Towards a Multi-Objective Modularization Approach for Entity-Relationship Models

Dominik Bork[1], Antonio Garmendia[2], and Manuel Wimmer[2]

[1] TU Wien, Business Informatics Group, Vienna, Austria, dominik.bork@tuwien.ac.at
[2] Johannes Kepler University Linz, CDL-MINT, Linz, Austria
{antonio.garmendia,manuel.wimmer}@jku.at

Abstract. Legacy systems and their associated data models often evolve into large, monolithic artifacts. This threatens comprehensibility and maintainability by human beings. Breaking down a monolith into a modular structure is an established technique in software engineering. Several previous works aimed to adapt modularization also for conceptual data models. However, we currently see a research gap manifested in the absence of: (i) a flexible and extensible modularization concept for Entity Relationship (ER) models; (ii) of openly available tool support; and (iii) empirical evaluation. With this paper, we introduce a generic encoding of a modularization concept for ER models which enables the use of meta-heuristic search approaches. For the efficient application we introduce the *ModulER* tool. Eventually, we report on a twofold evaluation: First, we demonstrate feasibility and performance of the approach by two demonstration cases. Second, we report on an initial empirical experiment and a survey we conducted with modelers to compare automated modularizations with manually created ones and to better understand how humans approach ER modularization.

Keywords: Entity-Relationship · Modularization · Meta-Heuristic Search · Genetic Algorithms

1 Introduction

During conceptual modeling, one is applying abstraction to reduce the complexity of a specific domain, normally to address a certain purpose. The aimed purpose can be manifold, such as the value attached to a conceptual model [2]. Traditionally, conceptual modeling was aimed to support communication amongst human beings to derive at a common understanding [21]. In model-based software engineering [4], conceptual models not only serve as a design blueprint of an information system, but furthermore may enable code generation.

Independently of the ultimate purpose a conceptual model supports, we expect a human being involved in the creation and analysis process. Therefore, comprehension of conceptual models is a prerequisite for enabling model value. For example, Entity Relationship (ER) models have been applied since several decades in order to create and analyse an abstract, conceptual representation of a data model for human beings. Thus, ER models apply abstraction to the technical details of database implementation, thereby providing a clear vision on the conceptual structures of a data schema.

However, legacy data models often evolve into large, unstructured, monolithic artifacts that impede efficient comprehension by overwhelming the cognitive processing capabilities of human beings [18]. This increasing complexity thus cannot be addressed solely by abstraction. Adding decomposition of the abstract models is one option for further decreasing complexity toward a manageable level. "The most common way of reducing complexity of large systems is to divide them into smaller parts or subsystems: This is called modularization." [18] Modularization is an established concept in software engineering [16] where software monoliths are decomposed into small and flexible modules that can be efficiently updated, extended, and retired. Modularization not only reduces complexity, it is also related to software quality [27]. Recently, modularization has been applied in conceptual modeling and ontology development aiming to improve comprehension [13, 22, 31]. Generally, benefits of modularization include *"division of labor, scalability, partial reuse, and broadened participation"* [22, p. 504].

Modularization is a very complex task that calls for automation as the number of possible modularizations increases exponentially following the Bell number. For a model with 10 modularizable elements there exist already 115.975 alternative modularizations. Mathematically, the possible modularizations can be calculated as follows [11]:

$$B_{n+1} = \sum_{k=0}^{n} \binom{n}{k} B_k \qquad B_0 = 1 \tag{1}$$

In the paper at hand, we propose an ER meta-model which is transformed into a native encoding that forms the input for the meta-heuristic search that enables the partitioning of complex, monolithic ER models into a modular structure. Our solution uses Genetic Algorithms (GA) to automatically compute good modularizations from an initial overarching ER model while aiming to optimize a set of objectives. Following this approach, a much more narrow declarative formulation of the objectives of a modularization can be used instead of defining precise procedures to derive a good modularization. However, the selection of objectives and their quantification remains challenging which is why an initial empirical study was conducted that enables us to reason on the expectations of humans on a modularized ER model. Our contributions thus extend the body of knowledge by proposing an openly available tool implementation and by an initial empirical study. We believe making the tool openly available enables the scientific community to efficiently contribute toward progressing this research line.

The rest of this paper is organized as follows: Section 2 reports on previous research. A modularization concept for ER models is then proposed in Section 3 and evaluated in Section 4. We conclude this paper in Section 5 with a brief summary and an outlook to future research directions.

2 Previous Research

Modularization of data models is not new in itself. This section briefly summarizes the most relevant works on the modularization of data models (Section 2.1) and modularization in conceptual modeling (Section 2.2).

