A Generic and Customizable Genetic Algorithms-based Conceptual Model Modularization Framework

Syed Juned Ali, Jan Michael Laranjo, Dominik Bork

A Generic and Customizable Genetic Algorithms-based Conceptual Model Modularization Framework

Syed Juned Ali[1[0000-0002-0710-8052]] Jan Michael Laranjo and Dominik Bork[1[0000-0001-8259-2297]]

TU Wien, Business Informatics Group, Vienna, Austria
{syed.juned.ali, dominik.bork}@tuwien.ac.at

Abstract. Conceptual models need to be comprehensible and maintainable by humans to exploit their full value in faithfully representing a subject domain. Modularization, i.e. breaking down the monolithic model into smaller, comprehensible chunks has proven very valuable to maintain this value even for very large models. The quality of modularization however often depends on application-specific requirements, the domain, and the modeling language. A well-defined generic modularizing framework applicable to different modeling languages and requirements is lacking. In this paper, we present a customizable and generic multi-objective conceptual models modularization framework. The multi-objective aspect supports addressing heterogeneous requirements while the framework's genericity supports modularization for arbitrary modeling languages and its customizability is provided by adopting the modularization configuration up to the level of using user-defined heuristics. Our approach applies genetic algorithms to search for a set of optimal solutions. In this paper, we present the details of our Generic Genetic Modularization Framework with a case study to show $i)$ the feasibility of our approach by modularizing models from multiple modeling languages, $ii)$ the customizability by using different objectives for the modularization quality, and, finally, $iii)$ a comparative performance evaluation of our approach on a dataset of ER and ECore models.

Keywords: Modularization · Genetic Algorithm · Generic Framework

1 Introduction

Conceptual modeling enables domain understanding and communication among stakeholders and is integral to enterprise and information systems engineering [30]. The usefulness of any conceptual model for humans is inversely proportional to the size of the model depicted. Models with more than 30 nodes/edges are considered to be already challenging for easy comprehension. The more relationships in a model, the less comprehension is possible due to the accompanying increase in complexity [41]. Therefore, the increased size and complexity can make models *cognitively intractable* [12].

Clustering or modularizing conceptual models[1] into smaller chunks provides benefits in that very large models become easy to communicate, validate, and maintain [41].

[1] Note, we use graph clustering/partitioning and modularization interchangeably in this paper.

However, to be truly effective in improving human understanding, clustering approaches must be based on sound principles of human information processing [25].

Due to the NP-hardness of the modularization problem, search-based algorithms treat modularization as an optimization problem over an objective function [28] using techniques like genetic algorithms (GA) [9,26]. However, several limitations exist, such as choosing the heuristic for modularization cannot be generalized well given the diversity of conceptual models designed, domains modeled, modeling languages used, and requirements being addressed. Finally, several steps are involved in GA-based solutions i.e., selection, crossover, and mutation which should be customizable depending on the modularization case. Therefore, there is a need for a modularization framework that $i)$ can be adapted to multiple modeling languages, $ii)$ provides multiple heuristics and allows extending the set of heuristics; and $iii)$ allows GA parameter (re-)configuration based on the modeler's requirements.

In order to achieve modularization genericity in terms of modeling languages, we need a generic representation of conceptual models that can be used as input for the modularization techniques. Knowledge graphs (KGs) can encapsulate knowledge in a graph structure, creating new processing possibilities, such as knowledge reasoning. KGs provide a foundation for data integration, fusion, analytics, and sharing [34] based on linked data and semantic metadata. Conceptual models can be treated as graphs with nodes and edges capturing conceptual model-specific information. We can transform a general conceptual model into KGs and utilize the benefits of a KG-based representation of conceptual models. Therefore, we can use knowledge graphs as a generic representation of conceptual models that captures the domain semantics of the model and the model's graph structure.

Once we have a generic conceptual model representation, we need to drive the modularization toward optimization objectives suitable for the modeling requirements. Multiple objectives can measure the modularization quality. Moody and Flitman [25] proposed nine principles for decomposing data models, including the number and the size of cognitively manageable clusters, which are also valid for conceptual models. Moreover, users should also be able to define objectives based on their requirements. E.g., a custom objective to optimize the modularization for the number of abstract classes in a module. Furthermore, using an objective as a combination of several objectives can be beneficial.

To that end, we present the main contributions of this paper as follows - $i)$ we introduce the Generic Genetic Modularization Framework (GGMF) for conceptual models that provides a multi-objective, generic, customizable framework for modularizing conceptual models; $ii)$ we provide a UI platform for modeling experts to modularize conceptual models where users can upload a conceptual model, define configuration parameters for the GA-based optimization, and select the objectives for which the GA should optimize. Furthermore, the user can weigh each objective based on its expected importance. Therefore, the user can execute the optimization as a multi-objective optimization or aggregate the objective into a single objective in a weighted or unweighted manner. Finally, (iii), we present a comparative analysis of our approach with existing approaches. We evaluate our framework based on the following research questions -

RQ1- Is developing a modularization framework that supports multiple modeling languages feasible?
RQ2- How does the framework support modularization for different requirements?
RQ3- How does the modularization perform compared to existing approaches?

Note that the focus of our paper is the genericity and customizability provided by GGMF. Therefore, we do not focus on the details of the optimization mechanism of genetic algorithms and use the jenetics [2] library. The rest of the paper is structured as follows: Section 2 presents the foundational concepts involved in our work. Section 3 discusses the relevant literature while Section 4 introduces our framework and the developed tool support. Section 5 evaluates our approach. We discuss the threats to validity in Section 6 and finally, we conclude this paper in Section 7.

2 Foundations

2.1 Modularization

Modularization in conceptual modeling concerns the decomposition of a monolith, potentially overarching model, into smaller, more comprehensive model chunks—called modules. The module components depend on the intended purpose of modularization while fulfilling the definition of a module. E.g., if the purpose of the module is solely to answer a query, then it should only be composed of the necessary concepts and relations that can answer the considered query [21].

2.2 Conceptual Knowledge Graphs

Knowledge Graphs (KGs) represent a collection of interlinked descriptions of entities—e.g., objects, events, and concepts. KGs provide a foundation for data integration, fusion, analytics, and sharing [34] based on linked data and semantic metadata. Recently, a generic approach has been proposed to transform arbitrary CMs into CKGs called *CM2KG* [36]. However, CM2KG focuses only on the element labels and metamodel information. Ali et al. [3] define the notion of Conceptual Knowledge Graphs (CKGs) by adding semantics from external sources such as language metamodel, domain, and foundational ontologies to the KG-based representation. These enriched semantics can be used as information sources for many tasks. In our work, CKGs act as the intermediary representation of CMs based on which the modularization is performed.

