第2章　　本　　論

1. SOAP をベースとするミドルウェアアーキテクチャに関する仕様策定

1.1.　概要

Webサービス環境構築のためのミドルウエアとして､本プロジェクトではSOAPに基づくミドルウエアを開発した。SOAP は、IBM や Microsoft といったベンダが集まり、W3C[1]を中心に活発な議論がなされ、今なお標準化の過程にある。

本プロジェクト開始当初（平成１３年４月)、また本プロジェクトが終了する現段階においても、具体的な Web サービスのサービスモデルや、それらの基盤となるべきミドルウェアアーキテクチャについての議論が始まったばかりで、メッセージ仕様のみが標準化の過程にあり、Web サービスの全体的な枠組みも未だに明確になっていない。

このようなことから、本サブグループでは、本プロジェクトで開発する Web サービス・プラットフォーム“OpenSOAP”に求められる機能・性能を、各種の関連仕様はもとより、他の SOAP 実装を調査の上、アーキテクチャの根本から議論し、OpenSOAP ミドルウェアアーキテクチャ及び本プロジェクトで実現すべき機能、メッセージ仕様を決定した。

本章では、各種関連仕様や他の実装に関する調査のほか、ミドルウェアアーキテクチャの仕様策定までの議論について、そしてミドルウェアアーキテクチャの全体的な仕組みについて述べる。また本サブグループでは、ミドルウェアアーキテクチャの設計の他に、機能毎の開発スケジュールの設定、各サブテーマ実施機関との技術的な連絡調整、一般への公開に係る作業についても担当したので、その状況についても報告を行う。

[1] World Wide Web コンソーシアム　WWW で用いられる技術の標準化と推進を目的とする国際学術研究機関。W3C は IETF（Internet Engineering Task Force）のワークグループで、WWW に関する具体的な標準化項目について議論している。

1.2. SOAP 仕様および関連仕様に関する調査

W3C は、本ミドルウエアのアーキテクチャ設計当初当初、以下の SOAP 関連の文書を公開している[2]のみであった。

- Simple Object Access Protocol specification 1.0
- Simple Object Access Protocol specification 1.1
- XML Protocol Requirements Working Draft
- XML Protocol Working Group Charter

これらは、いずれも W3C の中でも Working Draft ないし Note という位置付けである。また、開発中に、以下の文書も出されている。

- SOAP Version 1.2 Part 0: Primer
- SOAP Version 1.2 Part 1: Messaging Framework
- SOAP Version 1.2 Part 2: Adjuncts
- XML Protocol Abstract Model Working Draft
- XML Protocol (XMLP) Requirements

このように、メッセージ仕様に関しては標準化に向けて作業中の段階であったが、それを具体的にどのように使って Web サービス環境を実現するのかまでは議論が及んでいない状態である。また、Web サービスの実現のために議論・研究されているプロトコル仕様に XML-RPC, WDDX, XMI, ebXML, BizTalk などさまざまなものがあるが、これもほぼ SOAP と同様程度の水準であり、本プロジェクトでも各々について調査を行っているが、本報告書ではその詳細については触れないこととする。

SOAP 仕様は、メッセージ（エンベロープ）構造、エンコーディングスタイル、HTTP トランスポートを用いた RPC[3]の実現を規定しているのみであるが、

本プロジェクトが SOAP に注目したのは、以下の点で優位と判断したためである。

[2]当然ながら、SOAP 仕様のベースとなる XML や HTTP 仕様は除外している。
[3] Remote Procedure Call　ネットワーク上の異なるマシンで処理を実行する手続き。

第一に、W3C というインターネットコミュニティーで最も中心的な組織が標準化を進めていること。すなわち、ここで標準化された仕様は、インターネット標準となることが最も有力であると判断されること。

第二に、XML や HTTP という、既存の既にインターネットよく利用されている技術をベースとしており、関連する要素技術の詳細仕様がすべて公開されており明確であること。

第三に、大手のコンピュータベンダーが標準化作業の中心として W3C に参加し、SOAP を実装した製品が出回り初めていたこと。

などがある。すなわち、Web サービス環境実現のためのプロトコルとして最も有力と判断した、言うことにつきる。

しかしながら、例えば電子商取引への適用で必須となるセキュリティの確保に関する方策がないなど、実環境に Web サービスを適用するためには、まだまだ不十分な状態であると認識せざるを得なかった。