2.1 Modularization of Data Models

Feldman and Miller [10] propose *Clustered Entity Models*, by referring to an ER model as one that *"is a hierarchy of successively more detailed entity relationship diagrams, with a lower-level diagram appearing as a single entity type at the next higher-level diagram."* The approach is specified informally and its effectiveness heavily depends on the expertise of the modeler. No precise rules for either the number of cluster levels and the decisions of which ER elements should form part of which cluster are given. Moreover, the approach employs redundant clustering which has been evaluated empirically as being inferior with respect to comprehensiveness [17].

A *Structured Data Models* is proposed by Simsion [25]. One overarching data model (positioned at the bottom level) can be presented at multiple, more abstract levels following a generalization semantics. Evaluations of this approach showed that models at higher levels tend to be too generic for human comprehension, and that the procedure lacks formalization as it mostly relies on the expertise of the modeler [17].

Moody et al. described and evaluated an approach entitled *Levelled Data Models* [17, 19, 20]. This work applied the principles of human information processing for the decomposition of large data models. As a result, a hierarchy of sub-models of a manageable size are obtained.

2.2 Modularization in Conceptual Modeling

Tzitzikas and Hainaut [30] propose an approach that aims to automatically reverse engineer and visualize smaller diagrams from one overarching ER diagram. Their approach is based on EntityRank, an adapted PageRank [5] algorithm for identifying the most relevant Entities and Relationships in an ER model. The algorithm only aims at satisfying the single goal of the highest EntityRank. It can be customized to identify and visualize the top-k ER diagrams. Tooling is provided on a proprietary basis [23].

A similar approach is presented by Villegas et al. [32] that focuses on filtering relevant information to a requester as a subset of an overarching ER model. *"In order to select this subset, our method measures the interest of each entity type with respect to the focus set based on the importance and closeness."* [32, p. 258]. First, a user needs to specify her interest by means of: (i) denoting entities of interest, (ii) entities which are out of scope, and (iii) by providing the number of entities the resulting ER model shall comprise at maximum.

Guedes et al. describe an evolutionary database modularization design process based on ER diagrams that are extended by procedures [14]. Following an iterative approach, existing modules are edited or new modules are created by analyzing information sharing aspects. The concept of module interface for handling information sharing and resource access is introduced.

Garmendia et al. [13] propose abstract modularity meta-model patterns which can be applied to any meta-model and is therefore independent of a specific modeling language. The approach aims at improving the management of large models, particularly their validation.

The approach presented by [34] addresses modularization on the meta-model level from the objective of flexibly orchestrating large meta-models. A meta-model module concept is introduced together with mixins and extenders [34] for module integration.

Modularization can also be applied to overarching ontologies [22, 29]. The approach presented by Özacar et al. [22] aims to optimize multiple goals. A supporting tool automates the application of the approach by applying a set of ontology development patterns and allowed operations on ontology modules.

From the above can be concluded, that while there are several modularization approaches available, we are not aware of any approach that is based on a *generic encoding*, utilizes a *multi-objective search*, comes with proper *tool support*, and provides an *empirical evaluation*.

3 A Modularization Approach for Entity-Relationship Models

Based on the analysis of previous research in Section 2, we now present the details of our approach for modularizing ER models. The approach aims at mitigating the identified shortcomings while also enabling flexible extension in the future.

Fig. 1 illustrates the many-objective proposal to partition ER models. The rectangles with grey background are the main building blocks of genetic algorithms [8]. As a first step (label 1), the algorithm takes as an input an ER model whose structure and elements are explained in Subsection 3.1. Starting from the input model, an *Initial Population* is created (label 2) with the encoding proposed by Rizzi [24] (cf. Subsection 3.2). The *Evaluation* of each individual of the population (i.e., chromosome) is performed using a set of five functions based on Moody's principles for clustering ER models [20] (label 3). These functions are explained in detail in Subsection 3.3. If the termination condition is fulfilled, the Pareto set of optimal solutions is returned (label 4).

The results in the Pareto set are selected using the NSGA II algorithm [6], which ensures diversity in the Pareto set solutions (label 5). If the termination condition is not fulfilled, the algorithm creates the next generation of the population in order to find better solutions. The algorithm uses two *Crossover* (label 6) and two *Mutation* (label 7) operators. In particular, *Partially-Matched Crossover* (PMX) and *Multi-point Crossover* (MX) as well as *Swap* and *Flip* mutations are used, respectively (see Subsection 3.4 for a detailed description of these operators). Finally, each chromosome is tested against the constraints (label 8). In case an individual does not fulfil a constraint, a repair process is triggered which is described in Subsection 3.5.

