ブロックチェーンに基づく SLA 契約の自動化方法と そのプラットフォームの提案と評価

M2015SE007 中島啓貴

指導教員 青山幹雄

1. 背景

Web リソースを利用しエンドユーザにサービスを提供するアプリケーションが増加している. 一方で, 利用の前提となる SLA 契約や, アプリケーションに対する実装の追加は自動化されていない. このことから, アプリケーションが利用できるリソースは, 契約と実装に制約されていることが明らかである.

2. 研究課題

SLA 契約と実装による制約を解決しアプリケーションが自動的に必要なリソースを活用できるようにするために, 以下を研究課題とする.

(1) 当事者の環境に依存すること無く, 共通インタフェースを用いて SLA 契約を実行可能なプラットフォームの実現

(2) SLA 契約と Web API の形式的な仕様記述の定義

3. 関連研究

3.1. ブロックチェーン

P2P ネットワークにおける分散データベースと, ハッシュチェーンを応用した分散型台帳技術である. 改ざん耐性と無停止の可用性を持つ.

3.2. Ethereum[3]

ブロックチェーンを応用した分散型アプリケーションプラットフォームである. EVM と呼ばれる仮想マシン上で, アプリケーション(コントラクト)を実行する. コントラクトのデプロイやコントラクト関数の呼び出しは, アカウント情報を含めすべてブロックチェーンに記録される. 操作を行うアカウントは, 公開鍵暗号による署名で保証される. コントラクトでは, 仮想通貨 Ether のユーザ間の送金も扱うことができる. 文書の保証, 契約の自動化などへの応用が期待されている.

3.3. Web Service Level Agreement (WSLA) [6]

Web サービスの SLA 仕様記述言語であり XML 形式でプロバイダとコンシューマ間の合意と当事者の責務を定義するための言語である. サービス自体の表現には, WSDL[2] を利用することが定義されており SOAP over HTTP による通信を前提としている.

3.4. Web API の仕様記述

API Blueprint [1] などの, Web API の仕様記述方法が提案されている. これらには HTML ドキュメントの生成ツールやエディタ, モックアップツールなどが提供され開発, 利用支援が行われている.

3.5. SHACL (Shapes Constraint Language)[5]

RDF グラフに対する制約定義シェイプ (Shape) を定義するための言語である. シェイプの制約により構造が定義された RDF グラフをシェイプのデータグラフと呼ぶ.

SHACL では, すべての制約に対して RDF のクエリ言語 SPARQL[4] のクエリと期待される出力が定義されており, これらを応用することで制約充足の検証が可能であるとしている. SHACL の仕様の一部として, シェイプを用いた RDF グラフの検証方法や検証結果の表現形式も定義される方針である.

3.6. シェイプを用いた RDF 文書検証 [7]

SPARQL を応用しシェイプ対する RDF グラフの充足性を検証する機構を提案している. この検証機構は, 制約違反箇所を出力する機構を備えており, データグラフの記述支援への応用が可能である.

4. アプローチ

4.1. 分散型アプリケーションに基づく SLA 契約

コントラクト(分散型アプリケーション)に基づく SLA 契約により, 当事者の実行環境から独立した環境で契約と決済が実行可能となる.

また, プラットフォーム上の複数の Web API と共通のインタフェースを介して SLA 契約を実行でき, 契約の自動化が可能となる(図 1).

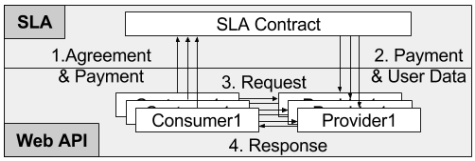

図 1 共通インタフェースによる SLA 契約

4.2. RDF 形式の仕様記述とシェイプの定義

RDF 形式の仕様記述を用いることで Web API および SLA の仕様記述は機械実行可能かつ, 拡張可能となる. また, 仕様記述の仕様のシェイプを定義することで仕様記述は, 検証可能, クエリ可能となる.

5. SSLAP (Smart SLA Platform)

分散型アプリケーションアーキテクチャに基づく, SLA 契約プラットフォーム SSLAP を提案する.

5.1. SSLAP の機能

SSLAP には, 以下の 7 つの機能が求められる, 機

能とアクタ間の関係を(図 3)に示す.

