オントロジーを用いた OpenAPI Document の制約推薦システム

柴田 晃[1,a)]　石橋 勇人[1,b)]

概要：Web API の仕様を記述するための記法として広く利用されている OpenAPI Specification にはリクエストパラメータの制約（最大値や最小値など）を記述する記法が用意されているにもかかわらず，制約を記述していない仕様書が多く見られる．仕様書に制約が記述されていない場合であっても開発者は何らかの制約を見いだし制約を実装しているが，制約が明示されていないために仕様書の記述者と開発者の間で解釈の齟齬が発生する可能性がある．認識の齟齬をなくすためには仕様書に制約を明示することが必要である．本稿では，開発者が自らの知識をもとに制約を見いだせることに注目し，Web API の知識を記述したオントロジーに基づいて仕様書に定義されていない制約を発見，推薦することによって仕様書の記述を支援するシステムを提案する．

キーワード：OpenAPI Specification, オントロジー, Web API, 設計支援

A constraints recommendation system for OpenAPI Documents using ontologies

Abstract: *The OpenAPI Specification* is widely used for describing Web API specifications. While it provides description syntax for constraints on request parameters such as maximum values and minimum values, many specification documents do not describe constraints. Even in such cases, human developers find some constraints and implement them as computer programs. On the other hand, it may lead to misunderstandings between the specification designers and the developers. It means that providing clear and strict definitions of constraints specification documents is inevitable for proper development. This paper proposes a constraints recommendation system for designers that discovers and recommends constraints not defined in the specification based on ontology describing knowledge on Web APIs.

Keywords: OpenAPI Specification, Ontology, Web API, Design assistance

1. はじめに

Web API と呼ばれる HTTP 経由で呼び出される API は現在広く利用されており，Web 上で公開されているサービス以外にもスマートフォン用のアプリケーションなどでも利用されている．Web API において仕様を記述するための記法として OpenAPI Specification（OAS）[1] が広く利用されている [2]．OAS の特徴としては，1）API の仕様を記述するために必要な多くのプロパティが用意されている，2）OAS に沿って記述される仕様書（OpenAPI Document）のファイルフォーマットが YAML または JSON 形式であり，人と機械の双方に読みやすい，3）サードパーティのツール群が豊富，といったことがあげられる．

一方，OpenAPI Document をもとにテストを行う既存の研究 [3], [4], [5] において，OpenAPI Document に必要なパラメータが記載されていない，パラメータの制約がコメントとして存在しているなど，OpenAPI Document の記述内容に関する問題が指摘されている．また Hosono らの調査では約 65.5%の公開されているエンドポイントで実装と OpenAPI Document の記述に差異があることをあげている [6]．

我々は文献 [7] において，OpenAPI Document に記述さ

[1] 大阪市立大学大学院工学研究科
Graduate School of Engineering, Osaka City University
[a)] d20tb501@xg.osaka-cu.ac.jp
[b)] h-ishibashi@osaka-cu.ac.jp

れているリクエストパラメータの定義をもとにテスト用のリクエストパラメータを生成する手法を提案した．そこでは OpenAPI Document に記載されているパラメータの型情報と制約（最大値や最小値，文字列長など）をもとに条件に合う入力値を生成しており，制約が記述されていない OpenAPI Document では型情報だけを頼りに生成せざるを得なかった．また，OpenAPI Document には制約の記述がなくとも，実装された Web API には入力値に制約がある場合に対応できない課題があった．そこで，本稿では OpenAPI Document に制約が記述されていないという問題に対応する．

OpenAPI Document の記述者が制約を定義していない理由としては，単純に記述者の怠慢である場合や，記述者にとっては自明でありわざわざ記述する必要はないと考えている場合などが考えられる．そこで，制約の候補を例示することで記述者に制約の記述が必要なことを伝えると同時に，候補の存在によって記述の難易度を下げ，制約の記述を容易にすることが有効であると考えられる．

