Automated Analysis of Inter-Parameter Dependencies in Web APIs

Alberto Martin-Lopez
Universidad de Sevilla

ABSTRACT

Web services often impose constraints that restrict the way in which two or more input parameters can be combined to form valid calls to the service, i.e. *inter-parameter dependencies*. Current web API specification languages like the OpenAPI Specification (OAS) provide no support for the formal description of such dependencies, making it hardly possible to interact with the services without human intervention. We propose specifying and automatically analyzing inter-parameter dependencies in web APIs. To this end, we propose a domain-specific language to describe these dependencies, a constraint programming-aided tool supporting their automated analysis, and an OAS extension integrating our approach and easing its adoption. Together, these contributions open a new range of possibilities in areas such as source code generation and testing.

KEYWORDS:

Web service,
DSL,
interdependency,
CSP,
automated analysis

1 RESEARCH PROBLEM AND MOTIVATION

Web APIs enable the consumption of services and data over the network, and they are heavily used nowadays for integrating distributed and heterogeneous systems. This is reflected in the size of popular API directories such as ProgrammableWeb [11], which currently indexes over 22K web APIs. Modern web APIs usually follow the REST architectural style [20], being referred to as RESTful web APIs. RESTful APIs can be described using languages such as the OpenAPI Specification (OAS) [9], which has arguably become an industry standard. An OAS document describes a RESTful API in terms of the elements it consists of, namely paths, operations, resources, request parameters and API responses. It is both human- and machine-readable, making it possible to automatically generate, for instance, documentation, source code or even basic test cases.

Web APIs often present *inter-parameter dependencies*, i.e. constraints that restrict the way in which two or more input parameters can be combined to form valid calls to the service. For instance, in the YouTube API [15], when using the parameter `videoDefinition` (e.g. to search videos in high definition) the `type` parameter must be set to `'video'`, otherwise a HTTP 400 code (bad request) is returned. Current API specification languages such as OAS or RAML [12] provide no support for the description of these dependencies. For example, the OAS documentation states [10]: "*OpenAPI 3.0 does not support parameter dependencies and mutually exclusive parameters. (...) What you can do is document the restrictions in the parameter description and define the logic in the 400 Bad Request response*". This makes it hardly possible to interact with the services without human intervention. For example, it would be extremely difficult, possibly infeasible, to automatically generate test cases for the YouTube API without an explicit and machine-readable definition of its dependencies. We recently conducted an exhaustive review of 40 web APIs and more than 2.5K operations [22], and found that inter-parameter dependencies are extremely common in practice, being present in 85% of the APIs analyzed. In addition, we classified all the dependencies into a catalogue of seven different patterns. Industry has shown great interest in this issue, as reflected in an open feature request in OAS entitled "Support interdependencies between query parameters", created in January 2015. To date, it has received 250 votes and 51 comments from 32 participants [4].

In this paper, we present an approach to automatically analyze inter-parameter dependencies in web APIs. We propose a set of tools, including a domain-specific language (DSL) for the specification of dependencies, an extension for the OAS language, and a constraint programming-aided tool supporting the automated analysis of dependencies. We have a prototype of the tool and have evaluated its usefulness with a case study on the YouTube web API: we automatically generated 100 times more valid test cases than a random approach, thanks to the management of dependencies.

2 BACKGROUND AND RELATED WORK

Wu et al. [25] presented an approach for the automated inference of inter-parameter dependencies in web services. Oostvogels et al. [23] proposed a DSL for the description of inter-parameter dependencies in OAS. Other authors [18, 21, 26] have partially addressed this issue in other type of web services such as WSDL [2] or OWL-S [1], technologies in disuse nowadays. In the context of testing, current approaches [16, 17, 19, 24] do not handle inter-parameter dependencies and therefore show limited applicability in practice. Our

*Work supported by the European Commission (FEDER) and Spanish Government under projects BELI (TIN2015-70560-R) and HORATIO (RTI2018-101204-B-C21), and by the Spanish Ministry of Education under FPU scholarship (FPU17/04077).

work is the first to attack the root of the problem, with a thorough study of 40 web APIs, and so the conclusions derived from it differ from others (e.g. the DSL from [23] does not support all dependency types from our catalogue). More importantly, our solution is the first to fully address both the specification and *automated* analysis of inter-parameter dependencies. It is also technology- and specification-independent, meaning that it can be integrated into any web API design framework or specification language.

