A first prototype of a new repository for feature model exchange and knowledge sharing

David Romero, José Á. Galindo, Jose-Miguel Horcas and David Benavides
University of Seville
Seville, Spain
{drorganvidez,jagalindo,jhorcas,benavides}@us.es

ABSTRACT

Feature models are the “de facto” standard for variability modelling and are used in both academia and industry. The MODEVAR initiative tries to establish a common textual feature modelling language that can be used by different communities and can allow information sharing. Feature model related researches use different models for different purposes such as analysis, sampling, testing, debugging, teaching, etc. Those models are shared in private repositories and there is a risk that all that knowledge is spread across different platforms which hinder collaboration and knowledge reuse. In this paper, we propose a first working version of a new feature model repository that allows to centralise the knowledge generated in the community together with advanced capabilities such as DOI generation, an API, analysis reports, among others. Our solution is a front end interface that uses the popular open science repository Zenodo as an end point to materialise the storage of all the information. Zenodo is enhanced with characteristics that facilitate the management of the models. The idea of our repository is to provide existing but also new features that are not present in other repositories (e.g., SPLOT). We propose to populate our repository with all the existing models of many sources including SPLOT.

CCS CONCEPTS

• **Software and its engineering** → **Software product lines**; *Requirements analysis*; *Software design engineering*; *Software implementation planning*;

KEYWORDS

feature model repository,

characteristics, variability,

requirements

1 INTRODUCTION

Feature models are the “de facto” standard for variability modelling and are used in both academia and industry. Feature models were introduced in the FODA (Feature-Oriented Domain Analysis) technical report [20] in 1990, by the Software Engineering Institute. Nowadays, feature model purposes ranges between multiple activities such as analysis [7], sampling [18], testing [19], debugging [15], and teaching [21], among others.

Currently, feature model researchers share their models among their software artefacts to improve results' replicability and fit open-science standards [4]. However, those models are often shared in private repositories and are commonly spread across different platforms, as for example, GitHub. Despite some feature models repositories exist such as SPLOT [14], ESPLA [13], or LVAT[1], they present several drawbacks. First, there is no common template or software APIs to access the repositories, making difficult to search for, obtain metrics, download, or even automate the analyses of the feature models. For example, SPLOT does not provide a REST API service to consume the knowledge provided. In addition, the creation, uploading and subsequent maintenance of feature models are rigid and with limited interaction. Second, there is limited knowledge sharing. Concretely, since there is no common organisational structure nor standard guidelines for the uploaded feature models, it is difficult to share knowledge in the community. To address this issue, the MODEVAR [1] initiative tries to establish a common textual feature modelling language that can be used by different communities and enables interoperability. However, there is a lack of sharing mechanisms for such feature modelling language. Finally, existing repositories are outdated very soon because of their technical debt [9]. Technology used for the creation of such repositories has been discontinued and has not been adapted to the technological requirements that users demand nowadays, such as a responsive-web design [10], secure http protocol (https), or user load balancing among other modern features. Ultimately, a new repository can be envisioned if the community wants to easily share knowledge in the following years.

To ease off those limitations, we have designed *FMRepo*, a first prototype of a new repository for feature model exchange and knowledge sharing. FMRepo aims to integrate both the common variability language and feature model analysis capabilities that are in consonance with other MODEVAR initiatives, such as the proposed universal variability language (UVL) [1], and the framework for automated analysis of feature models (AAFM) [6], as well as adhering to open science principles [16]. The main motivation of ~~building up a~~ new repository is to share feature modelling related

[1]https://gsd.uwaterloo.ca/feature-models-in-the-wild.html

artefacts and enable the reproducibility of experiments with feature models. However, there are other factors that promote the creation of such repository:

Meet open science standards. There are numerous guidelines within the framework of Open Science: Open Access, Open Data, Open Reproducible Search, Open Science Evaluation, Open Science Policies and Open Science Tools [4], to name a few, are part of the fundamental pillars. Our repository tries to cover all the minimum requirements to be classified in this taxonomy and to collaborate in the community.

Teaching and education. Knowledge sharing of feature models allows the significant reduction of various academic studies in relation to this technology. Building the repository within the scope of free and unrestricted access can promote the use of educational and instructional content.

Knowledge aggregation. This repository tries to bring together the dispersed knowledge that exists about feature models generated in a knowledge base that serves as support for a more in-depth and extensive study of this technology, saving time and effort.

