UVL.js: Experiences on using UVL in the JavaScript Ecosystem

Victor Lamas
Universidade da Coruña, CITIC Research Center, Database Lab.
A Coruña, Spain
victor.lamas@udc.es

Maria-Isabel Limaylla-Lunarejo
Universidade da Coruña, CITIC Research Center, Database Lab.
A Coruña, Spain
maria.limaylla@udc.es

Miguel R. Luaces
Universidade da Coruña, CITIC Research Center, Database Lab.
A Coruña, Spain
miguel.luaces@udc.es

David Romero-Organvidez
drorganvidez@us.es
I3US, Universidad de Sevilla
Seville, Spain

José A. Galindo
jagalindo@us.es
I3US, Universidad de Sevilla
Seville, Spain

David Benavides
benavides@us.es
I3US, Universidad de Sevilla
Seville, Spain

ABSTRACT

The Universal Variability Language (UVL) was developed as a community-driven effort to create a simple yet extensible language for feature modeling, promoting tool interoperability within the software product line community. Although UVL is supported by several tools like FeatureIDE, Flamapy, and Pure::variants, it currently lacks direct support for web environments. To address this, we introduce a JavaScript-based UVL parser built with the ANTLR framework. This parser makes UVL models accessible directly within browser-based environments, eliminating the need for extra installations and enhancing UVL's usability for web-based tools. Furthermore, the parser can be used in back-end environments with JavaScript runtime environments such as Node.js. The parser has been successfully tested with more than 1,000 UVL models available on UVLHub and supports various UVL language levels and conversion strategies. We demonstrate its integration through two use cases: UVLHub, a public repository for UVL models developed using open science principles, and an application lifecycle management tool for software product lines. This JavaScript UVL parser is the first of its kind, unlocking new possibilities for web and JavaScript applications to take advantage of the advancements in UVL technology.

CCS CONCEPTS

• **Software and its engineering** → **Software product lines**; **Reusability**; **Software libraries and repositories**; *Abstraction, Modeling and Modularity*; *Context specific languages.*

ACM Reference Format:
Victor Lamas, Maria-Isabel Limaylla-Lunarejo, Miguel R. Luaces, David Romero-Organvidez, José A. Galindo, and David Benavides. 2025. UVL.js: Experiences on using UVL in the JavaScript Ecosystem. In *19th International Working Conference on Variability Modelling of Software-Intensive Systems (VaMoS 2025), February 04–06, 2025, Rennes, France.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3715340.3715437

VaMoS 2025, February 04–06, 2025, Rennes, France

ACM ISBN 979-8-4007-1441-2/25/02
https://doi.org/10.1145/3715340.3715437

1 INTRODUCTION

The Universal Variability Language (UVL), a textual notation for variability models, was recently developed due to community effort [2]. The MODEVAR initiative [1] aims to achieve widespread adoption of a single language (UVL), as the variety of languages available limits tool interoperability [6]. To support this, the UVL parser [10] introduces an extension mechanism designed to handle different levels of complexity. It includes two main components: language levels, which define a simple core language with optional and more complex extensions, and conversion strategies that allow tools to translate between these levels. These strategies replace complex constructs with simpler and semantically equivalent expressions, enabling better tool interoperability. The parser allows tool developers to select the language level they support while automatically converting unsupported features.

JavaScript is a dynamic, high-level programming language that has become crucial for modern web development. Its primary role as a web application language is to allow developers to design responsive and interactive user interfaces directly within the browser. Its flexibility extends far beyond scripting on the client side. JavaScript has evolved into a powerful platform for server-side development, supporting a whole ecosystem of server-side components, especially with the introduction of runtime environments like Node.js. This includes frameworks for building services, REST APIs, and full-fledged back-ends. Because of its seamless integration with HTML and CSS, JavaScript is widely used and indispensable for creating intricate, feature-rich web applications that work on various platforms.

In this work, we enhance UVL's accessibility for web platforms by introducing a JavaScript-based parser for the language. Using JavaScript's widespread adoption and client-side runtime capabilities, the parser makes UVL accessible in web-based environments without requiring additional tooling or installations. This facilitates more straightforward integration with current web technologies and encourages wider adoption of UVL by enabling developers to process and work with variability models within web applications. The parser also supports UVL's language levels and conversion strategies to ensure compatibility with different toolchains.