2.3 Genetic Algorithms

Genetic algorithms (GA) are randomized search-based optimization algorithms inspired by the principles and mechanics of natural selection and natural genetics [15]. GAs map the optimization problem into the concepts involved in GA-based search and apply the search to find the optimal (a set of) solutions. We briefly describe the involved concepts.

Genetic Encoding. The genetic encoding i.e., the *genotype* is the representation of the optimization problem that the GA uses as input for executing the optimization. In modularization, a *genotype* could be a binary string or an array where each element represents the module assignment of a node in the graph. A *gene* is a specific element within a *genotype*, e.g., a gene would represent the module assignment of a specific node in the graph. The value stored in a gene is called an *allele*. A *chromosome* is a collection of genes that forms a complete genetic representation. It represents an individual or candidate solution in the genetic algorithm, e.g., a complete set of module assignments for all nodes in the graph. Commonly used encoding schemes are binary encodings, tree encodings, or matrix encodings [19]. A *population* is a set of currently present chromosomes. Linear Linkage Encoding (LLE) is a genetic encoding for grouping problems, such as graph coloring or data clustering. It represents a problem solution as a fixed-length chromosome, which encodes the assignment of elements to a group. This encoding ensures there is only one representation for a grouping solution [20], thereby mitigating redundancy and isomorphism.

Fitness Functions and Objectives. The fitness function is used to determine the fitness value of a chromosome [15]. The fitness value quantifies the quality of chromosome w.r.t a fitness function. In the context of modularization, the fitness function would evaluate the quality of a chromosome by considering factors such as intra-module similarity and inter-module dissimilarity. GA can use multiple objectives during optimization. However, multi-objective problems often have conflicting objectives, i.e., maximizing one objective may lead to minimizing another, e.g., maximizing cohesion and minimizing coupling. Pareto optimal solution sets or Pareto front is a solution for conflicting objectives. This set consists of possible Pareto optimal solutions.

We explain modularity and MQ in Eq. 1.

$$\begin{aligned} modularity &= \frac{1}{2m}\sum_{i,j}\left(A_{ij} - \frac{k_i k_j}{2m}\right)\delta(c_i, c_j) \\ MQ &= \sum CF_i \text{ where} \\ CF_i &= \frac{2\mu_i}{2\mu_i + \epsilon_i} \end{aligned} \tag{1}$$

For a graph G = (V,E) where V is the set of nodes in the graph and E is the set of edges. A is the adjacency matrix of a graph with n nodes and m edges. Modularity is defined in Eq. 1, where A_{ij} is an element of the adjacency matrix A, $\frac{d_i d_j}{2m}$ is the expected number of edges between nodes i and j, d_i is the degree of node i, c_i is the community to which node i belongs and δ is the Kronecker delta function that returns 1 if $c_i = c_j$ and 0 otherwise. The MQ score is obtained from the sum of all module factors. CF$_i$ denotes the module factor for each module *i*, where μ_i represents the count of intra-module edges and ϵ_i represents inter-cluster between two modules [28].

Selection. Once we have a genetic encoding, the selection step selects a subset of chromosomes from the current population for reproduction based on their quality defined by a fitness function and creating the offspring [19]. The fitness function guides the selection process by favoring individuals with higher fitness scores, increasing the probability of their genetic material being passed on to the next generation. The selected individuals for the reproduction process are also called the mating pool. There are different types of selection operators in GA. The most common variations for single-objective GA are roulette wheel section, rank selection, and tournament selection [19]. The roulette wheel selects chromosomes randomly for the reproduction process [15] based on the number of copies of each chromosome present in the population. Tournament selector uses a tournament process, where randomly selected individuals compete against each other in pairs or larger tournaments. The winner is the individual with the highest fitness, and the winner is added to the mating pool with chromosomes with higher average fitness compared to the population's average fitness. The tournament is repeated until the desired size for the mates is reached.

Alterers. *Crossover* combines the genetic material from two parent chromosomes to create the offspring. In the context of modularization, the crossover could involve combining the module assignments of two parent chromosomes to generate a new set of module assignments for the offspring. *Mutation* is an operator that introduces chromosome alterations. It helps to introduce diversity in the population and explore new solutions. In modularization, the mutation could involve randomly reassigning the module of a node to a different cluster.

3 Related Works

We now discuss the different modularization approaches, separating GA and non-GA approaches, and the modularization metrics proposed in the literature.

3.1 Conceptual Models Modularization

Non-GA-based Approaches. Stuckenschmidt and Klein [37] propose a method that clusters models based on the structure of the class hierarchy for real-world ontologies like SUMO and the NCI cancer ontology. Saruladha et al. [33] propose two new neighbor-based structural proximity measures, TNSP and DNSP to decompose ontologies into disjoint clusters. They consider concept pairs with common neighbors for clustering. Doran et al. [11] present a language-independent ontology module extraction approach implemented as ModTool. Andritsos et al. [4] present LIMBO, a hierarchical clustering algorithm based on minimizing information loss that will be incurred on merging two nodes in a cluster, in the context of a software system. In [24] a hierarchical clustering-based weighted linkage clustering (WLC) approach is presented. They associate a new feature vector using the feature vectors of a set of cluster entities. Two entities are merged based on their types, globals, and routines. Hence, the new feature vector correctly reflects relationships between the entities. Pourasghar et al. [28] present a GMA (Graph-based Modularization Algorithm) modularization technique. Their work uses relationship depth to compute the similarity between model entities. Furthermore, they propose several metrics to evaluate the modularization quality that uses structural features of the modularized model.

Metrics. Sarkar et al. [32] propose a set of metrics for determining the quality of modularization of large-scale object-oriented software using intra-module dependencies, the modules' APIs, and object-oriented inter-module dependencies, e.g., inheritance. Using model slicing, Bae et al. [5] modularize UML metamodels. The Model Slicing approach uses vital elements to generate the modularized metamodel. A key element is a model element of M. The slicing is executed in two phases using the edges of the multigraph and the key elements to determine the slices. Hinkel et al. [16] apply the modularization quality metrics proposed by [32] on metamodels. Like class diagrams, metamodels can be organized into packages, making the metrics appealing for application on metamodels. Based on the results of this work, they propose an entropy-based approach. Their metric measure the degree of classes that are stored in different packages [17].

