OpenAPItoUML: A Tool to Generate UML Models from OpenAPI Definitions[*]

Hamza Ed-douibi[1][0000−0003−4342−4818], Javier Luis Cánovas
Izquierdo[1][0000−0002−2326−1700], Jordi Cabot[1,2][0000−0003−2418−2489]

[1] UOC. Barcelona, Spain
{hed-douibi,jcanovasi}@uoc.edu
[2] ICREA. Barcelona, Spain
jordi.cabot@icrea.cat

Abstract. REpresentational State Transfer (REST) has become the prominent architectural style for designing Web APIs. This increasing adoption has triggered the creation of languages to formally describe REST APIs, thus facilitating and promoting their usage. In particular, a consortium of companies has created the OpenAPI Initiative, which aims at creating a vendor neutral, portable, standard and open specification for describing REST APIs. OpenAPI specification has become the choice of reference for describing REST APIs, and its adopters can benefit from a plethora of tools for documenting, developing and integrating REST APIs. However, current documentation tools for OpenAPI only describe REST APIs in HTML pages using text and code samples, thus requiring a considerable effort to visualize and understand what the APIs offer. In this paper, we propose a tool called OpenAPItoUML, which generates UML models from OpenAPI definitions, thus offering a better visualization of the data model and operations of REST APIs.

Keywords: OpenAPI, UML, REST API

1 Introduction

The REpresentational State Transfer (REST) is an architectural style which allows relying on URIs and HTTP messages to build interoperable Web applications. Due to its lightweight nature, adaptability to the Web, and scaling capacity, REST has become the preferred style for building Web APIs. For instance, reports from ProgrammableWeb[3], the biggest repository of public Web APIs with more than 19,000 APIs, show that more than 80% of the registered APIs are REST[4].

The growing importance of REST APIs has been supported by the proposal of several languages aimed at formally describing REST APIs and therefore facilitating their discovery and integration (e.g., Swagger, API Blueprint, and RAML). Recently, and aiming at standardizing the way to describe REST APIs, several vendors have announced the OpenAPI Initiative (OAI)[5]. OAI has succeeded in attracting major companies (e.g.,

---

[3] http://www.programmableweb.com

[4] https://tinyurl.com/yd8gnuer
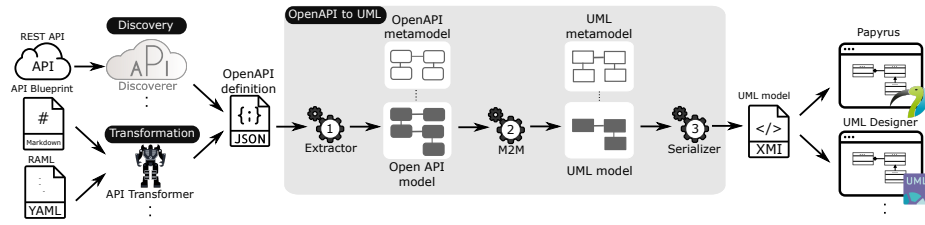
[5] https://openapis.org

Fig. 1. Our approach.

Google, Microsoft or IBM) and has created the OpenAPI specification (initially based on Swagger), which has become the choice of reference to describe REST APIs. For instance, *APIs.guru*[6], a repository of OpenAPI definitions, lists more than 800 APIs.

There is therefore a need for creating an ecosystem of supporting tools to facilitate the integration, development, and documentation of REST APIs described by OpenAPI (e.g., generating client SDKs, documentation pages or server skeleton). In this paper we propose a tool, called OpenAPItoUML, which contributes to this ecosystem by allowing the visualization of OpenAPI definitions as UML Class diagrams (including both the structure and behavior of the API), thus offering a better visualization of the capabilities of REST APIs. To the best of our knowledge, current documentation tools for OpenAPI (e.g., ReDoc[7] and Swagger UI[8]) display the operations and data structures of the definitions in HTML pages using only text and code samples, which complicate the understanding and visualization of REST APIs. Only JSONDiscoverer [2] allows visualizing the data schema of JSON-based REST APIs but focus on the inputs/outputs of the operations and does not model the operations themselves nor supports OpenAPI descriptions.

The rest of the paper is organized as follows. Section 2 describes our approach and Section 3 presents the tool. Section 4 concludes the paper and presents the future work.

2 The OpenAPItoUML Approach

We propose a model-based approach to visualize OpenAPI definitions as UML Class diagrams. From an input OpenAPI definition, our approach extracts first an OpenAPI model which is then transformed into a UML model (i.e., Class diagram) showing the data structure and operation signatures of the API. While the intermediate OpenAPI model is useful to perform other kinds of advanced analysis on the OpenAPI definition, it is more convenient to generate a UML model for visualization and comprehension purposes. Being a standard UML model, our result can be automatically rendered and modified using any of the plethora of UML modeling tools (e.g., Papyrus or UML designer).