(1) プロバイダが SLA の情報と購読プラン(料金と期間)を登録できること
(2) バリデータが SSLAP に登録された SLA の情報を取得でき, SLA の仕様記述の検証結果が登録できること
(3) コンシューマが SSLAP に登録された SLA の情報と検証結果の情報を取得でき, SLA の仕様記述, 検証結果が取得できること
(4) コンシューマが SLA 契約を実行(プラットフォームへの送金)することで, プロバイダへの送金が行われること
(5) プロバイダが SLA 契約済みのコンシューマの情報を取得でき, 契約済みのコンシューマからの Web API リクエストを識別できること
(6) SLA により保証された Web API の利用に不具合が生じた場合, モニタによってプロバイダに対し保証を請求できること
(7) プロバイダが保証の請求を受理(プラットフォームへの送金)することで, コンシューマへの送金が行われること

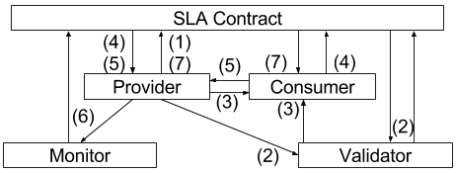

図 3 SSLAP の機能とアクタ間の関係

SSLAP は, 機能が呼び出された際, 呼び出し元の実行権限を確認することが求められる. すべての機能はトリガとなるアクタの任意のタイミングで開始できることが求められる.

5.2. 仕様記述が満たすべき性質

SLA 仕様記述は, その特性上必ず, Web API の仕様記述を参照することを前提としている. SLA 契約を実行と Web API の利用を結びつけるためには, これらの文書に以下の 4 つの性質が求められる.

(1) 仕様変更の表現可能性
(2) リソース連携のための拡張可能性
(3) 記述の検証可能性
(4) クエリ可能性

5.3. SSLAP の内部要素

SSLAP は以下の 5 つの要素の状態を扱う.

(1) SLA: SLA 文書の登録情報
(2) Plan: Web API の購読プラン
(3) Validation Result: 検証結果の登録情報
(4) Subscription: SLA 契約の情報
(5) Refund Request: 保証の情報

5.4. SSLAP コントラクト

SSLAP コントラクトを図 2 に示す. SSLAP コントラクトは, 18 のコントラクト関数と 5 つの構造体のコレクションにより, SLA 契約に関する状態を表現する.

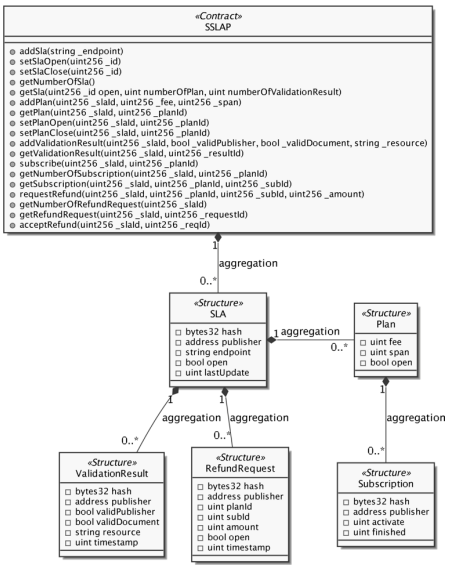

図 2 SSLA コントラクトと構成要素

5.5. URI 参照によるリソースの特定

コントラクト関数の引数および戻り値は, 固定長に限られる. そのため, SLA の仕様記述と検証結果の文書は, SSLAP に URI を登録するものとした.

5.6. 署名を用いた文書発行者の保証

リソースを URI により参照する場合, 発行者以外のアカウントが文書を登録する可能性が否定できない. この問題に対し, SSLAP では文書の登録情報をもとにハッシュを生成, 発行者はハッシュに対しプラットフォームの自身のアカウントの秘密鍵で署名を行い, 文書に添付するものとした.

5.7. Web API に対するリクエスト発行者の検証

SLA 契約成立時, 購読情報のハッシュを生成する. コンシューマは, 自身の秘密鍵とハッシュとナンスより署名を生成する. 署名とハッシュ, ナンスをリクエストに添付する. プロバイダは, リクエストの署名をハッシュとナンスを用いて復号しリクエスト発行者のアドレス

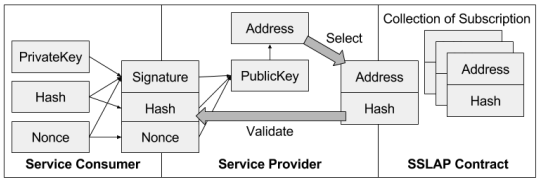

図 4 リクエスト発行者の検証

を生成する. SSLAP の SLA 契約者情報より対応するハッシュを特定し比較する(図 4).

