L-015

プロトコル解析におけるソフトウェアアーキテクチャ

Software Architecture for Protocol Analyzing

山田 博之† Hiroyuki Yamada
南澤 吉昭† Yoshiaki Minamisawa
畠山康博† Yasuhiro Hatakeyama

1．はじめに

コンピュータが普及している現在，通信ネットワークは社会基盤となり非常に重要なインフラであり，通信が失敗するということは社会全体へ計り知れない影響を与える．通信が失敗する原因の 1 つはコンピュータ同士で通信を行う際の基準となる，プロトコルと呼ばれる通信規約の相違である．技術標準化団体が策定するプロトコルの仕様に準拠することでコンピュータは相互通信を行うことができるが，現実には一部ソフトウェアベンダがプロトコルの中でも通信メッセージに関する規約に関して，独自の解釈でソフトウェアを開発しているため，通信に障害が生じてしまう．この場合，ソフトウェアがプロトコルに準拠するよう実装を改良すればよいのだが，そのためには通信メッセージのどの箇所がプロトコルに非準拠であるかを調査しなければならない．したがって，品質と作業効率を向上させるためにも，その調査のためにプロトコル解析用ソフトウェアが必要となる．

本論文では，通信メッセージに関するプロトコルに準拠しているか否かの適合性を解析するための解析要件をまとめ，プロトコル全般に適用可能な解析ソフトウェアのためのアーキテクチャを提案する．また，提案アーキテクチャを XML-Web サービス検証ソフトウェアに適用することで，提案アーキテクチャを，プロトコルを特定した解析モジュールに実装可能であることを検証する．さらに，提案アーキテクチャを適用した場合と既存技術を用いた場合との比較を行う．

2．プロトコル 解析の要件

本章では，通信メッセージに関するプロトコル全般を解析する際の「基本的な要件」，「通信の現状を踏まえた要件」，「解析ツールとしての要件」を示す．

2.1 基本要件

プロトコルの仕様は HTTP[1][2]など，さまざまなものが広く公開されている．その仕様書に記載されている規約のうち，通信メッセージに関する規約については，以下の 2 種類の規約に区別できる．

- 通信メッセージの構文規約
- 通信メッセージの意味規約

構文規約は，そのプロトコルではどのような文法構造を許すかを示したもので，その構造は一般的に BNF で表記されている．意味規約は，本論文では構文規約以外の通信メッセージに関する規約を指す．

† 北海道大学大学院 情報科学研究科

構文規約，意味規約ともに適合性を解析する対象となるため，基本要件として構文解析・意味解析の両方を行う必要がある．

2.2 現状を踏まえた要件

プロトコルの仕様については技術標準化団体が仕様書を公開しているのが一般的であるが，各ソフトウェアベンダはその仕様書に準拠した実装をしているとは限らない．実際プロトコルの仕様に一部準拠していないソフトウェアが，業界のデファクトスタンダードになっているという現状がある．そのソフトウェア独自のルール（以降，慣習ルール）が公開されている仕様書に基づいたルール（以後，標準ルール）のように振舞うため，コンピュータの相互通信性という観点からは慣習ルールに従う必要性がある．したがって，解析を行う際には慣習ルールを考慮するかどうかを吟味する必要があり，考慮する必要がある場合はそのルールチェックを実装しなければならない．

また，通信メッセージは同時に複数のプロトコルを基準としている場合がある．よってメッセージが複数のプロトコルにそれぞれ準拠しているかを解析しなければならない．その際，これらは互いに共通の対象領域を持つ場合があるので，効率の良い処理を行う必要がある．

2.3 解析ツールとしての要件

解析ツールとしてのプロトコル解析ソフトウェアは，ユーザの利便性を考えると，解析結果がより多く得られるべきである．そのためには，解析中にメッセージ内にプロトコル非準拠エラーが存在しても，そのエラーの影響の出ない範囲で解析を復帰・継続する必要がある．

以上，本章より導かれる通信メッセージに関するプロトコル解析の要件は以下の通りである．

- 構文解析
- 意味解析
- 慣習ルールの考慮
- 複数のプロトコル解析
- エラー復帰

3. 提案アーキテクチャ

本章では，２章に示したプロトコル解析の要件に対してのアプローチを示し，そのアプローチに基づき，通信メッセージに関するプロトコル全般に適用可能な，解析ソフトウェアのアーキテクチャを提案する．

3.1 解析要件へのアプローチ

前章で述べた解析の要件それぞれについて，以下のアプローチをとる．

- 構文解析・意味解析
  構文解析と意味解析の処理はそれぞれ分割して行う．構文解析，意味解析の順番で処理を行う．
- 複数のプロトコル解析
  各プロトコル解析用のモジュールをそれぞれ作成し，順に解析を行う．その際，各プロトコルで対象領域が重複している箇所があるので，その部分の重複した処理を共通化する．
- 慣習ルールの考慮
  標準ルールと慣習ルールのチェック処理を分割する．
- エラー復帰
  これを容易に行うために，構文解析を段階的に行う．そうすることで，メッセージのある箇所に構文エラーがあってもエラー復帰し，解析を続けることが出来る．