GA-based approaches. Bork et al. [9] introduce the ModulER tool for modularizing entity-relationship models. The tool follows a meta-heuristic search approach using genetic algorithms. Multiple objectives, defined as fitness functions, aim to minimize or maximize specific properties of the modularization of each individual, resulting in a Pareto Set of optimal solutions [9].

Mu et al. [26] propose a hybrid genetic algorithm (HGA) using a heuristic based on edge contraction and vectorization techniques to generate feature-rich solutions and subsequently implant these solutions as seeds into the initial population. Finally, a customized genetic algorithm (GA) improves the solution quality. Tabrizi et al. [38] combine hierarchical clustering with genetic algorithms, where they first modularize the model using GA and then further improve the solution using hierarchical clustering. Bavota et al. [6] propose Interactive Genetic Algorithms (IGAs) to integrate the developer's knowledge in a re-modularization task. Their approach uses automatically evaluated fitness functions and a human evaluation to penalize cases where a developer considers module assignments meaningless.

3.2 Modularization Metrics

Metrics are needed to quantify the modularization quality and different metrics have been proposed in the past. Some metrics use the characteristics between modules and relationships between elements in modules. Others utilize concepts from information theory or network theory. Lastly, some metrics consider specific conceptual models' properties and are only tailored for a specific conceptual modeling language.

Moody et al. [25] propose nine principles for decomposing data models using network theory and cognitive science principles. Sarkar et al. [32] present metrics for cohesion and coupling between modules by defining an entropy metric that measures the extent to which classes are used together and should be clustered. The properties or structure of specific conceptual models can also indicate which elements belong together in a module. Prajapati et al. [29] modularize by properly distributing classes among various packages in a model with minimum perturbation. Hinkel et al. [16] adapt some of the proposed metrics in the context of metamodels. Hinkel et al. [17] propose an entropy-based modularization metric that quantifies class distribution in packages [17]. Dazhou et al. [18] use weights for relations in UML classes. Each UML class relation gets weight assigned to build weighted class dependence graphs. Singh et al. [35] similarly assign dependency weights to BPMN models.

In network theory, many metrics can be used to measure modularity. The importance of an edge is captured by determining the shortest paths of all vertex pairs that go through the edge. The edges have higher values in communities as there are more shortest paths between the vertex pairs. This metric is called Edge Betweenness Centrality [13]. Similarly, Vragovic et al. [42] use the idea that neighbors in communities are close to each other, even when they are removed. They introduced the concept of loop coefficients. It takes the number of smallest loops running through a node into consideration. High coefficients indicate main nodes in a community, whereas low values indicate peripheral nodes in communities. Newman [27] defines a spectral method for modularity optimization. His approach uses Eigenvectors to express modularity characteristics and optimize accordingly.

In the related work, we note a need for a customizable and generic modularization framework, i.e., applicable to multiple modeling languages, extensible to user-defined metrics. Multiple modularization metrics available can be used depending on the requirements. With this motivation, we propose our generic and customizable conceptual model modularization framework using genetic algorithms.

4 The Generic Genetic Modularization Framework (GGMF)

In the following, we introduce our proposed Generic Genetic Modularization Framework (GGMF) in detail. We show in Fig. 1 the end-to-end approach from a monolithic conceptual model to a modularized one. The conceptual model is first transformed into a CKG, which the GA then uses to produce a Pareto Set of optimal solutions based on the user's selected objectives and requirements.

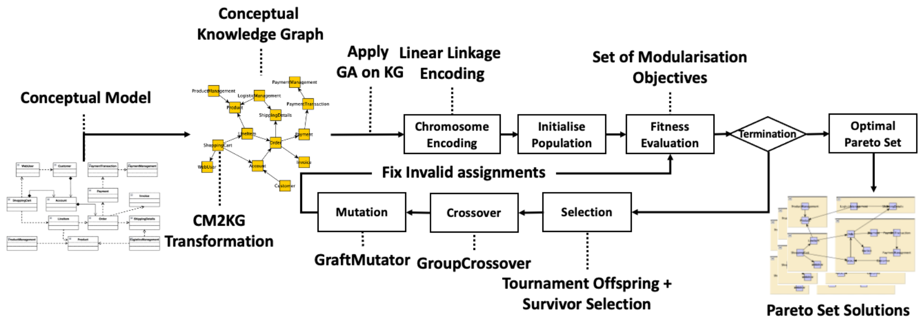


Fig. 1: End-to-end Generic Genetic Modularization Framework

4.1 Model Transformation and Genetic Encoding

In GGMF, we first transform a conceptual model of any modeling language into a CKG. We show in Fig. 2 the transformation of an Online Shop UML class diagram into the corresponding CKG. The CKG is represented as a labeled property graph [14] with each node and edge capturing their corresponding properties, e.g., association, and dependency relationships. The GA can use these properties during optimization (see Fig. 4).

Once we have transformed the model into a CKG, we create a genetic encoding that serves as input to the GA. We use Linear Linkage Encoding of the CKG in Fig. 3. LLE represents the elements in each module as a linked list with elements sorted in the order of their indices. The sorting is done to avoid isomorphic representations of a module and thereby avoid duplicate solutions. Fig. 3 shows LLE as a chromosome representing the CKG where each vertex and edge is assigned to a module. A gene is associated with

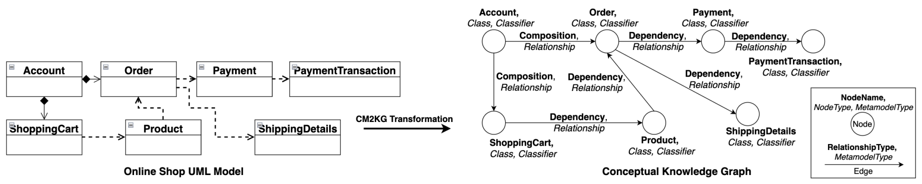

Account
Order
Payment
PaymentTransaction
ShoppingCart
Product
ShippingDetails
Online Shop UML Model
CM2KG Transformation
Account, Class, Classifier
Order, Class, Classifier
Payment, Class, Classifier
PaymentTransaction, Class, Classifier
ShoppingCart, Class, Classifier
Product, Class, Classifier
ShippingDetails Class, Classifier
Composition, Relationship
Dependency, Relationship
Conceptual Knowledge Graph
NodeName, NodeType, MetamodelType
Node
RelationshipType, MetamodelType
Edge

Fig. 2: Conceptual Model to CKG transformation

each vertex and an edge. The allele values of each gene are the index of the next vertex or edge in the same module. So each gene is linked to the next element in the module except for the last element. The last element points to itself. Each module is denoted by the last element, i.e., the ending node. The modularization yields a set of modules where each module is connected to the other. Note that in the subsequent subsections, we provide high-level details of the aspects involved in the optimization process of GAs. These aspects are invariants during optimization in GGMF and do not contribute to the genericity and customizability of GGMF; therefore, the in-depth details about these aspects are outside the scope of our paper.