In this paper, we present the design and implementation of FMRepo to fulfil new demands of researchers and developers. This is a first working version of a new feature model repository that allows to centralise the knowledge generated in the community together with advanced capabilities that do not exist in other repositories such as SPLOT [14] or the ESPLA catalog [13]. The main contributions of FMRepo are:

- Integration of Open Science technologies. FMRepo has been built on top of open-science technologies supported by the European Research Council [11]. To grant access to uploaded models in the future for the sake of replicability.
- As part of the Open Science technologies, FMRepo provides DOI generation for feature models through the Zenodo repository [16]. The generation of DOI can be interesting for the work of research articles. Researchers can upload their models to FMRepo and directly obtain a universal identifier to be able to integrate it into their academic work. This reduces the time and complexity of generating new knowledge using a single tool.
- An REST API that allows a series of advantages to enable the use of FMRepo as a base system in future projects, providing automatic access to the feature models.
- Integration with feature model analysis technologies *as a service*. This enables researchers and developers to consume the existing models within the repository and validate their research.
- Population of FMRepo with real-world and academic feature models from many sources, including existing repositories (e.g., SPLOT).

The remainder of this paper is organised as follows: Section 2 describes the necessary concepts of the technologies mentioned in this paper; Section 3 lists the characteristics and needs that we cover with the functionalities implemented in the repository; Section 4, make a general summary of the main modules that make up the architecture and the reason for its presence; Section 5, details the main problems of the modules and the solution we have given; and Section 6 where we explain the real reason of creating this repository and why it would be convenient for the community to get involved.

2 OPEN SCIENCE

Open Science [16] is a movement promoted by the OECD[2] countries and promoted by the European Commission. It advocates free access by citizens to the results of scientific research, data, resources, results, thoughts, as well as that the results and discoveries of scientific research are universally accessible and unrestricted. Some of the existent efforts to support open science are the European OpenAIRE[3] project and the Invenio[4] framework to build open-science repositories. Invenio is an open source project by CERN that covers a repository/document management platform (InvenioRDM), an integrated library system (InvenioILS) and a code library to build large-scale information systems (such as InvenioRDM and InvenioILS).

FMRepo promotes the open-science standards through the use of Zenodo. Zenodo[5] is an Open Science repository that has been developed on top of the Invenio framework which provides persistence to the data uploaded as well as offers a DOI (*Digital Object Identifier*) to enable easy citation of data. Zenodo also works as a backup of all the data uploaded.

3 REPOSITORY CHARACTERISTICS

In previous work [5], we defined 12 characteristics that a feature model repository should have. Those characteristics were defined observing existing repositories and were presented as user stories following the template of of *"As a [persona], I [want to], [so that]."*. In this section, we detail how FMRepo addresses those characteristics and discuss the limitations of existing repositories like SPLOT.

1. *As a researcher, I want to upload models to the repository, so that my models are available for others.* This is one of the most basic characteristics, however, existing repositories do not provide simple uploading of models and complicate this operation. For instance, while SPLOT requires to upload models within a URL based on non-secure http, ESPLA requires to create an issue or a pull request in GitHub, in order to upload the feature model. FMRepo allows uploading feature models through a simple web form.
2. *As a researcher, I want to syntax check my models before uploading in the repository, so that my models are syntax error free.* FMRepo will provide support for several languages and notations, including principally the universal variability modelling language promoted by the MODEVAR initiative [1].
3. *As a researcher or user, I want to download models from the repository, so that I can replicate experiments or use existing models.* The way models are downloaded could vary from one by only batch downloads. For instance, SPLOT supports downloads one by one, forcing researchers to perform hard-manual task or to build custom scrapping scripts to download

[2]https://www.oecd.org/
[3]https://www.openaire.eu/
[4]https://inveniosoftware.org/
[5]https://zenodo.org/

all models at once. FMRepo will provide support for flexible downloading of models, from individual models to set of models.

(4) *As a researcher, I want to generate a universal identifier for my models in the repository, so that my models are citable and easily identified.* This characteristic is a new contribution of FMRepo which provides DOI generation thanks to Zenodo. The DOI is specific for a feature model so that it can be integrated into the research papers.

(5) *As a researcher or user, I want to manage versions of my models, so that I can compare different versions.* Existing repositories store different versions of the feature models as independent (new upload) files. While this characteristic could make the repository more complex to maintain, our objective is to provide version control in FMRepo.