3.2 提案アーキテクチャ概要

アプローチに基づいたアーキテクチャの概要を図 1 に示す．

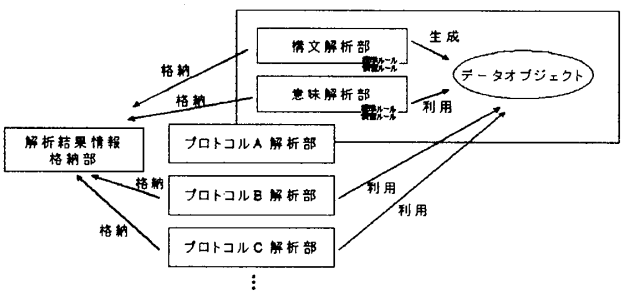

図 1　提案アーキテクチャのイメージ

提案アーキテクチャは解析したいプロトコルの数だけ，プロトコル解析部を持つ．各プロトコル解析部は構文解析部・意味解析部を持つが，解析を行うプロトコルの性質によってはどちらかしか持たない場合もある．また，他のプロトコル解析部で生成されたデータオブジェクトを利用するかしないかは各プロトコルの性質により異なる．各プロトコル解析部の処理順序は，他のプロトコル解析部で生成されたデータオブジェクトを利用するかどうかによる．そのため解析したいプロトコルの性質をよく吟味した上で処理順序を決めなければならない．

図 1 では，解析処理は以下のような流れになる．

1. 「プロトコル A 解析部」の「構文解析部」で，通信メッセージの構文解析を行い，解析結果を「解析結果情報格納部」に格納する．その際，副産物として通信メッセージ文が格納されたデータオブジェクトを生成する．このデータオブジェクトには通信メッセージの任意の情報を参照するための API を実装している．
2. 「プロトコル A 解析部」の「意味解析部」で，通信メッセージの意味解析を行い，解析結果を「解析結果情報格納部」に格納する．その際，先に生成されたデータオブジェクトから，必要な情報を参照する．
3. 「プロトコル B 解析部」，「プロトコル C 解析部」でも同様の処理を行う．必要な場合は他のプロトコル解析部で生成したデータオブジェクトから情報を参照する．

これらの処理を行うことで，解析要件である「構文解析」，「意味解析」，「複数のプロトコル解析」要件を満たしている．

また，各プロトコルの構文解析は，メッセージの最大構成要素から最小構成要素へと段階を踏んで行う．複数の文から構成されたメッセージの解析を例に取ると，以下の順で行われる．ただし，必要条件は各プロトコルによって異なる．

1. メッセージ全体において，メッセージが成り立つための必要条件について文法チェックを行う．文法エラーがあれば解析を終了する．
2. メッセージの各文において，文が成り立つための必要条件について文法チェックを行う．ある文に文法エラーがあれば，その文の解析を終了して，次の文の解析を行う．
3. 文法エラーがなかった文の各語句を解析する．語句に文法エラーがあった場合も，続けて次の語句を解析する．

このように構文解析を行うことで，文法にエラーがあった場合のエラー復帰が容易になる．よって解析要件である「エラー復帰」を満たしている．また，各解析段階で，解析処理を複数実装することで，慣習ルールを解析する．意味解析は，データオブジェクトから必要な情報を参照することで行う．各意味ルールのチェック処理を個別に実装するので，慣習ルール解析も追加することができる．よって解析要件である「慣習ルールの考慮」要件を満たしている．

以上より，本アーキテクチャは先に述べた解析要件を全て満たしている．

4．検証

本章では，先に提案したアーキテクチャを XML-Web サービス検証ソフトウェアに適用することで，提案アーキテクチャが特定のプロトコル解析モジュールで実装可能かを検証する．また，プロトコル解析を行うための既存技術との比較を行う．

4.1 XML-Web サービス検証ソフトウェアへの適用

XML-Web サービス用 SOAP 監視ソフトウェア（以後，SOAP Inspector と呼ぶ）に対して，3 章で提案したアーキテクチャの適用を行った．SOAP Inspector とは株式会社テクノフェイスで開発研究を進めているソフトウェアであり，リアルタイムで行われる XML-Web サービス通信で用いられる SOAP メッセージを取得し，その検証を行うシステムである．本研究では SOAP Inspector の開発研究に参加し，

HTTP・WS-I Basic Profile[3]解析モジュールの設計・実装を行った．

SOAP Inspector のシステム全体像を図 2 に示す．

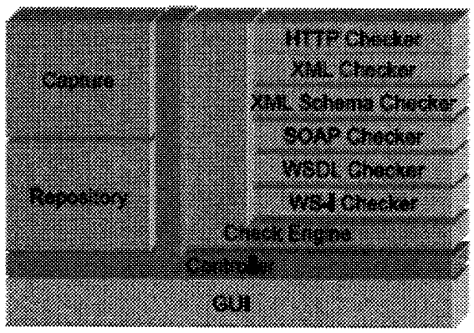

図 2　SOAP Inspector 全体像

SOAP Inspector のシステムは全部で 6 つのプロトコル解析を行うことができる．本論文ではこのうち HTTP Checker と WS-I Checker について，提案アーキテクチャを適用した．