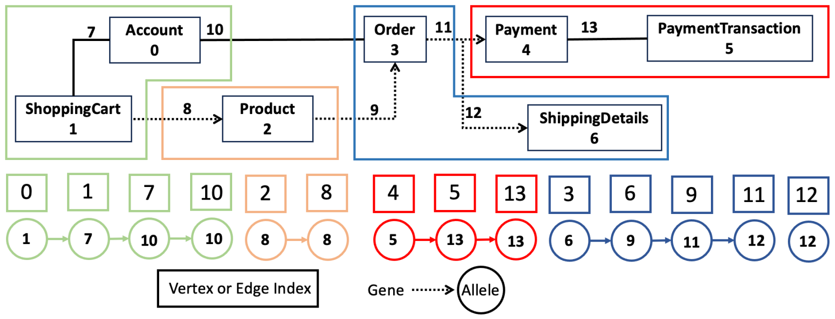


Fig. 3: Linear Linkage Encoding Example

4.2 Objective functions

Given an LLE chromosome, we can evaluate the module assignment of each element of the CKG and therefore evaluate the modularization quality on the objectives selected for modularization. Table 1 shows multiple objectives to measure modularization quality from a chromosome found in the literature. We categorized the objectives into *module-based*, *semantics-based*, *entropy-based*, and *graph-based*. The module-focused objectives aim to measure the module-related properties. The semantics-based objectives use language model-based word embeddings of an element. The entropy-based types only apply entropy techniques to the string values. Lastly, graph-based objectives employ graph properties for characterization. The categorization is not mutually exclusive, i.e., one objective has side effects on other objectives. For example, graph-based objectives can capture cohesion.

We can use the structural features of the graph to evaluate the module and graph-based objectives. Note that module and graph-based objectives can have edge weights

Table 1: List of implemented objective functions

Category	**Objective**	**Objective Description**
Module	Cohesion	Maximise the sum of intra-module edges
Module	Coupling	Minimise the sum of inter-module edges
Module	Balancedness	Minimize the standard deviation of module size from a threshold[2]
Module	Average Module Size	Minimise the average module size
Semantics	Semantic cohesion	Maximise the semantic similarity of intra-module vertices
Semantics	Semantic coupling	Minimise the similarity of inter-module vertices
Entropy	String similarity	Maximise the average string similarity per module
Entropy	String difference	Maximise the average string difference between modules
Graph	node closeness	Minimise the average of vertex closeness centrality per module
Graph	edge betweenness	Minimise the average edge betweenness centrality per module
Graph	Modularity	Maximise the modularity score [27]

depending on the relation type (see Fig. 4). To evaluate the entropy and semantics-based objectives, we use the natural language semantics captured by the node labels using a language model-based representation of the nodes that capture the natural language semantics of its labels. Recent contextualized NLP models such as BERT [10], with bidirectional attention-based mechanism, i.e., transformers [40], can extract essential features from textual sequences and learn high-quality contextualized representations. Pre-trained BERT can be effectively employed for knowledge transfer and has produced impressive results in various downstream tasks such as open-domain question answering [45] and aspect-based sentiment analysis [43]. Therefore, pre-trained BERT embeddings can represent the conceptual model elements' terms. Therefore, we use the vector-based representations of each node to capture the natural language semantics of each node label. To evaluate the semantic similarity between two nodes, we use the cosine similarity measure between the vector embeddings of the node labels.

Finally, it is important to note that we can execute the GA using multiple objectives. However, we can also combine the objectives as a weighted sum of the objectives to create a single objective. We multiply the maximization objectives by minus one, reducing the single objective to a minimization objective. By default, we treat the weights of all objectives as equal to one.

4.3 Selection

Once we have transformed the CKG into LLE and we have an initial population of chromosomes, we need to perform a selection of candidates for the offspring generation. We use two selection operators. The tournament selection is used for choosing the chromosomes for the offspring generation when both the single- or multi-objective functions are applied. These selected chromosomes go through the altering process i.e., crossover and mutation to create new altered individuals. The roulette wheel selector is used for choosing the chromosomes for the survivor population. The altered offspring and survivor population get merged. Some individuals are removed during this merge to simulate the killing process [2].

4.4 Alteration

After selecting the chromosomes for offspring generations, we apply crossover with the selected chromosomes. The parents are randomly chosen from the offspring population to create new offsprings. The group crossover guarantees the creation of valid LLE instances. The central idea of the group crossover is to treat the ending node of the parent individuals as the central element of a module during the crossover. These central elements are then shared in the offspring instances [20]. The crossover operation generates two children, which share the ending nodes of the parents. They are inserted into the offspring population. The modules are built in the offspring from these new ending nodes after crossover. The group crossover approach also ensures that each module is a connected element, which follows the self-contained principle described in [25].

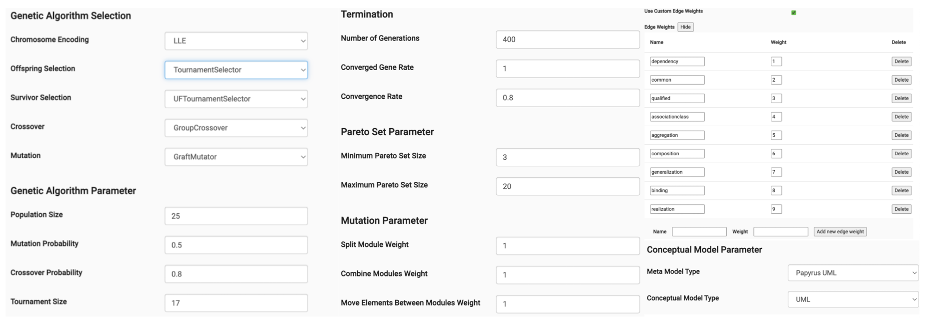
Mutation randomly takes and alters any individual from the offspring population and creates an offspring. The alteration focuses on changes in the modularization solution instance where modules will be merged, split, or elements in a module get assigned to a different module. The graft mutation operation randomly determines which of the three possible mutation types is applied. The first type of mutation divides a random module consisting of multiple elements into multiple connected submodules. The main idea involves multiple random walks to split modules into multiple connected submodules. This approach only operates inside the respective module—other modules' elements are unaffected. The next type of mutation is the combination of two random modules. In this case, only the modules with neighboring modules are used. The last mutation type moves one element in a module to a neighboring module. Elements are candidates if they are directly incident to a different module. If the element is a vertex, an incident edge must be in a different module. When the element is an edge, then one of the incident vertices must be assigned to a different module. This element is removed from the source to the target module's linked list.