一般に，OpenAPI Document には制約が記述されていない場合であっても，開発者はそこに何らかの制約を見いだし実装している．しかし，制約が明示されていないために OpenAPI Document の記述者と開発者の間で解釈の齟齬が発生する可能性がある．

本稿では，制約が記述されていない状態を曖昧な状態と定義し，OpenAPI Document が曖昧な状態にあることによって生じる解釈の齟齬を排除することを目的として，OpenAPI Document の記述者をサポートするシステムを提案する．OpenAPI Document 内に制約が記述されていない場合であっても開発者は制約を解釈できていることから，開発者の知識を利用することによって暗黙の制約を見つけることができると考えられる．

人が持つ知識をコンピュータで解釈し利用するための表現の一つとしてオントロジーがある．そこで，Web API のパラメータを理解するのに必要な知識をオントロジーとして記述し，それをもとにパラメータの内容を推論することによって，OpenAPI Document の記述に不足している制約を推薦するシステムを構築する．オントロジーを用いることで，次のような推薦を行うことができる．

- *instance-id* というパラメータ定義があった時，Amazon Web Services（AWS）の知識から「`^i-[0-9a-z]{17}$`」というパターンを制約として推薦する
- *name*（氏名），*surname*（氏），*given_name*（名）というパラメータ定義があった時，パラメータ間の関係と氏名に関する知識から *surname* の最大長と *given_name* の最大長を足した値が，*name* の文字数の最大長を越えないように推薦する

本稿では，このような制約を推薦できるシステムを実現する最初の段階として，型情報と Web API のパラメータについてのオントロジーから，パラメータの制約を推薦するシステムを提案する．

2. 関連研究

Ed-Douibi らは OpenAPI Document からモデルを生成し，モデルからパラメータを推論することによってテストケースを生成する手法を提案している [3]．APIs.guru に公開されている OpenAPI Document を用いて実験した結果，取得した Document には OAS が提供する制約の記法ではなく自然言語によってパラメータの制約が記述されていることがあり，それがテストケースの生成を妨げるという OpenAPI Document の記述に起因する問題をあげている．

Martin-Lopez はパラメータ間の依存関係を記述するための Inter-parameter Dependency Language (IDL) と呼ばれるドメイン固有言語を提案し，IDL をもちいて OAS を拡張することを提案している [8]．Web API では 2 つ以上の入力パラメータを組み合わせてサービスを呼び出す制約（パラメータ間の依存性）が課されることがよくあるが，OAS にはこのような依存性を記述するためのサポートを提供していない．そこで，IDL を使用して OpenAPI Document に依存関係を記述した後に依存関係を制約充足問題（CSP）にマッピングして，CSP に対して分析操作を実行することで依存性の自動分析を可能にするシステムを提案している．Martin-Lopez はパラメータ間の組み合わせによる制約に注力しており，我々の様に個々のパラメータが持つべき制約については研究の対象としていない．また，OpenAPI Document の記述者は OAS 以外に IDL の理解が必要なため，学習コストが高くなるという問題がある．

Karavisileiou らは次に述べる Mainas らの研究をもとに，OpenAPI Document をオントロジーのインスタンスに変換する手順を提案している [9]．そこでは，OpenAPI Document において同じプロパティに異なる名前をつけていたり，その意味が定義されていない場合があるといった曖昧さが存在すること，OAS にそれらの曖昧さを検出し対処するための方法が提供されていないことを問題としてあげている．これらの曖昧さは我々が本稿で対象とするスコープとは異なるが，これらに対応できれば同じプロパティで異なる制約が定義されていないかなどを確認できるようになり，より良い制約の推薦ができると考えられる．