Listing 1 shows an example in UVL format of a feature model to obtain different smartwatch configurations. There are two main blocks: features and constraints. Thanks to the indentations, we can create blocks that group the features under a common hierarchy. For

example, in the mandatory block, at the first level of indentation, there are the features screen and energy management. In turn, the feature screen is of the alternative type, i.e., there is a choice between two types of screen: touch or standard. The constraints block allows the definition of restrictions between features, using the features' identifiers as a reference.

Listing 1: Feature model example in UVL

```
namespace smartwatch

features
    smartwatch
        mandatory
            screen
                alternative
                    touch
                    standard
            "energy management"
                alternative
                    basic
                    "advanced solar"
        optional
            payment
            gps
            "sports tracking"
                or
                    running
                    skiing
                    hiking

constraints
    !(payment & standard)
    "sports tracking" => gps
```

2 JAVASCRIPT PARSER FOR UVL

The UVL Parser [10] for Python and Java has already been published in a GitHub repository[1]. The repository encourages community collaboration by accepting contributions in other languages implementing UVL parsing. ANTLR is the foundation for the grammar parsing [7]. Our work involved adding new definitions to the base grammar already established for other languages in the repository, specifically incorporating elements such as opening and closing parentheses and brackets, considering the different features' depth. We then used the ANTLR4 JavaScript generator to generate the corresponding JavaScript classes based on UVL grammar automatically. These classes produce interfaces programmers can implement to process, read, and export the Abstract Syntax Tree (AST) from any UVL feature model definition. The parser initially passed 70% of the UVLs when we first tested it with all UVLs available on UVLHub. The errors were not due to our parser but rather the improper use of quotation marks in certain UVLs. We informed the UVLHub administrators so they could correct the issue. After they resolved it, we revalidated our parser with all UVLs from UVLHub, and it successfully passed their 1,515 feature models.

```
import { FeatureModel } from 'uvl-parser';
const featureModel = new FeatureModel('
    example.uvl');
const tree = featureModel.getFeatureModel();
```

Listing 2: Example usage of JavaScript UVL Parser

[1]https://github.com/Universal-Variability-Language/uvl-parser

Listing 2 demonstrates a basic usage example. The process begins by importing the `FeatureModel` class. Next, an instance of the class is created, with the UVL file location passed as an argument to the constructor. The location of the UVL file is provided as an argument to the constructor, which creates an instance of the class. The constructor will directly parse the input text if the file is not found at the specified path. In this case, it will treat the parameter as if the UVL were provided in plain text rather than as a file path. The method `getFeatureModel()` within the `FeatureModel` class parses the UVL file into an AST. An error is raised if the file does not conform to the UVL grammar.

To further enhance the usability of this new parser, it has been added to the standard package manager for Node.js: the npm registry [2]. This allows developers to easily include the library as a dependency on their existing Node.js projects.

3 INTEGRATION INTO WEB-BASED TOOLS

We integrated the parser into two distinct tools: UVLHub and SPLALM. The following sections describe these integrations.

3.1 UVLHub, an Online Repository for UVL Models

UVLHub is a repository for feature models in UVL format [9]. It adheres to open science principles, promoting the dissemination and sharing of knowledge [8]. The repository is integrated with Zenodo [3] to ensure the permanent storage of the models. This integration enables each model to receive a Digital Object Identifier (DOI) for easy citation. UVLHub also provides an REST API, allowing programmatic access to the models and facilitating their integration into other analysis tools. By unifying and standardizing access to feature models, UVLHub contributes to open science. It promotes transparency and replicability in software engineering variability research, enabling researchers to access and share data effectively.

Figure 1 shows the integration of uvl-parser with UVLHub. The uvl-parser package is available through the Node.js NPM manager. UVLHub maintains a package log within its package.json file, including the uvl-parser package. Due to the modular architecture of UVLHub, each module specifies how its JavaScript scripts should be compiled using Webpack. Webpack is a module bundler for JavaScript that combines and optimizes code files and resources—such as CSS, images, and scripts—into one or more files [3]. This process enhances development efficiency and improves the performance of web applications.