4.5 Constraints and Termination

The mutation operation can produce invalid chromosomes due to randomly assigning a gene to a module. Therefore, constraints are enforced to remove invalid chromosomes. We enforce firstly that modules must be a connected subgraph to be self-contained [25], i.e., each element in the module must be reachable. Secondly, both the endpoints of an edge must be present in one of the modules. Fig. 1 shows the genetic algorithm process. The process is terminated if the population's fitness reaches a pre-defined threshold based on the selected objectives. The convergence of the algorithm and the number of solutions in the Pareto set is dependent on the CKG complexity and the fitness functions. Currently, GGMF ranks all the different solutions in the Pareto set equally.

4.6 Modularization tool

We now present the Web-based tool we developed for GGMF. Fig. 4 shows the configuration parameters including the edge weights. The edge weights denote the importance of a conceptual model's different edge types of. Fig. 4 shows the edge weights for a UML model, however, the edge weights as well as labels get updated depending on the type of modeling language of the input model. The user can freely customize the weights.



Genetic Algorithm Selection

Chromosome Encoding	LLE
Offspring Selection	TournamentSelector
Survivor Selection	UFTournamentSelector
Crossover	GroupCrossover
Mutation	GraftMutator

Genetic Algorithm Parameter

Population Size	25
Mutation Probability	0.5
Crossover Probability	0.8
Tournament Size	17

Termination

Number of Generations	400
Converged Gene Rate	1
Convergence Rate	0.8

Pareto Set Parameter

Minimum Pareto Set Size	3
Maximum Pareto Set Size	20

Mutation Parameter

Split Module Weight	1
Combine Modules Weight	1
Move Elements Between Modules Weight	1

Use Custom Edge Weights ✓

Edge Weights

Name	Weight	Delete
dependency	1	Delete
common	2	Delete
qualified	3	Delete
associationclass	4	Delete
aggregation	5	Delete
composition	6	Delete
generalization	7	Delete
binding	8	Delete
realization	9	Delete

Conceptual Model Parameter

Meta Model Type	Papyrus UML
Conceptual Model Type	UML

Fig. 4: Configuration parameters

The web UI is developed with Angular [1]. The web application utilizes a web component to create the user interface's visual part. In the background, a service creates an HTTP request and waits for the HTTP response. The UI is built so that it can be extended via a configuration. The default parameters, the edge weights, and the objectives are specified in a configuration file. Especially for the weights and objectives, a new entry can be added to the configuration and is immediately displayed. The rationale behind the UI was to ease the configuration of modularization experiments. The implementation of the tool can be publicly available in the GitHub Repository [3]

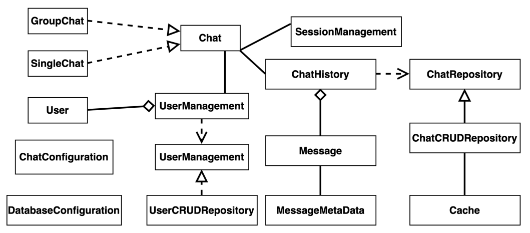
5 Evaluation

In the following, we respond to the three research questions as defined in Section 1 by first showing the genericity of our framework in terms of modeling languages supported (RQ1). We perform an impact analysis of the different kinds of objectives used for modularization (RQ2), and, finally, we show the performance of GGMF by comparing it to three other approaches proposed in the literature (RQ3).

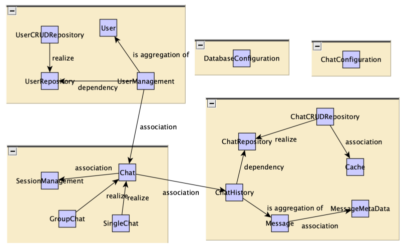
5.1 Generic modularization

Fig. 5 shows the modularization result of applying our approach to different modeling languages. The figures show modularized results of the models from three different modeling languages, i.e., ER, UML and ADOxx-based models. The modularized results produce valid models and do not violate any constraints. These results support the feasibility of our proposed approach. Note that we conducted experiments on the dataset of 555 UML[4], 42 ER[5] models but due to limited space, we show the three representative cases in Fig. 5.

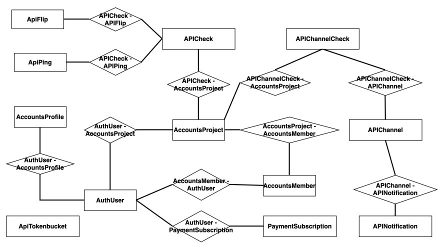
[3] `https://github.com/me-big-tuwien-ac-at/GGMF`
[4] `https://zenodo.org/record/2585456#.YM5ziSbtb0o`
[5] `https://drawsql.app/templates`



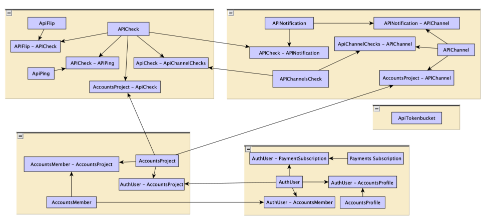
(a) Monolithic ADOxx-based Model



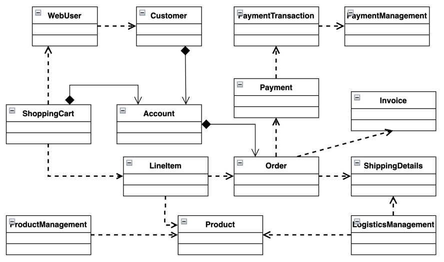
(b) Modularised ADOxx-based model


ApiFlip
APICheck - APIFlip
APICheck
APIChannelCheck
ApiPing
APICheck - APIPing
APICheck - AccountsProject
APIChannelCheck - AccountsProject
APIChannelCheck - APIChannel
AccountsProfile
AuthUser - AccountsProject
AccountsProject
AccountsProject - AccountsMember
APIChannel
AuthUser - AccountsProfile
AccountsMember - AuthUser
AccountsMember
AuthUser
APIChannel - APINotification
ApiTokenbucket
AuthUser - PaymentSubscription
PaymentSubscription
APINotification