(6) *As a user, I want to search for models in the repository, so that I can find the models I am interested in.* Only ESPLA allows searching its catalog using tags. In fact, the information is stored in a .csv file. In contrast, our FMRepo provides different search types based on tag, metadata (e.g., authors) or even metrics (e.g., size of the feature model).

(7) *As a user, I want to display models in the repository, so that I can have a glance at the model in the case I want to use it.* Existing repositories like SPLOT displays the model as a tree visualisation along with a very limited number of metrics. The ESPLA catalog does not provide any visualisation of the models. Our proposal, in addition to displaying a feature model through a graphical or textual visualisation, also provides insight on feature model metrics so that users can have a quick characterisation of the model before using.

(8) *As a user or researcher, I want to know some indicators about the models in the repository such as ratings or number of downloads, so that I can compare models.* FMRepo stores this information as a metadata of the feature model. Other repositories like SPLOT do not provide metadata information such as how many times a model has been downloaded or rated.

(9) *As a developer or researcher, I want to have an API that can interact with the repository, so that I can programmatically access repository's information.* The contents of FMRepo can be accessed programatically in the form of an API. This allows a developer to call a method to download some models with some characteristics for example a given number of features.

(10) *As a user, I want to have recommendations according to my profile, so that I can do better model selections.* Since FMRepo stores metrics of the models according to downloads or user ratings, a recommendation system based on user profiles and ratings can be developed to better select models using the previous information stored in Zenodo. However, this seems to be a more long term feature to be considered.

(11) *As a researcher, I want to create communities, so that I can have a common space to manage my models.* The underlying Zenodo system of FMRepo allows managing multiple users which can be grouped around a research group, university or community. However, this characteristic has been discussed also in depth among the authors and no consensus has been already reached.

(12) *As a user, I want to see model's metrics, so that I can have extra information about the models.* The integration of feature model analysis technologies as a service within FMRepo allows providing implicit information of feature models such as the number of features, the number of relationships, structural metrics [2], or other analysis operations (e.g., dead features, number of configurations,...) [3].

4 ARCHITECTURE OVERVIEW

This section presents the architecture of FMRepo as well as the underline technology to provide support for knowledge sharing and feature models exchange. Figure 1 shows an overview of the architecture divided in two main parts: the interface (top of Figure 1) and the three main modules (bottom of Figure 1).

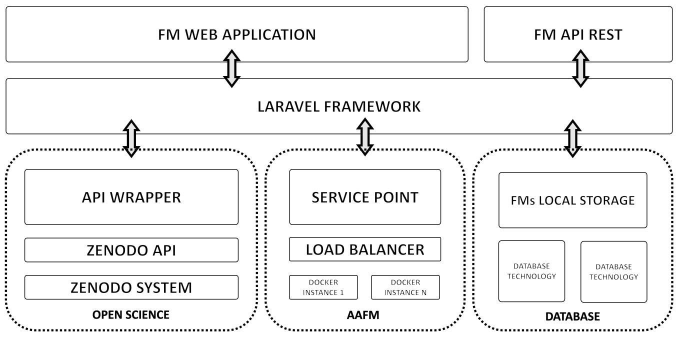


Figure 1: FMRepo architecture.

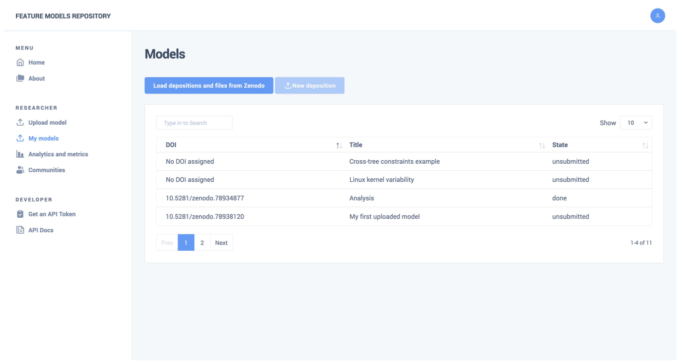
4.1 FMRepo interface

The interface and communication between the different components and the user have been developed using the `Laravel framework`. Laravel[6] is a PHP framework based on the MVC [12] architecture. It provides various convenience utilities such as a template engine, a dependency manager for PHP to include third-party libraries in your own project and a command interface program that allows you to create predefined software artefacts. This allows the repository to be instantiated on any web server with a minimum of configuration.

`Laravel` serves as the entry point for the web application and the REST API consumption managing both modules equally but independently so they can be used separately. Also, Laravel allows us to modularize [8] new functionality without needing to refactor code.