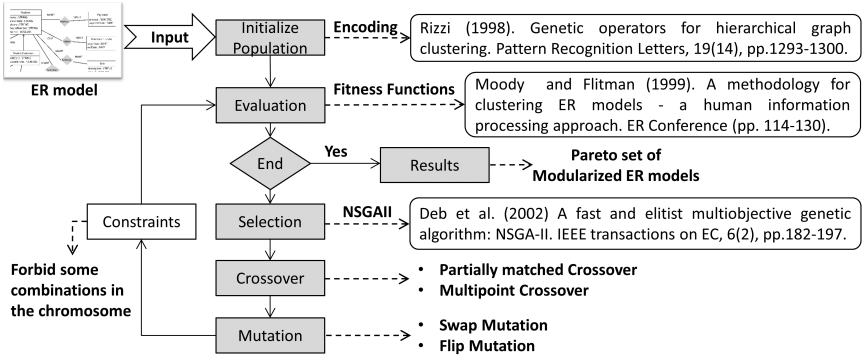


Fig. 1: Overview of the ModulER approach for automatically modularizing ER models

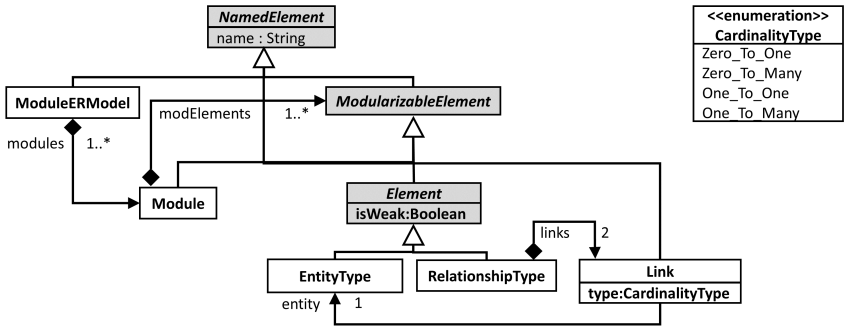


Fig. 2: ModulER meta-model (grey $\widehat{=}$ abstract classes, white $\widehat{=}$ concrete classes)

3.1 Conceptual Modeling

The design of a modeling language consists in formalizing the abstract syntax, concrete syntax, and semantics [4]. For formalizing the abstract syntax, one usually refers to meta-models [3] which define the syntactic concepts of a language as well as the allowed combinations thereof. In the case our approach, we extend the core *ER* language concepts that enable a modular structure.

Fig. 2 shows the ModulER meta-model in terms of an UML class diagram. A ModuleERModel instance contains only Module elements. Objects of type Module may contain ModularizableElements as well as other modules. A ModularizableElement is either an EntityType or a RelationshipType. Objects of type RelationshipType must contain two Links and each one references to an EntityType instance. This language definition for *ER* is designed to account for different configurations of an *ER* model and for efficient extension in the future. For instance, we may have an *ER* model represented in one module or partitioned into a set of either flattened or hierarchically structured modules.

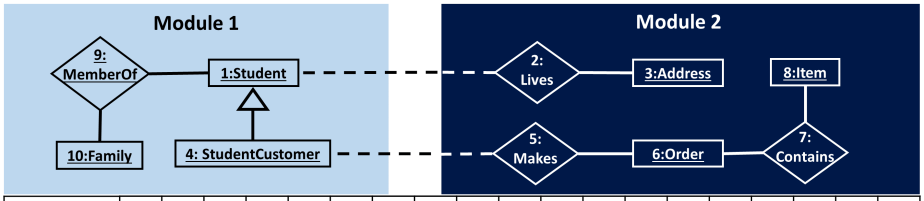
3.2 Encoding scheme

We applied the encoding proposed by Rizzi [24] to represent each chromosome in the population. This encoding proposes a chromosome that consists of different types of genes depending on where it is located. The genes in odd positions should represent the integer identifier of the ModularizableElement, and the genes in even positions would take binary values 0 or 1. These binary values are called *separators*, and the sum of them is the denoted by the letter n. The number of clusters is calculated summing n+1.

In order to exemplify the application of this encoding to our language, Fig. 3 shows an ER excerpt of the Karate School example proposed by Ambler [1] which we will also use in our experiments (see Section 4). In this figure, the chromosome that represents the ER model of the Karate School is shown, which is depicted as an object diagram. This chromosome has one separator set to 1 at the 8th position, which means that the model should be partitioned into two modules.

3.3 Fitness Functions

We use five fitness functions based on the principles proposed by Moody et al. [20]. Table 1 summarizes these fitness functions.