OpenAPI Document をオントロジーを用いて解釈をしようとする研究として Mainas らの研究がある [10]．OAS の各要素をオントロジーで定義した概念に関連付けるため，OpenAPI Document に記述された内容にアノテーションを付加する Semantic OAS (SOAS) を提案しているが，Martin-Lopez の提案と同様 SOAS の理解やオントロジーの理解が必要となり，学習コストが高くなるという問題がある．

3. 提案方式

本稿では，曖昧な状態にある OpenAPI Document によって発生する解釈の齟齬を排除することを目的に，OpenAPI Document に記述されているパラメータに不足している制約の記述をオントロジーを用いて発見，推薦することで OpenAPI Document の記述を支援するシステムを提案する．

この目的を達成するためには，OpenAPI Document 内に OAS の記法に沿って制約を記述する必要があり，それには対象のパラメータに指定できる制約の種類と制約の値が必要になる．

我々が提案する方式では，Web API のパラメータに関するオントロジーから，パラメータに設定できる制約の種類と制約の値を発見する．発見した制約の情報から OpenAPI Document の記述者に対して制約を記述するためのプロパティとプロパティに設定する値をあわせて推薦する．

これによって，OpenAPI Document の記述者は推薦された値が妥当であればそのまま採用でき，他に正しい値があるならば自ら記述することもできる．

3.1 提案システムの概要

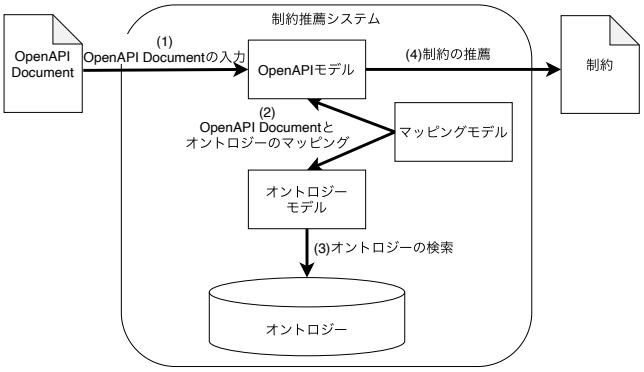

図 1: 提案システムの概要

提案システムの概要を図 1 に示す．その動作は概ね次の様になる．

(1) OpenAPI Document を受け取る．
(2) 受け取った OpenAPI Document からパラメータの情報を抜き出し，パラメータの型情報をもとにオントロジーのクラスにマップする．
(3) マップしたクラスをもとにオントロジーを検索し，利用できる制約を見つける．
(4) OpenAPI Document に記述されていない制約を推薦する．

3.2 OpenAPI Document

OpenAPI Document は OAS に沿って Web API の仕様を記述した仕様書である．リスト 1 およびリスト 2 に OpenAPI Initiative が GitHub で公開している OpenAPI Document[*1]から一部を抜き出した例をあげる．

```yaml
paths:
  /pets:
    post:
      description: Creates a new pet in the store.
      ↪  Duplicates are allowed
      operationId: addPet
      requestBody:
        description: Pet to add to the store
        required: true
        content:
          application/json:
            schema:
              $ref: "#/components/schemas/NewPet"
      responses:
        "200":
          description: pet response
          content:
            application/json:
              schema:
                $ref: "#/components/schemas/Pet"
```

リスト 1: 新しいペットを登録する API の定義

```yaml
components:
  schemas:
    NewPet:
      type: object
      required:
        - name
      properties:
        name:
          type: string
        tag:
          type: string
```

リスト 2: NewPet スキームの定義

リスト 1 は新しいペットを登録する API の定義で*/pets* というパスに NewPet スキーマに沿ったデータを POST 形式のパラメータとして送ることによってペットを登録できることを表している．リスト 2 は NewPet スキーマの定義である．プロパティとして文字列型の *name* と *tag* が含まれており，*name* は必須のパラメータであることを表している．

[*1] https://github.com/OAI/OpenAPI-Specification/blob/master/examples/v3.0/petstore-expanded.yaml