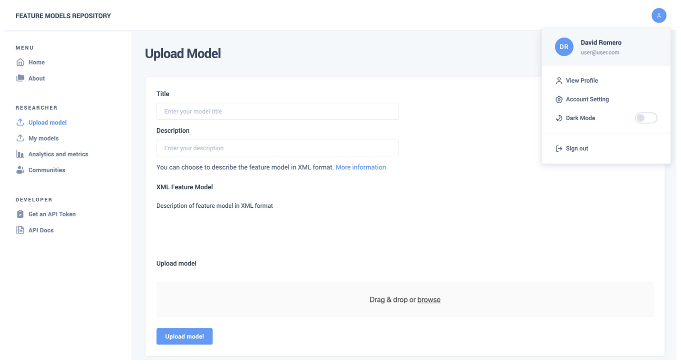
FM web application. To interact with repository tools, we provide a web interface making the system more accessible without relying on a command interface. In addition, the frontend allows several technologies to be brought together through a control panel common to all users. Figure 2 illustrates the web interface with a list of models uploaded to the repository, maintaining synchronicity with Zenodo. Feature models are uploaded using a web form as shown in Figure 3.

[6]https://laravel.com/



DOI	Title	State
No DOI assigned	Cross-tree constraints example	unsubmitted
No DOI assigned	Linux kernel variability	unsubmitted
10.5281/zenodo.78934877	Analysis	done
10.5281/zenodo.78938120	My first uploaded model	unsubmitted

Figure 2: Listing of models in FMRepo.



You can choose to describe the feature model in XML format. More information

Description of feature model in XML format

Figure 3: Uploading model in FMRepo.

FM REST API. FMRepo has a RESTful service to provide different operations such as consulting a feature model, updating an associated file, or reporting metrics. The usefulness of this module is to provide access and development to new tools created in the Software Product Line framework. The repository REST API is versioned, which allows to increase its functionality with new features that will be included in the system in the future.

4.2 FMRepo modules

This section presents the three main modules of the architecture: the Open Science module; the Automated Analysis of Feature Models (AAFM) module, and the Database module.

4.2.1 Open Science using Zenodo API. To facilitate the integration of the `Zenodo API` with the repository, we have designed an `API wrapper`. The `API wrapper` is a software artefact that works as an interface to abstract the implementation details of the service, in this case, the `Zenodo API`. This allows the API to be used as if it was an internal repository service. The `Zenodo API` can be extended to other APIs to include more Open Science projects.

4.2.2 Automated Analysis of Feature Models (AAFM). Most repositories work only as static storage. We have designed the architecture so that feature models are post-processed thanks to an internal connection with a `Docker instance`. The AAFM is running on this machine, which receives the user's feature models and performs various analysis operations or extracts metrics.

Instead of using just one machine, we have decided to replicate the AAFM module on multiple machines. Therefore, a `Load Balancer` distributes the work in N instances to alleviate the workload. The purpose of this module is not only to work with Docker and AAFM machines running on them, but future feature model processing technologies, such as benchmarking, can be integrated. The `Service Point` is in charge of providing support for integrating new software that can run in parallel.

4.2.3 Storage and database. The main storage is delegated to the `Zenodo system`. To avoid over-consuming the API, all feature models are stored locally once Zenodo has returned a positive response to the upload. The architecture supports different types of databases. Laravel has an object-relational mapping capable of switching between different `database technologys`. We take advantage of this feature to demand different services. For example, the metadata information is stored in a non-SQL database, in our case MongoDB, due to the non-relational nature of this information. However, the different internal transactions, such as the upload of feature models or the results generated from the AAFM are stored in a SQL database.

5 IMPLEMENTATION DETAILS

In this section, we briefly describe the implementation details of the different modules of the FMRepo architecture.

5.1 Zenodo as abstraction layer

The original way of working with Zenodo is through a *deposition*. A deposition is made up of a series of metadata and attachments. To facilitate the implementation of Zenodo in the use of FMRepo, we have decided place it as an abstraction layer. The important thing about deposition is that they generate a unique and permanent DOI. For practical purposes, all feature models of a user are uploaded to the same Zenodo account to group them for later usage.

FMRepo communicates with Zenodo by using its REST API, which allows the comsumption of depositions and their attachments. Some of the end points that we have defined to consume the Zenodo API are:

- *GET /api/deposit/depositions*. List all depositions for the currently authenticated user.
- *GET /api/deposit/depositions/:id/files*. List all deposition files for a given deposition.
- *POST /api/deposit/depositions*. Create a new deposition resource.
- *POST /api/deposit/depositions/:id/files*. Upload a new file.
- *POST /api/deposit/depositions/:id/actions/publish*. Publish a deposition.