The parser integration with the actual JavaScript code has been done thanks to the snippet shown in Listing 2. The integration occurs on the client side to optimize resources. Whenever a UVL file is uploaded, the system parses its content. If any syntax errors are detected, they are displayed in the interface, and the upload is canceled. This parser ensures that invalid UVL models are not uploaded. Figure 2 shows the visual interface of UVLHub and the error message that the uvl-parser throws when trying to upload

[2]https://npmjs.com/package/uvl-parser
[3]https://zenodo.org/

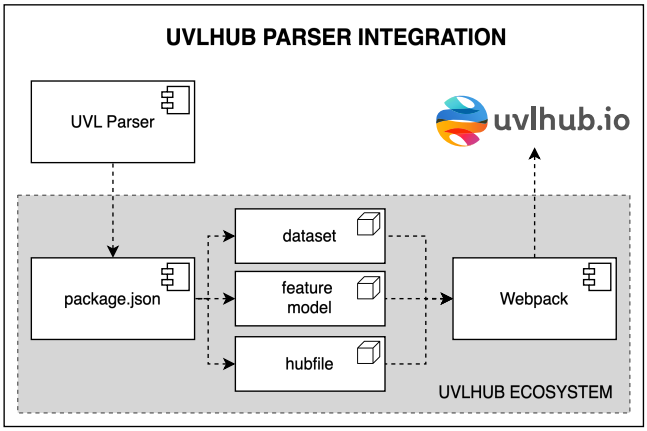

Figure 1: Integration of the UVL parser into UVLHub ecosystem

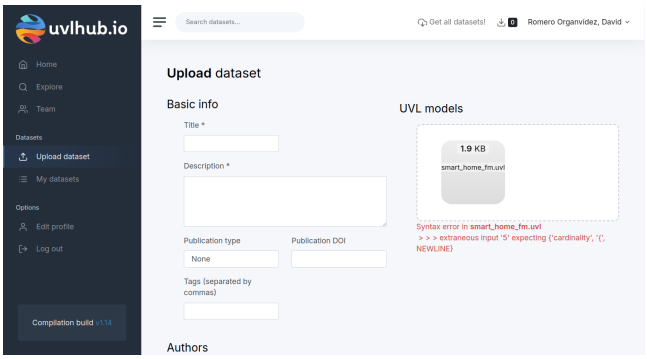

Figure 2: Integration of the UVL parser into UVLHub graphic interface

a UVL model with syntactical errors. This integration is already available in the production version of UVLHub [4].

3.2 SPLALM, a Web-based PLE Factory

SPLALM [4] is an application lifecycle management tool designed for software product lines. SPLALM enables users to manage multiple software product lines, each characterized by its feature model in UVL and a collection of source code assets. Within SPLALM, users can also maintain a portfolio of products, with each product having its configuration and associated source code. SPLALM uses GitLab as a version control repository for product source code and product line assets. It implements a Git branching model to track configuration history effectively. SPLALM uses spl-js-engine [5], a JavaScript library created to produce source code for finished products using an annotative approach. This is accomplished by combining a product specification, a feature model for a product line, and annotated code. Spl-js-engine verifies the product specification against the feature model before producing the code.