3.3 提案システムで利用するオントロジー

提案システムでは Web API のリクエストで渡されるパラメータとその制約の関係を表現することを目標に独自のオントロジーを構築した．

オントロジーでは概念をクラスとして表現し，クラスの具体的な事実はインスタンスとして表現する．クラスやインスタンス間の関係をプロパティを使って表現する．例えば，人間クラスが人間という概念を表し，徳川家康と徳川秀忠と言う人間クラスのインスタンスが実際の人物を表す．子というプロパティを使って徳川家康の子が徳川秀忠であることを表す．

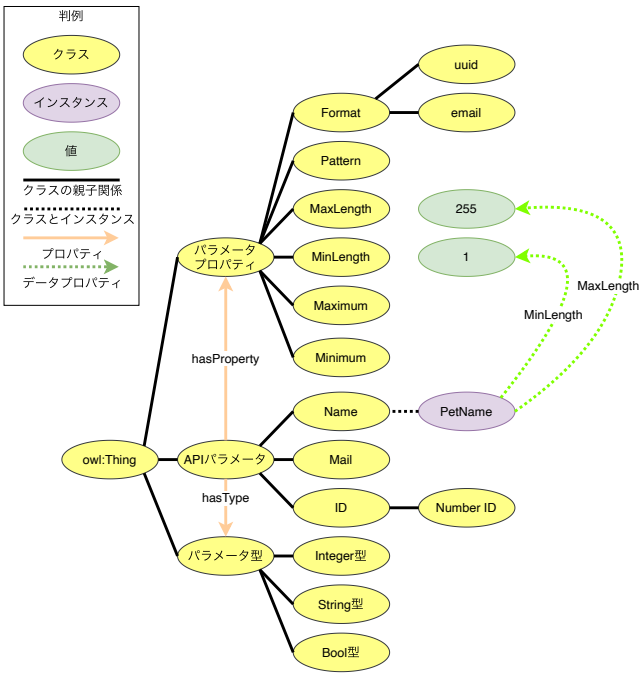

図 2: 提案システムで用いるオントロジーの概要

図 2 に構築したオントロジーの一部を示す．黄色の楕円で囲まれている項目がクラスを表す．API パラメータクラスは API のパラメータという概念を表しており，同様にパラメータ型クラスはパラメータのデータ型の概念を，パラメータプロパティクラスはパラメータの制約の概念を表している．クラス間をつなぐ黒い実線はスーパークラスとサブクラスの関係（親子関係）を表している．図 2 では owl:Thing に近い側がスーパークラスである．サブクラスはスーパークラスの性質を受け継いでいる．紫の楕円で囲まれている項目はインスタンスを表しており，点線で繋がっているクラスの具体的な個体を表現している．クラス間を結ぶ矢印はプロパティで，API パラメータクラスが hasType というプロパティを通じてパラメータ型クラスを所有している関係を表している．緑の矢印はデータプロパティで，MinLength の値が 1 であることを表している．

API パラメータクラスとそのサブクラス，インスタンスは著者らの経験をもとに定義した．API パラメータクラスのサブクラスとして，API パラメータを解釈するうえでよくある概念を定義している．例えば Name クラスは名前を表す概念のパラメータで，Mail はメールアドレスを表すパラメータである．これらの基本的な概念に加えて，API パラメータクラスのサブクラスからインスタンスを作成し，具体的な制約の値を定義している．

例えば PetName はペットの名前というパラメータを表すインスタンスで，MinLength が 1 で MaxLength が 255 と制約の値を定義している．

パラメータプロパティクラスは OAS で定義されている Schema Object のプロパティをもとに定義した．同様にパラメータ型クラスは OAS で定義されている Data Types をもとに定義した．

3.4 OpenAPI Document とオントロジーのマッピング

提案システムではオントロジーを用いて制約を推薦するために，オントロジーのクラス間の関係を利用する．そのため，OpenAPI Document に記述されているパラメータを何らかの方法でオントロジー内のクラスとマッピングさせる必要がある．