However, the Zenodo REST API needs authentication to be used. To simplify user experience, FMRepo has an internal token store that is automatically used to avoid excessive consumption of. Zenodo REST API.

5.2 Running FM analysis

Another utility provided by FMRepo is the possibility of analysing feature models uploaded directly from the web application. This completely eliminates the need to install AAFM software on the user's machine. Currently we are relying on the Python framework for automated analysis [6] presented in MODEVAR [1]. The AAFM tool runs in parallel on a machine instantiated by Docker[7] for security reasons [17]. We have decided to rely on a Docker instance to avoid the execution of malicious code. This instance is capable of analysing feature models from various web clients, allowing FMRepo to be running on several servers and increasing the demand for resources. FMRepo acts as an intermediary between the Docker machine and the uploaded feature models so that end users never have access to the internal servers.

We have also designed a load balancer that allows having multiple instances of Docker running. Using a priority algorithm, the repository decides which instance has the lowest workload and chooses a candidate throughout the execution cycle.

To execute any analysis over a feature model, most AAFM tools require access to files in a certain manner. To solve this, FMRepo promotes the use of the same hierarchical file structure for each upload that are willing to provide AAFM support. This, also gives us the possibility to add metadata or relevant information. Then a packaging process is carried out following this structure:

- The **/model** directory stores the original file in XML (or similar) format of the feature model.
- The **/metadata** directory stores the original file in JSON (or similar) format of the feature model metadata.
- The **feature.json** file (optional) describes the feature model using the visual editor of the graphical interface instead of uploading an XML file.

5.3 FMRepo API

To allow future expansions of the system, we implemented an REST API. FMRepo works as a RESTful service for the main functionalities. When a user logs in as a *developer*, the system provides them with a unique access token. This token allows consuming the API. Our API is versioned and the documentation can be consulted.

We were inspired by the structure of Zenodo, facilitating thus, the usage of the platform to users that had previously worked with it. The main actions that can be performed with the API are:

- *GET /api/v1/models*. List all models for the currently authenticated user.
- *GET /api/v1/models/:model/files*. List all files for a given model.
- *GET /api/v1/models/:model/structure*. Retrieves the original structure of the uploaded model and its metadata.
- *POST /api/v1/models*. Create a new feature model.

6 CONCLUSION AND VISION

We believe that FMRepo is a starting point to create a more active community in the exchange of knowledge. This new repository is more aligned with current software technologies and in line with the MODEVAR initiatives. FMRepo also integrates automated analysis being able to use the technology that runs in the background and encouraging its use if more advanced operations are desired. The repository attempts to promote the community's use of Open Science and its relevance to research. The architecture of the repository is modular, which will allow to extend the functionality in a simple and collaborative way in the future.

Our vision of the repository is to provide a useful tool for the study of feature models in terms of meta-management, analysis and sharing. We appeal to all researchers and encourage the use of this repository. We believe that community support will favour the development, expansion and support of future characteristics. To avoid an outdated, abandoned and deprecated repository, we urge the community to upload their own feature models that cover all possible scenarios and study their performance as an analysis tool. We believe that this will encourage the use of the repository, becoming a product built on the Open Science guidelines.

[7]https://www.docker.com/

ACKNOWLEDGEMENTS

This work was supported by the Project (RTI2018-101204-B-C22, OPHELIA), funded by: FEDER/Ministry of Science and Innovation - State Research Agency; the TASOVA network (MCIU-AEI TIN2017-90644-REDT); and the Junta de Andalucia COPERNICA project; and the Spanish Government under Juan de la Cierva—Formación 2019 grant.

REFERENCES

[1] Mathieu Acher, Philippe Collet, David Benavides, and Rick Rabiser. 2020. Third International Workshop on Languages for Modelling Variability (MODEVAR@SPLC 2020). 1–1. https://doi.org/10.1145/3382025.3414948

[2] Ebrahim Bagheri and Dragan Gasevic. 2011. Assessing the maintainability of software product line feature models using structural metrics. *Software Quality Journal* 19, 3 (2011), 579–612. https://doi.org/10.1007/s11219-010-9127-2

[3] David Benavides, Sergio Segura, and Antonio Ruiz-Cortés. 2010. Automated analysis of feature models 20 years later: A literature review. *Information Systems* 35 (09 2010), 615–636. https://doi.org/10.1016/j.is.2010.01.001