Position	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Chromosome	1	0	4	0	9	0	10	1	2	0	3	0	5	0	6	0	7	0	8

Fig. 3: Encoding scheme using the Karate School example

First of all, Equation 2 shows the definition of a link in an ER model, which is basically the relation between instances of type Entity and Relationship. In order to obtain the sum of all links in an ER model, we use Equation 3. The first fitness function (cf. Equation 4) is *Cohesion* (COH), which measures the internal links within a module. Cohesion must be maximized in order to obtain modules where the elements are closely related to each other. The second fitness function (cf. Equation 5) is *Coupling* (COP), which represents the number of inter-module links. Coupling must be minimized, since we would like to reduce those links.

$$DER(e_i, r_j) = \begin{cases} 1 \text{ if entity } e_i \text{ has a Link to a relationship } r_j \\ 0 \text{ otherwise} \end{cases} \tag{2}$$

$$ERI(m_i, m_j) = \sum_{\substack{e_i \in E(m_i) \\ r_j \in R(m_j)}} DER(e_i, r_j) \tag{3}$$

$$Cohesion = \sum_{m_i \in Modules} ERI(m_i, m_i) \tag{4}$$

$$Coupling = \sum_{\substack{m_i, m_j \in Modules \\ m_i \neq m_j}} ERI(m_i, m_j) \tag{5}$$

Table 1: Summary of the fitness functions

ID	Description	Goal
COH	Cohesion: the sum of links within modules.	MAXIMIZE
COP	Coupling: the sum of links between modules.	MINIMIZE
NMOD	Number of modules.	MINIMIZE
AVGMODEL	Average number of modularizable elements per module.	MINIMIZE
BAL	The standard deviation of module size of all modules.	MINIMIZE

The objective of the third function is to minimize the number of the new created modules (NMOD). In this sense, it is not desirable to produce a sparse representation of the ER model. The fourth and fifth functions are metrics to measure the balance between modules of an ER model. An ER model has a good balance if every module is approximately equal in size [20]. In order to accomplish this, we propose two fitness functions, which are the average number of modularizable elements (AVGMODEL) represented in the Equation 7, and the standard deviation represented in the Equation 8. To calculate AVGMODEL we sum all the model elements using the Equation 6.

$$NMODELE(i) = \sum_{m_i \in Modules} E(m_i) + R(m_i) \tag{6}$$

$$AVGMODEL = \frac{\sum_{i=1}^{NMOD}(NMODELE(i))}{NMOD} \tag{7}$$

$$BAL = \sqrt{\frac{\sum_{i=1}^{NMOD}(NMODELE(i) - AVGMODEL)}{NMOD}} \tag{8}$$

3.4 Alterers

In order to create the offsprings, we use different alterer methods. Starting from the chromosome explained in Section 3.2, we make a transformation as shown in Fig. 4. The transformation consists in making a new chromosome that consists of two parts. The first part of the chromosome will be the genes that represent ModularizableElements, which is usually called an EnumerableChromosome and the second part will be the genes that represent the separators, called a BitChromosome.

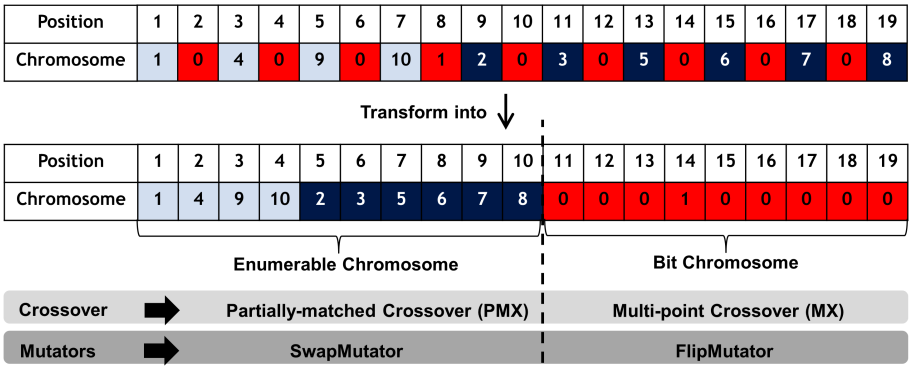


Fig. 4: Overview of the Alterer Operators

Crossover. For changing the produced initial population, in each iteration with some probability, two crossover algorithms will be applied: *Partially Matched Crossover* and *Multi Point Crossover*. Whereas the former is applied on the genes part that relates *ModularizableElement*s to a module, the latter is applied to the part that entails the module separators (Fig. 4).

The *Partially Matched Crossover* takes two selected parent individuals (P1 and P2) and two random position cut-points. The information given in the range between the two random positions will be stored in a mapping table. All occurrences of elements in this mapping table in P1 and P2 will be replaced by the corresponding counterpart thereby creating new offspring individuals P1' and P2' that will combine information coming from both parents. Fig. 5 (left) visualizes how the Partially Matched Crossover works.

