Contents lists available at ScienceDirect

The Journal of Systems & Software

journal homepage: www.elsevier.com/locate/jss

UVLHub: A feature model data repository using UVL and open science principles[☆]

David Romero-Organvidez [a,d,*], José A. Galindo [a,d], Chico Sundermann [b], Jose-Miguel Horcas [c], David Benavides [a,d]

[a] *Department of Computer Languages and Systems, University of Seville, Spain*
[b] *Institute of Software Engineering and Programming Languages, Ulm University, Germany*
[c] *ITIS Software, Universidad de Málaga, Spain*
[d] *I3US Research Institute, Universidad de Sevilla, Seville, Spain*

ARTICLE INFO

Dataset link: https://github.com/diverso-lab/uvlhub, https://www.uvlhub.io

Keywords:
Feature models
Software product line
Variability
Dataset
Uvl

ABSTRACT

Feature models are the *de facto* standard for modelling variabilities and commonalities in features and relationships in software product lines. They are the base artefacts in many engineering activities, such as product configuration, derivation, or testing. Concrete models in different domains exist; however, many are in private or sparse repositories or belong to discontinued projects. The dispersion of knowledge of feature models hinders the study and reuse of these artefacts in different studies. The Universal Variability Language (UVL) is a community effort textual feature model language that promotes a common way of serializing feature models independently of concrete tools. Open science principles promote transparency, accessibility, and collaboration in scientific research. Although some attempts exist to promote feature model sharing, the existing solutions lack open science principles by design. In addition, existing and public feature models are described using formats not always supported by current tools. This paper presents `UVLHub`, a repository of feature models in UVL format. `UVLHub` provides a front end that facilitates the search, upload, storage, and management of feature model datasets, improving the capabilities of discontinued proposals. Furthermore, the tool communicates with Zenodo – one of the most well-known open science repositories – providing a permanent save of datasets and following open science principles. `UVLHub` includes existing datasets and is readily available to include new data and functionalities in the future. It is maintained by three active universities in variability modelling.

1. Introduction

Feature models are widely used for variability modelling in many domains, especially in software product line engineering (Felfernig et al., 2024). Feature models have been widely adopted in practice and academia since their invention in 1990 (Sochos et al., 2004). Applications of feature models include operating systems (Galindo et al., 2010, 2016; Sincero et al., 2007), content-management systems (Halin et al., 2020; Rodas-Silva et al., 2019) and the automotive industry (Felfernig et al., 2018; Le et al., 2023) among many others. Feature models are used in those domains for many engineering tasks such as automated analysis (Galindo et al., 2019b), sampling (Segura et al., 2007), testing (Segura et al., 2014), debugging (Noorian et al., 2011), and even teaching (Webb and Kuzmycz, 1995).

Although widely used, feature models are often shared in private web pages or are spread across different platforms such as code repositories (e.g., GitHub), personal websites, links to the Zenodo repository (Ramachandran et al., 2021) or discontinued projects. In the past, there were some efforts aiming feature model sharing, such as SPLOT (Mendonça et al., 2009b), ESPLA (Martinez et al., 2017), and

☆ Editor: Laurence Duchien.
* Corresponding author.
E-mail addresses: drorganvidez@us.es (D. Romero-Organvidez), jagalindo@us.es (J.A. Galindo), chico.sundermann@uni-ulm.de (C. Sundermann), horcas@uma.es (J.-M. Horcas), benavides@us.es (D. Benavides).
[1] LVAT: https://gsd.uwaterloo.ca/feature-models-in-the-wild.html.

https://doi.org/10.1016/j.jss.2024.112150
Received 27 July 2023; Received in revised form 18 February 2024; Accepted 24 June 2024
Available online 1 July 2024

LVAT.[1] While currently available, these repositories come with limitations and miss today's enforcement for open science because, maybe, they were not designed with this principle in mind. In addition, the maintenance of such repositories is resource-consuming, and in the case of shutting down the server, all the data would be lost. Finally, existing repositories soon become outdated because of their technical debt (Horcas et al., 2023). A large part of a project's resources is spent on technical debt management (Ernst et al., 2015).

Since 2018, there have been efforts in the community to develop a variability language to facilitate knowledge sharing. Concretely, three main software artefacts are proposed by the MODEVAR community (Benavides et al., 2019). First, a textual feature modelling language called *Universal Variability Language* (UVL) (Benavides et al., 2024; Sundermann et al., 2021a, 2023b), so the feature models can be shared in the same format and be easily understood by the practitioners with little effort.[2] Second, tools supporting the usage of UVL and, for that, parsers and adaptations to various well-known feature model modelling and analysis tools such as FeatureIDE (Sundermann et al., 2021b) and `flamapy` (Galindo et al., 2023). Finally, a central open science repository for the UVL language sharing.

Open Science (Ramachandran et al., 2021) initiative was championed by the OECD[3] nations and the European Commission. It underscores the importance of unrestricted and global access to scientific research outcomes, ideas, and resources. Different initiatives were presented, such as OpenAIRE (Czerniak et al., 2019), an abbreviation for Open Access Infrastructure for Research in Europe, which offers online services that help researchers meet open access requirements. Also, Invenio (Caffaro and Kaplun, 2010), another initiative promoted by CERN, is an open-source software architecture that facilitates the creation and management of digital libraries. This technological infrastructure finds widespread application in institutions that require a robust solution to manage voluminous digital data and documents. One of the most well-known tools is Zenodo.[4] This open-access research repository enables researchers in all disciplines to store, share and expose a wide range of research outputs, including datasets, software, publications, and other research material (Nielsen et al., 2022). With features like the assignment of Digital Object Identifiers (DOIs), Zenodo enhances resource traceability and visibility and simplifies the citation process. It guarantees the long-term persistence of stored data, thus ensuring their future availability for research purposes.

In this paper, we present `UVLHub`[5] with the following main characteristics:

- **Adherence to open science principles**. `UVLHub` is built upon open science technologies backed by the European Research Council (Katoch et al., 2019), which ensures future accessibility of uploaded models to maintain replicability. Using Zenodo as a persistent data layer also grants this open–science adherence and offers benefits such as having *DOIs* for feature model datasets or having a persistent accessible *backup copy*. Although `UVLHub` is supported by three different universities, in the event of a discontinuation of the initiative, the datasets will still be available in Zenodo, a project supported by several international organizations.
- **Specifically designed for feature models**. Although open science tools are available for all researchers, there is a lack of specialization for feature model sharing. `UVLHub` offers a front-end for researchers and practitioners to facilitate feature model sharing. It uses the UVL language to have a homogenized file format compatible with a bigger set of tools. Only UVL feature model files can be uploaded to `UVLHub`. This check prevents all kinds of files, such as video, audio or scripts Zenodo allows. With this functionality, it is easy to offer advanced search options specific to feature models, such as looking for models with a range of features, relationships, or specific characteristics. Also, additional capabilities can be added, such as performing automated analysis or online editions of feature models.
- **Existing feature models already available**. Many models are dispersed around different sources and in different formats. `UVLHub` group real-world and academic feature models from diverse sources such as the SPLOT repository in UVL format. First, we scrapped all the repositories of models that we were aware of. Second, we translated those models to UVL, and finally, we populated the datasets to `UVLHub` and automatically synchronized them with Zenodo.

We also examine the adherence of `UVLHub` to Open Science principles and its usability by addressing the following two research questions:

1. **RQ 1:** *Can feature model sharing be modernized following open science principles?* This research question attempts to compare the inclusion of new features following Open Science principles to other similar solutions that do not have such features.
2. **RQ 2:** *Is the use of a specialized repository for feature models in UVL format useful for the community?* In this research question, we study whether designing a solution that focuses on UVL models and provides additional features is useful for the scientific community.

The remainder of this paper is organized as follows. Section 2 gives a brief introduction to feature models and UVL. Section 3 describes the methodology used to design the architecture, functionality and evaluation of the proposal. Section 4 describes the main elements of the solution and lists the various feature model datasets that are used to populate the tool. Section 5 evaluates our proposal detecting deficiencies and areas for improvement, and discusses research questions. Section 6 compares existing repositories. Finally, Section 7 provides some conclusions and proposes future work.

2. Feature models and UVL

A *software product line* (SPL) can be used to manage related products (Clements and Northrop, 2001). Each product comprises separate and reusable units called *features*. Not every combination (or *configuration*) of such features typically results in a consistent product. *Feature models* are commonly employed to specify the valid configurations of a (software) product line. The specification of the valid configurations with a feature model is typically achieved by imposing constraints that disallow certain combinations of features (Batory, 2005; Kang et al., 1990). For example, including a feature may automatically require to exclude another feature.

2.1. Structure of feature models

Feature models are typically separated into a tree structure describing the feature hierarchy and additional cross-tree constraints. Commonly, cross-tree constraints are expressed as formulas in propositional logic (Acher et al., 2011, 2013; Batory, 2005; Meinicke et al., 2017; Mendonça et al., 2009a; Plazar et al., 2019; Pohl et al., 2011; Sundermann et al., 2021a, 2023a). While, in some cases, more sophisticated expressions, such as constraints over numerical values, are also considered (Munoz et al., 2019; Siegmund et al., 2017; Sprey et al., 2020; Sundermann et al., 2023c), we consider cross-tree constraints as arbitrary propositional boolean formulas that are those currently supported in UVL. Fig. 1 shows an example feature model specifying the valid configurations of a simple chat service. The selection of a child feature always requires the selection of its parent. For example,

[2] UVL website: https://universal-variability-language.github.io/.
[3] OECD: https://www.oecd.org/.
[4] Zenodo: https://zenodo.org/.
[5] UVLHub: www.uvlhub.io.

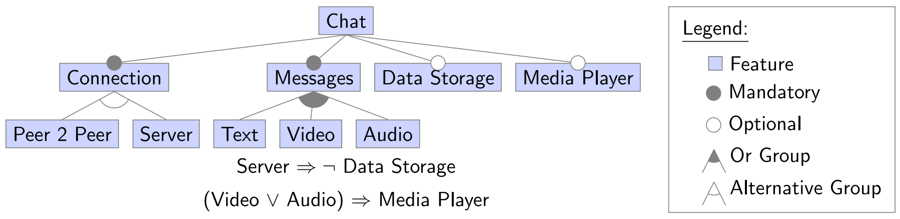

Fig. 1. Feature model specifying a simplified chat service.

selecting the `Text` feature always requires selecting `Messages`. The following types of hierarchical constraints are typically supported in feature models (Knüppel et al., 2017a):

- Mandatory/Optional: A *mandatory* feature must always be selected if its parent is selected. For instance, each chat service requires a `Connection` and support `Messages`.
- Or: An *or*-group between a parent feature and its children indicates that at least one child needs to be selected if the parent is selected. For instance, messages of at least one type are required.
- Alternative: An *alternative*-group between a parent feature and its children indicates that exactly one child needs to be selected if the parent is selected. For instance, the connection can be either `Peer 2 Peer` or `Server` but not both.
- Group Cardinality: A *cardinality* [n..m] indicates that between n and m child features need to be selected if the parent is selected. For instance, the or-relation below `Messages` can also be encoded as [1..3] cardinality.