6. SLAMS (SLA Metadata Specification)

RDF 形式で SLA の仕様記述を行う仕様 SLAMS を定義, SHACL を用いて SLAMS シェイプを定義した.

6.1. RDF 形式の利用

形式の特性上リソースの付加可能性を満たす. また, 更新情報もリソースとして表現することで, 仕様変更の表現を実現する.

6.2. SLAMS シェイプの定義

SLAMS における仕様記述の検証は, シェイプを用いた文書検証[5][7]を適用するものとする. データグラフは, シェイプを前提とすることで SPARQL によるクエリが可能である.

6.3. SLA 仕様表現に求められる表現

SLA 契約プラットフォームに基づく SLA 契約を実行するために SLA の仕様記述には, 以下の 3 つの表現が求められる.

(1) Web API の仕様記述への参照
(2) サービス水準と保証の定義
(3) 契約の当事者の情報

6.4. SLAMS の構造

SLAMS のモデルを図 6 に示す. SLAMS は WSLA に基づき仕様を定義した. SLA の仕様は, slams:SLAMS[1] リソースをルートリソースとする RDF グラフで表現される. slams:SLAMS リソースは, APIMS のリソース, サービスプロバイダのリソース, 及びサービス水準のリソース REST API の仕様記述への参照を持つ. サービス水準の達成判断に, モニタを利用する場合, サービス水準リソースはモニタリソース (slams:SupportingParty) を参照する.

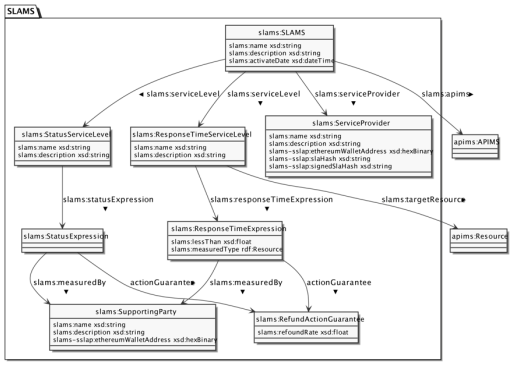

図 6 SLAMS のモデル

[1] slams: は APIMS の名前空間を表す.

7. APIMS (API Metadata Specification)

RDF 形式で Web API の仕様記述を行う仕様 APIMS を定義, SHACL を用いて APIMS シェイプを定義した.

7.1. REST API 仕様記述に求められる表現

SLA 契約プラットフォームに基づく Web API を利用するためには, 少なくとも既存の Web API 仕様記述と同等の表現能力が求められる.

7.2. APIMS の構造

APIMS のモデルを図 5 に示す. APIMS は API Blueprint に基づき仕様を定義した. APIMS の仕様は, apims:APIMS[2] リソースをルートリソースとする RDF グラフで表現される. apims:Content リソースは, tson:TSON[3] リソースへの参照を持つ.

API Blueprint では, JSON と JSON-Schema の表現に MSON (Markdown Syntax for Object Notation) 形式が利用されている. これに対応するため, RDF を用いた表現 TSON (Triples Syntax for Object Notation) を定義, SHACL を用いて TSON のシェイプを定義した.

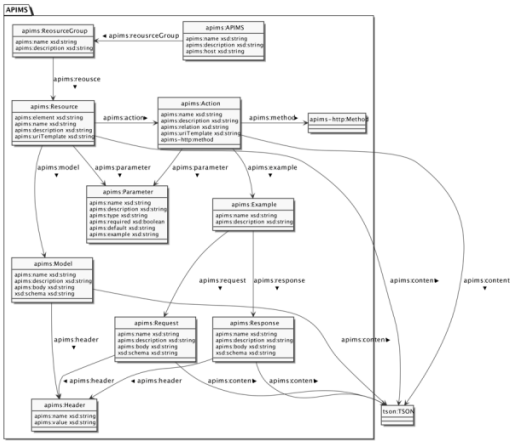

図 5 APIMS のモデル

7.3. TSON の構造

TSON は RDF 形式で JSON 及び JSON-schema を表現する記述仕様である. MSON 及び, Standard ECMA-40[8]を基にシェイプも合わせて定義した.

8. プロトタイプ実装と例題への適用

8.1. SSLAP コントラクトの実装

SSLAP コントラクトを Ethereum 上で動作するコントラクトを Solidity を用いて実装(300 (LOC)), EVM エミュレータ上にデプロイし, 利用シナリオを

[2] apims: は APIMS の名前空間を表す.

[3] tson: は TSON の名前空間を表す.

実行し実現可能性を確認した.