[4] Isabel Bernal. 2018. Open Science: Principios, políticas, componentes, buenas prácticas. (11 2018). https://doi.org/10.20350/12794

[5] José A. Galindo and David Benavides. 2019. Towards a New Repository for Feature Model Exchange. In *Proceedings of the 23rd International Systems and Software Product Line Conference - Volume B* (Paris, France) *(SPLC '19)*. Association for Computing Machinery, New York, NY, USA, 170–173. https://doi.org/10.1145/3307630.3342405

[6] José A. Galindo and David Benavides. 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, 52–55. https://doi.org/10.1145/3382026.3425773

[7] José Angel Galindo, David Benavides, Pablo Trinidad, Antonio Manuel Gutiérrez-Fernández, and Antonio Ruiz-Cortés. 2019. Automated analysis of feature models: Quo vadis? *Computing* 101, 5 (2019), 387–433. https://doi.org/10.1007/s00607-018-0646-1

[8] Jesse Griffin. 2021. *Modularizing Laravel*. 237–284. https://doi.org/10.1007/978-1-4842-6023-4_8

[9] José-Miguel Horcas, Mónica Pinto, and Lidia Fuentes. 2019. Software product line engineering: a practical experience. In *Proceedings of the 23rd International Systems and Software Product Line Conference, SPLC 2019, Volume A, Paris, France, September 9-13, 2019*. ACM, 25:1–25:13. https://doi.org/10.1145/3336294.3336304

[10] Jason C. Hung and Chun-Chia Wang. 2021. Exploring the website object layout of responsive web design: results of eye tracking evaluations. *J. Supercomput.* 77, 1 (2021), 343–365. https://doi.org/10.1007/s11227-020-03283-1

[11] Sameeksha Katoch, Kowshik Thopalli, Jayaraman J. Thiagarajan, Pavan K. Turaga, and Andreas Spanias. 2019. Invenio: Discovering Hidden Relationships Between Tasks/Domains Using Structured Meta Learning. *CoRR* abs/1911.10600 (2019). arXiv:1911.10600 http://arxiv.org/abs/1911.10600

[12] Glenn E Krasner, Stephen T Pope, et al. 1988. A description of the model-view-controller user interface paradigm in the smalltalk-80 system. *Journal of object oriented programming* 1, 3 (1988), 26–49.

[13] Jabier Martinez, Wesley K. G. Assunção, and Tewfik Ziadi. 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, 38–41. https://doi.org/10.1145/3109729.3109748

[14] Marcílio Mendonça, Moises Branco, and Donald Cowan. 2009. S.P.L.O.T. - Software product lines online tools. *Proceedings of the Conference on Object-Oriented Programming Systems, Languages, and Applications, OOPSLA*, 761–762. https://doi.org/10.1145/1639950.1640002

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

[16] Rahul Ramachandran, Kaylin Bugbee, and Kevin Murphy. 2021. From Open Data to Open Science. *Earth and Space Science* 8 (05 2021). https://doi.org/10.1029/2020EA001562

[17] Edwin Sarmiento. 2020. *Docker Images and Containers*. 69–98. https://doi.org/10.1007/978-1-4842-5826-2_5

[18] Sergio Segura, David Benavides, Antonio Ruiz-Cortés, and Pablo Trinidad. 2007. Automated Merging of Feature Models Using Graph Transformations. 489–505. https://doi.org/10.1007/978-3-540-88643-3_15

[19] Sergio Segura, Ana B. Sánchez, and Antonio Ruiz-Cortés. 2014. Automated Variability Analysis and Testing of an E-Commerce Site. An Experience Report. *ASE 2014 - Proceedings of the 29th ACM/IEEE International Conference on Automated Software Engineering*. https://doi.org/10.1145/2642937.2642939

[20] Periklis Sochos, Ilka Philippow, and Matthias Riebisch. 2004. Feature-Oriented Development of Software Product Lines: Mapping Feature Models to the Architecture. In *Object-Oriented and Internet-Based Technologies*, Mathias Weske and Peter Liggesmeyer (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 138–152.

[21] Geoffrey I. Webb and Mark Kuzmycz. 1995. Feature Based Modelling: A methodology for producing coherent, consistent, dynamically changing models of agents' competencies. *User Modeling and User-Adapted Interaction* 5, 2 (01 Jun 1995), 117–150. https://doi.org/10.1007/BF01099758