Cross-tree constraints can further limit the set of valid configurations. These are typically limited to propositional logic (Acher et al., 2013; Batory, 2005; Meinicke et al., 2017) or even further restricted to `REQUIRES` (i.e., force selection) and `EXCLUDES` (i.e., forbids selection) constraints (Kang et al., 1990). In our running example, having a `Server` is always required to have a `Data Storage`. If the chat service supports at least one of `Video` or `Audio`, a `Media Player` is required.

2.2. Universal variability language

The *Universal Variability Language (UVL)* is a textual specification for feature models (Sundermann et al., 2021a). UVL has been developed in tight cooperation with the community within the MODEVAR initiative (Sundermann et al., 2021a).

Listing 1 shows our running example in UVL notation. The UVL model is separated into two main parts describing the feature tree and cross-tree constraints, respectively. The feature tree is introduced with the `features` keyword. As one design goal of UVL is its readability in textual format, UVL mimics the tree hierarchy of feature models with an indentation-based format. The hierarchical constraints are realized via the keywords `mandatory`, `optional`, `or`, and `alternative`.

Listing 1: Running example in UVL.

```
features
    Chat
        mandatory
            Connection
                alternative
                    "Peer 2 Peer"
                    Server
            Messages
                or
                    Text
                    Video
                    Audio
        optional
            "Data Storage"
            "Media Player"

constraints
    Server => "Data Storage"
    Video | Audio => "Media Player"
```

Cross-tree constraints are introduced with the `constraints` keyword in UVL models. To specify the propositional cross-tree constraints, `=>` (imply), `<=>` (bi-imply), `&` (and), `|` (or), and `!` (negation), and parentheses are available. Note that the constraints can be arbitrarily nested.

3. Methodology

We designed, evaluated, and refined `UVLHub` using the methodology described in this section. Our methodology was inspired by design science (Wieringa, 2014), with one evaluation and refinement cycle. First, we synthesized the requirements of `UVLHub` by observing repositories of features models (e.g., SPLOT) and general repositories (e.g., Zenodo), as well as after some brainstorming discussions among the authors. We shared and got feedback from the community (Galindo and Benavides, 2019). We elicited the main functionalities and a first version of the architecture from the requirements and discussed and validated with the community (Romero-Organvidez et al., 2021). Next, we combined hands-on and feedback from the SPL community through our participation in the different editions of the MODEVAR workshop (from 2019 to 2022) (Benavides et al., 2019) to design and develop the current version of `UVLHub`. Last, we performed a practical evaluation with practitioners in the last edition of MODEVAR workshop (2024)[6] and based on the feedback from the community evaluation, we refined `UVLHub`.

3.1. Data sources

Our methodology relied on the following data sources:

Repository requirements. In a previous work (Galindo and Benavides, 2019), we presented 12 characteristics that a feature model repository should have. These characteristics were defined by observing SPLOT and other repositories such as Zenodo and after some brainstorming discussion among the authors. Then, the characteristics were discussed with the community in the first edition of the MODEVAR Workshop (Benavides et al., 2019). We synthesized `UVLHub` to align with and fulfil these characteristics, considering the feedback received (Section 5.1.1).

[6] 6th MODEVAR 2024: https://modevar.github.io/.

Author's experiences. Our experience with SPL and feature models ranges between 3 and 20 years. We have collaborated with many partners in funded research projects, and developed large framework projects (Benavides et al., 2007; Galindo et al., 2023). We are also embedded in the research community, having contributed tools (Galindo et al., 2023; Horcas et al., 2022; Sundermann et al., 2021b), as well as systematic studies (Benavides et al., 2010; Galindo et al., 2019a).

Experts in feature models and SPL. We used qualitative feedback from experts known in the academic and SPL community who participated in the MODEVAR workshops. Through the proceedings of the SPLC, VaMoS, and Configuration Workshop, we also sought the flagship and well-recognized venues in SPL and variability modelling. We searched for experience reports and tool track papers where authors reported their experiences working and managing feature models in their experiments or mentioned repositories and datasets of feature models.

3.2. Architecture design

We structured the architecture of `UVLHub` into five main components. The components comprised (1) a web application, (2) a local storage, (3) an external stable repository, (4) support for automated analysis of feature models, and (5) a REST API. We started by carefully analysing our requirements to identify the technological stack and build the language infrastructure. The architectural design is based on the following requirements:

Light. `UVLHub` should rely on a small core of functionalities due to the number of optional characteristics identified that a feature model repository may have.

Reliable. The core component `UVLHub`, should be backed up by a larger project to mitigate the effects of the technical debt and avoid the proliferation of multiple discontinued repository projects.

Extensible. `UVLHub` should provide other functionality related to the feature models stored in the repository, such as the evaluation of metrics for the models, the user's rating of models, or providing a REST interface to access the models.

Open science. `UVLHub` should be built upon open science technologies to ensure accessibility and replicability.

We made the following design decisions to align and adhere to these requirements. First, we chose to use the micro-framework Flask,[7] which provides a light and extensive core for a web application and is implemented in Python. Python was chosen as the primary programming language due to the rise of machine learning and data science, along with the robust Python-based libraries that currently exist that provide support for the development of websites, task automation, data analysis, and data visualization. The Flask web application supports basic functionality, such as accepting requests and building our application. Second, the local storage component acts as a core asset that provides the basic functionality of hosting and organizing the models. At this point, for reliability, we choose Zenodo as an external repository for permanent persistence and preventing data loss. The advantages of using Zenodo have been discussed in Section 1. Moreover, Zenodo is a solution stack relying on Invenio, which was initially developed at CERN and holds an open-source licence, which aligns well with the requirement of reusing and promoting open-science previous efforts. Then, to provide extensibility that allows incorporating additional functionality and integration with different tools, we use a plugin system through an REST API interface to consume it following the standard HTTP protocol. We instantiate an extension point with an essential component supporting the automated feature model analysis. This component allows obtaining different metrics and analyses used in the repository to present and search for models. The resulting architecture is detailed in Section 4.3.

3.3. Experts evaluation

Following the development of `UVLHub`, we evaluated it at the latest edition of the MODEVAR workshop (as part of the VaMoS 2024 conference). After an introductory meeting, we provided the participants with a description of `UVLHub` to familiarize them with it and asked them to complete a questionnaire. The questionnaire comprised three questions about the different functionalities based on the requirements of `UVLHub`. The questionnaire aimed to determine how `UVLHub` was perceived by the participants in different aspects (e.g., usability, usefulness, effort, clarity of the website and forms). The first ten questions were designed to be yes/no answers and the completion time of each of the ten proposed tasks. Ten questions (see Section 5.3) were designed to be answered on a five-point Likert scale (strongly disagree, disagree, neutral, agree, strongly agree) and aligned with a satisfaction survey about the aspect we wanted to evaluate. We also incorporated four questions aimed at getting subjective opinions from the participants and were therefore designed to be open-ended. The details of our evaluation are presented in Section 5.

4. `UVLHub` Description

This section provides an overview of the main elements of the `UVLHub`. We present the different facets considered when developing the `UVLHub`. First, we describe the goal, the participants and the usage scenarios. Then, we describe the main parts of the `UVLHub`, the functionality and how it aligns with Open Science principles. Third, we describe the project architecture in which we show the different modules that compose the different functionalities of the repository. Fourth, we present the domain model of the data we store; and finally, the life cycle of the dataset. Source code is publicly available on GitHub.[8]

4.1. Goals, participants and usage scenarios

We surveyed existing repositories to identify those functionalities needed in our repository (see Section 6). We follow the user story pattern: *"As a [persona], I [want to], [so that]."*. That pattern represents a characteristic or functionality within our tool. In this section, we go through these functionalities. First, let us describe the functionalities that have already been implemented:

1. *As a researcher, I want to upload feature models so that I can share my work easily*. For researchers, a hassle-free model uploading functionality is essential for sharing their work.
2. *As a researcher, I want to perform syntax checking before uploading the model so that my models are error-free*. Before uploading, researchers need to ensure that their models are free from syntax errors. `UVLHub` supports this requirement, accommodating the UVL notation (Sundermann et al., 2021a,b).
3. *As a user or researcher, I want different download options so that I can use existing models effectively*. A user-friendly model download system is necessary for replicating experiments or reusing models. For example, we are interested in selecting specific dataset models and downloading them all at once.

[7] Flask: https://flask.palletsprojects.com/.

[8] UVLHub source code: https://github.com/diverso-lab/uvlhub.

4. *As a researcher, I want unique identifier generation for my models so that they can be cited and identified easily*. Uniquely identifying models with a universal identifier allows for efficient citation and identification. In this aspect, UVLHub is innovative, relying on the Zenodo DOI generation specific to each feature model dataset.
5. *As a user, I want enhanced model searching capabilities to find models I am interested in*. Advanced search options enable users to locate relevant models quickly. UVLHub enhances search functionality with options like metadata and metrics. For example, listing the models ranging from 10 features to 100 would be required.
6. *As a user or researcher, I want access to model popularity indicators to compare models*. Indicators such as download counts or ratings allow users to gauge a model's popularity. UVLHub records this information as part of the model's metadata.
7. *As a developer or researcher, I want API access for repository interaction to access the repository's information programmatically*. An API that allows programmatic access to the repository is a valuable characteristic for developers and researchers. UVLHub offers an API enabling developers to programmatically access and search repository information.
8. *As a user, I want a detailed display of model metrics so that I have extra information about the models*. Displaying model metrics enables a more detailed selection of the models within the repository. Integrating feature model analysis technologies within UVLHub can furnish detailed metrics like the number of features, structural metrics (Bagheri and Gasevic, 2011), and other analysis operations (Benavides et al., 2010).

The functionalities currently under active development are the following:

1. *As a researcher, I want to compare different model versions*. Managing model versions is an essential characteristic for comparison. Existing solutions store versions as separate files, whereas UVLHub plans to offer in-built version control.
2. *As a user, I want comprehensive model visualization and metrics display to understand the model before using*. The visual representation and insight on metrics give users a quick understanding of a model. UVLHub delivers graphical and textual representations supplemented with valuable metrics. This functionality integrates the *FM Fact Label* visualization proposed by Horcas et al. (2022) with rights visualization (Romero-Organvidez et al., 2024).
3. *As a user, I want profile-based model recommendations to make better model selections*. A personalized recommendation system can enhance the model selection. Though this is a long-term characteristic, UVLHub is equipped to develop such a system by leveraging stored metrics and user profiles.

4.2. UVLHub Overview

Our proposal seeks to add value by creating an intermediary between the researcher or UVL model developer and Zenodo, which follows Open Science principles. Fig. 2 shows these principles' alignment and each layer's functionality. UVLHub already provides the principles of Open Source (available code), Open Methodology (the user can provide extra information about the models and how they have been generated) and Open Access (any user can download, visualize and redistribute the models). For its part, Zenodo carries the weight of the principles of Open Educational Resources (publicly available material), Open Peer Review (uploaded datasets have helpful metadata such as the name, surname and ORCID of the authors) and Open Data (data available openly and free of charge).