8.2. API Blueprint から APIMS への変換

API Blueprint のパーサを応用し, API Blueprint ドキュメントより SLAMS ドキュメントを生成するドキュメントコンパイラを Node.js で実装した(742 (LOC)).

9. 評価

9.1. SSLAP コントラクトの評価

利用シナリオを実行することで, 5.1 節で定義した機能が実現可能であることが確認できた.

送金を伴うコントラクト関数において, 呼び出し時の送金量が十分でない場合や, コントラクト関数を実行権限の無いアカウントが実行した場合は実行を例外により停止する実装が実現できた.

文書の発行元の検証, Web API に対するリクエストの発行元の保証は, Ethereum の署名関数 ecsign と複合関数 ecrecover により実現できた.

9.2. API Blueprint から APIMS への変換の評価

API Blueprint の仕様定義に用いられている 20 の example ドキュメントの変換を行い, リソースの欠落は発生せず, 正しい出力が得られた.

SLAMS は API Blueprint からの互換性をもち少なくとも同等の表現能力を持つことを示した.

10. 考察

10.1. オンデマンドな Web API の利用

Web API を用いるアプリケーションで新たなリソースを利用する場合, 開発者が契約と実装の追加を行う必要があった. 共通インタフェースの SLA 契約プラットフォームと機械実行可能な仕様記述により, 必要な契約のみアプリケーションがオンデマンドに契約可能になった. SLA 契約の単位をシステム全体から, 利用者に細分化し, アプリケーション構築がより柔軟になる.

10.2. 実績に基づく Web API の評価

Web API の公開情報は, プロバイダ毎に異なり比較は困難であった. SSLAP に基づく SLA 契約では, プラットフォーム上の情報と仕様記述はすべてアクセス可能でありため, 実績基づく Web API 評価が可能となった.

10.3. 拡張可能な仕様記述

SSLAP に基づく SLA 契約は, コンシューマが主導するので契約が成立するため, 仕様変更の履歴が利用可能である必要がある. さらに, 仕様記述は Web リソースを規定する文書であり, リソース間の連携のための拡張記述も必要である.

現在の Markdown に基づく仕様記述[1]では, 既存のスキーマを前提なり, 記述の拡張は困難である. RDF に基づく仕様記述は, 仕様変更履歴を表現するリソースや新たに付加したい情報を表現するリソースへの参照により, リソースの拡張が可能である.

10.4. 検証可能な仕様記述

自然言語による仕様記述では, 仕様に対する文書の充足性を判断できない. RDF を用いた仕様記述は, シェイプを用いた文書検証を適用できる. 検証可能性は, 現在提案されている仕様記述同様に HTML ドキュメントの生成ツールやエディタ, モックアップツールの提供が可能であることを示す[7].

さらに, シェイプに基づき定義した SPARQL クエリを利用することが可能であり, 汎用のクエリを利用した仕様の比較や発見の実装が可能である[5].

11. 今後の課題

(1) コントラクト関数の実行コストは, 当事者の負担となるため, 最小化が求められる.

(2) SHACL の仕様は策定段階でありシェイプを用いた文書検証の実装は未確立である.

(3) SLAMS では, 表現可能な保証, メトリクスの取得方法は限定的である.

12. まとめ

Web リソース利用をアプリケーションの増加に対し SLA 契約や, リソース連携のため実装の追加は自動化されておらず, リソース利用の制約となっている. 本稿では分散型アプリケーションを応用した SLA 契約プラットフォームと RDF を用いた SLA と Web API の仕様記述を提案し, プロトタイプを実装することで実現可能性を示した. これらより, オンデマンドに SLA 契約を締結し Web リソースを利用するアプリケーションが実現可能である.

13. 参考文献

[1] API Blueprint, https://apiblueprint.org .

[2] Basic Profile Version 1.0, http://www.ws-i.org/profiles/BasicProfile-1.0-2004-04-16.html, May 2004.

[3] Ethereum Foundation, Ethereum Project, https://www.ethereum.org/ .

[4] S. Harris et al., SPARQL 1.1 Query Language, https://www.w3.org/TR/sparql11-query/, Mar 2013 .

[5] H. Knublauch et al., Shapes Constraint Language (SHACL), https://www.w3.org/TR/shacl/, Aug 2016 .

[6] H. Ludwig et al., Web Service Level Agreement (WSLA) Language Specification, IBM, 2003.

[7] 中島 他., シェイプに基づく RDF 文書検証定義と検証方法の提案, FOSE 2015, 2015.

[8] Standard ECMA-404, https://www.ecma-international.org/publications/standards/Ecma-404.htm , Oct 2013 .