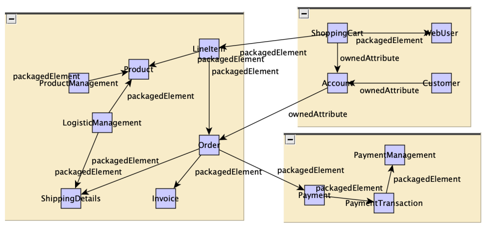
(c) Monolithic ER Model


ApiFlip
APICheck
APIFlip - APICheck
APICheck - APIPing
APICheck - APIChannelChecks
ApiPing
AccountsProject - ApiCheck
APINotification
APINotification - APIChannel
APICheck - APINotification
ApiChannelChecks - APIChannel
APIChannel
APIChannelsCheck
AccountsProject - APIChannel
ApiTokenbucket
AccountsMember - AccountsProject
AccountsProject
AuthUser - AccountsProject
AccountsMember
AuthUser - PaymentSubscription
Payments Subscription
AuthUser
AuthUser - AccountsProfile
AuthUser - AccountsMember
AccountsProfile

(d) Modularised ER model



(e) Monolithic UML Model

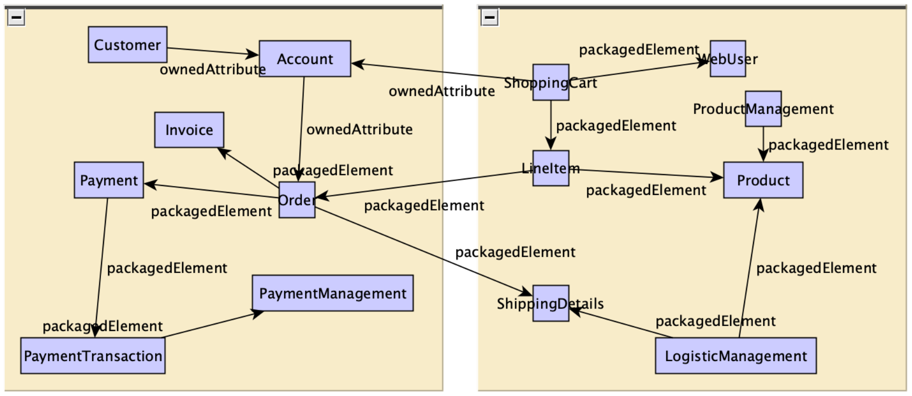


(f) Modularised UML model

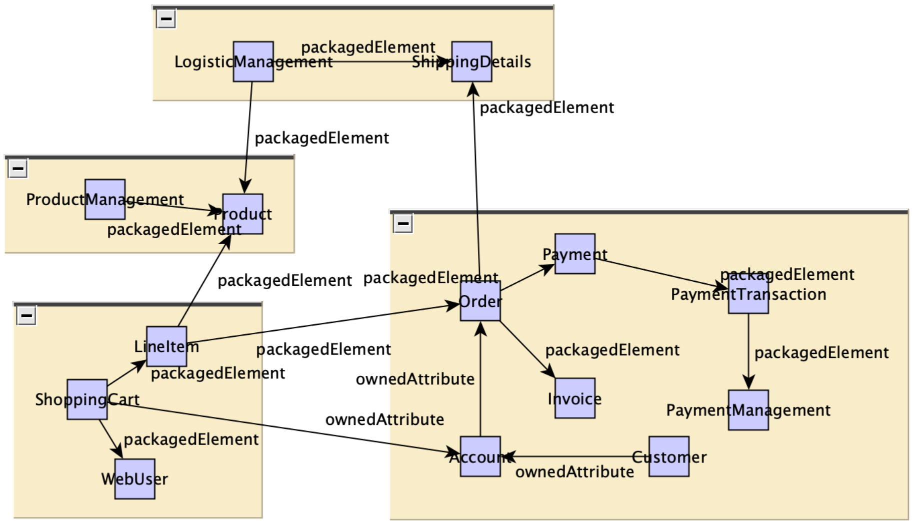
Fig. 5: Three exemplary modularization experiments with different models

5.2 Objective Impact Analysis

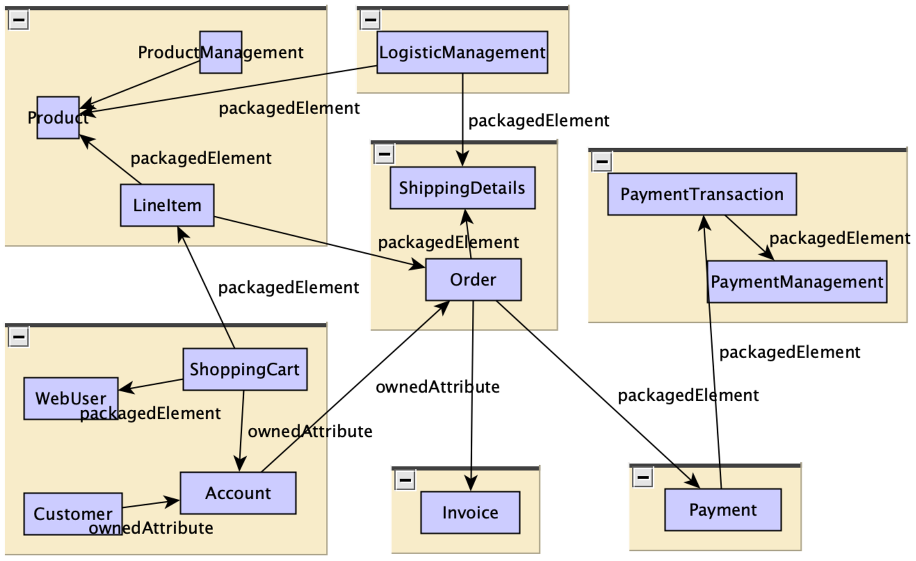
To respond to RQ-2, i.e., how the framework caters to different requirements, we show the impact of different objectives on modularization. We show, that changing objectives affect the modularization quality, therefore, our framework can support modularization in the context of changing requirements given that the user can choose appropriate objectives. Fig. 6 shows the impact of using *balancedness*, *modularity*, and *semantic cohesion* as the objectives for modularization. We choose these three objectives to show the effect of different types of objectives, with balancedness focusing on individual module size, modularity focusing on the entire model, and semantic cohesion focusing on the natural language semantics of the model. We see, that using balancedness as the objective produces two modules with seven elements as per the expectation of the objective to produce modules with a size close to Miller's magic number seven [31]. In the case of modularity, we see more modules with uneven sizes. This modularization results from maximizing the modularity score as the objective given by Eq. 1. Finally, we see even more modules for semantic coupling to minimize the similarity between terms in different modules. We see that the terms in each module are semantically closer and dis-



(a) Balancedness



(b) Modularity



(c) Semantic Betweenness

Fig. 6: Effect of different objectives on the modularization

Table 2: Modularization approaches comparison

NumNodes	**Modularity**				**MQ**			
	WLC	GMA	Louvain	Ours	WLC	GMA	Louvain	Ours
48	0.55	0.55	0.66	0.62	3.55	4.34	5.13	4.34
56	0.15	0.15	0.21	0.62	2	2	3.80	2
97	0.55	0.07	0.42	0.39	3.91	1.76	5.13	3.27
98	0.42	0.42	0.51	0.62	2	2	3.49	2.84
103	0.26	0.55	0.59	0.62	2	2	3.76	4.52

similar in different modules. Each objective has strengths and weaknesses, depending on the use case. Therefore, GGMF allows users to apply any objectives they want.