UVLHub integrates an abstraction layer on top of Zenodo, allowing exclusive upload of model datasets in UVL format. This layer ensures control over content and users, paving the way for future expansions, including software artefacts. Zenodo, supported by CERN and powered by Invenio, ensures permanent storage.

Highlights of UVLHub's functionalities include an advanced search by various criteria such as author, model type or date, and an automatic validation that ensures the syntactic correctness of the supported UVL models.

4.3. Architecture

We have followed the methodology described in Section 3 and for the design of our architecture (see Section 3.2). This architecture design allows to implement a solution that satisfies the requirements of a light, reliable, extensible tool that follows Open Science principles.

The architecture of UVLHub is composed of five main components. These components are (1) web application, (2) local storage, (3) Zenodo, (4) automated analysis of feature models, and (5) REST API. All these components are interconnected through the web application, which can be extended to more services. Fig. 3 shows the architecture overview of UVLHub along with the interconnection of its main components and the role played by each one.

1. **Web application**. The main access to the web application is through a web browser. The user accesses a domain that conveniently redirects to the application (1) developed in Flask.[9] UVLHub combines four different services, which are (2) the local storage of UVL models and relevant information, (3) the Zenodo service for permanent persistence, (4) the automatic analysis, and (5) a RESTful service to extend the functionality to other domains.
2. **Local storage**. Users upload their models in UVL format. All uploaded models must be syntactically valid, i.e., present a validated syntax concerning UVL grammar. However, models can contain semantic errors such as dead features or contradicting constraints (Benavides et al., 2010). Note that UVL files are stored locally, while all relevant information, such as title, description, authors, and tags, are stored in a relational database.
3. **Zenodo**. While we store some data from the models, the local storage is backed up in the Zenodo general repository. Moreover, thanks to this, the UVL datasets get a DOI, simplifying the process of obtaining the identifier. Eventually, if UVLHub becomes unavailable, the datasets are still in Zenodo, and the local storage can be rebuilt without data loss.
4. **Automated analysis of feature models (AAFM)**. Eventually, the user can analyse (Benavides et al., 2010) the models of each of his datasets uploaded to UVLHub. This analysis relying on the flamapy tool suite (Galindo et al., 2023) could result in, for example, whether a model uploaded to UVLHub is valid, the number of features the model has, or the number of different products that can be obtained. This component ensures that each model uploaded by the user is syntactically correct, delegating the responsibility of the syntax verification to UVLHub, thus avoiding the need for the user to perform this check.
5. **REST API**. Open Science promotes open access to research data and seeks to make data generated in scientific research freely available and accessible to other researchers, practitioners, and the general public. To push in that direction, UVLHub provides a REST API accessible to any user who registers with the developer role. This access to any dataset allows freely integrating models already validated and eventually analysed in (4) to apply them to other domains.

[9] Flask: https://flask.palletsprojects.com.

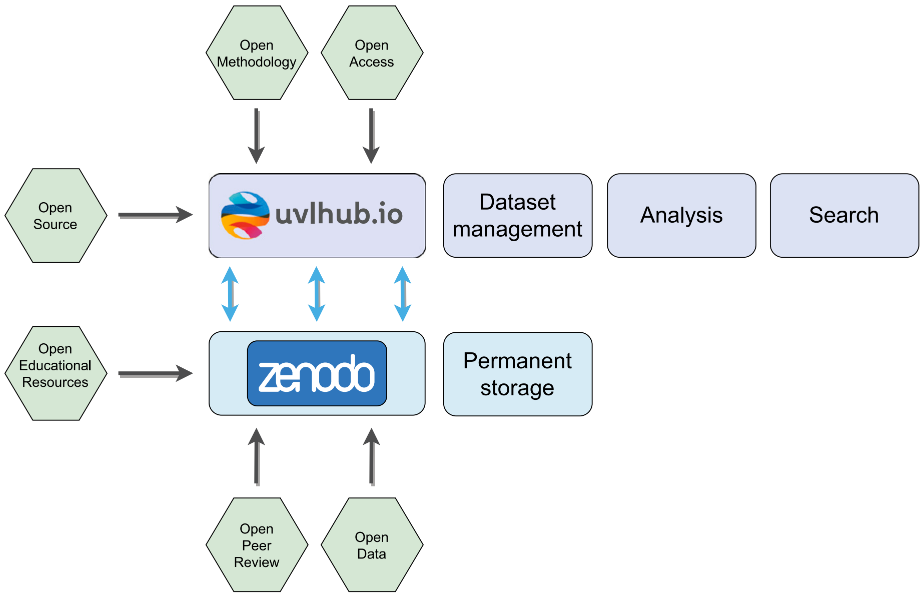

Fig. 2. UVLHub overview.

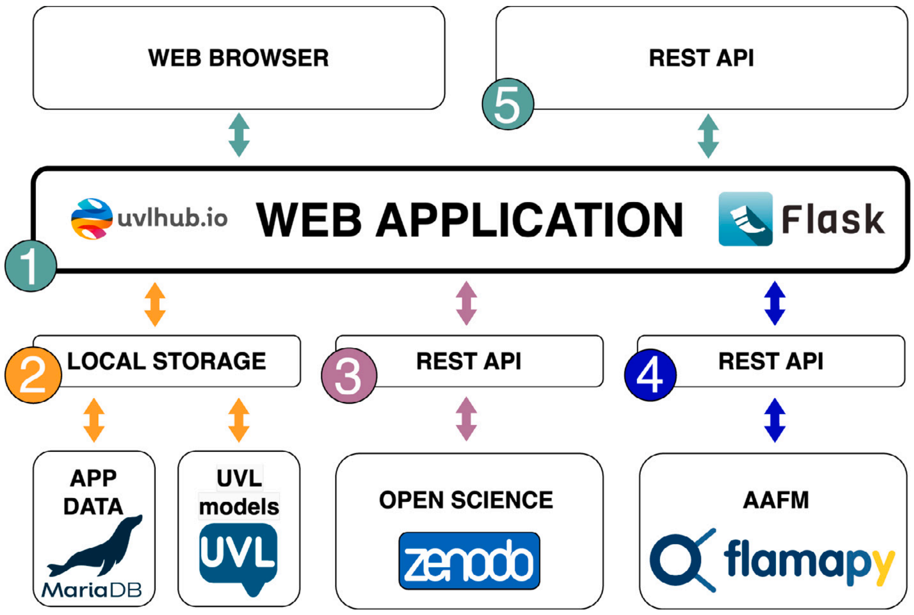

Fig. 3. UVLHub architecture overview.

Fig. 4 shows the application deployment architecture. The deployment is based on *Docker* containers to isolate each service and replicate the system configuration on any server. A main *nginx* container acts as a reverse proxy. This container redirects traffic to the main application and, in turn, to a load balancer. This load balancer dynamically creates and destroys flamapy tool containers to automatically analyse feature models based on demand. The main application container is permanently connected to the Zenodo REST API, storing the datasets.

4.4. Domain model

The main entity in the Zenodo data domain is a *deposition*. A deposition is an artefact comprising software, images, documents, audio, video, or any digital media. A deposition contains metadata. In the context of our problem, metadata contain a title, publication type, authors, description, keywords, related identifiers, and notes. The related identifiers are all the DOIs of each of the feature models associated with a dataset, while the notes are all the metadata associated with those feature models.

Similarly, the main entity in UVLHub is a dataset. A dataset comprises one or more UVL files that, in turn, have relevant metadata such as the title of each UVL, a description, and an independent DOI, among others. Fig. 5 compares the Zenodo and UVLHub domain models. A dataset in UVLHub is a deposition in Zenodo. As far as possible, all the information stored in UVLHub is also stored in Zenodo. In case our tool crashes, this allows us in the future to reconstruct all the information from the deposition created in Zenodo.

4.5. Dataset life cycle

A dataset is the minimum unit of information in UVLHub uploaded by a user. A dataset is composed of one or more feature models in UVL

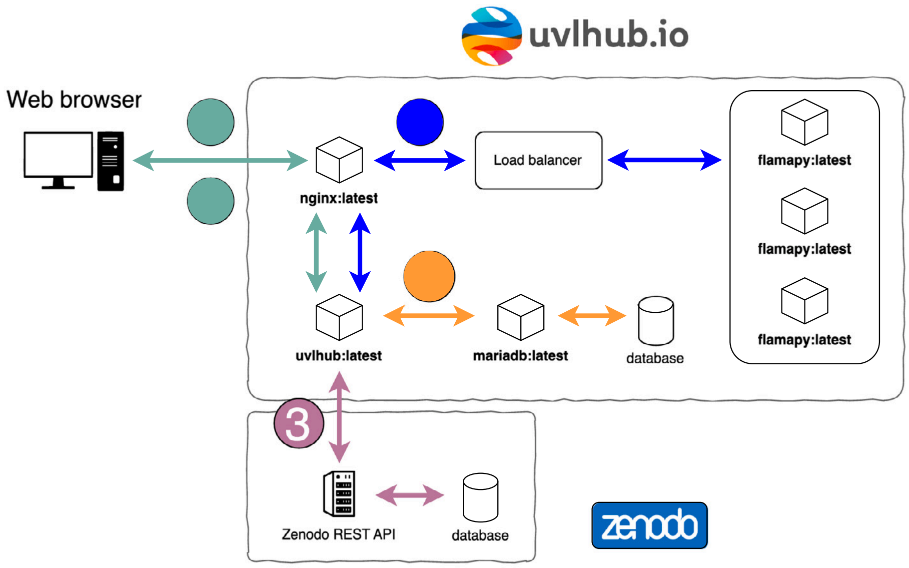

Fig. 4. Components detail in UVLHub architecture.

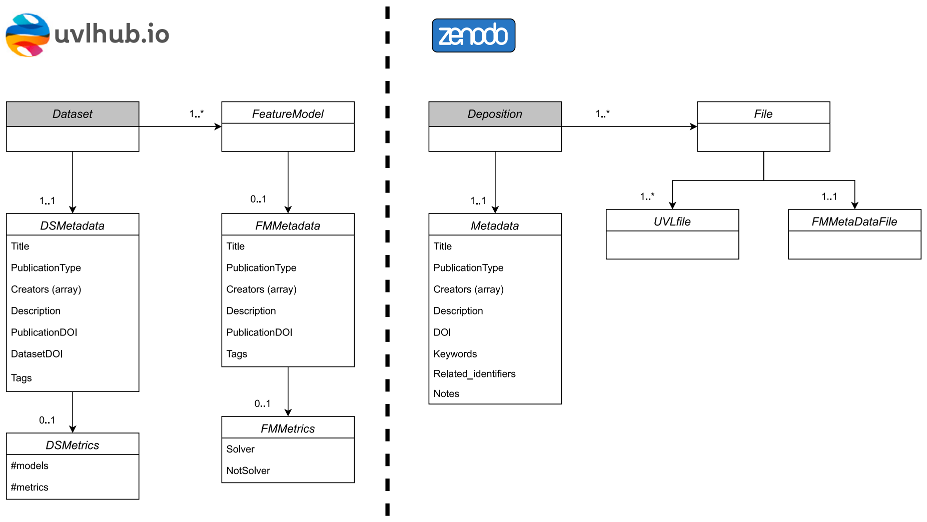

Fig. 5. UVLHub domain model.

format. A dataset has metadata: the title, description, tags, publication type, publication DOI, and authors (see Section 4.4).