The *Multi-Point Crossover*, shown in Fig. 5 (right), also takes two selected individuals P1 and P2 and also two random positions. The offspring individuals are then produced by only swapping the information within the boundaries of the two positions.

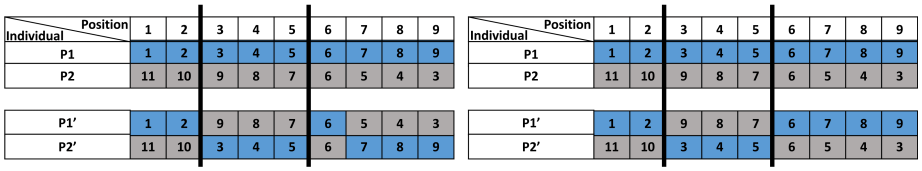


Fig. 5: Partially Matched Crossover (left) and Multi-Point Crossover (right)

Mutation. Mutation is another important part of genetic algorithms to encounter the natural mutation of species. In our approach, we use a *Swap Mutator* that applies to the first chromosome part and a *Flip Mutator* that applies to the second part of the chromosome (see Fig. 4).

By applying the *Swap Mutator*, the order of genes in a chromosome is changed, i.e., two genes within one chromosome are swapped, potentially changing the assignment of a *ModularizableElement* to a module. The *Flip Mutator* changes the number of modules by flipping the separator chromosomes either from 0 to 1, thereby adding a module and moving affected elements of the previous module into the new module, or from 1 to 0, thereby removing a module and merging the elements. Consequently, also the *Flip Mutator* changes the assignment of `ModularizableElements` to modules.

3.5 Constraints

Our approach limits the allowed combinations produced by the GA using constraints. Each produced individual is checked against the constraints, and, if it violates any of the them, an automated repair mechanism is applied. Consequently, the search space is significantly reduced and the search process is guided toward valid solutions as follows.

RelationshipToEntity. This constraint is violated in cases where a RelationshipType is not in one of the modules the two entities it relates are in. If a violation is detected, a repair function moves the RelationshipType instance to one of the two modules comprising the related entities. To ensure a good balance, the algorithm first selects the module that has fewer elements. In case both modules are equal in size, the algorithm selects the module randomly.

NumberOfModules. This constraint is violated in cases where Moody's principle that each diagram should have "seven plus or minus two" elements [20] is not met. The algorithm calculates the optimal number of modules and then only allows individuals with a number of modules in the range of the optimal number $\pm$ 50%. In case the number of modules is greater, the repair mechanism deletes randomly the minimum number of separators in the BitChromosome. Otherwise, separators are added at random positions in the BitChromsome until the criteria is met.

3.6 Tool Support

We developed an Eclipse plug-in, called *ModulER* [12], that supports the automatic modularization of ER models. *ModulER* uses the Eclipse Modeling Framework [28] and the Sirius [26] framework to implement a graphical editor. The editor allows creation and editing of modularized ER models and also provides the interface to execute the genetic algorithm. The algorithm was implemented with Jenetics[3]. *ModulER* is publicly available as a Github repository [12]. Fig. 6 visualizes the former example in our ModulER tool because this example was also used during the empirical experiments.

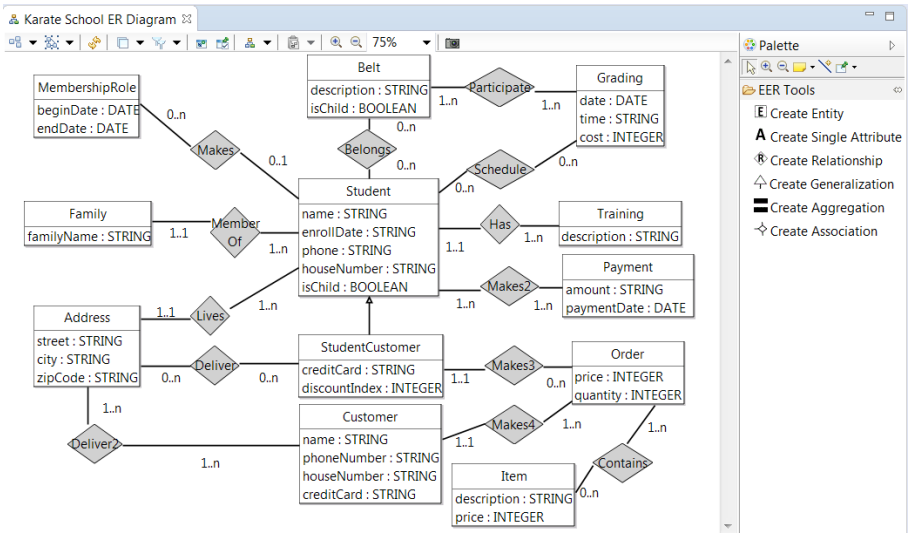


Fig. 6: ModulER graphical editor which shows in the canvas the Karate School example

[3] https://jenetics.io/, last visited: 24.09.2020

4 Evaluation

For the evaluation, we applied a multi-method approach that comprises a feasibility evaluation using two demonstration cases as well as an empirical evaluation using experiments with practitioners. In the next subsections, we describe our research questions, the experimental setup [12], and the metrics we use to answer these questions.