5.3 Comparative analysis

In order to evaluate RQ-3, we compare the quality of the GGMF modularization results with three different approaches i.e., *Weighted Linking Clustering* (WLC) [24], *Graph-based Modularization* (GMA) [28], and the *Louvain* algorithm [8] on a set of five UML models. We use a distance matrix created from the adjacency matrix of the CKG as an input to WLC and a similarity matrix constructed from the distance matrix as $1 - distance$ as input to GMA. We compare the quality of the results using modularity and MQ score. Note that the purpose of this comparison is primarily to show that the results from our approach are reasonably good in the context of modularization, which can be further optimized depending on the customizations provided by GGMF based on the user requirements.

We show the comparison based on modularity and MQ Score in Table 2. We see the modularized results from our approach provide better modularity and MQ scores for the five models compared to GMA and WLC and also perform comparable to the Louvain algorithm. However, Louvain can suffer from drawbacks such as it may yield arbitrarily badly connected communities and communities may even be internally disconnected [39], which our approach explicitly avoids as part of the constraints (see Section 4). Moreover, we cannot apply natural language semantics-based objectives with Louvain. The results show that our GA-based optimization approach successfully finds good-quality modules of a model based on heterogeneous objectives.

6 Threats to Validity

We now elaborate on the threats to validity according to the widely accepted categories introduced by Wohlin et al. [44]. **Conclusion validity** concerns the relationship between the treatment and the outcome. We mitigated this threat by testing our framework on models of multiple modeling languages and with multiple combinations of objective functions to perform impact analysis of using a specific objective on modularization. **Internal Validity** - Parameter tuning of search algorithms is still considered an open research challenge [7]. In our work we set the configuration parameters for modularization based on experience with the modularization tool. However, we make our results reproducible with a set of configuration parameters and we expose all the configuration parameters through our web UI. **External Validity** - The quality of conceptual models used in our experiments also threatens the validity of our work. However, we mitigated this by using the dataset of models used by several works in the literature [22, 23].

7 Conclusion

In this paper, we presented a generic, customizable framework for conceptual model modularization using genetic algorithm optimization techniques. We showed, that using Conceptual Knowledge Graphs as the intermediary representation of conceptual models can be used as a generic intermediary representation before applying GA-based modularization. We presented our end-to-end approach that takes as input a conceptual model and provides the modularized model. We showed the feasibility of generic modularization by executing our approach on models from three modeling languages i.e., ER, UML and ADOxx-based models. Different requirements require targeting different objectives, therefore, we showed the effect of using different objectives on the modularization results. GGMF allows extending the list of objectives that can be used by the GA for optimization, thereby supporting modularization depending on different requirements. We compared the quality of the solutions produced by our approach with two other approaches. The results show that our approach provides modules of comparable quality. Finally, we showed the interface of the web-based modularization tool that we developed, which allows users to configure the parameters for the modularization and also adjust the importance (weights) of different types of edges present in a model, which allows using a weighted contribution of an edge while evaluating the quality of a module during optimization. Consequently, GGMF enables an entirely new level of flexibility and customizability of GA-based model modularization which is also applicable for non-technical users.

In the future, we focus on the following potential improvements of GGMF: $i)$ advanced qualitative evaluation of the resulting modules in relation to the initial modularization goals, which is currently quantitatively evaluated using the fitness scores of the involved fitness functions; $ii)$ statistical-analysis based GGMF configuration parameter values recommendation for diverse models; and $iii)$ comparative analysis with ML-based modularization techniques that use structure and semantics for modularization.

References

1. Angular. `https://angular.io/`, accessed: 2022-07-30
2. Jenetics. `https://https://jenetics.io/`, accessed: 2022-07-09
3. Ali, S.J., Guizzardi, G., Bork, D.: Enabling representation learning in ontology-driven conceptual modeling using graph neural networks. In: CAiSE 2023. Lecture Notes in Computer Science, vol. 13901. Springer (2023). https://doi.org/10.1007/978-3-031-34560-9_17
4. Andritsos, P., Tzerpos, V.: Information-theoretic software clustering. IEEE Transactions on Software Engineering **31**(2), 150–165 (2005)
5. Bae, J.H., Lee, K., Chae, H.S.: Modularization of the uml metamodel using model slicing. In: Fifth International Conference on Information Technology: New Generations (itng 2008). pp. 1253–1254 (2008). https://doi.org/10.1109/ITNG.2008.179