OpenAPI Document とオントロジーのマッピングのためには前述の Mainas ら [10] のように OpenAPI Document の記述にオントロジーの情報を追記する方法があるが，この方法は OpenAPI Document の記述者がオントロジーを理解する必要がある．提案システムでは OpenAPI Document の記述者がオントロジーの内容を理解する必要をなくすため，OAS で定義されている利用可能なパラメータのデータ型（string や integer など）と，オントロジー内でパラメータのデータ型を表すクラスの対応表を用意し，OpenAPI Document に定義されているパラメータのデータ型からオントロジー内のデータ型クラスにマッピングする．

3.5 制約の検索

提案システムで用いるオントロジーでは，hasType プロパティや hasProperty プロパティを通じて，API パラメータクラスがパラメータ型クラスとパラメータプロパティクラスを所有する関係を定義している．また，API パラメータクラスのサブクラスによって具体的な API パラメータを表しており，親クラスの関係を引き継ぎ具体的な型や制約を，パラメータ型クラスとパラメータプロパティクラスのサブクラスを所有することで表現している．

これらの各クラスの関係をもとにオントロジーのクラスを検索することで，API のパラメータが持つべき制約を見つけることができる．

図 3 の例では OpenAPI Document の *name* が string のため，事前に用意したマッピングモデルから String 型クラスにマッピングする．次に API パラメータクラスのサブクラスを走査し，hasType プロパティを通じて String 型ク

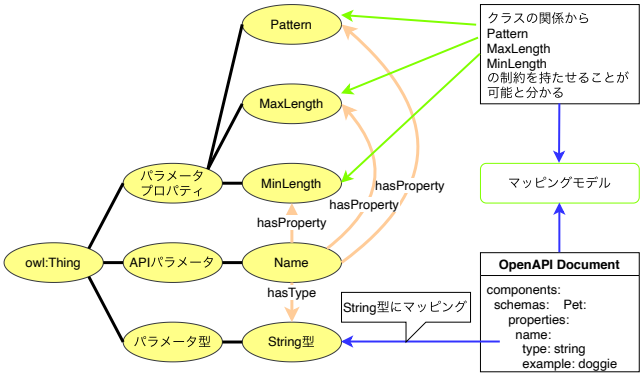

図 3: オントロジーとのマッピングと制約の検索

ラスを所有しているクラスを検索することで Name クラスを見つける．続いて Name クラスを走査して hasProperty プロパティを通じて所有しているパラメータプロパティクラスのサブクラス，Pattern クラス，MaxLength クラス，MinLength クラスを見つける．マッピングモデルから見つけた各パラメータプロパティクラスのサブクラスを OpenAPI Document のパラメータプロパティ（制約の記述）にマッピングする．最後に OpenAPI Document に既に書かれているパラメータプロパティと，検索して取得したパラメータプロパティを比較し，OpenAPI Document 側にないパラメータプロパティの記述を推薦する．図 3 の例では *name* に見つけた *maxLength*，*minLength*，*pattern* を記述するよう推薦する．

さらに，Name クラスのインスタンスがある場合，インスタンスが持つ具体的な制約の値を合わせて提案する．図 2 では Name クラスのインスタンスとして PetName を定義しており，MinLength の具体的な値として 1 を，MaxLength の具体的な値として 255 を定義している．この定義を利用する事で MinLength を推薦する場合は，単純に *minLength* を記述するよう推薦するのではなく *minLength: 1* を記述することを推薦する．

4. 実装

提案システムは図 1 に示すように，OpenAPI Document を操作する OpenAPI モデルと，オントロジー内を検索し制約を見つけるオントロジーモデル，OpenAPI モデルとオントロジーモデルの間でデータを交換するためのマッピングモデル，Web API のパラメータについての知識を記述したオントロジーで構成される．