4.1 Research Questions & Evaluation Methods

Our study responds to three major research questions which may be further decomposed into sub-questions using different evaluation methods.

RQ1 Search Validation: Is the GA able to find valid modularization solutions efficiently? We use two diverging demonstration cases, (i) a quite compact ER model representing a student management system and (ii) a financial institution comprising much more elements. We then apply a sanity check and a quantitative analysis of the results. Furthermore, we measure the runtime performance of the algorithm. These tests were performed using Eclipse DSL Tools 4.14 in Windows 7 SP1, an Intel Processor Intel(R) Core(TM) i7-2600, 3.40GHz and 8GB of RAM memory. The Java virtual machine was configured with an initial and maximum memory size of 4GB using JAVA 11 (jdk-11.0.2) as an execution environment. We choose commonly used parameters to configure our GA [7]. We use a high probability (0.9) for the crossovers operators and a low probability (0.1) for our mutator operators. The population will consist of 200 individuals and in the Pareto Set will be 75 to 100 optimal solutions. The algorithm stops after 300 iterations.

RQ2 Solution Correctness

RQ2.1 How good are the GA solutions based on manual inspection? We manually inspect the solutions and evaluate, whether they are good in the sense of providing a heterogeneous Pareto set of solutions that optimize individual objectives while still fulfilling the constraints (cf. Section 3.5). Many approaches exist for selecting solutions form the Pareto set (e.g., knee-point analysis). In our initial study, we selected a sample solution s based on the minimum sum of ranks of all objectives O.

$$\text{RankSum}(s) = \sum_{o \in O} |s_k \in \text{Solution}, o(s_k) < o(s)|$$

RQ2.2 How good are the GA solutions compared to solutions a conceptual modeller would create? We implement an ad-hoc comparison using EMF Compare [9] to calculate the distance between the models created by 7 experiment participants and the Pareto set of solutions. Furthermore, we compare the GA and participant solutions based on the search objectives values.

RQ3 Search Objectives

RQ3.1 How do conceptual modelers quantify the different search objectives? We conduct a survey with the experiment participants and ask them to rate the perceived relevance of individual search objectives based on a scale from 1 (not relevant) to 10 (highly relevant).

RQ3.2 Does the perceived relevance correlate with the modularization? We compare the calculated metrics of the search objectives for the solution developed by each participant with his/her quantified relevance in the survey.

4.2 Evaluation Results

Search Validation. We can respond to **RQ1** by stating, that our approach is capable of modularizing even large ER models into valid modules. Table 2 shows the characteristics and the execution performance of the two cases. To measure the average computation time, we run the algorithm 30 times, and in both cases, the algorithm generated 75 solutions. It can be derived, that our approach is also very efficient in modularizing even large ER models like the Finance example that we received from an industrial partner.

Table 2: Characteristics of the two example cases and modularization performance.

Case	Entity Types	Relationship Types	Inheritance	Attributes	Avg. Computation time [Seconds]
Karate	12	13	1	29	1.8
Finance	116	17	97	67	6.8

Solution Correctness. By computing the Rank sums for both studies and investigating the results, we can confirm that the GA produces heterogeneous modularizations, each of which optimizing the different objectives to different extends. The Pareto set comprises modules with excellent balance, modules with high cohesion, and modules with low coupling. All solutions in the Pareto set fulfill the constraints. Thus, we can positively respond to **RQ2.1**.

In response to **RQ2.2**, we measured that on average, transforming a participants model into the closest one that exists in the Pareto set would need only a minimum of 5 movements. In the opposite direction, the Pareto set comprises solutions that outperform the participant's solutions in all objectives, i.e., cohesion, coupling, and balance (cf. Table 3 (right)). We thus state, that the solutions produced by the GA are reasonably similar while they are even better w.r.t. the search objectives compared to those solutions created by the conceptual modelers.