6. Bavota, G., Carnevale, F., De Lucia, A., Di Penta, M., Oliveto, R.: Putting the developer in-the-loop: an interactive ga for software re-modularization. In: 4th International Symposium on Search Based Software Engineering (SSBSE 2012). pp. 75–89. Springer (2012)
7. Bill, R., Fleck, M., Troya, J., Mayerhofer, T., Wimmer, M.: A local and global tour on momot. Software & Systems Modeling **18**, 1017–1046 (2019)
8. Blondel, V.D., Guillaume, J.L., Lambiotte, R., Lefebvre, E.: Fast unfolding of communities in large networks. Journal of statistical mechanics: theory and experiment **2008**(10), P10008 (2008)
9. Bork, D., Garmendia, A., Wimmer, M.: Towards a multi-objective modularization approach for entity-relationship models. In: ER Forum, Demo and Poster 2020. pp. 45–58. CEUR (2020)
10. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint:1810.04805 (2018)
11. Doran, P., Tamma, V., Iannone, L.: Ontology module extraction for ontology reuse: an ontology engineering perspective. In: Proceedings of the sixteenth ACM conference on Conference on information and knowledge management. pp. 61–70 (2007)
12. Figueiredo, G., Duchardt, A., Hedblom, M.M., Guizzardi, G.: Breaking into pieces: An ontological approach to conceptual model complexity management. In: 2018 12th International Conference on Research Challenges in Information Science (RCIS). pp. 1–10. IEEE (2018)
13. Freeman, L.: A set of measures of centrality based on betweenness. Sociometry **40**, 35–41 (03 1977). https://doi.org/10.2307/3033543
14. Glaser, P.L., Sallinger, E., Bork, D.: Model-based construction of enterprise architecture knowledge graphs (2022 (Under Review))
15. Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, New York (1989)
16. Hinkel, G., Strittmatter, M.: On using sarkar metrics to evaluate the modularity of metamodels. In: Proceedings of the 5th International Conference on Model-Driven Engineering and Software Development. pp. 253–260. Springer (2017)
17. Hinkel, G., Strittmatter, M.: Predicting the perceived modularity of mof-based metamodels. In: 6th International Conference on Model-Driven Engineering and Software Development (MODELSWARD 2018), Funchal, P, January 22-24, 2018. pp. 48–58. SciTePress (2018)
18. Kang, D., Xu, B., Lu, J., Chu, W.: A complexity measure for ontology based on uml. In: Proceedings. 10th IEEE International Workshop on Future Trends of Distributed Computing Systems, 2004. FTDCS 2004. pp. 222–228 (2004)
19. Katoch, S., Chauhan, S.S., Kumar, V.: A review on genetic algorithm: past, present, and future. Multim. Tools Appl. **80**(5), 8091–8126 (2021). https://doi.org/10.1007/s11042-020-10139-6
20. Korkmaz, E.: Multi-objective genetic algorithms for grouping problems. Appl. Intell. **33**, 179–192 (10 2010). https://doi.org/10.1007/s10489-008-0158-3
21. LeClair, A., Marinache, A., El Ghalayini, H., MacCaull, W., Khedri, R.: A review on ontology modularization techniques-a multi-dimensional perspective. IEEE Transactions on Knowledge and Data Engineering **35**(5), 4376–4394 (2022)
22. López, J.A.H., Cánovas Izquierdo, J.L., Cuadrado, J.S.: Modelset: a dataset for machine learning in model-driven engineering. Software and Systems Modeling pp. 1–20 (2022)
23. López, J.A.H., Cuadrado, J.S.: An efficient and scalable search engine for models. Software and Systems Modeling **21**(5), 1715–1737 (2022)
24. Maqbool, O., Babri, H.: Hierarchical clustering for software architecture recovery. IEEE Transactions on Software Engineering **33**(11), 759–780 (2007)
25. Moody, D.L., Flitman, A.: A methodology for clustering entity relationship models - a human information processing approach. In: 18th International Conference on Conceptual Modeling. pp. 114–130. Springer (1999)

26. Mu, L., Sugumaran, V., Wang, F.: A hybrid genetic algorithm for software architecture re-modularization. Information Systems Frontiers **22**, 1133–1161 (2020)
27. Newman, M.E.J.: Modularity and community structure in networks. Proceedings of the National Academy of Sciences **103**(23), 8577–8582 (2006). https://doi.org/10.1073/pnas.0601602103
28. Pourasghar, B., Izadkhah, H., Isazadeh, A., Lotfi, S.: A graph-based clustering algorithm for software systems modularization. Information and Software Technology **133**, 106469 (2021)
29. Prajapati, A., Kumar Chhabra, J.: Optimizing software modularity with minimum possible variations. Journal of Intelligent Systems **29**(1), 1135–1150 (2020). https://doi.org/doi:10.1515/jisys-2018-0231
30. Proper, H.A., Guizzardi, G.: Modeling for enterprises; let's go to rome via rime. hand **1**, 3 (2022)
31. Saaty, T.L., Ozdemir, M.S.: Why the magic number seven plus or minus two. Mathematical and computer modelling **38**(3-4), 233–244 (2003)
32. Sarkar, S., Kak, A.C., Maskeri Rama, G.: Metrics for measuring the quality of modularization of large-scale object-oriented software. IEEE Transactions on Software Engineering **34**(05), 700–720 (sep 2008). https://doi.org/10.1109/TSE.2008.43
33. Saruladha, K., Aghila, G., Sathiya, B.: Neighbour based structural proximity measures for ontology matching systems. In: Proceedings of the International Conference on Advances in Computing, Communications and Informatics. pp. 1079–1085 (2012)
34. Sequeda, J., Lassila, O.: Designing and building enterprise knowledge graphs. Synthesis Lectures on Data, Semantics, and Knowledge **11**(1), 1–165 (2021)
35. Singh, P., Jonkers, H., Iacob, M.E.: Modeling value creation with enterprise architecture. ICEIS 2014 - Proceedings of the 16th International Conference on Enterprise Information Systems **3**, 343–351 (2014)
36. Smajevic, M., Bork, D.: Towards graph-based analysis of enterprise architecture models. In: Int. Conference on Conceptual Modeling. pp. 199–209. Springer (2021)
37. Stuckenschmidt, H., Klein, M.: Structure-based partitioning of large concept hierarchies. In: The Semantic Web–ISWC 2004: Third International Semantic Web Conference, Hiroshima, Japan, November 7-11, 2004. Proceedings 3. pp. 289–303. Springer (2004)
38. Tabrizi, A.H.F., Izadkhah, H.: Software modularization by combining genetic and hierarchical algorithms. In: 2019 5th Conference on Knowledge Based Engineering and Innovation (KBEI). pp. 454–459. IEEE (2019)
39. Traag, V.A., Waltman, L., Van Eck, N.J.: From louvain to leiden: guaranteeing well-connected communities. Scientific reports **9**(1), 5233 (2019)
40. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems **30** (2017)
41. Villegas Niño, A.: A filtering engine for large conceptual schemas. Doctoral thesis (2013)
42. Vragović, I., Louis, E.: Network community structure and loop coefficient method. Physical review. E, Statistical, nonlinear, and soft matter physics **74**, 016105 (08 2006). https://doi.org/10.1103/PhysRevE.74.016105
43. Wang, Y., Chen, Q., Wang, W.: Multi-task bert for aspect-based sentiment analysis. In: 2021 IEEE International Conference on Smart Computing. pp. 383–385. IEEE (2021)
44. Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslén, A.: Experimentation in software engineering. Springer Science & Business Media (2012)
45. Yang, W., Xie, Y., Lin, A., Li, X., Tan, L., Xiong, K., Li, M., Lin, J.: End-to-end open-domain question answering with bertserini. arXiv preprint arXiv:1902.01718 (2019)