup \text{Tomato}) && (11)\\
\text{HealthyBurger} &\sqsubseteq \text{Burger} && (12)\\
\text{Lettuce} &\sqsubseteq \text{Filling} && (13)\\
\text{Medium} &\sqsubseteq \text{PattyCook} && (14)\\
\text{PattyCook} &\equiv \text{Medium} \sqcup \text{Rare} \sqcup \text{WellDone} && (15)\\
\text{Rare} &\sqsubseteq \text{PattyCook} && (16)\\
\text{Sauce} &\sqsubseteq \text{Filling} && (17)\\
\text{Tomato} &\sqsubseteq \text{Filling} && (18)\\
\text{WellDone} &\sqsubseteq \text{PattyCook} && (19)\\
\text{WhiteBun} &\sqsubseteq \text{BurgerBun} && (20)\\
\text{WholeWheatBun} &\sqsubseteq \text{BurgerBun} && (21)\\
\text{WholeWheatBun} &\sqsubseteq \neg\text{WhiteBun} && (22)\\
\text{Func(hasBun)} & && (23)\\
\exists\ \text{hasBun.}\top &\sqsubseteq \text{Burger} && (24)\\
\top &\sqsubseteq \forall \text{hasBun.BurgerBun} && (25)\\
\exists\ \text{hasPatty.}\top &\sqsubseteq \text{Burger} && (26)\\
\top &\sqsubseteq \forall \text{hasPatty.Patty} && (27)\\
\exists\ \text{hasPattyCook.}\top &\sqsubseteq \text{Patty} && (28)\\
\top &\sqsubseteq \forall \text{hasPattyCook.PattyCook} && (29)\\
\text{MarthasBurger} &\neq \text{MyBurger} && (30)\\
\text{MarthasBurger} &: \text{Burger} && (31)\\
\text{MyBurger} &: \text{Beefburger} && (32)\\
\text{MyBurger} &: \text{Burger} && (33)\\
\text{MyBurger} &: \text{Beefburger} && (34)\\
\text{ChefRose} &: \text{Chef} && (35)\\
& \text{cookedBy(MyBurger, ChefRose)} && (36)
\end{aligned}$$

# WEIGHTED ABSTRACTION ALGORITHM

- Let us assume we wish to create a module whereby we remove 25% of the entities. To achieve this, we set the threshold value to 25%.
- First, we we weigh each class in the ontology with its number of referencing axioms and we store both the number of referencing axioms and each class in two arrays with corresponding indices.
- We sort the weight array values from low to high and the class array such that it matches the weight array.
- A limit variable is calculated as the product of the threshold percentage (.25) and the number of classes in the ontology (21) which is rounded off to a value of 5.
- The classes with the 5 lowest values are removed; these are deemed less-important than the rest and are to be removed due to having the least number of referencing axioms in the ontology.

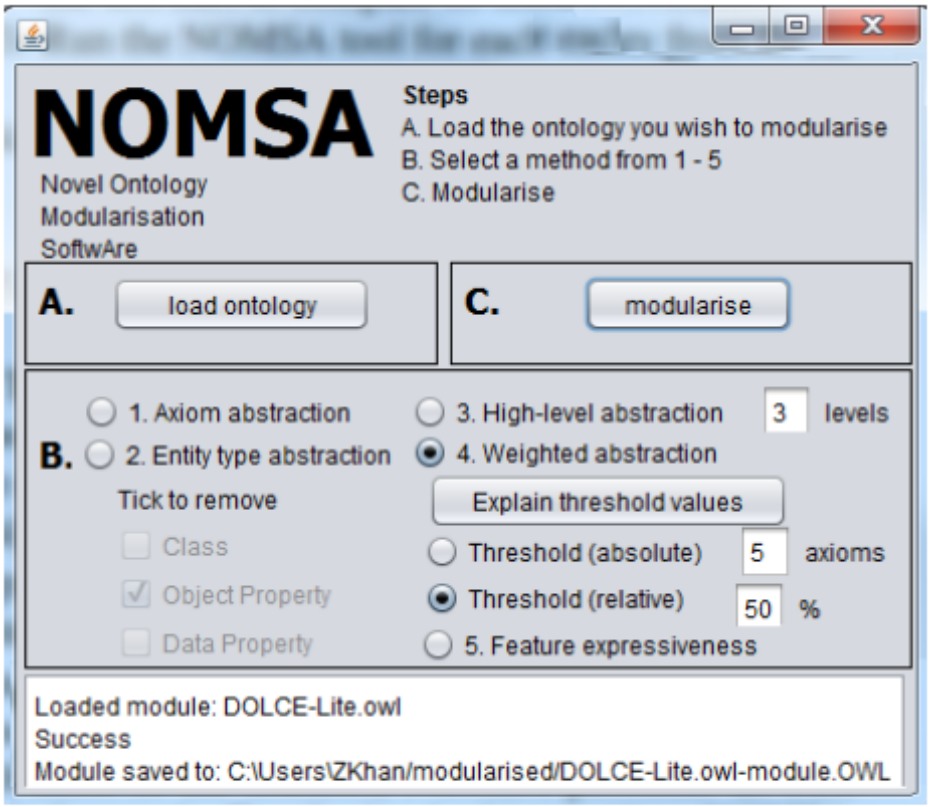
# WEIGHTED ABSTRACTION ALGORITHM

- The classes in bold font are those that are to be removed because they have the least number of referencing axioms.

| | | | | | |
|---|---|---|---|---|---|
| WhiteBun | **2** | Medium | 3 | Patty | 4 |
| Customer | **2** | Lettuce | 3 | BeefBurger | 4 |
| Cheese | **2** | HealthyBurger | 3 | BurgerBun | 4 |
| Sauce | **2** | BeefPatty | 3 | Hamburger | 4 |
| Chef | **2** | Tomato | 3 | Filling | 5 |
| WholeWheatBun | **2** | WellDone | 3 | PattyCook | 6 |
| Person | 3 | Rare | 3 | Burger | 7 |

# NOMSA

- Algorithms implemented in NOMSA (Novel Ontology Modularisation SoftwAre).
- Allows users to upload an ontology file, select an algorithm, and input parameters and automatically generate a module.