3 APPROACH

Figure 1 depicts the proposed approach. First, dependencies are described using Inter-parameter Dependency Language (IDL), a novel DSL designed for the specification of inter-parameter dependencies in web APIs. Then, IDL dependencies are mapped to a constraint satisfaction problem (CSP), using a constraint modeling language such as MiniZinc [8]. The CSP must also contain the information about the parameters and their domains, this is extracted from the API specification (e.g. an OAS document). Finally, a catalogue of analysis operations can be run on the resulting CSP, thereby allowing the automated analysis of dependencies.

Our proposal opens a new range of specification-driven applications in web APIs. For example, API gateways could automatically reject requests violating dependencies, without even redirecting the call to the corresponding service, saving time and user quota. Test case generators could automatically generate valid test cases (those satisfying all inter-parameter dependencies) rather than using brute force or writing specific input grammars for each API under test. Source code generators could generate web API clients including built-in assertions to deal with invalid input combinations. The range of new applications is promising.

4 RESULTS AND CONTRIBUTIONS

Next, we describe the current results and expected contributions.

State of practice. Fully understanding how dependencies arise in practice is essential for the design of a DSL that can be used to formally express all types of dependencies. To address this issue, we performed a survey on 2,557 operations from 40 APIs, constituting the largest systematic review on the presence of inter-parameter dependencies in web APIs [22]. As a result, we managed to classify all dependencies identified (over 600) into seven general patterns.

Domain-specific language. We propose IDL [7], a DSL for the specification of dependencies among input parameters in web APIs. IDL has been designed to express the seven types of dependency patterns found in our review of industrial APIs. It is specification-independent, meaning that it can be integrated into any existing API specification language (e.g. OAS, RAML and WSDL). This has enabled the creation of IDL4OAS, an OAS extension supporting the description of dependencies in this language. A sample IDL4OAS document is available on [3]. IDL is implemented with Xtext [14], a popular framework for the development of programming languages and DSLs. We have also developed an editor and a parser for IDL, facilitating its integration in other tools. The IDL grammar is available on [3]. As an example, Listing 1 depicts the IDL specification of the dependencies in the Google Maps Places API [5].

Automated analysis. We propose translating IDL specifications to CSPs. The proposed IDL-to-CSP mapping is available on

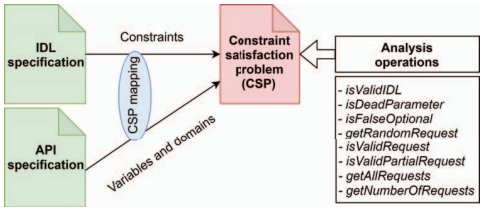

Figure 1: Proposed approach.

[3]. Then, the CSP can be leveraged to automatically extract information from IDL specifications through a catalogue of analysis operations using state-of-the-art CSP solvers. So far, we have identified eight operations (see Figure 1), and these are implemented in IDLReasoner (available on GitHub [6]), a MiniZinc-based Java library supporting the integration of our approach in any external project. Among others, IDLReasoner can automatically check whether a given API request satisfies all dependencies (operation *isValidRequest*), whether a given parameter is dead and therefore cannot be selected due to inconsistencies in the specification (operation *isDeadParameter*), or it can even generate random valid requests satisfying all dependencies (operation *randomRequest*). For example, consider the following request to the *Search* operation of the Google Maps API (Listing 1, lines 2-4): `{location=0,0; rankby=distance}`. IDLReasoner identified the request as *invalid* in 71 milliseconds, since it violates dependency in line 3, i.e. if `rankby` equals `'distance'`, then `keyword`, `name` or `type` must be set. An API gateway supporting this operation could reject requests like this (and much more complex ones) without even calling the actual service, saving time and user quota, and making service-based applications more reliable.

```
1  // Operation 1: Search for places within specified area:
2  ZeroOrOne(radius, rankby=='distance');
3  IF rankby=='distance' THEN keyword OR name OR type;
4  maxprice >= minprice;
5  // Operation 2: Query information about places:
6  AllOrNone(location, radius);
7  Or(query, type);
8  maxprice >= minprice;
9  // Operation 3: Get photo of place:
10 OnlyOne(maxheight, maxwidth);
11 // Operation 4: Automcomplete place name:
12 IF strictbounds THEN location AND radius;
```

Listing 1: IDL dependencies in the Google Maps Places API.