The OpenAPItoUML process is depicted in Figure 1. As can be seen, the process takes as input an OpenAPI definition, which can be (1) provided by the API provider; (2) generated by tools such as APIDiscoverer [3], which allows discovering OpenAPI definitions from API call examples; or (3) derived from other API definition formats
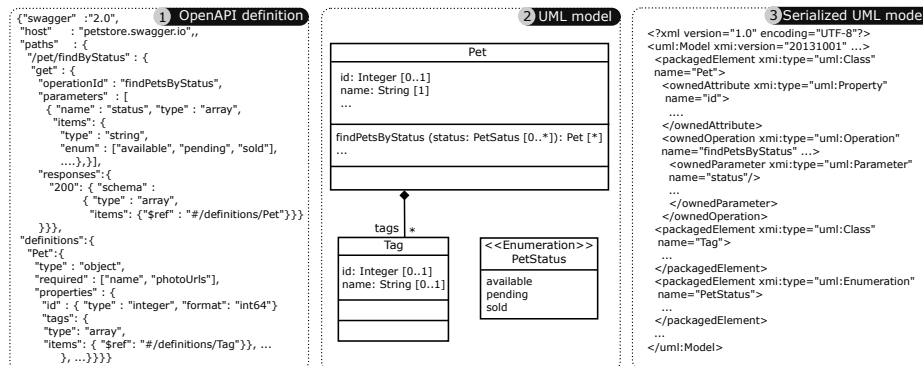
---

[6] https://apis.guru/openapi-directory/

[7] http://rebilly.github.io/ReDoc/

[8] https://swagger.io/swagger-ui/

Fig. 2. The Petstore example.

(e.g., API Blueprint, RAML) using tools such as API Transformer[9], which allows converting API definitions.

The OpenAPItoUML process generates UML models in three steps (see the steps 1, 2, and 3 in Figure 1), which we will illustrate with the Petstore API example, a REST API for a pet store management system, released by the OpenAPI community as a reference. Figure 2.1 shows an excerpt of the Petstore OpenAPI definition including the operation *findPetsByStatus* and the schema definition *Pet*.

The first step (see step 1 in Figure 1) extracts a model conforming to our OpenAPI metamodel from the input OpenAPI definition. More details about this metamodel can be found in our previous work [3]. Since the OpenAPI metamodel conforms to the OpenAPI specification, the generation of OpenAPI models from OpenAPI definitions is almost straightforward and only special attention had to be paid to deal with JSON references. We omitted to show this model for the Petstore example as it mirrors the OpenAPI definition shown in Figure 2.1.

The second step (see step 2 in Figure 1) performs a model-to-model transformation to generate a model conforming to the UML metamodel from the previously extracted OpenAPI model. This transformation iterates over the operations and definitions of the OpenAPI model in order to generate classes, properties, operations, data types, enumeration, and parameters, accordingly. This process relies on a set of heuristics to identify the most adequate UML class to attach each OpenAPI operation to. Heuristics are based on the analysis of the *tags*, *parameters* and *responses* of the operation[10]. The full list of heuristics can be found in the tool website [1]. Figure 2.2 shows an excerpt of generated UML model for the Petstore API. As can be seen, the OpenAPI schema `Pet` is transformed to the UML class `Pet`, while the OpenAPI operation `findPetsByStatus` is transformed into the UML operation `findPetsByStatus` in the `Pet` class.

The last step of the process (see step 3 in Figure 1) serializes the generated UML model as an XMI file (standard XML format for UML tool interoperability). Users can rely on tools such as Papyrus and UML designer to open and visualize such file.

---

[9] https://apimatic.io/transformer

[10] When no class is a good fit for the operation, an artificial class is created to host the operation.
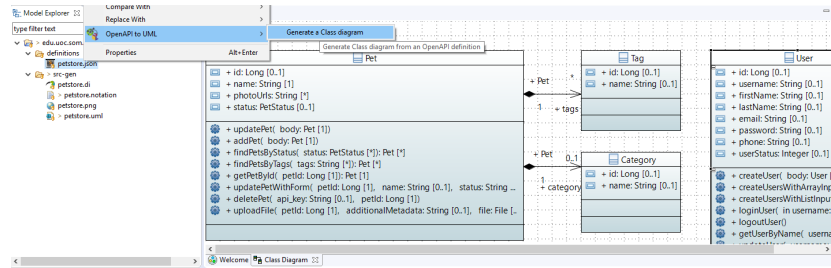
Fig. 3. A screenshot of the OpenAPItoUML plugin.

3 Tool

OpenAPItoUML has been implemented in Java as a plugin for the Eclipse platform [1]. The plugin extends the platform to provide a contextual menu to obtain a UML model from an OpenAPI definition (using its JSON representation format). Figure 3 shows a screenshot of our plugin including the created contextual menu (on the left side) and the generated Class diagram for the Petstore API displayed using Papyrus (on the right side), the "de facto" UML tool for Eclipse. The OpenAPI metamodel has been implemented using the Eclipse Modeling Framework (EMF), while UML models rely on UML2[11], an EMF-based implementation of the UML 2.5 OMG metamodel.

4 Conclusion

We have presented OpenAPItoUML, a tool to generate UML models from OpenAPI definitions. We believe our approach contributes to the ecosystem of tools for OpenAPI by offering developers the opportunity to understand and easily visualize the capacities of REST APIs. OpenAPItoUML is available as an Open Source Eclipse plugin [1]. The plugin repository includes a *Get started* guide which explains the steps to install the plugin and generate and visualize UML models.

As further work, we would like to extend our approach in order to support the newly released version of OpenAPI (i.e., OpenAPI v3.0) once it starts to get more attraction and adoption. This v3.0 version includes some interesting new features (e.g., explicit links between operations) that could be exploited to generate other types of UML diagrams (e.g. sequence diagram showing the suggested execution order). We would like also to release our tool as a Web application to visualize the generated UML models on-the-fly using Javascript.

References

1. OpenAPItoUML https://github.com/SOM-Research/openapi-to-uml
2. Cánovas Izquierdo, J., Cabot, J.: JSONDiscoverer: Visualizing the schema lurking behind JSON documents. Knowl.-Based Syst. 103, 52–55 (2016)
3. Ed-Douibi, H., Cánovas Izquierdo, J., Cabot, J.: Example-driven Web API Specification Discovery. In: Eur. Conf. on Modelling Foundations and Applications (2017)

---

[11] https://wiki.eclipse.org/MDT/UML2