

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

中島 啓貴† 青山 幹雄†

南山大学大学院 理工学研究科 ソフトウェア工学専攻†

1. 背景

Web リソースの利用が拡大している一方で、SLA 契約やデータ連携のための実装は自動化されておらず、開発者に依存している。

2. 研究課題

- (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 API の形式的な仕様記述の定義

3. 関連研究

3.1 Ethereum[1]

分散型台帳技術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。Web リソースとの連携方法は未確立である。

3.2 Web Service Level Agreement (WSLA) [2]

SLA 仕様記述言語であり、XML 形式でサービスの当事者間の合意と責務の定義を表現する。

3.3 Web API 仕様定義

API Blueprint [3] などの、Web API の仕様記述が提案されており、開発支援ツールも提供されている。

3.4 SHACL (Shapes Constraint Language)[4]

RDF に対する制約(シェイプ)を定義する言語である。RDF のクエリ言語 SPARQL を応用した RDF 文書検証方法が仕様の一部として定義される。

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)。

4.2 RDF を用いた仕様記述の利用

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

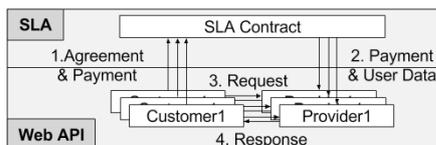


図 1 共通インタフェースによる SLA 契約

An Automation Method of SLA Contract and its Evaluation
† HIROKI NAKASHIMA, MIKIO AOYAMA, Graduate School of Science and Engineering, Nanzan University

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を提案する。SSLAP には、(1)SLA 契約

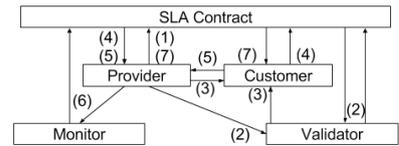


図 2 機能とアクタ間の関係

文書の登録(2)SLA 契約文書の検証結果の保証の登録(3)SLA 文書の一覧(4)SLA 契約および決済(5)契約済みコンシューマの一覧(6)モニタによる保証要求(7)保証要求の受理、返金 これらの機能が求められる。アクタの関係を図 2 に示す。

5.1 仕様記述の性質と検証

SLA 仕様記述及び、Web API の仕様記述には、リソースの拡張性、リソースのクエリ可能性、および、リソースの検証可能性が求められる。

5.2 SSLAP コントラクト

18 のコントラクト関数と 5 つの構造体により、SLA 契約を実現する(図 3)。

5.3 署名を用いた文書発行者の保証

SSLAP が発行する文書情報のハッシュと、発行者の秘密鍵による署名で、文書の発行者を保証する。

5.4 Web API に対するリクエスト発行者の検証

Web API リクエストを受信する際、SLA 契約が成立しているコンシューマであることを検証する必要がある。SSLAP は、SLA 契約成立時、契約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。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検証を行う。

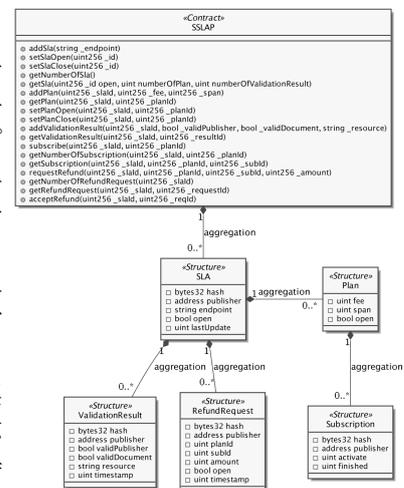


図 3 SSLA コントラクトと構成要素

6. SLAMS (SLA Metadata Specification)

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

情報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。

仕様記述の検証は、シェイプを用いた文書検証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。

6.2 SLA 仕様表現に求められる表現

SLA 契約を実行するために SLA の仕様記述には、(1)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の表現が求められる。モデルを図 4 に示す。

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。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。APIMS のモデルを図 5 に示す

API Blueprint では、MSON 形式が利用されているため、RDF を用いた表現 TSON (Triples Syntax for Object Notation) も定義し、APIMS 同様にシェイプを定義した。

8. プロトタイプ実装と例題への適用と評価

8.1 SSLAP コントラクトの実装

Ethereum のコントラクトとして SSLAP コントラクトを Solidity を用い実装し(300 (LOC))可能性を確認した。送金を伴うコントラクト関数と権限を伴う関数では、異常を検知し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には、Ethereum の署名関数で実現できた。

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コンパイラを Node.js で実装した(742 (LOC))。

API Blueprint が提供している 20 の example ドキュメントを変換し、リソースの欠落がない正しい出力を得

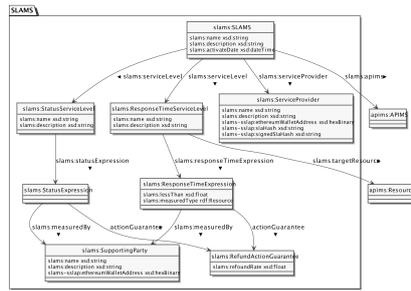


図 4 SLAMS のモデル

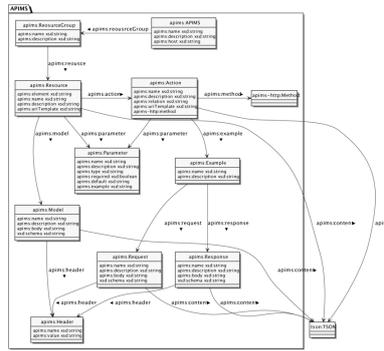


図 5 APIMS のモデル

た。この結果から、SLAMS は API Blueprint と互換性を持ち、同等の表現能力があるといえる。

9. 考察

9.1 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。

9.2 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 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、実績に基づく Web API 評価が可能となった。

9.3 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 に基づく仕様記述では、既存のスキーマを前提としており、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。

9.4 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、シェイプを用いて文書検証が可能である。

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。

10. まとめ

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。

参考文献

- [1] Ethereum Foundation, Ethereum Project, <https://www.ethereum.org/>.
- [2] H. Ludwig et al., Web Service Level Agreement (WSLA) Language Specification, IBM, 2003.
- [3] API Blueprint, <https://apiblueprint.org>.
- [4] H. Knublauch et al., Shapes Constraint Language (SHACL), <https://www.w3.org/TR/shacl/>, Aug 2016.