The interface of NOMSA.

# NOMSA EVALUATION

mparison of three features of several modularisation tools against NOMSA and the average running times of the respective algorithms for the test tologies (excluding the time of manual modularisation tasks such as loading the ontology and setting the parameters).

| | Level of interaction | Algorithm complexity | Technique | Time (seconds) |
|---|---|---|---|---|
| OOP orithm 1 | semi-automatic | quadratic | locality-based | 1 |
| OOP orithm 2 | automatic | quadratic | graph-partition | 6 |
| L module actor | semi-automatic | quadratic | locality-based | 1 |
| OMPT | user-driven | unknown | query-based | - |
| tégé orithm 1 | semi-automatic | unknown | locality-based | 1 |
| tégé orithm 2 | automatic | unknown | language-based | 1 |
| tégé orithm 3 | semi-automatic | unknown | locality-based | 1 |

| | Level of interaction | Algorithm complexity | Technique | Tim (sec |
|---|---|---|---|---|
| **Protégé Algorithm 4** | semi-automatic | unknown | language-based | 1 |
| **TaxoPart** | automatic | linear | graph-partition | 15 |
| **PATO** | automatic | unknown | graph-partition | 16 |
| **NOMSA AxAbs** | automatic | linear | semantic-based abstraction | 3 |
| **NOMSA VocAbs** | automatic | linear | semantic-based abstraction | 2 |
| **NOMSA HLAbs** | automatic | quadratic | semantic-based abstraction | 2 |
| **NOMSA WeiAbs** | automatic | linear | semantic-based abstraction | 4 |
| **NOMSA FeatExp** | automatic | quadratic | language-based | 3 |

# SUMMARY OF CONTRIBUTIONS FOR MODULARISATION

- Provided a new, exhaustive definition for modularisation
- Identified dimensions for modularisation
- Created dependencies between modularity dimensions
- Determine how to evaluate a module
- Improve modularisation techniques

# CONCLUSION

- Solved the problems that:
  - 1. Existing techniques are not sufficient for modularisation
    - performing a classification on a set of ontology modules to determine which techniques are lacking in tools
    - by designing and implementing novel algorithms to perform modularisation
  - 2. A user has no guidance on how to initiate modularisation for an ontology
    - by identifying dimensions of modularity
    - classifying a set of modules with dimensions
    - linking various dimensions together to create dependencies
  - 3. How to determine if the module is of good quality
    - by identifying new and existing evaluation metrics
    - providing equations for those that did not have any equations
    - the development of a tool to compute the metrics for an ontology module
    - performing an investigation to determine which metrics can be used to measure which module types

# CONCLUSION

- We provided a foundation for modularity encompassing:
  - a framework for modularity
  - new algorithms for modularisation
  - a method and tool for evaluating the quality of a module
- The foundation successfully solves several problems concerning modularity.

# REFERENCES


- [1] van der Vet, Paul E., and Nicolaas JI Mars. "Bottom-up construction of ontologies: the case of an ontology of pure substances." *Memoranda informatica* (1995): 95-35.
- [2] Rector, Alan L. "Modularisation of domain ontologies implemented in description logics and related formalisms including OWL." *Proceedings of the 2nd international conference on Knowledge capture*. ACM, 2003.
- [3] Grau, Bernardo Cuenca, et al. "Modularity and Web Ontologies." *KR*. 2006.
- [4] Stuckenschmidt, Heiner, and Michel Klein. "Structure-based partitioning of large concept hierarchies." *International semantic web conference*. Springer, Berlin, Heidelberg, 2004.
- [5] Noy, Natalya F., and Mark A. Musen. "Specifying ontology views by traversal." *International Semantic Web Conference*. Springer, Berlin, Heidelberg, 2004.
- [6] Keet, C. Maria. "Using abstractions to facilitate management of large ORM models and ontologies." *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*. Springer, Berlin, Heidelberg, 2005.
- [7] Grau, B. Cuenca, et al. "Modular reuse of ontologies: Theory and practice." *Journal of Artificial Intelligence Research* 31 (2008): 273-318.
- [8] Parent, Christine, and Stefano Spaccapietra. "An overview of modularity." *Modular ontologies*. Springer, Berlin, Heidelberg, 2009. 5-23.
- [9] d'Aquin, Mathieu, et al. "Criteria and evaluation for ontology modularization techniques." *Modular ontologies*. Springer, Berlin, Heidelberg, 2009. 67-89.
- [10] Del Vescovo, Chiara, et al. "The modular structure of an ontology: Atomic decomposition." *Twenty-Second International Joint Conference on Artificial Intelligence*. 2011.
- [11] Abbes, Sarra Ben, et al. "Characterizing modular ontologies." 2012.

# THANK YOU

- http://www.thezfiles.co.za/modularisation/
- http://www.thezfiles.co.za/Modularity/TOMM.zip