Uploading a dataset always considers the Zenodo REST API service. That is, every dataset that is uploaded to UVLHub is also uploaded to Zenodo at the same time. This way, the datasets are still available if UVLHub becomes unavailable.

Fig. 6 shows the life cycle of a dataset through its five essential states. Let us describe each of them:

- *S0: Initial state*. The user prepares his feature models in UVL format and all the relevant information he wants to make available to the public.
- *S1: Local metadata stored*. The user provides basic data about the dataset or each model within. These data are but are not limited to the title, description, type of publication, and authors, among others.
- *S2: Remote metadata stored*. Metadata is uploaded to Zenodo. This process is used to obtain a deposition equivalent to a dataset in UVLHub. This deposition has a permanent identifier (ID) that identifies it in Zenodo. This ID matches it with the dataset previously stored in UVLHub.
- *S3: Local UVL files uploaded*. In this part of the process, the user uploads attachments to the dataset. These files are models in UVL format. All models are parsed to verify that they are valid and have no formatting problems. The number of models that can be attached to a dataset is limited to the limited size of the Zenodo depositions, which is virtually infinite for the case of UVL files (50 GB maximum for deposition).
- *S4: Remote UVL files uploaded*. Thanks to the deposition identifier obtained in Section 3, we can attach the files previously uploaded to the deposition. For each file, a call is made to the corresponding endpoint. Zenodo returns success or error codes depending on whether there has been any problem with the upload. In this section, the files attached to the dataset are the same as those attached to the deposition.
- *S5: Inmutable dataset*. To obtain a DOI, publishing a deposition in Zenodo is mandatory. To publish is to make the deposition permanent, i.e., no changes are allowed in our files since this originates a new version and, therefore, a new DOI.
- *S6: DOI stored*. Once the deposition has been successfully published, the obtained DOI is stored in UVLHub.

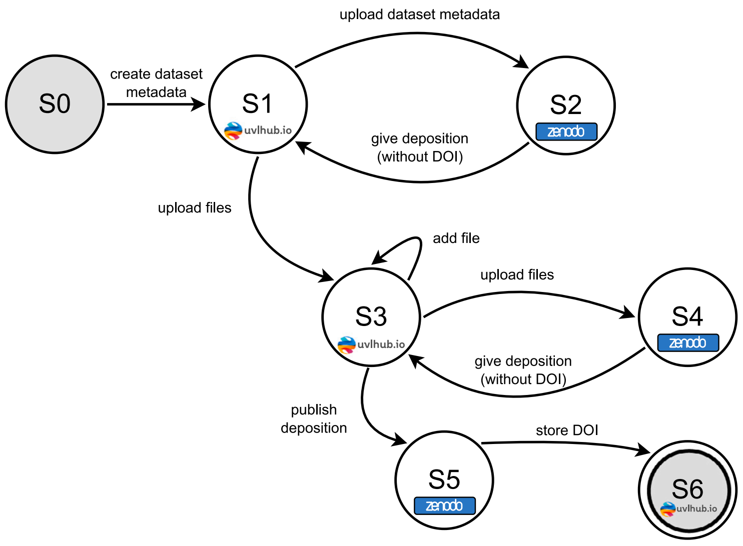

Fig. 6. Life cycle of a dataset and its states.

The point of failure of the dataset life cycle may be due to the eventual crash of the Zenodo API. The philosophy is that those models uploaded to UVLHub are also uploaded to Zenodo. However, if the Zenodo API, at the time of file upload, is not available, UVLHub does a local save. This problem corresponds to states 1 and 3. If the API problem happens when uploading files to Zenodo, a *cron job* replicates states 2, 4, and 5 every 24 h to get UVLHub and Zenodo back in sync. Using a cron job eliminates the user's responsibility of manually synchronizing its dataset or uploading it again.

4.6. Feature model dataset

As an initial set of UVL models, we added some UVL models initially published in previous work (Sundermann et al., 2021a) to UVLHub. Still, we consider the added models instead as a starting point. We envision the model collection in UVLHub continuously growing due to exchange between authors. Our goal is that users can easily access available models and publish their own.

Initial collection. In previous work, Sundermann et al. (2021a) translated several feature models, decision models, and OVM models to UVL (Sundermann et al., 2021a). We added all transformed UVL models published within that work (Sundermann et al., 2021a) to UVLHub. We also collected additional metadata, such as publishing authors of the UVL models and UVLHub-tags, and published it together with the models.

Table 1 provides an overview of the added UVL models. The overall 1427 (excluding histories) resulting UVL models contain 10–18,617 features and 0–3455 cross-tree constraints.[10] The models cover several domains, such as automotive, operating systems, and finance. For three systems, namely Automotive02, BusyBox, and FinancialServices, a history of multiple UVL models, each representing a timestamp, is available. We have also extracted the entire set of SPLOT models and converted them to UVL.

Table 1
Overview initial dataset in UVLHub.

Original source	#Models	#Features	#Constraints	Domains
SPLOT	1291	10–451	2–451	SPLOT research
Feature model	127	76–18,616	20–3455	Automotive, operating system, database, finance
Decision model	6	11–116	0–96	Dopler, phone, website, automation
OVM	3	10–37	1–11	Radio, phone, website

5. Evaluation

UVLHub provides an interface encompassing several features that attempt to meet the community's needs. These needs are reflected in three critical points, which are (1) the search for models under specific criteria, (2) the access and download of these models manually or programmatically, and (3) the management of the feature models in terms of uploading, storing, processing and visualization of metrics of these models.

We need a way to evaluate the usability and usefulness of the proposal, so we rely on Jakob Nielsen's heuristic evaluation (Nielsen and Molich, 1990), which uses usability inspection techniques. With this evaluation, we expect to find deficiencies, added value and improvement points. The selected phases of this evaluation that fit into the study of our proposal are:

1. **Preparation**, where we define which elements we are going to evaluate and under what conditions, as well as the objective and methods.
2. **Individual evaluation** (*tasks realization*), where we ask volunteers to perform a series of tasks defined by the development team.
3. **Consolidation of findings** (*conducting satisfaction surveys*), where the information obtained from all volunteers is centralized and reported to the development team and the main problems are analysed.
4. **Evaluation report** (*analysis of metrics and surveys*), where we analyse all the individual evaluations obtained and draw conclusions from them.

For the first and second phases, preparation and individual evaluation, we followed an iterative process:

1. **Pilot test.** To test the effectiveness of the evaluation, a pilot test was conducted with 7 members of the Diverso Lab research

[10] UVL models: https://github.com/Universal-Variability-Language/uvl-models.

group[11] at the University of Seville. During the course of the test, deficiencies were detected, all related to an inadequate task description that caused confusion. As a result of this test, a second, improved version of the evaluation was prepared.[12]
2. **Evaluation in MODEVAR workshop**. The corrected version of the evaluation was carried out at the Sixth International Workshop on Languages for Modelling Variability (MODEVAR 2024),[13] a workshop that takes place within the organization of 18th International Working Conference on Variability Modelling of Software-Intensive Systems (VAMOS 2024).[14] The surveys were conducted by 8 workshop participants.
3. **Evaluation with local colleagues**. Furthermore, we did an evaluation at the University of Malaga with 2 colleagues from one of the author's lab.

5.1. Preparation

The first phase of the evaluation based on Nielsen (1999) defines the objectives to be met, the methods to be followed, the tasks to be performed by the users, and the metrics to be collected.

5.1.1. Goal

The main objective is to study the usefulness of the tool. We define the tool is sufficiently helpful if the results are in the middle range or higher (see Section 5.4.1). We interpret this rank as users seeing that the need to use a feature model data repository is aligned and realized with the functionality provided by `UVLHub`.

5.1.2. Methods

The method of collecting information for subsequent analysis will be through surveys. The surveys have two distinct sections; on the one hand, the user fills in information regarding each task, which is then used to obtain the metrics. On the other hand, once the tasks have been completed, the user will fill in a satisfaction survey that allows us to obtain the degree of usefulness based on the range of the average score obtained.

5.2. Tasks realization

For a proper evaluation of the aforementioned critical points (search, access and management of feature models), we detail a total of 10 tasks that users complete in a time of finished. For each task, we specify with *(S)* the user history and with *(T)* the task in question to be performed by the user.

1. *Sign up*. (S) A researcher needs to create an account to share their own models. (T) Create an account.
2. *Upload feature models*. (S) A researcher needs to share their latest feature model. (T) Upload a model to the system and check its availability to other users.
3. *Ensure syntax check*. (S): A user wants to make sure that their model is error free before uploading it. (T): Upload a model to the system and verify that the syntax is correct.
4. *Ensure valid files*. (S): A user wants to make sure that only UVL files can be uploaded. (T): Upload a file with an extension other than UVL.
5. *Add metadata*. (S): A user wants to add dataset metadata or models. (T): Upload a dataset by modifying metadata.
6. *Modify authors*. (S): A user wants to add or remove authors from the dataset or models. (T): Upload a dataset adding or removing authors before uploading it.
7. *Download in different ways*. (S): A user needs to download several models for research. (T): Select and download a specific set of models.
8. *Generate unique identifiers (DOI)*. (S): A researcher wants their model to be easily citable. (T): Generate a unique identifier for a new model and test that it works by accessing it.
9. *Search dataset*. (S): A user wants to search their own dataset to check if it is available. (T): Search the uploaded dataset using the author name as search criteria.
10. *Search for models under criteria*. (S): A user searches for a specific model based on certain criteria. (T): Perform an advanced search and locate a specific model.

5.2.1. Key metrics

At the same time users perform tasks, we will collect valuable metrics to detect those aspects of functionality that present problems. These problems may be reflected in indicators during the completion of tasks or excessive time to complete them.

- *Average time to complete tasks*. There is a log that stores the time from when the user starts a task until it is completed.
- *Ratio of completed tasks*. This metric captures the number of tasks that were successfully completed.
- *Number of tasks that presented some help from developers*. This metric reflects the number of tasks that could not be performed independently and required help from an expert or developer of the tool.
- *Rate of errors or technical problems experienced*. This metric captures the number of errors experienced by the tool in meeting the objectives of a task, whether it is a client error or a server error.