まず HTTP Checker について説明する．HTTP では構文解析と意味解析を行う必要がある．HTTP Checker 全体のクラス図を図 3 に示す．

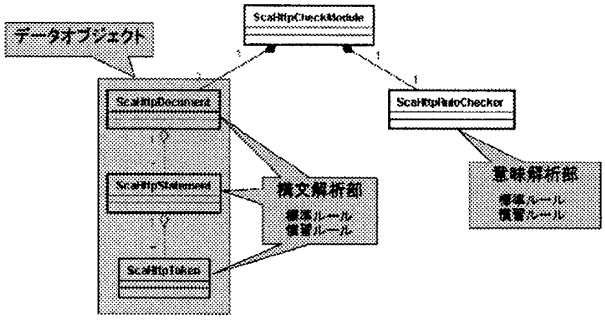

図 3　HTTP Checker のクラス図

ScaHttpCheckModule クラスは HTTP Checker 全体を統括しているクラスである．ScaHttpDocument クラス・ScaHttpStatement クラス・ScaHttpToken クラスはそれぞれメッセージ全体・文・語句の文字情報を持ったデータオブジェクトである．これらにメッセージ情報参照用の API を実装しているので，意味解析を行う際にその API を使うことでメッセージのメソッドや Header-Field などの情報を容易に参照できる．また，これらデータオブジェクトにそれぞれ文法チェック処理を実装することで，段階的に構文解析を行う．また，構文解析における慣習文法チェック処理もこれらデータオブジェクトに実装する．ScaHttpRuleChecker クラスは意味解析を行うクラスである．チェックすべき意味規約は，このクラスでそのチェック処理を実装する．その際，慣習意味ルールのチェックも ScaHttpRuleChecker クラスに実装可能である．

次に WS-I Checker について説明する．WS-I には構文解析規約は含まれていないので，意味解析部のみを実装した．意味解析を行う際は HTTP メッセージの情報が必要となるので，HTTP 解析部で生成されたデータオブジェクトを利用する．

以上より，SOAP Inspector に適用することができた．

4.2　既存技術との比較

メッセージの解析を行うための一般的な既存ソフトウェアとして Flex・Bison がある．Flex は字句解析プログラム生成ツールであり，Bison は BNF で表記された文脈自由文法を扱える構文解析プログラム生成ツールである．提案アーキテクチャと比較すると，Flex・Bison から生成されたプログラムを組み合わせたモジュールはメッセージの構文解析を行うことができる．また Bison の機能によりメッセージ内の任意の語句を参照できるため，意味解析を行うことができる．さらに，Flex・Bison を用いて複数のプロトコル解析モジュールを作成し，メッセージが複数のプロトコルに準拠しているかを解析することができる．

しかし，Bison を用いるとデフォルトでは解析するメッセージに 1 箇所でも文法エラーがある場合，解析が止まってしまう．エラー復帰を行うには，error トークンと呼ばれる Bison 独自の機能を，仕様書で BNF 表記されている構文木に適切に挿入しなければならない．したがってエラー復帰処理を実装することは困難である．また，仕様書で BNF 表記された構文木に慣習ルールを追加してしまうと，あいまいな構文木となったり，Bison では解析できない構文木となる可能性がある．その場合慣習ルールを解析できない．

以上より，Flex・Bison で作成したモジュールは「慣習ルールの考慮」，「エラー復帰」要件に対しての実装に問題がある．

5．おわりに

本論文では，通信メッセージに関するプロトコルに準拠しているか否かの適合性を解析するための解析要件をまとめ，プロトコル全般に適用可能な解析ソフトウェアのためのアーキテクチャを提案した．また，提案アーキテクチャを HTTP・WS-I Basic Profile 用解析モジュールに適用することができた．さらにその適用したモジュールと既存の技術を比較したことで，適用モジュールのほうが優れていることを示した．

今後の課題としては，提案アーキテクチャを他のプロトコル解析ソフトウェアに適用し，その有効性を検証することが挙げられる．

6．謝辞

本研究を行うにあたり株式会社テクノフェイスの皆様には様々なアドバイスを頂きました．この場を借りて，深く感謝致します．

参考文献

[1] R. Fielding, J. Gettys, et al., “Hypertext Transfer Protocol - HTTP/1.1 (RFC 2616)”, IETF, 1999

[2] T. Berners-Lee, R. Fielding, H. Frystyk, “Hypertext Transfer Protocol .HTTP/1.0 (RFC 1945)", IETF, 1996

[3] Keith Ballinger, David Ehnebuske, et al., "Basic Profile Version 1.0", WS-I, 2004