Search Objectives. Table 3 summarizes the results to **RQ3.1** by means of the survey responses. As can be seen, cohesion (COH) was considered more relevant than coupling (COP), whereas the balance (BAL) was considered least relevant for a good modularization. Although not yet implemented in our approach, we also wanted to know whether a thematic grouping is deemed relevant. The results indicate a high relevance, which is why we put it on our future research agenda.

Moreover, we are interested to check whether the perceived relevance denoted by a participant in the survey is also reflected in his/her modularization **(RQ3.2)**. From calculating the objectives values of cohesion (COH), coupling (COP), and balance (measured as standard deviation of module size (cf. Std. Dev. in Table 3)) for the participants'

Table 3: Search objectives relevance and computed participant's solution objectives

Participant	Search Objectives Relevance				Solution Objectives Values		
	COH	COP	BAL	Thematic	COH	COP	Std. Dev.
1	7	5	8	9	21	6	0.9428
2	10	7	3	10	20	7	0.9428
3	8	5	4	10	22	5	2.4944
4	8	6	3	8	21	6	2.357
5	7	6	4	8	22	5	3.2998
6	7	9	5	9	21	6	0.9428
7	7	6	2	10	20	7	4.9888

models, we see that all participants rated cohesion very high and this is also reflected in the modularization they produced (i.e., the cohesion of their modularized ER model was also high). For participant 1, balance was very relevant and the modularization had also the minimal standard deviation. All others denoted balance moderately or less relevant (values ranging from 2 to 5) but their module balance was very different. The values and measures for coupling were very homogeneous, thus we are not able to report any significant findings. Further studies need to investigate these aspects in more detail.

4.3 Critical Discussion

As any empirical study, the empirical evaluation presented in this paper is suspect to threats to validity [33]. From an internal and concept validity point of view, we made sure that all experiments were conducted in a controlled and equal environment. Participation was voluntarily and no reliability relationship exists between the participants and the researchers. A particular threat we could not mitigate is the limited sample size which hampers the conclusion validity and the external validity. Moreover, while the trial-and-error approach for finding parameters of search algorithms is frequently used [7], other parameters might yield different results and may allow further tuning for our given problem.

Besides these limitations, conducting these first experiments yielded meaningful insights that will be considered in our future research agenda. From observing the participants we learnt a great deal about different strategies they applied for identifying modules and/or moving ER elements into existing modules. One participant focused on the processes behind the data, another participant focused on the different purpose addresses and the corresponding stakeholders involved. Amongst all participants, a clear focus was on thematic grouping which also correlates to the high relevance it received in the survey (see Table 3).

5 Conclusion and Outlook

In this paper, we presented a flexible and efficient approach for automatically modularizing ER models based on a generic encoding and by employing a multi-objective

search. We reported two example case studies to evaluate the feasibility and the performance of the approach. Moreover, a survey with conceptual modelers yielded insights into how humans approach the modularization of ER models.

Future work aims at extending the approach with further objectives, e.g., thematic clustering, to extend the approach toward Extended ER concepts, and to further incorporate modelers in empirical studies. Moreover, we consider realizing a reinforcement learning approach inspired by [15]. Alternative modularizations from the Pareto set can be shown to the modelers of each GA generation in order to provide feedback that steers the algorithm towards the preferred solutions. Finally, we invite the conceptual community to join our efforts in developing an open environment for performing and understanding modularizations in conceptual models by making our tool open source [12].

References

1. Ambler, S.: http://www.agiledata.org/essays/agileDataModeling.html, (last visited: 24.09.2020)
2. Bork, D., Buchmann, R.A., Karagiannis, D., Lee, M., Miron, E.T.: An Open Platform for Modeling Method Conceptualization: The OMiLAB Digital Ecosystem. Communications of the Association for Information Systems **44**, pp. 673–697 (2019)
3. Bork, D., Karagiannis, D., Pittl, B.: A survey of modeling language specification techniques. Information Systems **87**, 101425 (2020)
4. Brambilla, M., Cabot, J., Wimmer, M.: Model-Driven Software Engineering in Practice, 2nd Edition. Synthesis Lectures on Software Engineering, Morgan & Claypool Publishers (2017)
5. Brin, S., Page, L.: The anatomy of a large-scale hypertextual web search engine. Computer networks and ISDN systems **30**(1-7), 107–117 (1998)
6. Deb, K., Agrawal, S., Pratap, A., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. **6**(2), 182–197 (2002)
7. Eiben, A.E., Smit, S.K.: Parameter tuning for configuring and analyzing evolutionary algorithms. Swarm Evol. Comput. **1**(1), 19–31 (2011)
8. Eiben, A., Smith, J.: Introduction to Evolutionary Computing, 2nd Edition. Springer (2015)
9. EMF Compare: EMF Compare. https://www.eclipse.org/emf/compare/, (last visited: 24.09.2020)
10. Feldman, P., Miller, D.: Entity model clustering: Structuring a data model by abstraction. The Computer Journal **29**(4), 348–360 (1986)
11. Fleck, M., Troya Castilla, J., Wimmer, M.: The class responsibility assignment case. In: Proc. of TTC 2016: 9th Transformation Tool Contest. pp. 1–8. CEUR-WS (2016)
12. Garmendia, A., Bork, D., Wimmer, M.: jku-win-se/module-eer: Second release of ModulER (Jul 2020), https://doi.org/10.5281/zenodo.3962651
13. Garmendia, A., Guerra, E., De Lara, J., García-Domínguez, A., Kolovos, D.: Scaling-up domain-specific modelling languages through modularity services. Information and Software Technology **115**, 97–118 (2019)
14. Guedes, G.B., Baioco, G.B., de Oliveira Moraes, R.L.: Evolutionary Database Design: Enhancing Data Abstraction Through Database Modularization to Achieve Graceful Schema Evolution. In: Proc. of the International Conference on Database and Expert Systems Applications. pp. 355–369. Springer (2016)
15. Kessentini, W., Wimmer, M., Sahraoui, H.: Integrating the designer in-the-loop for metamodel/model co-evolution via interactive computational search. In: Proc. of the 21th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems. pp. 101–111 (2018)