[4]https://www.uvlhub.io

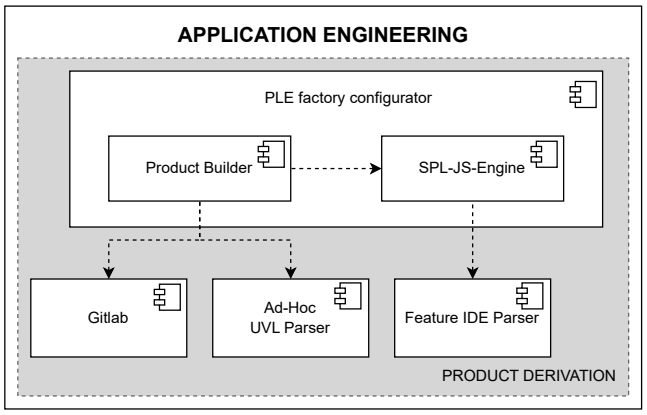

Figure 3: Architecture of SPLALM product derivation before UVL parser changes

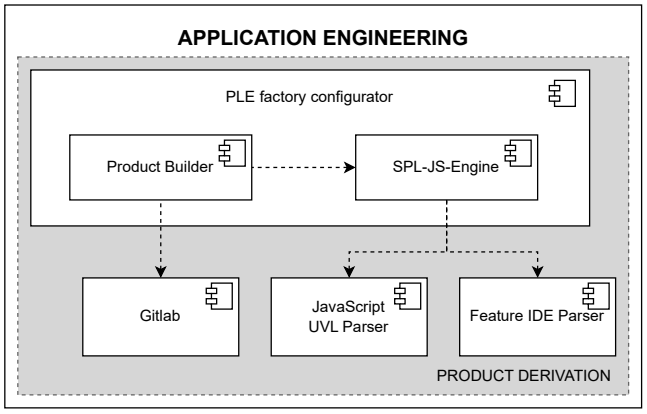

Figure 4: Architecture of SPLALM product derivation after UVL parser changes

In SPLALM, the product derivation component plays a critical role in managing and generating software products based on defined feature models. When providing a feature model for the SPL, SPLALM accepts both FeatureIDE files and UVL files. However, the product derivation component relied on a custom, ad-hoc UVL parser because the official UVL parser did not support JavaScript. Moreover, the component had to convert UVL files into the FeatureIDE language because it was the only format accepted by the spl-js-engine for code generation, as shown in Figure 3.

The introduction of a new, integrated JavaScript UVL parser marked a significant improvement. As shown in Figure 4, the ad-hoc UVL parser was removed, and the new JavaScript UVL parser was directly integrated into the spl-js-engine derivation engine. This integration allowed the derivation engine to autonomously determine which parser to use based on the input file format, whether FeatureIDE or UVL. As a result, the engine can now directly generate the Abstract Syntax Tree (AST) from the input, improving the process of producing the final source code for the required products. This transition from a standalone ad-hoc UVL parser to an integrated parser within the derivation engine yielded several

advantages. It eliminated an unnecessary processing step, reducing dependencies on external components. By integrating the UVL parser, the overall performance and simplicity of SPLAM were significantly improved. This change not only simplifies maintenance but also alleviates some technical debt. Figure 4 shows a screenshot of the Graphical User Interface of the UVL editor in SPLALM, which UVL.js supports.

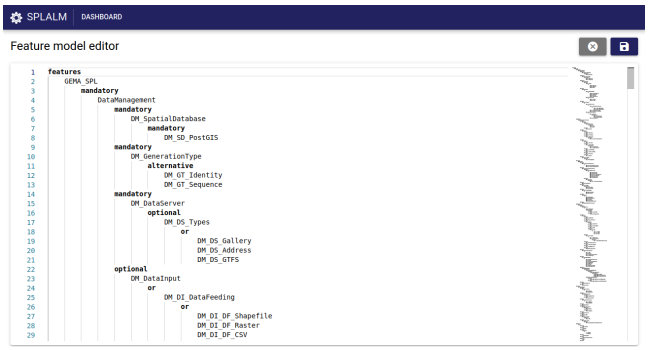

Figure 5: SPLALM Graphical User Interface: UVL editor

4 CONCLUSIONS AND FUTURE WORK

In this paper, we improved the accessibility and usability of the Universal Variability Language (UVL) in web-based environments by introducing a parser for UVL based on JavaScript. We showcased the parser's seamless integration into two web-based tools, utilizing JavaScript's widespread application in web development to encourage broader UVL adoption without requiring extra software installations. The parser supports UVL's language levels and conversion strategies, ensuring interoperability across different toolchains.

For future work, we propose developing more complex libraries around the parser to expand its capabilities and utility in diverse applications.

MATERIAL

We provide the code related to the paper. You can find the JavaScript version of the parser at this link: https://github.com/Universal-Variability-Language/uvl-parser, and the UVLHub code for integration at https://github.com/diverso-lab/uvlhub.

