Automatic Inference of Smart Data Discovery Interfaces for Rare Disease Datasets

Artur Boronat[1], Adekunle Ademeyo, Mehdi Mehtarizadeh[2] and Steffen Zschaler[3]

[1]School of Computing and Mathematical Sciences, University of Leicester, Leicester, UK
[2]e-Research Department, King's College London, London, UK
[3]Department of Informatics, King's College London, London, UK

Abstract

Café Variome is a flexible, web-based, data-discovery tool that offers a query language for checking the existence of biomedical data held in a federation of heterogenous data sources of rare diseases. Currently, there is a growing need for user-friendly graphical interfaces that are web-based to assist bioinformatics researchers look for cohorts of patients, across widely heterogenous data sources. Moreover, data owners who are usually clinicians, hospitals, or local trusts, lack the technical programming skills to create an interface that can help their data be queried and hence discovered.

In this work, we present the design of VForms, a platform for automatically inferring smart user interfaces (UIs) from heterogeneous datasets about rare diseases. This platform consists of a domain-specific modeling language (DSL) for specifying UIs for genomic datasets. A VForm is realized as a ReactJS web form that allows rare disease medical researchers define optimized queries over rare disease datasets. Hence each VForm is a domain-specific form-based query language. VForms infers the conceptual model from a given dataset, from which a domain-specific query UI in ReactJS is automatically generated using model-to-model transformations in YAMTL. The generation process is parameterized so that medical researchers can customize the generation of UIs.

The goal of this work is to demonstrate the potential of using model-driven engineering (MDE) to improve the development of UIs for genomic datasets of patients with rare diseases. By using VForms, data owners who lack the technical programming skills to create a UI that can help their data be queried and discovered, will be able to do so with minimal effort and in a more user-friendly way. Additionally, VForms allows for quick adaptation of the implementation of query interfaces to new user requirements.

Keywords

health data discovery, user interface generation, model-driven engineering, low-code development

1. Introduction

The difficulty in accessing and utilizing genomic data from rare diseases is often due to the information being spread across various heterogeneous data sources. While tools like Café

HEDA 2023: the 3rd International Workshop on Health Data (https://conf.researchr.org/home/staf-2023/heda-2023). Co-located with STAF 2023, 18–21 July, Leicester, United Kingdom.

† This research was partly funded by the UK Engingeering and Physical Sciences Research Council (EPSRC) through a seedcorn fund from MDENet (EP/T030747/1).

artur.boronat@leicester.ac.uk (A. Boronat); kunleadeyemo@rocketmail.com (A. Ademeyo); mehdi.mehtarizadeh@kcl.ac.uk (M. Mehtarizadeh); szschaler@acm.org (S. Zschaler)
https://arturboronat.github.io/ (A. Boronat); http://www.steffen-zschaler.de/ (S. Zschaler)
0000-0003-2024-1736 (A. Boronat); 0000-0001-9062-6637 (S. Zschaler)

CEUR Workshop Proceedings (CEUR-WS.org)

Variome [1] have made strides in enabling biomedical data querying, adapting such tools to new requirements is often labor-intensive and prone to errors. This situation is even more challenging for researchers without programming skills or those needing rapid modifications to query interfaces.

To overcome these limitations, we propose a Domain-Specific Modelling Language (DSML) and a proof-of-concept no-code development platform. Our solution, VForms, leverages model-driven engineering (MDE) [2, 3] to generate smart, customizable User Interfaces (UIs) from heterogeneous rare disease datasets, which define queries that are interpreted in the Café Variome API in SQL. The forms specified with VForms are automatically generated from the conceptual model of the dataset through YAMTL (Yet Another Model Transformation Language) model-to-model transformations [4, 5] and can be customized by researchers.

With VForms, we aim to enhance the development of UIs for genomic datasets, allowing data owners without technical programming expertise to create a UI and facilitating data querying and discovery. Moreover, it enables rapid adaptation of query interfaces to new user requirements.