16. Mkaouer, W., Kessentini, M., Shaout, A., Koligheu, P., Bechikh, S., Deb, K., Ouni, A.: Many-objective software remodularization using NSGA-III. ACM Trans. Softw. Eng. Methodol. **24**(3), 17:1–17:45 (2015)
17. Moody, D.: A multi-level architecture for representing enterprise data models. In: Proc. of the International Conference on Conceptual Modeling. pp. 184–197. Springer (1997)
18. Moody, D.: The 'physics' of notations: toward a scientific basis for constructing visual notations in software engineering. IEEE Transactions on Software Engineering **35**(6), 756–779 (2009)
19. Moody, D.L.: Comparative evaluation of large data model representation methods: The analyst's perspective. In: Proc. of the International Conference on Conceptual Modeling. pp. 214–231. Springer (2002)
20. Moody, D.L., Flitman, A.: A methodology for clustering entity relationship models—a human information processing approach. In: Proc. of the International Conference on Conceptual Modeling. pp. 114–130. Springer (1999)
21. Mylopoulos, J.: Conceptual modelling and Telos. Conceptual Modelling, Databases, and CASE: an Integrated View of Information System Development pp. 49–68 (1992)
22. Özacar, T., Öztürk, Ö., Ünalır, M.O.: Anemone: An environment for modular ontology development. Data & Knowledge Engineering **70**(6), 504–526 (2011)
23. Rever Data Engineers: DB-MAIN, the data modeling and data architecture software (2019), `https://www.rever.eu/en/db-main/`, last visited: 24.09.2020
24. Rizzi, S.: Genetic operators for hierarchical graph clustering. Pattern Recognition Letters **19**(14), 1293–1300 (1998)
25. Simsion, G.: A structured approach to data modelling. Australian Computer Journal **21**(3), 108–117 (1989)
26. Sirius: Sirius. `https://www.eclipse.org/sirius/`, (last visited: 24.09.2020)
27. Sommerville, I.: Software engineering. Addison-Wesley (2011)
28. Steinberg, D., Budinsky, F., Merks, E., Paternostro, M.: EMF: Eclipse Modeling Framework. Pearson Education (2008)
29. Stuckenschmidt, H., Klein, M.: Reasoning and change management in modular ontologies. Data & Knowledge Engineering **63**(2), 200–223 (2007)
30. Tzitzikas, Y., Hainaut, J.L.: How to tame a very large ER diagram (using link analysis and force-directed drawing algorithms). In: Proc. of the International Conference on Conceptual Modeling. pp. 144–159. Springer (2005)
31. Verdonck, M., Hedblom, M., Guizzardi, G., Figueiredo, G., Gailly, F., Poels, G.: An Empirical Study Concerning Ontology-Driven Modularization and Ontology-Neutral Modularization Techniques. In: Proc. of the 13th Int. Workshop on Value Modelling and Business Ontologies (2019)
32. Villegas, A., Olivé, A.: A method for filtering large conceptual schemas. In: International Conference on Conceptual Modeling. pp. 247–260. Springer (2010)
33. Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslén, A.: Experimentation in software engineering. Springer Science & Business Media (2012)
34. Živković, S., Karagiannis, D.: Mixins and extenders for modular metamodel customisation. In: Proc. of the 18th International Conference on Enterprise Information Systems. pp. 259–270. SciTePress (2016)