5.3. Conducting satisfaction surveys

We have designed a questionnaire using the System Usability Scale (SUS) (Aaron Bangor and Miller, 2008) to design the questions. Positive and negative statements are alternated to prevent bias. For each statement, the user marks a value on a Likert scale (Joshi et al., 2015) from 1 to 5, where 1 means "Strongly disagree" and 5 means "Strongly agree".

1. *I need to learn many concepts before I can use UVLHUB effectively.*
2. *UVLHUB met all my needs in terms of functionality.*
3. *The information provided by UVLHUB was irrelevant to my tasks.*
4. *I was able to complete my tasks efficiently using UVLHUB.*
5. *I can't find an easy way to access the models.*
6. *I find the metrics that accompany the models useful.*
7. *I consider that UVLHUB does not add value to my work or study.*
8. *Finding a model that met my needs was relatively easy.*
9. *I think that the way of organizing the models by means of datasets is not the correct one.*
10. *I would recommend UVLHUB to others based on its usefulness.*

5.4. Analysis of metrics and surveys

During the evaluation we have collected different metrics on the use of the tool by users. For each of the tasks, we collected the following data:

5.4.1. Metrics results

In Table 2 metrics have been collected to analyse which tasks present difficulties. Each column represents a task, T_i, and the columns are (1) average task completion time in minutes, (2) ratio of completed tasks to total tasks, (3) ratio of tasks that required developer assistance to total tasks, and (4) ratio of tasks that had a software error during use to total tasks.

[11] Diverso Lab: https://grupo.us.es/diversolab/.
[12] Satisfaction survey: https://zenodo.org/doi/10.5281/zenodo.12519589.
[13] 6th MODEVAR 2024: https://modevar.github.io/.
[14] 18th VaMoS conference 2024: https://vamos2024.inf.unibe.ch/.

Table 2
Summary of metrics obtained.

Task	Minutes (average)	Completed ratio	Assisted ratio	Incidents ratio
T_1	1.1	1	0	0
T_2	2.2	1	0	0
T_3	1.1	0.9	0.1	0.1
T_4	1.2	1	0	0
T_5	1.4	1	0.1	0.2
T_6	1.3	0.9	0.2	0.3
T_7	1.4	1	0	0
T_8	1.1	1	0	0
T_9	1.1	1	0	0
T_{10}	1.1	1	0	0

Table 3
Survey results.

User	Q_1	Q_2	Q_3	Q_4	Q_5	Q_6	Q_7	Q_8	Q_9	Q_{10}
U_1	1	4	1	5	2	4	1	5	1	4
U_2	2	4	2	5	2	4	3	4	2	3
U_3	1	2	1	1	1	4	2	4	3	3
U_4	2	4	2	5	1	4	2	3	2	4
U_5	1	4	1	4	1	3	1	5	1	4
U_6	1	–	1	4	1	3	1	5	1	5
U_7	1	4	1	5	2	–	1	2	2	3
U_8	4	4	1	5	2	3	2	4	3	5
U_9	1	4	3	4	2	3	2	5	1	4
U_{10}	3	3	2	4	1	1	2	3	3	4

The task that takes the longest time to complete is T_2 because users must upload a dataset, add metadata, add models and upload them to Zenodo. The remaining tasks have a similar completion time.

Regarding the ratios, it is observed that the task T_3 (*Ensure syntax check*) presents some problems to be performed. These problems are due to a visualization problem in the interface, where it is unclear that the models have been syntactically validated, as there is a barely noticeable and very brief animation.

The task T_5 (*Add metadata*) has some difficulty to be performed. In this case, it is not transparent to some users whether the metadata can also be associated with the model (it is) and not only with the dataset. This problem should be corrected in a later version.

The task T_6 (*Modify authors*) has an issue rate of up to 30%. It is unclear whether the authors can be modified after the dataset has been uploaded or whether the task consists of adding them when creating the dataset. In future versions of the tool, we will allow the editing of metadata associated with a dataset.

Overall, all tasks were completed in an average of 13 min for an evaluation estimated to take 20 min to complete.

5.4.2. Results of satisfaction surveys

According to Nielsen and Landauer (1993), only the presence of 5 users is necessary to discover 75% to 99% of problems. Even so, we have achieved the participation of 10 users with the results in Table 3. Each of the rows represents a user U_i and each column a question Q_j (see Section 5.3) where the user has responded with a Likert score from 1 to 5.

After having collected the satisfaction surveys filled out by the users, we carry out a normalization process according to whether the affirmation is negative or positive.

1. **Score adjustment for questions with negative statements (odd-numbered questions: 1, 3, 5, 7, 9)**. Subtract the score given by the user from 5. Since negative statements are rated more favourably with low responses (a 1 indicates disagreement with a negative statement), subtracting the user's score from 5 inverts the scale, aligning it with the maximum positive value on the adjusted 0–4 scale.

Table 4
Normalized survey results.

User	Q_1	Q_2	Q_3	Q_4	Q_5	Q_6	Q_7	Q_8	Q_9	Q_{10}	Total
U_1	4	3	4	4	3	3	4	4	4	3	36
U_2	3	3	3	4	3	3	2	3	3	2	29
U_3	4	1	4	0	4	3	3	3	2	2	26
U_4	3	3	3	4	4	3	3	2	3	3	31
U_5	4	3	4	3	4	2	4	4	4	3	35
U_6	4	–	4	3	4	2	4	4	4	4	33
U_7	4	3	4	4	3	–	4	1	3	2	28
U_8	1	3	4	4	3	2	3	3	2	4	29
U_9	4	3	2	3	3	2	3	4	4	3	31
U_{10}	2	2	3	3	4	0	3	2	2	3	24
Average	3.3	2.7	3.5	3.2	3.5	2.6	3.3	3.3	3.1	3	30.2

2. **Score adjustment for questions with positive statements (even questions: 2, 4, 6, 8, 10)**. Subtract 1 from the user's score. Higher responses (a 5, for example) are more favourable in positive statements. Subtracting 1 converts the 1–5 scale to a 0–4 scale, where 4 represents the maximum positive value.
3. **Calculation of the final score**. To calculate the final SUS score for `UVLHub`, all adjusted response scores are first summed. Then, this total sum is multiplied by 2.5. This multiplication factor is derived from the fact that the maximum possible score with ten questions is 40 (since each question, on its adjusted scale from 0 to 4, can contribute a maximum of 4 points, resulting in 10*4 = 40 for all questions). By multiplying the total sum of the adjusted scores by 2.5 (which equals 100 divided by 40), the score is converted to a 100-point scale, making it easier to interpret the results.

The normalized data are collected in Table 4. The last row collects each column's mean score, corresponding to the mean score of each task in a range [0-4]. The last column collects the total score for each user in a range [0-40]. The average SUS score is 30.2. If we translate it into a range of [0-100] as the procedure described indicates, we obtain a SUS score of 75.5.

After having subjected the scoring of the surveys to the normalization process, we can design the following ranges according to Lewis and Sauro (2018) and Sauro and Lewis (2012):

- 0–68: **Low rank**. This range indicates significant usability problems and areas needing improvement. According to Sauro and Lewis (2012), a SUS score below the 50th percentile, which generally falls around a score of 68, indicates less than acceptable usability.
- 68–80: **Medium range**. Acceptable usability, but with room for improvement. Based on Sauro's interpretation of percentiles, an SUS score in this range would be approximately between the 50th and 70th percentile, indicating that usability is acceptable but not outstanding.
- 80–100: **High rank**. Indicates excellent usability with few or no significant problems. According to Sauro, a score within this range suggests that users find the system highly usable and satisfactory.

Our evaluation based on SUS and the above ranges gives us a score of 77.5, which is in the *medium range*. This score means that, although we are on the borderline between a medium and a high range, there are still usability and usability deficiencies that we need to address.

The lowest-scoring question (2.6) is the Q_6 question, *I find the metrics that accompany the models useful*. That is, there are no visualized metrics that provide value. This problem is solved, for example, by the work of Horcas et al. (2022) by generating useful metrics such as the number of features, valid products and constraints associated with a feature model or a dataset.

Table 5
Comparison of existing repositories/catalogues of feature models.

Characteristic	SPLOT (Mendonça et al., 2009b)	DyMMer 2.0 (Bezerra et al., 2021)	ESPLA (Martinez et al., 2017)	UVLHub
Public feature models available	■	■	□	■
Include metadata of feature models	■	■	■	■
Creation/Edition of feature models	■	■	□	□[a]
Visualization of the feature diagram	□	□	□	■
Support for UVL	□	□	□	■
Export models in different formats	□	□	□	□[a]
Support dataset concept	□	□	□	■
REST API	□	□	□	■
DOI association/generation	□	□	□	■
Open Science	□	□	□	■
Number of feature models	991	323	151	1489
Last update date	Jan. 2015	May. 2023	Jun. 2023	Jul. 2024
Active development	□	■	■	■

■ It supports the characteristic. □ It does not support the characteristic.
[a] Under development.

The second-lowest scoring question (2.7) is for question Q_2, *UVLHub met all my needs in terms of functionality*. We conclude that the product needs to evolve by adding functionality that brings value to UVLHub.

5.5. Discussion and limitations

This section addresses the two research questions introduced in Section 1, which have been explored throughout this paper. The aim of this section is not only to summarize the key findings and studies but also to discuss the limitations and potential threats to validity.

RQ 1: *Can feature model sharing be modernized following open science principles?* Table 5 compares the characteristics present in each repository/catalogue, including the number of feature models and the last active development date. This comparison addresses research question RQ 1, demonstrating the feasibility of creating a modern repository of feature models based on Open Science principles. A prototype repository has been designed, separating permanent storage with Zenodo and introducing functionalities to assist the community. Except for UVLHub, none of the other three repositories offer REST API functionality, DOI generation, dataset support, or export in different formats (the latter is under development). Additionally, UVLHub is the only repository adhering to the six Open Science principles.

RQ 2: *Is the use of a specialized repository for feature models in UVL format useful for the community?* Thanks to Section 5, research question RQ 2 is addressed. The final SUS score of 77.5 (see Section 5.4.2), straddling the boundary between medium and high ranges, highlights the utility and usability of the proposal. However, there are areas for improvement, particularly in tasks T_3 (*Ensure syntax check*) and T_5 (*Add metadata*), which impact the overall high ranking of the final score. Despite this, during the evaluation, volunteers provided valuable feedback on the treatment of the datasets that we will study, which serve as a starting point for new versions of our approach. Some minor changes have been made to the deployed software[15] to correct deficiencies encountered during performing tasks.

The evaluation was limited to 10 volunteers in a community workshop, with an average duration of 13 min per participant. This small sample and the short period of interaction may restrict the generalizability of the results and the depth of insights obtained, underscoring the need to expand the study in future research. The average interaction of 13 min per participant may not be enough time to explore all the features and capabilities of UVLHub. This limited time may lead to users not fully experiencing the platform.

The proposed use of a repository with reliance on data contributed by the community and Zenodo's permanent storage service suggests the following threats to validity:

Inadequate data management. Errors in the management of stored data, such as the loss of critical information, errors in updating feature models or the incorporation of erroneous data, can diminish the reliability of UVLHub.