提案システムの構築には Python3 と owlready2 パッケージ [11] を利用した．owlready2 はオントロジーを Python から操作するためのパッケージで，オントロジーのクラスやインスタンスを Python のオブジェクトとして操作できる．また，システムを構成する各モデルは Python のクラスとして構成し，データの構造に加えていくつかの処理を実行するメソッドを内包する．

4.1 OpenAPI モデル

OpenAPI モデルの主な役割は，ユーザから入力された OpenAPI Document からパラメータの定義リストを生成することである．リストには対象のパラメータ名，定義されている行番号，パラメータ定義をモデル化したオブジェクトの 3 つが保持されている．

OAS では Path Object と呼ばれるスキーマでパラメータを定義する．直接 Path Object 内でパラメータの定義を記述する方式と，Components Object と呼ばれるスキーマでパラメータの定義を記述し，$ref フィールドによって Path Object から参照する方式がある．Components Object を利用する場合，実質的なパラメータ定義は Components Object 側にあるため，パラメータの定義リストには，Components Object 側の定義のみ登録する．

$ref を利用する例としては，リスト 1 の 12 行目の`$ref: "#/components/schemas/NewPet"`がある．この場合パラメータの定義はリスト 2 で行われているため，OpenAPI モデルはパラメータが定義されている 8 行目や 10 行目をパラメータの定義リストに保持する．

4.2 オントロジーモデル

オントロジーモデルの主な役割は，オントロジーから制約を検索することである．オントロジーの検索には，owlready2 がサポートしている SPARQL[12] と呼ばれる問い合わせ言語を使用している．

リスト 3 に SPARQL で String 型を所有するクラスを検索するクエリの例を示す．SPARQL のクエリではほとんどの形式で，主語，述語，目的語の 3 つで構成されるトリプルパターンが含まれる．*?*で始まる単語は変数であり，オントロジー上の何らかの記述が入る．

```
1 select ?class
2 where {
3     ?s owl:allValuesFrom
      ↪  api_ontology:ParameterTypeOfString .
4     ?class ?p ?s
5 }
```

リスト 3: SPARQL による String 型を所有するクラスの検索例

3 行目は，主語が *?s*，述語が *owl:allValuesFrom*，目的語が *api_ontology:ParameterTypeOfString* のオントロジーを表す．ここで *api_ontology:ParameterTypeOfString* は String 型の値を表し，*owl:allValuesFrom* が目的語の値のみを持つことを表すことから，*?s* は，String 型の値を持つ何らかのプロパティを表す．

4 行目は，先に見つけたプロパティ *?s* を目的語にし，その他は変数になっていることから，String 型の値を持つ何

らかのプロパティと関係を持つクラスを表す．

最後に 1 行目の `select ?class` により，条件に合うクラスのみが抽出される．

4.3 マッピングモデル

マッピングモデルの主な役割は，OAS の記述とオントロジーのクラス間の相互変換である．このためには，OAS に定義されているデータタイプと対応するオントロジーのクラス，Schema Object のプロパティと対応するオントロジーのクラスでそれぞれリストを作成した．

例として表 1 に OAS のデータタイプとオントロジーのクラスの対応を示す．

表 1: OAS のデータタイプとオントロジーのクラスの対応

OAS のデータタイプ	オントロジーのクラス
string	ParameterTypeOfString
integer	ParameterTypeOfInteger
number	ParameterTypeOfNumber
boolean	ParameterTypeOfBoolean