The datasets we use comprise sensitive and heterogeneous data about patients with rare diseases, which present challenges in data complexity and privacy, requiring a user-centered design approach. Addressing these challenges benefits from a low-code software development approach for data discovery, which we implement in VForms.

The rest of this article presents a case study illustrating our approach, details the modeling languages utilized, explores how we used model-to-model transformation for generating smart UIs, and discusses the potential implications and future directions of our research.

2. Outline

In this section, we elaborate on the structure of our low-code development platform, VForms, and the various model management tasks that it enables for enhanced user interaction with large datasets.

Our approach is primarily built on three elements: a data description metamodel, enriching dataset features with descriptive statistics; a VForms Domain-Specific Language (DSL), for specifying the UI elements of a web form; and an edit metamodel, for modeling changes to the UI to track and represent changes.

We utilize Yet Another Model Transformation Language (YAMTL)[4, 5] as a critical tool for model-to-model transformation. YAMTL offers transformative features for models defined with the Eclipse Modeling Framework (EMF) [6]. It enables concise and efficient model transformations, pattern matching, trace management, and integrates with Java IDEs and libraries. Importantly, YAMTL allows for the seamless import of models from semi-structured datasets, reducing the need for boilerplate code and facilitating the analysis of diverse datasets.

The process of inferring a UI from a dataset follows two main data transformation steps, shown in Figure 2 (left). First, after loading the dataset into YAMTL, an object-based representation is produced, which informs the data description model extraction (`csv_to_dm`), involving the computation of descriptive statistics for each feature. Simultaneously, an initial edit model is built (`csv_to_edit`). This edit model will later be used by the UI front-end to record user-defined

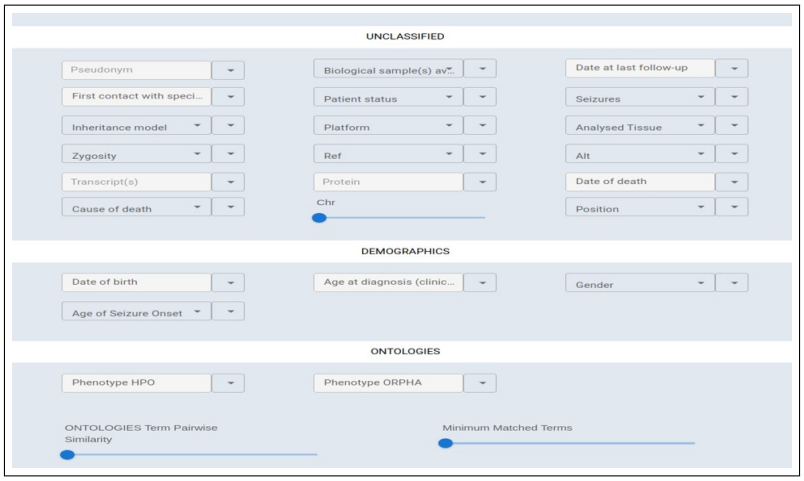

Figure 1: Example of a web form specified using VForms.

manual changes (e.g. the name to be used or the type of UI element chosen) to the generated UI. A second transformation (`dm_to_vforms`) creates a web form UI specification using the VForms metamodel, ensuring each feature is displayed using an appropriate UI element and enhancing user interaction with the data. Additionally, the edit metamodel allows for tracking UI changes, offering a detailed modification history, and simplifying undoing or redoing changes.

This transformation process is illustrated in a hypothetical genomic dataset. Let's assume the existence of a numerical feature `seizure_age`.When the data description model is obtained from the dataset, the feature `seizure_age` is augmented with statistics metadata, like the minimum age (e.g., 2 years), the maximum age (e.g., 80 years), the mean age (e.g., 45 years), and the standard deviation (e.g., 15 years). Specific patient ages are obfuscated for data protection by generating a frequency table that classifies patients into age ranges, defined by standard deviation intervals from the mean. This technique conceals individual data points, while preserving important statistical information.. In the second transformation, the `seizure_age` feature is translated into a VForms `FormInput` that allows users to input a numerical value or select a range, depending on the descriptive statistics associated with it. Lastly, let's say a researcher wants to change the label for the `seizure_age` input feature to `Age at Seizure` for clarity. They can make this change through the generated UI, which will update the JSON file that is sent to the backend and used to regenerate the UI.