Funding and resource limitations. Since universities maintain the project, it may face funding and human resources limitations for its development and ongoing maintenance.

UVL dependency. UVL represents a community effort to establish a universal variability modelling language. The purpose of UVLHub is to promote the adoption of this language. However, there is a risk that UVL might overshadow emerging languages in the future, potentially diminishing UVLHub's popularity and usefulness.

Changes in the access policies of Zenodo or other services. Changes in policies or the availability of services such as Zenodo or other components outside the development of UVLHub could affect the ability to ensure permanent storage of datasets or functionality.

Community dependence. The platform relies heavily on the active contribution and participation of the community in providing new models and updating existing ones. Decreased interest or participation could negatively affect the relevance and usefulness of the UVLHub.

Competition and technological evolution. The emergence of new tools or platforms offering similar or better functionality could divert the attention of the UVLHub community.

6. Related work

From 2018, the MODEVAR (Benavides et al., 2019) initiative promotes the definition of a unified language for feature models (materialized in UVL) and an ecosystem of tools supporting this language. Recent contributions include the integration of UVL into well-known SPL tools such as FeatureIDE (Sundermann et al., 2021b), transformation approaches such as TRAVART (Feichtinger et al., 2021), a framework for automatic analysis (Galindo and Benavides, 2020; Galindo et al., 2023), or a visualization for metrics (Horcas et al., 2022). The contributions also include some previous work (Romero-Organvidez et al., 2021; Galindo and Benavides, 2019) pointing to the definition of a

[15] UVLHub: https://www.uvlhub.io.

repository of feature models. In Galindo and Benavides (2019), Galindo and Benavides discussed the requirements and characteristics a feature model repository should have. In Romero-Organvidez et al. (2021), Romero et al. implement an early prototype (*FMRepo*) to show the viability of the repository to the SPL community. In our paper, we extend the previous work to develop a stable repository from scratch, taking into account the feedback provided by the SPL stakeholders during MODEVAR 2019–2024.

The idea of a repository of feature models is not new; few repositories provide a collection of feature models. Here, we discuss and compare the most relevant existing repositories of feature models or tools managing collections of feature models. SPLOT (Mendonça et al., 2009b)[16] was originally a web-based reasoning and configuration system for SPLs, but has been widely known and used in the field for the last ten years as a repository of feature models containing accurate and generated models to encourage knowledge sharing among researchers. Despite feature models from the SPLOT repository being the most commonly used within the SPL community, 98% of feature models are small, having less than 100 features. In comparison, SPLOT's most significant feature model has less than 500 features. Nowadays, SPLOT suffers from technical debt and has not been updated since 2015. For instance, SPLOT does not provide local uploading of feature models and requires uploading models within a URL based on a non-secure HTTP protocol. Moreover, it does not support UVL, and the support for modelling variability is limited to feature models containing the most basic variability modelling concepts (Horcas et al., 2023) (e.g., it does not support extended feature models with attributes).

It is worth mentioning the COfFEE and MAcchiATO catalogs (Bezerra et al., 2016)[17] (a.k.a. DyMMer) that despite not being pure feature models' repositories, there are datasets containing the values of 32 measures to evaluate the quality of 218 feature models, extracted from the SPLOT repository. Later, the same authors released the DyMMer 2.0 repository. *DyMMer 2.0* (Bezerra et al., 2021)[18] is a free and open source web-based tool for creating and editing feature models built on Eclipse and also has a public and private feature model repository of SPLs and dynamic SPLs. In the repository, the feature models are organized by ID, name, type (SPL or DSPL), number of features, user name, creation date, and origin (DyMMer, SPLOT, DyMMer 2.0), but DyMMer 2.0 does not support UVL. In addition, the public repository is managed by DyMMer administrators, while the feature models from users are in a private local repository for each user. Thus, feature models cannot be shared within the community to avoid inconsistencies within the public feature models. An exciting feature of DyMMer 2.0 is that it uses data from the repository to derive thresholds for feature models' metrics. In this respect, we plan to extend `UVLHub` to incorporate a more standard way of visualizing feature models' metrics using an *FM Fact Label* (Horcas et al., 2022).

The *ESPLA* Catalogue (Martinez et al., 2017)[19] is a collaborative collection of case studies on extractive SPL adoption. This catalogue and a repository of feature models often get mixed up. However, it is essential to note that ESPLA does not hold any feature model. Instead, ESPLA offers an informative catalogue that details SPL case studies, including references to associated papers and links to code repositories. In some cases, these code repositories may include the variability model for the respective case study.

Apart from those repositories, multiple works have gathered feature models and metadata about those feature models without explicitly defining a repository. To mention a few, *Feature Models in the Wild* (LVAT) (Berger et al., 2013)[20] is a project that investigated and studied accurate world feature models in the operating systems domain, including large-scale feature models such as the Linux kernel, with more than 6000 features, and Ecos, with more than 1000 features. These feature models were initially defined in the Linux KConfig language and translated into the Clafer language (Juodisius et al., 2019). More recently, Knüppel et al. (2017b) provide newer versions of these feature models used for evaluation,[21] and they are available in UVL due to the integration of UVL into the FeatureIDE tool (Sundermann et al., 2021b). Finally, Horcas et al. (2023) empirically analysed the tool support for SPLs, including an analysis of up to 20 case studies of variability models in six domains. In our `UVLHub` repository, we have included all available feature models of these works.

7. Conclusions and future work

This paper presents `UVLHub`, a novel online tool designed to aid researchers and practitioners in managing and analysing UVL models. The platform aims to facilitate the accessibility and sharing of UVL models among the research community, leading to a better understanding of variability modelling. It is built with a simple, user-friendly interface with various functionalities such as search, filter, visualization, and sharing of UVL models. Moreover, an initial dataset of UVL models drawn from previous work is included to initialize the model collection within `UVLHub`.

`UVLHub` provides an inclusive platform where users can easily access, publish and compare UVL models, fostering knowledge sharing and empirical research. Its use of the UVL language allows the convenient use of other tools. Also, the backup and sync with Zenodo enable a security layer in case of failure and adherence to open-science standards. Each dataset receives an automatic and invariant DOI from Zenodo, which improves model referencing and ensures a stable backup of datasets. However, `UVLHub` is also adapted to better suit the needs of feature model practitioners. For example, `UVLHub` ensures that only UVL-formatted feature models are loaded, improving model extraction and search. The metadata model of `UVLHub` allows models to be easily categorized, identified, and compared. All these characteristics underline the open design of `UVLHub`, making it a powerful and versatile tool for the feature modelling community.

Despite the numerous benefits, a few areas where `UVLHub` could be improved. Currently, `UVLHub` focuses primarily on UVL models, which could limit the scope of its application in the broader domain of variability modelling. The initial collection, though significant, primarily consists of models that have been previously published, and the diversity of the dataset will heavily depend on community contributions moving forward.

Overall, `UVLHub` represents a significant contribution to the research community, providing a centralized, easy-to-use platform for managing and analysing UVL models. It aims to foster the exchange and growth of knowledge in the area of variability modelling, helping to overcome the current fragmentation of models across different sources and formats. Extending the tool to support more variability modelling languages and implementing reverse transformation capabilities would be beneficial in the future. By doing so, `UVLHub` could become a comprehensive solution for variability modelling and research, empowering researchers and practitioners alike.

7.1. Future work

Several pending features in `UVLHub` open opportunities for future enhancements and improvements. Firstly, we plan to incorporate a version management system for models (*"As a researcher, I want version management for my models so that I can compare different versions"*). This

[16] SPLOT: http://www.splot-research.org/.
[17] MAcchiATO: http://carlabezerra.great.ufc.br/macchiato/.
[18] DyMMer 2.0: https://dymmerufc.github.io/.
[19] ESPLA: https://but4reuse.github.io/espla_catalog/.
[20] LVAT: https://gsd.uwaterloo.ca/feature-models-in-the-wild.html.

[21] Large models: https://github.com/AlexanderKnueppel/is-there-a-mismatch.

feature will allow users to compare and contrast different versions of their models without storing them as separate files. This feature will significantly ease the process of model comparison and analysis.

Secondly, we aim to provide a more comprehensive model visualization and metrics display (*"As a user, I want a comprehensive model visualisation and metrics display, so that I can understand the model before using it"*). We plan to leverage the *FM Fact Label* methodology proposed by Horcas et al. (2022) to present graphical and textual representations of models and valuable metrics. This feature will allow users to understand a model quickly and comprehensively before using it.

Lastly, we plan to incorporate a profile-based model recommendation system (*"As a user, I want profile-based model recommendations to make better model selections"*). While this is a long-term feature, we believe it will significantly enhance the model selection process. Using stored metrics and user profiles, `UVLHub` could offer personalized model recommendations, ensuring users can access models that best fit their needs.

Material

All the source code and data can be downloaded and executed from the following repository: https://github.com/diverso-lab/uvlhub. A deployed version of the tool and set-up can be viewed at https://www.uvlhub.io.

CRediT authorship contribution statement

David Romero-Organvidez: Conceptualization, Investigation, Methodology, Software, Supervision, Validation, Writing – original draft, Writing – review & editing. **José A. Galindo:** Investigation, Supervision, Writing – original draft, Writing – review & editing. **Chico Sundermann:** Investigation, Writing – original draft, Writing – review & editing. **Jose-Miguel Horcas:** Methodology, Validation, Writing – original draft, Writing – review & editing. **David Benavides:** Conceptualization, Investigation, Supervision, Validation, Writing – original draft, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

All data is available at https://github.com/diverso-lab/uvlhub and accesible through https://www.uvlhub.io. Any other data can be made available on request.

Acknowledgements

This work was partially supported by FEDER/Ministry of Science, Innovation and Universities/Junta de Andalucía/State Research Agency/CDTIwith the following grants: *Data-pl* (PID2022-138486OB-I00), TASOVA PLUS research network (RED2022-134337-T) and MIDAS (IDI-20230256). Also, the work from the Universidad de Málaga is supported by the project IRIS PID2021-122812OB-I00 (co-financed by FEDER funds). David Romero-Organvidez is supported by PREP2022-000335, financed by MICIN/AEI/10.13039/501100011033 and by FSE+

References

Aaron Bangor, Philip T. Kortum, Miller, James T., 2008. An empirical evaluation of the system usability scale. Int. J. Hum.–Comput. Interact. 24 (6), 574–594. http://dx.doi.org/10.1080/10447310802205776, arXiv:https://doi.org/10.1080/10447310802205776.

Acher, Mathieu, Collet, Philippe, Lahire, Philippe, France, Robert B., 2011. Slicing feature models. In: Proc. Int'L Conf. on Automated Software Engineering. ASE, IEEE, pp. 424–427.

Acher, Mathieu, Collet, Philippe, Lahire, Philippe, France, Robert B., 2013. Familiar: A domain-specific language for large scale management of feature models. Sci. Comput. Program. (SCP) 78 (6), 657–681.