4.4 オントロジーの構築

提案システムではオントロジーの構築にも owlready2 を使用した．owlready2 を使うことによって，Python のクラス定義を利用してオントロジーを記述できる．また，同パッケージを利用して推薦システムを構築しているため，オントロジーの構築とシステム開発との親和性も高い．

```
1  with onto:
2      class API_Parameter(Thing):
3          label = "API Parameter"
4          comment = "API のパラメータを表す"
5
6      class Name(API_Parameter):
7          hasType = ParameterTypeOfString
8          hasProperty = [Property_MaxLength,
9                         Property_MinLength,
                          ↪ Property_Pattern]
10
11     pet_name = Name("PetName",
12                     hasProperty=[Property_MaxLength,
                       ↪ Property_MinLength],
13                     Minlength=1, Maxlength=255)
```

リスト 4: オントロジー定義例

リスト 4 にオントロジーの定義の一部を示す．2 行目や 6 行目の `class` で始まる行は．Python のクラス定義でそのままオントロジーのクラスを表す，クラス定義の括弧内は親クラスを表しており，6 行目の定義では Name クラスは API_Parameter クラスの子クラスであることを表している．

11 行目は Python でのインスタンス生成部分で，Name クラスのインスタンスを生成している．この行はオントロジーでの PetName インスタンスを表す．

4.5 動作検証

提案システムの動作検証には，Swagger が GitHub で公開している OpenAPI Document[*2]を利用した．

例として，入力の一部をリスト 5 に示す．この場合，リスト 6 のような推薦内容が得られる．

```
1  id:
2    type: integer
3    format: int64
4    example: 10
5  name:
6    type: string
7    example: doggie
```

リスト 5: 動作検証用の入力例

```
1  line:742
2  name:id
3  maximum: 2147483647
4  minimum: 1
5  exclusiveMaximum:
6  exclusiveMinimum:
7
8  line:746
9  name:name
10 minLength: 1
11 maxLength: 255
12 pattern:
```

リスト 6: 推薦内容のデバッグ出力例

出力例の 1 行目 `line: 742` と 2 行目の `name: id` によって，OpenAPI Document の 742 行目に定義されている *id* というプロパティに対する推薦であることを示している．3 行目から 6 行目までが *id* に対する制約の推薦内容で，“maximum: 2147483647” は *maximum* という OAS で定義されているプロパティに，2147483647 を指定することを推薦する出力である．*exclusiveMaximum* や *exclusiveMinimum* についてはオントロジー内に値を定義したインスタンスが記述されていないため項目名のみが出力されている．

[*2] https://github.com/swagger-api/swagger-petstore/blob/master/src/main/resources/openapi.yaml

5. まとめと今後の課題

本稿では，OpenAPI Document に記述されている型情報に基づいて Web API のパラメータについての概念を定義したオントロジーを検索し，OpenAPI Document に記述されていない制約を発見し推薦することで OpenAPI Document の記述を支援するシステムを提案した．

提案システムでは，OpenAPI Document に不足している制約を発見し指摘するだけではなく，どのような制約値を指定するべきかも推薦する．これによって OpenAPI Document の記述者は，推薦内容が正しいかを判断するだけで制約を記述できる．本システムによって OpenAPI Document の記述者は簡単に不足している制約を記述でき，制約が記述されていない曖昧な状態によって発生する認識の齟齬という問題をなくすことができる．

提案システムによって OpenAPI Document に記述されているパラメータに不足している制約を推薦できることは確認したが，1）準備したオントロジーが小さく推薦される内容にバリエーションが少ない，2）型情報をもとにオントロジーとマッピングするため，適切でない値の推薦が行われる可能性がある，3）推薦を OpenAPI Document の記述者に提示するインタフェースが用意できていない，などいくつかの課題が残されている．

推薦内容のバリエーションが少ないという課題については，今後オントロジーの定義を拡充することに加えて，将来的には既存のオントロジーと組み合わせることを検討している．例えばデータベースの知識について扱うオントロジーと組み合わせることで DB の持つデータ型の知識をもとに最大値や最小値の制限を推薦するなど，組み合わせたオントロジーが持つ情報から制約の値を推薦できると考えられる．