ACKNOWLEDGMENTS

Partly funded by: TED2021-129245B-C21 (PLAGEMIS): partially funded by MCIN/AEI/10.13039/501100011033 and "NextGenerationEU"/PRTR; PID2022-141027NB-C21 (EarthDL): partially funded by MCIN/AEI/10.13039/501100011033 and EU/ERDF A way of making Europe; CITIC is funded by the Xunta de Galicia through the collaboration agreement between the Department of Culture, Education, Vocational Training and Universities and the Galician universities for the reinforcement of the research centers of the Galician University System (CIGUS). This work was also partially supported by FEDER/Ministry of Science, Innovation and Universities/Junta de Andalucía/State Research Agency/CDTI with the following grants: *Data-pl*(PID2022-138486OB-I00) , TASOVA PLUS research network (RED2022-134337-T), AquaIA (GOPG-SE-23-0011) and MIDAS (IDI-20230256). David Romero-Organvidez is supported by PREP2022-000335, financed by MICIN/AEI/10.13039/501100011033 and by FSE+.

REFERENCES

[1] David Benavides, Rick Rabiser, Don Batory, and Mathieu Acher. 2019. First International Workshop on Languages for Modelling Variability (MODEVAR 2019). In *Proceedings of the 23rd International Systems and Software Product Line Conference - Volume A* (Paris, France) *(SPLC '19)*. Association for Computing Machinery, New York, NY, USA, 323. https://doi.org/10.1145/3336294.3342364

[2] David Benavides, Chico Sundermann, Kevin Feichtinger, José A Galindo, Rick Rabiser, and Thomas Thüm. 2024. UVL: Feature Modelling with the Universal Variability Language. *Available at SSRN 4764657* (2024).

[3] Mohamed Bouzid. 2020. *Webpack for Beginners: Your Step-by-Step Guide to Learning Webpack 4.* https://doi.org/10.1007/978-1-4842-5896-5

[4] Alejandro Bujan, A. Cortiñas, and M. R. Luaces. 2024. Development of a PLE Factory Environment with GitLab Integration and following ISO/IEC 26580. In *Proceedings of the 28th ACM International Systems and Software Product Line Conference (SPLC 2024)*. Luxemburgo, 34–37.

[5] A. Cortiñas, M. R. Luaces, and O. Pedreira. 2022. spl-js-engine: a JavaScript tool to implement Software Product Lines. In *Proceedings of the 26Th ACM International Systems And Software Product Line Conference (SPLC 2022)*. Graz, 66–69.

[6] José A. Galindo, José Miguel Horcas, Alexander Felfernig, David Fernández-Amorós, and David Benavides. 2023. FLAMA: A collaborative effort to build a new framework for the automated analysis of feature models. In *Proceedings of the 27th ACM International Systems and Software Product Line Conference - Volume B, SPLC 2023, Tokyo, Japan, 28 August 2023- 1 September 2023*, Paolo Arcaini, Maurice H. ter Beek, Gilles Perrouin, Iris Reinhartz-Berger, Ivan Machado, Silvia Regina Vergilio, Rick Rabiser, Tao Yue, Xavier Devroey, Mónica Pinto, and Hironori Washizaki (Eds.). ACM, 16–19. https://doi.org/10.1145/3579028.3609008

[7] T. J. Parr and R. W. Quong. 1995. ANTLR: A predicated-LL(k) parser generator. *Software: Practice and Experience* 25, 7 (1995), 789–810. https://doi.org/10.1002/spe.4380250705 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/spe.4380250705

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

[9] David Romero-Organvidez, José A. Galindo, Chico Sundermann, Jose-Miguel Horcas, and David Benavides. 2024. UVLHub: A feature model data repository using UVL and open science principles. *Journal of Systems and Software* 216 (2024), 112150. https://doi.org/10.1016/j.jss.2024.112150

[10] Chico Sundermann, Stefan Vill, Thomas Thüm, Kevin Feichtinger, Prankur Agarwal, Rick Rabiser, José A. Galindo, and David Benavides. 2023. UVLParser: Extending UVL with Language Levels and Conversion Strategies. In *Proceedings of the 27th ACM International Systems and Software Product Line Conference - Volume B* (Tokyo, Japan) *(SPLC '23)*. Association for Computing Machinery, New York, NY, USA, 39–42. https://doi.org/10.1145/3579028.3609013