The project implementing this proposal is available at https://github.com/arturboronat/mde-rare-diseases. Figure 1 depicts an example of a web form derived from a VForm model for one of the sample datasets used for the evaluation of the platform. Further details are elaborated in subsequent sections.

3. Conceptual Architecture

Our low-code development platform's conceptual architecture is constituted by three central elements: a data description metamodel, the VForms metamodel, and the edit metamodel.

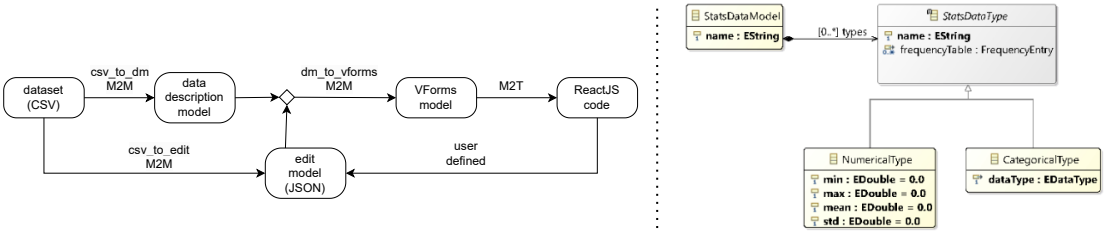

Figure 2: Outline of approach (left) and data description metamodel (right).

The *data description metamodel* enhances dataset features with descriptive statistics, facilitating an understanding of the data and pattern identification. These statistics provide an overview of the dataset values' distribution, outlining their shape, spread, and central tendency. Figure 2 (right) displays this metamodel, including the `StatsDataType` class which describes a dataset's value distribution and can provide minimum, maximum, mean, and standard deviation values. Its subclasses `CategoricalType` and `NumericalType` are employed for categorized data and numerical data, respectively.

VForms is a DSL that allows the specification of user interfaces (UIs) for genomic datasets, which are compiled to ReactJS web form. VForms facilitates data exploration across multiple sources and is adaptable to customize query interfaces based on user requirements. The main modeling primitives of VForms are shown in the metamodel in Figure 3. `FormInput`, a key component of the DSL, corresponds to an input element in a form and is specialized to capture the most representative UI elements that may appear in a genomic data discovery UI, such as those appearing in the screenshot in Figure 1. For example, `Age of Seizure` is a `FormInputSelect`, which shows as options the different age ranges, and a `FormInputGroup` is used to define the groups of form input elements appearing on the UIs.

The *edit metamodel* details the types of changes that can be made on a VForms model, facilitating UI element customization based on domain needs. This includes changes to the name displayed for a UI element or the type of UI element. The UI enables the modification of such elements and records user-defined changes in a JSON file that conforms to this metamodel. This JSON file is sent to the back-end, where it is taken into account for regenerating the UI.

4. Inference of VForm UIs from datasets

In this section, we explain the transformation process from datasets to UIs that help end users interact with the data. This process typically involves multiple steps, including data cleaning, data completion, and computation of descriptive statistics. These steps are important because they help to ensure the quality and accuracy of the data, as well as provide context and understanding for the end user. Once the data is in a suitable format, it is transformed into a UI model, such as VForms, which is to generate a web form in ReactJS.