適切でない推薦が行われる可能性があるという課題については，OpenAPI Document に記述されている型情報以外のパラメータもオントロジーの検索に利用することを検討している．例えば人はパラメータの名前からどのようなパラメータかを考えられるように，パラメータの名前をクラスのマッピングに利用することで，より適切なオントロジー上のクラスとマッピングすることが可能と考えられる．適切なクラスとマッピングできれば，制約で提案する値についてもより適切な値が提案できる．

推薦を記述者に提示するインタフェースが用意できていないという課題には，エディタの拡張機能としてインタフェースを用意することを検討している．エディタの拡張機能としてインタフェースを用意することで，実際に制約を記述する必要がある行に推薦を提示することや，定義の記述中に制約を推薦することを検討している．

参考文献

[1] OpenAPI Initiative: OpenAPI Specification, OpenAPI Initiative (online), available from ⟨https://spec.openapis.org/oas/v3.1.0⟩ (accessed 2021-08-26).

[2] SmartBear Software.: The State of API Report — 2020, SmartBear Software. (online), available from ⟨https://static0.smartbear.co/smartbearbrand/media/pdf/smartbear_state_of_api_2020.pdf⟩ (accessed 2021-08-26).

[3] Ed-douibi, H., Canovas Izquierdo, J. L. and Cabot, J.: Automatic Generation of Test Cases for REST APIs: A Specification-Based Approach, *2018 IEEE 22nd International Enterprise Distributed Object Computing Conference (EDOC)*, IEEE, pp. 181–190 (2018).

[4] Atlidakis, V., Godefroid, P. and Polishchuk, M.: RESTler: Stateful REST API Fuzzing, *Proceedings of the 41st International Conference on Software Engineering*, CSE '19, Vol. 2019-May, IEEE, pp. 748–758 (2019).

[5] Karlsson, S., Causevic, A. and Sundmark, D.: QuickREST: Property-based Test Generation of OpenAPI-Described RESTful APIs, *2020 IEEE 13th International Conference on Software Testing, Validation and Verification (ICST)*, IEEE, pp. 131–141 (2020).

[6] Hosono, M., Tokumoto, S., Monpratarnchai, S., Washizaki, H., Honda, K., Nagumo, H., Sonoda, H., Fukazawa, Y., Munakata, K., Nakagawa, T. and Nemoto, Y.: Inappropriate Usage Examples in Web API Documentations, *2019 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, IEEE, pp. 343–347 (2019).

[7] 柴田 晃，石橋勇人：QA テストのアクセスログを用いた Web システム負荷試験生成システムの提案，インターネットと運用技術シンポジウム論文集，pp. 103–104 (2020).

[8] Martin-Lopez, A.: Automated analysis of inter-parameter dependencies in web APIs, *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Companion Proceedings*, ICSE '20, ACM, pp. 140–142 (online), DOI: 10.1145/3377812.3382173 (2020).

[9] Karavisileiou, A., Mainas, N. and Petrakis, E. G.: Ontology for OpenAPI REST Services Descriptions, *2020 IEEE 32nd International Conference on Tools with Artificial Intelligence (ICTAI)*, Vol. 2020-Novem, IEEE, pp. 35–40 (2020).

[10] Mainas, N. and Petrakis, E. G.: SOAS 3.0: Semantically Enriched OpenAPI 3.0 Descriptions and Ontology for REST Services, *2020 IEEE 14th International Conference on Semantic Computing (ICSC)*, pp. 207–210 (2020).

[11] Lamy, J.-B.: Owlready: Ontology-oriented programming in Python with automatic classification and high level constructs for biomedical ontologies, *Artificial Intelligence in Medicine*, Vol. 80, pp. 11–28 (2017).

[12] W3C SPARQL Working Group: SPARQL 1.1 Overview, World Wide Web Consortium (W3C) (online), available from ⟨https://www.w3.org/TR/sparql11-overview/⟩ (accessed 2021-08-26).