Evaluation. Our current efforts focus on evaluating the developed tools in terms of correctness, expressiveness and performance. In order to show the potential of our approach, we assessed the usefulness of the tool for testing purposes with a case study on the YouTube web API. Its search operation [13] presents dependencies in 25 out of 31 parameters. We generated test cases using 26 of them, discarding others that are still unsupported by our tool (e.g. dates comparison). Two test suites of 100 test cases each were automatically generated, the first with a random approach and the second with IDLReasoner. The random approach generated only one valid request (i.e. a request obtaining a 200 status code in the response), while with IDLReasoner all requests were valid. Both test suites and the execution results are available on [3]. Additionally, we succeeded to find conformance errors in the form of dependencies not described in the documentation (e.g. if the parameter `channelType` is used, the `type` parameter must be set to `'channel'`).

REFERENCES

[1] 2004. *Semantic Markup for Web Services (OWL-S).* https://www.w3.org/Submission/OWL-S/

[2] 2007. *Web Services Description Language (WSDL) Version 2.0.* https://www.w3.org/TR/wsdl20/

[3] 2019. *Additional material of the paper.* https://github.com/isa-group/ICSE-2020-SRC---Supplementary-material

[4] 2019. *GitHub issue in the OpenAPI repository: "Support interdependencies between query parameters".* https://github.com/OAI/OpenAPI-Specification/issues/256

[5] 2019. *Google Maps Places API.* https://developers.google.com/places/web-service/intro

[6] 2019. *IDLReasoner.* https://github.com/isa-group/IDLReasoner

[7] 2019. *Inter-parameter Dependency Language (IDL).* https://github.com/isa-group/IDL

[8] 2019. *MiniZinc: Constraint Modeling Language.* https://www.minizinc.org/

[9] 2019. *OpenAPI Specification.* https://www.openapis.org

[10] 2019. *Parameter dependencies in OAS.* https://swagger.io/docs/specification/describing-parameters/#dependencies

[11] 2019. *ProgrammableWeb.* https://www.programmableweb.com

[12] 2019. *RESTful API Modeling Language (RAML).* https://raml.org

[13] 2019. *Search operation of the YouTube API.* https://developers.google.com/youtube/v3/docs/search/list

[14] 2019. *Xtext.* https://www.eclipse.org/Xtext/

[15] 2019. *YouTube API.* https://developers.google.com/youtube/v3/docs

[16] A. Arcuri. 2019. RESTful API Automated Test Case Generation with EvoMaster. *ACM Trans. on Software Engineering and Methodology* 28, 1 (2019), 3.

[17] V. Atlidakis, P. Godefroid, and M. Polishchuk. 2019. RESTler: Stateful REST API Fuzzing. In *Intern. Conference on Software Engineering*. 748–758.

[18] D. Cacciagrano, F. Corradini, R. Culmone, and L. Vito. 2006. Dynamic Constraint-based Invocation of Web Services. In *3rd Intern. Workshop on Web Services and Formal Methods*. 138–147.

[19] H. Ed-douibi, J.L.C. Izquierdo, and J. Cabot. 2018. Automatic Generation of Test Cases for REST APIs: A Specification-Based Approach. In *IEEE 22nd Intern. Enterprise Distributed Object Computing Conference*. 181–190.

[20] R. T. Fielding. 2000. *Architectural Styles and the Design of Network-based Software Architectures.* Ph.D. Dissertation.

[21] C. Gao, J. Wei, H. Zhong, and T. Huang. 2014. Inferring Data Contract for Web-based API. In *IEEE Intern. Conference on Web Services*. 65–72.

[22] A. Martin-Lopez, S. Segura, and A. Ruiz-Cortés. 2019. A Catalogue of Inter-Parameter Dependencies in RESTful Web APIs. In *Intern. Conference on Service-Oriented Computing*. 399–414.

[23] Oostvogels, N., De Koster, J., De Meuter, W. 2017. Inter-parameter Constraints in Contemporary Web APIs. In *17th Intern. Conference on Web Engineering*. 323–335.

[24] S. Segura, J.A. Parejo, J. Troya, and A. Ruiz-Cortés. 2018. Metamorphic Testing of RESTful Web APIs. *IEEE Trans. on Software Engineering* 44, 11 (2018), 1083–1099.

[25] Qian Wu, Ling Wu, Guangtai Liang, Qianxiang Wang, Tao Xie, and Hong Mei. 2013. Inferring Dependency Constraints on Parameters for Web Services. In *Proceedings of the 22nd Intern. Conference on World Wide Web*. 1421–1432.

[26] L. Xu, Q. Yuan, J. Wu, and C. Liu. 2009. Ontology-based Web Service Robustness Test Generation. In *IEEE Intern. Symp. on Web Systems Evolution*. 59–68.