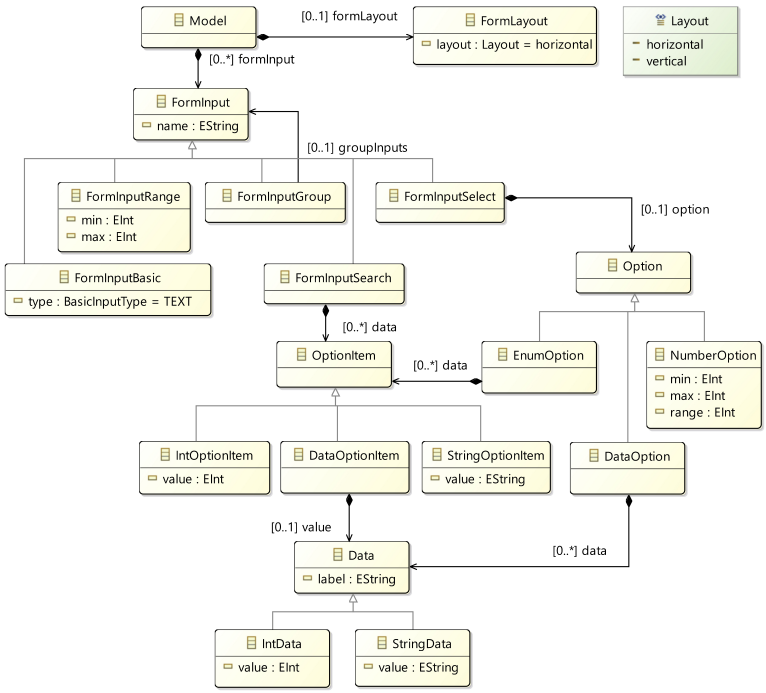

Figure 3: VForms DSL abstract syntax.

4.1. Generation of data description model

The first step in generating user interfaces for genomic datasets is to understand the structure and content of the data. The process begins by importing raw CSV data. An empty descriptive model structure, referred to as a `StatisticsDataModel`, is then created. This model serves as a container that will be populated with the transformed data. Next, each feature (or column) in the CSV dataset undergoes a transformation process. Two key transformation rules are applied, identifying features of type `CategoricalType` and of `NumericalType`. These transformations also generate frequency tables, capturing information about the distribution of the data values for each feature. For numerical features, each frequency entry corresponds to data values within ranges defined by standard deviation intervals from the mean. For categorical features, frequency entries are obtained for each unique category within the feature data. Data obfuscation is implicitly executed during these transformations, with the original data values being grouped into ranges or categories for frequency recording, thereby providing data protection.

The transformation process employs a simple classifier to effectively categorize features that are intended to appear together in a web form group. This classifier methodically examines the nature of each feature, including its data type and content, and assigns it to a particular group, such as demographics, DNA sequence, and ontologies[1], based on the patterns in the field values.

[1]The ontologies used in Café Variome are the Human Phenotype Ontology [7] and the OrphaNet Ontology [8], which are represented as directed acyclic graphs in a Neo4J database.

By doing so, the transformation process ensures the creation of intuitive and organized web forms where related features are collectively presented, thereby enhancing user navigation and interaction with the data. This process simplifies the task of editing the generated interface a posteriori embedding domain-specific knowledge in the transformation.

4.2. Generation of VForms UIs

The next phase is transforming the data description model into VForm UI models. This process uses a model-to-model transformation to create the VForms UI specification, which is then transformed into a web form using ReactJS. The transformation is parametrized with an edit model, which allows domain-specific UI rules to override default behaviour. The process involves several steps, including initializing the VForms model and layout, forming groups for various types of form inputs, creating different types of form inputs depending on the data type and characteristics, and forming ranges for numerical data. These steps help present complex genomic data in an intuitive, user-friendly format, thereby enhancing data discovery in rare disease research.

The transformation of the data description model into a VForms model is a key process that aids in presenting complex genomic data in a user-friendly and intuitive format. This automated and customizable transformation significantly enhances the process of data discovery in the context of rare disease research. The platform has been evaluated with the same mock datasets used to develop data discovery UIs within Café Variomé, mimicking real world data. The generated UIs exhibited comparable expressivity, successfully reflecting the data discovery facilities available in Café Variomé. In addition, the UI editing facilities avoid numerous iterations with software developers to refine the UIs used to investigate genomic data.