Bagheri, Ebrahim, Gasevic, Dragan, 2011. Assessing the maintainability of software product line feature models using structural metrics. Softw. Qual. J. 19 (3), 579–612. http://dx.doi.org/10.1007/s11219-010-9127-2.

Batory, Don, 2005. Feature models, grammars, and propositional formulas. In: Proc. Int'L Systems and Software Product Line Conf.. SPLC, Springer, pp. 7–20.

Benavides, David, Rabiser, Rick, Batory, Don S., Acher, Mathieu, 2019. First international workshop on languages for modelling variability (MODEVAR). In: 23rd International Systems and Software Product Line Conference (SPLC). Vol. A, ACM, p. 46:1. http://dx.doi.org/10.1145/3336294.3342364.

Benavides, David, Segura, Sergio, Ruiz-Cortés, Antonio, 2010. Automated analysis of feature models 20 years later: A literature review. Inf. Syst. 35, 615–636. http://dx.doi.org/10.1016/j.is.2010.01.001.

Benavides, David, Segura, Sergio, Trinidad, Pablo, Cortés, Antonio Ruiz, 2007. FAMA: Tooling a framework for the automated analysis of feature models. In: 1st International Workshop on Variability Modelling of Software-Intensive Systems (VaMoS). Lero Technical Report, Vol. 2007–01, pp. 129–134, http://www.vamos-workshop.net/proceedings/VaMoS_2007_Proceedings.pdf.

Benavides, D., Sundermann, C., Vill, S., Feichtinger, K., Galindo, José A., Rabiser, R., Thüm, T., 2024. UVL: Feature modelling with the universal variability language. J. Syst. Softw. http://dx.doi.org/10.2139/ssrn.4764657, https://ssrn.com/abstract=4764657, submitted for publication.

Berger, Thorsten, She, Steven, Lotufo, Rafael, Wasowski, Andrzej, Czarnecki, Krzysztof, 2013. A study of variability models and languages in the systems software domain. IEEE Trans. Softw. Eng. 39 (12), 1611–1640. http://dx.doi.org/10.1109/TSE.2013.34.

Bezerra, Carla I.M., Barbosa, Jefferson, Freires, Joao Holanda, Andrade, Rossana M.C., Monteiro, José Maria, 2016. DyMMer: A measurement-based tool to support quality evaluation of DSPL feature models. In: Proceedings of the 20th International Systems and Software Product Line Conference. SPLC '16, Association for Computing Machinery, New York, NY, USA, pp. 314–317. http://dx.doi.org/10.1145/2934466.2962730.

Bezerra, Carla, Lima, Rafael, Silva, Publio, 2021. DyMMer 2.0: A tool for dynamic modeling and evaluation of feature model. In: Proceedings of the XXXV Brazilian Symposium on Software Engineering. SBES '21, Association for Computing Machinery, New York, NY, USA, pp. 121–126. http://dx.doi.org/10.1145/3474624.3476016.

Caffaro, Jerome, Kaplun, Samuele, 2010. Invenio: A modern digital library for grey literature. In: GL-Conference Series: Conference Proceedings. Vol. 7.

Clements, Paul, Northrop, Linda, 2001. Software Product Lines: Practices and Patterns. Addison-Wesley.

Czerniak, Andreas, Schirrwagen, Jochen, Rettberg, Najla, Fava, Ilaria, Kuchma, Iryna, 2019. OpenAIRE: the open science pillar of the EOSC. http://dx.doi.org/10.5281/zenodo.3516519.

Ernst, Neil A., Bellomo, Stephany, Ozkaya, Ipek, Nord, Robert L., Gorton, Ian, 2015. Measure it? Manage it? Ignore it? Software practitioners and technical debt. In: Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering. In: ESEC/FSE 2015, Association for Computing Machinery, New York, NY, USA, pp. 50–60. http://dx.doi.org/10.1145/2786805.2786848.

Feichtinger, Kevin, Stöbich, Johann, Romano, Dario, Rabiser, Rick, 2021. TRAVART: An approach for transforming variability models. In: VaMoS. pp. 8:1–8:10. http://dx.doi.org/10.1145/3442391.3442400.

Felfernig, Alexander, Falkner, Andreas, Benavides, David, 2024. Feature Models: AI-Driven Design, Analysis and Applications. http://dx.doi.org/10.1007/978-3-031-61874-1.

Felfernig, Alexander, Walter, Rouven, Galindo, José A., Benavides, David, Erdeniz, Seda Polat, Atas, Müslüm, Reiterer, Stefan, 2018. Anytime diagnosis for reconfiguration. J. Intell. Inf. Syst. 51 (1), 161–182. http://dx.doi.org/10.1007/s10844-017-0492-1.

Galindo, José A., Benavides, David, 2019. Towards a new repository for feature model exchange. In: 23rd International Systems and Software Product Line Conference. SPLC, B, ACM, pp. 85:1–85:4. http://dx.doi.org/10.1145/3307630.3342405.

Galindo, José A., Benavides, David, 2020. A Python framework for the automated analysis of feature models: A first step to integrate community efforts. In: SPLC '20: 24th ACM International Systems and Software Product Line Conference, Montreal, Quebec, Canada, October 19-23, 2020, Volume B. ACM, pp. 52–55. http://dx.doi.org/10.1145/3382026.3425773.

Galindo, José A., Benavides, David, Segura, Sergio, 2010. Debian packages repositories as software product line models. Towards automated analysis. In: Dhungana, Deepak, Rabiser, Rick, Seyff, Norbert, Botterweck, Goetz (Eds.), Proceedings of the 1st International Workshop on Automated Configuration and Tailoring of Applications, Antwerp, Belgium, September 20, 2010. In: CEUR Workshop Proceedings, Vol. 688, CEUR-WS.org, pp. 29–34, https://ceur-ws.org/Vol-688/acota2010_paper5_galindo.pdf.

Galindo, José, Benavides, David, Trinidad, Pablo, Gutierrez, Antonio, Ruiz-Cortés, Antonio, 2019a. Automated analysis of feature models: quo vadis? [journal first]. p. 1. http://dx.doi.org/10.1145/3336294.3342373.

Galindo, José A., Benavides, David, Trinidad, Pablo, Gutiérrez-Fernández, Antonio Manuel, Ruiz-Cortés, Antonio, 2019b. Automated analysis of feature models: Quo vadis? Computing 101 (5), 387–433. http://dx.doi.org/10.1007/s00607-018-0646-1.

Galindo, José A., Horcas, José Miguel, Felfernig, Alexander, Fernandez-Amoros, David, Benavides, David, 2023. FLAMA. A collaborative effort to build a new framework for the automated analysis of feature models. In: SPLC '23: 27th ACM International Systems and Software Product Line Conference, Tokyo, Japan, August 28 - September 1, 2023.

Galindo, José A., Turner, Hamilton A., Benavides, David, White, Jules, 2016. Testing variability-intensive systems using automated analysis: an application to Android. Softw. Qual. J. 24 (2), 365–405. http://dx.doi.org/10.1007/s11219-014-9258-y.

Halin, Axel, Nuttinck, Alexandre, Acher, Mathieu, Devroey, Xavier, Perrouin, Gilles, Baudry, Benoit, 2020. Test them all, is it worth it?: assessing configuration sampling on the JHipster web development stack. In: Lopez-Herrejon, Roberto Erick (Ed.), SPLC '20: 24th ACM International Systems and Software Product Line Conference, Montreal, Quebec, Canada, October 19-23, 2020, Volume a. ACM, p. 12:1. http://dx.doi.org/10.1145/3382025.3414985.

Horcas, José Miguel, Galindo, José A., Pinto, Mónica, Fuentes, Lidia, Benavides, David, 2022. FM fact label: a configurable and interactive visualization of feature model characterizations. In: Felfernig, Alexander, Fuentes, Lidia, Cleland-Huang, Jane, Assunção, Wesley K.G., Quinton, Clément, Guo, Jianmei, Schmid, Klaus, Huchard, Marianne, Ayala, Inmaculada, Rojas, José Miguel, Le, Viet-Man, Horcas, José Miguel (Eds.), 26th ACM International Systems and Software Product Line Conference. SPL, B, ACM, pp. 42–45. http://dx.doi.org/10.1145/3503229.3547025.

Horcas, José Miguel, Pinto, Mónica, Fuentes, Lidia, 2023. Empirical analysis of the tool support for software product lines. Softw. Syst. Model. 22 (1), 377–414. http://dx.doi.org/10.1007/s10270-022-01011-2.

Joshi, Ankur, Kale, Saket, Chandel, Satish, Pal, Dinesh, 2015. Likert scale: Explored and explained. Br. J. Appl. Sci. Technol. 7, 396–403. http://dx.doi.org/10.9734/BJAST/2015/14975.

Juodisius, Paulius, Sarkar, Atrisha, Mukkamala, Raghava Rao, Antkiewicz, Michal, Czarnecki, Krzysztof, Wasowski, Andrzej, 2019. Clafer: Lightweight modeling of structure, behaviour, and variability. Program. J. 3 (1), 2. http://dx.doi.org/10.22152/programming-journal.org/2019/3/2.

Kang, Kyo C., Cohen, Sholom G., Hess, James A., Novak, William E., Peterson, A. Spencer, 1990. Feature-Oriented Domain Analysis (FODA) Feasibility Study. Technical Report CMU/SEI-90-TR-21, Software Engineering Institute.

Katoch, Sameeksha, Thopalli, Kowshik, Thiagarajan, Jayaraman J., Turaga, Pavan K., Spanias, Andreas, 2019. Invenio: Discovering hidden relationships between tasks/domains using structured meta learning. CoRR abs/1911.10600 arXiv:1911.10600 http://arxiv.org/abs/1911.10600.

Knüppel, Alexander, Thüm, Thomas, Mennicke, Stephan, Meinicke, Jens, Schaefer, Ina, 2017a. Is there a mismatch between real-world feature models and product-line research? In: Proc. Europ. Software Engineering Conf./Foundations of Software Engineering (ESEC/FSE). ACM, pp. 291–302.

Knüppel, Alexander, Thüm, Thomas, Mennicke, Stephan, Meinicke, Jens, Schaefer, Ina, 2017b. Is there a mismatch between real-world feature models and product-line research? In: 11th Joint Meeting on Foundations of Software Engineering (ESEC/FSE). ACM, pp. 291–302. http://dx.doi.org/10.1145/3106237.3106252.

Le, Viet-Man, Vidal Silva, Cristian, Felfernig, Alexander, Benavides, David, Galindo, José, Tran, Thi Ngoc Trang, 2023. FASTDIAGP: An algorithm for parallelized direct diagnosis. In: Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 37, pp. 6442–6449. http://dx.doi.org/10.1609/aaai.v37i5.25792, https://ojs.aaai.org/index.php/AAAI/article/view/25792, no. 5.

Lewis, James, Sauro, Jeff, 2018. Item benchmarks for the system usability scale, Vol. 13, pp. 158–167.

Martinez, Jabier, Assunção, Wesley K.G., Ziadi, Tewfik, 2017. ESPLA: A catalog of extractive SPL adoption case studies. In: Proceedings of the 21st International Systems and Software Product Line Conference, SPLC 2017, Volume B, Sevilla, Spain, September 25-29, 2017. ACM, pp. 38–41. http://dx.doi.org/10.1145/3109729.3109748.

Meinicke, Jens, Thüm, Thomas, Schröter, Reimar, Benduhn, Fabian, Leich, Thomas, Saake, Gunter, 2017. Mastering Software Variability with FeatureIDE. Springer.

Mendonça, Marcilio, Branco, Moises, Cowan, Donald, 2009b. S.P.L.O.T.: Software product lines online tools. In: Proceedings of the 24th ACM SIGPLAN Conference Companion on Object Oriented Programming Systems Languages and Applications. OOPSLA '09, Association for Computing Machinery, New York, NY, USA, pp. 761–762. http://dx.doi.org/10.1145/1639950.1640002.

Mendonça, Marcílio, Wąsowski, Andrzej, Czarnecki, Krzysztof, 2009a. SAT-based analysis of feature models is easy. In: Proc. Int'L Systems and Software Product Line Conf.. SPLC, Software Engineering Institute, pp. 231–240.

Munoz, Daniel-Jesus, Oh, Jeho, Pinto, Mónica, Fuentes, Lidia, Batory, Don, 2019. Uniform random sampling product configurations of feature models that have numerical features. In: Proc. Int'L Systems and Software Product Line Conf.. SPLC, ACM, pp. 289–301.

Nielsen, Jakob, 1999. Designing Web Usability: The Practice of Simplicity. New Riders Publishing, Indianapolis, IN, USA.

Nielsen, Lars, Gonzalez Lopez, Jose Benito, Smith, Tim, Ioannidis, Alex, 2022. Zenodo data repository: Providing practical solutions for data storage and data sharing. Regul. Aff. Watch 4, 25–27. http://dx.doi.org/10.54920/SCTO.2022.RAWatch.7.25.

Nielsen, Jakob, Landauer, Thomas K., 1993. A mathematical model of the finding of usability problems. In: Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems. CHI '93, Association for Computing Machinery, New York, NY, USA, pp. 206–213. http://dx.doi.org/10.1145/169059.169166.

Nielsen, Jakob, Molich, Rolf, 1990. Heuristic evaluation of user interfaces. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. CHI '90, Association for Computing Machinery, New York, NY, USA, pp. 249–256. http://dx.doi.org/10.1145/97243.97281.

Noorian, Mahdi, Ensan, Alireza, Bagheri, Ebrahim, Boley, Harold, Biletskiy, Yevgen, 2011. Feature model debugging based on description logic reasoning. Collection / Collection : NRC Publications Archive / Archives des publications du CNRC.

Plazar, Quentin, Acher, Mathieu, Perrouin, Gilles, Devroey, Xavier, Cordy, Maxime, 2019. Uniform sampling of SAT solutions for configurable systems: Are we there yet? In: Proc. Int'L Conf. on Software Testing, Verification and Validation. ICST, IEEE, pp. 240–251.

Pohl, Richard, Lauenroth, Kim, Pohl, Klaus, 2011. A performance comparison of contemporary algorithmic approaches for automated analysis operations on feature models. In: Proc. Int'L Conf. on Automated Software Engineering. ASE, IEEE, pp. 313–322.

Ramachandran, Rahul, Bugbee, Kaylin, Murphy, Kevin, 2021. From open data to open science. Earth Space Sci. 8, http://dx.doi.org/10.1029/2020EA001562.

Rodas-Silva, Jorge, Galindo, José A., García-Gutiérrez, Jorge, Benavides, David, 2019. Selection of software product line implementation components using recommender systems: An application to wordpress. IEEE Access 7, 69226–69245. http://dx.doi.org/10.1109/ACCESS.2019.2918469.

Romero-Organvidez, David, Galindo, José A., Horcas, José Miguel, Benavides, David, 2021. A first prototype of a new repository for feature model exchange and knowledge sharing. In: 25th SPLC. Vol. B, pp. 80–85. http://dx.doi.org/10.1145/3461002.3473949.

Romero-Organvidez, David, Horcas, Jose-Miguel, Galindo, José A., Benavides, David, 2024. Data visualization guidance using a software product line approach. J. Syst. Softw. 213, 112029. http://dx.doi.org/10.1016/j.jss.2024.112029.

Sauro, Jeff, Lewis, James R. (Eds.), 2012. Quantifying the user experience: Practical statistics for user research. In: Quantifying the User Experience. Morgan Kaufmann, Boston, p. v. http://dx.doi.org/10.1016/B978-0-12-384968-7.00016-3.

Segura, Sergio, Benavides, David, Ruiz-Cortés, Antonio, Trinidad, Pablo, 2007. Automated merging of feature models using graph transformations. pp. 489–505. http://dx.doi.org/10.1007/978-3-540-88643-3_15.

Segura, Sergio, Sánchez, Ana B., Ruiz-Cortés, Antonio, 2014. Automated variability analysis and testing of an E-commerce site. An experience report. In: ASE 2014 - Proceedings of the 29th ACM/IEEE International Conference on Automated Software Engineering. http://dx.doi.org/10.1145/2642937.2642939.

Siegmund, Norbert, Sobernig, Stefan, Apel, Sven, 2017. Attributed variability models: Outside the comfort zone. In: Proc. Europ. Software Engineering Conf./Foundations of Software Engineering (ESEC/FSE). ACM, pp. 268–278.

Sincero, Julio, Schirmeier, Horst, Schröder-Preikschat, Wolfgang, Spinczyk, Olaf, 2007. Is the linux kernel a software product line? In: Proc. Int'L Workshop on Open Source Software and Product Lines. OSSPL, IEEE, pp. 9–12.

Sochos, Periklis, Philippow, Ilka, Riebisch, Matthias, 2004. Feature-oriented development of software product lines: Mapping feature models to the architecture. In: Weske, Mathias, Liggesmeyer, Peter (Eds.), Object-Oriented and Internet-Based Technologies. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 138–152.

Sprey, Joshua, Sundermann, Chico, Krieter, Sebastian, Nieke, Michael, Mauro, Jacopo, Thüm, Thomas, Schaefer, Ina, 2020. SMT-based variability analyses in FeatureIDE. In: Proc. Int'L Working Conf. on Variability Modelling of Software-Intensive Systems (VaMoS). ACM, 6, 9 pages.

Sundermann, Chico, Feichtinger, Kevin, Engelhardt, Dominik, Rabiser, Rick, Thüm, Thomas, 2021a. Yet another textual variability language? A community effort towards a unified language. In: Proc. Int'L Systems and Software Product Line Conf.. SPLC, ACM, pp. 136–147.

Sundermann, Chico, Heß, Tobias, Engelhardt, Dominik, Arens, Rahel, Herschel, Johannes, Jedelhauser, Kevin, Jutz, Benedikt, Krieter, Sebastian, Schaefer, Ina, 2021b. Integration of UVL in featureIDE. In: 25th SPLC. Vol. B, pp. 73–79. http://dx.doi.org/10.1145/3461002.3473940.

Sundermann, Chico, Heß, Tobias, Nieke, Michael, Bittner, Paul Maximilian, Young, Jeffrey M., Thüm, Thomas, Schaefer, Ina, 2023a. Evaluating state-of-the-art #SAT solvers on industrial configuration spaces. Empir. Softw. Eng. (EMSE) 28.

Sundermann, C., Vill, S., Thüm, T., Feichtinger, K., Agarwal, P., Rabiser, R., Galindo, José A., Benavides, D., 2023b. UVLParser: Extending UVL with language levels and conversion strategies. In: 27th ACM International Systems and Software Product Line Conference (SPLC 2023). ACM, in press.

Sundermann, Chico, Vill, Stefan, Thüm, Thomas, Feichtinger, Kevin, Agarwal, Prankur, Rabiser, Rick, Galindo, José A., Benavides, David, 2023c. UVLParser: Extending UVL with language levels and conversion strategies. In: Proc. Int'L Systems and Software Product Line Conf.. SPLC, ACM.

Webb, Geoffrey I., Kuzmycz, Mark, 1995. Feature based modelling: A methodology for producing coherent, consistent, dynamically changing models of agents' competencies. User Model. User-Adapt. Interact. 5 (2), 117–150. http://dx.doi.org/10.1007/BF01099758.

Wieringa, Roel J., 2014. Design Science Methodology for Information Systems and Software Engineering. Springer, http://dx.doi.org/10.1007/978-3-662-43839-8.

David Romero-Organvidez is a graduate of the BSc in Software Engineering in 2021, MSc in Software Engineering in 2022 and currently a PhD student in Computer Science at the University of Seville. He has a pre-doctoral contract in the Data-pl project led by Professors David Benavides and José A. Galindo. His areas of interest are variability models, software product lines and model repositories based on Open Science principles. He is the technology leader of the UVLHub project (https://www.uvlhub.io). The main line of his thesis is directed to the intensive use of data in software product lines. He combines his thesis with a Teaching Assistant position at the University of Seville.

José Angel Galindo is a Tenured Associate Professor at the University of Seville, he has conducted research and taught in the United States, France, and Spain. Specializing in software product lines and product configuration, he initiated his research by developing variability management tools such as FaMa and BeTTy, which have gained widespread adoption in academia and industry. His thesis, focusing on the application of these techniques and tools to diverse domains beyond traditional product lines, earned him the prestigious SISTEDES Best Thesis Award at the national level. Nowadays, he continues to pioneer new variability analysis techniques and tools such as the flamapy Project (https://www.flamapy.org) as project leader, which aggregates the state of the art on variability analysis.

Chico Sundermann is Ph.D. student at University of Ulm. His researches mostly focuses on the analysis and specification of variability models. He has been involved in the design of UVL for several years.

José-Miguel Horcas is an Associate Professor at the Universidad de Málaga, Spain, where he received his M.Sc. degree in 2012 and his Ph.D. in Computer Sciences in 2018. He is a member of the CAOSD research group of the Universidad de Málaga. He carried out two postdoc stays at King's College London in 2019 for three months, and at the University of Seville in 2021-2022 for 18 months. His main research areas are related to software product lines, including variability and configurability, as well to quality attributes and modularity. More information available at https://sites.google.com/view/josemiguelhorcas.

David Benavides received the B.S. degree in information systems from the Institute Superieur d'Electronique de Paris, France, in 2000, the M.Sc. degree in computer engineering from the University of Seville, Spain, in 2001, and the Ph.D. degree in software engineering from the University of Seville, in 2007, where he is Full Professor of Software Engineering and leads the Diverso Lab (https://grupo.us.es/diversolab). He is in the direction board of UVL (https://universal-variability-language.github.io), flamapy (https://www.flamapy.org) and UVLHub (https://www.uvlhub.io). His main research interests include software product lines, feature modelling, variability-intensive systems, computational thinking and libre and open-source software development.