5. Conclusions

We have developed a system for automatic user interface generation from genomic datasets associated with rare diseases, emphasizing data privacy and obfuscation. Utilizing a low-code development approach, we use a model-driven procedural approach to understand the structure of genomic data and generate an effective, user-friendly interface, by augmenting data with descriptive statistics. This interface can be customized via edit models to suit specific data discovery tasks. Our approach demonstrates the potential of low-code software development for data discovery, offering a robust solution for modeling and querying rare disease genomic datasets. This paves the way for accessible and intuitive genomic data exploration, contributing to advancements in understanding and treating rare diseases.

MDE, and particularly YAMTL, was used to effectively abstract from low-level implementation specifics, including semi-structured data formats and ReactJS details, facilitating more focused data design. Additionally, model transformations have underscored the principle of separation of concerns, with each metamodel serving a distinct function. Morever, the current platform serves as the basis for exploring additional ML techniques that help learn from semistructured data. Future work includes a more thorough evaluation of the effectivenes and usability of the generated UIs, analysing the advantages and limitations of our approach.

References

[1] O. Lancaster, T. Beck, D. Atlan, M. A. Swertz, D. Thangavelu, C. D. Veal, R. Dalgleish, A. J. Brookes, Cafe variome: General-purpose software for making genotype–phenotype data discoverable in restricted or open access contexts, Human Mutation 36 (2015).

[2] D. C. Schmidt, Guest editor's introduction: Model-driven engineering, IEEE Computer 39 (2006) 25–31. URL: http://dx.doi.org/10.1109/MC.2006.58. doi:10.1109/MC.2006.58.

[3] D. D. Ruscio, D. Kolovos, J. de Lara, A. Pierantonio, M. Tisi, M. Wimmer, Low-code development and model-driven engineering: Two sides of the same coin?, Software and Systems Modeling 21 (2022) 437 – 446.

[4] A. Boronat, Expressive and efficient model transformation with an internal dsl of xtend, in: Proceedings of the 21th ACM/IEEE International Conference on MoDELS, ACM, 2018, pp. 78–88.

[5] A. Boronat, Incremental execution of rule-based model transformation, International Journal on Software Tools for Technology Transfer 1433-2787 (2020). URL: https://doi.org/10.1007/s10009-020-00583-y. doi:10.1007/s10009-020-00583-y.

[6] D. Steinberg, F. Budinsky, M. Paternostro, E. Merks, EMF: Eclipse Modeling Framework 2.0, 2nd ed., Addison-Wesley Professional, 2009.

[7] S. Köhler, M. Gargano, N. Matentzoglu, L. C. Carmody, D. Lewis-Smith, N. A. Vasilevsky, D. Danis, G. Balagura, G. Baynam, A. M. Brower, T. J. Callahan, C. G. Chute, J. L. Est, P. D. Galer, S. Ganesan, M. Griese, M. Haimel, J. Pazmandi, M. Hanauer, N. L. Harris, M. Hartnett, M. Hastreiter, F. Hauck, Y. He, T. Jeske, H. Kearney, G. Kindle, C. Klein, K. Knoflach, R. Krause, D. Lagorce, J. A. McMurry, J. A. Miller, M. Munoz-Torres, R. L. Peters, C. K. Rapp, A. M. Rath, S. A. Rind, A. Rosenberg, M. M. Segal, M. G. Seidel, D. Smedley, T. Talmy, Y. Thomas, S. A. Wiafe, J. Xian, Z. Yüksel, I. Helbig, C. J. Mungall, M. A. Haendel, P. N. Robinson, The Human Phenotype Ontology in 2021, Nucleic Acids Research 49 (2020) D1207–D1217. URL: https://doi.org/10.1093/nar/gkaa1043. doi:10.1093/nar/gkaa1043. arXiv:https://academic.oup.com/nar/article-pdf/49/D1/D1207/35364524/gkaa1043.pdf.

[8] A. Rath, A. Olry, F. Dhombres, M. M. Brandt, B. Urbero, S. Ayme, Representation of rare diseases in health information systems: The orphanet approach to serve a wide range of end users, Human Mutation 33 (2012) 803–808. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/humu.22078. doi:https://doi.org/10.1002/humu.22078. arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/humu.22078.