1.3.　他の SOAP 実装

プロジェクト開始当初、SOAP を実装したツールキットなどには、以下のようなものがある。

- Microsoft 社
  The SOAP Toolkit for Visual Studio 6.0
- IBM 社
  Web Service Toolkit
- Apache プロジェクト
  Apache SOAP
- その他
  SOAP4R(Ruby), SOAP/Perl, SOAP::Lite, physon など

これらはいずれも、RPC を実現する最も基本的な程度の実装しかなされていない。また、このように既にいくつもの SOAP の実装がコンピュータベンダーや他のプロジェクトでなされていたが、各々の SOAP 実装へ依存した部分が多く、元々クロスプラットフォームでアプリケーション間通信を行うことを目的とした SOAP でもあるにも関わらず、相互接続性が確保されていないのが実状であった。

1.4. 開発目標

実環境におけるアプリケーション間通信、特に Web サービスを前提とした本格的なビジネスアプリケーションへ適用するには、SOAP 仕様に規定されていない範囲において、いくつもの機能や拡張が必要となるため、直ちに適用できるものとはなっていない。

前項までに示した調査結果を元に、本プロジェクトでは、以下に列挙する項目を開発目標に掲げ進められた。これらは当然ながら、設計および仕様策定に反映されている。

- W3C　SOAP 仕様に準拠すること。
- 複数ＯＳ、複数言語をサポートすること。
- セキュリティの確保
- 非同期処理(メッセージング)機能
- トランザクション機能
- メッセージルーティング

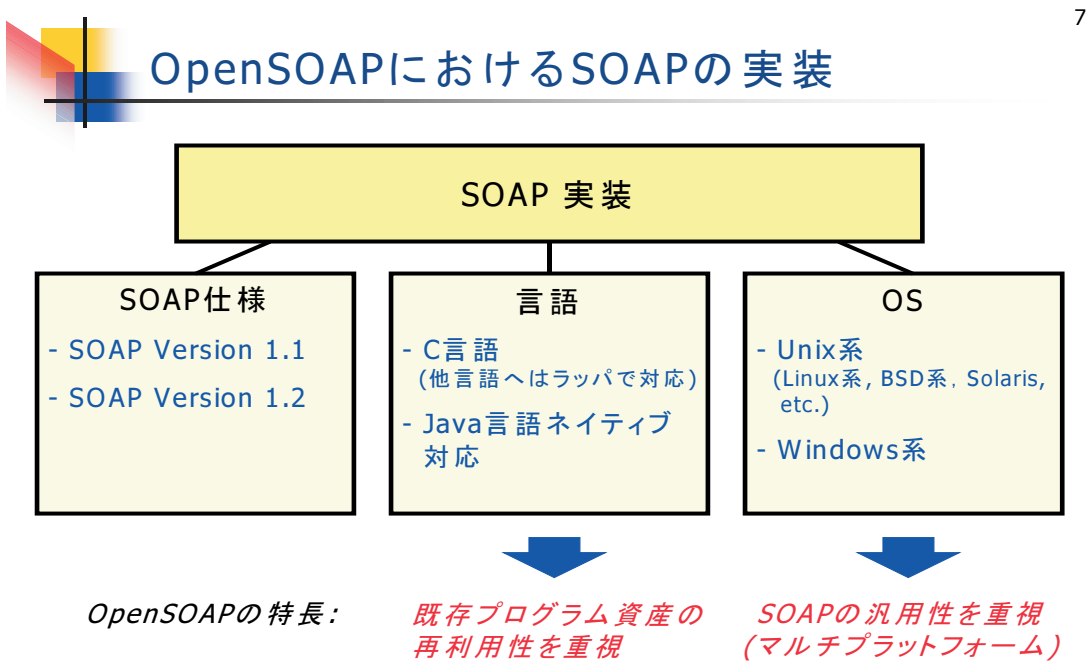

図 1.4.1 OpenSOAP にける SOAP の実装

- 多国語対応
- 拡張性・プラグインモジュール化
- 既存のＩＴ資産の有効活用
- デバッグ・トレース機能のサポート
- 高速化
- 開発ツール（ＡＰＩ）、ドキュメント、サンプルコードを含む、パッケージ・ソースコードの無償配布する。

1.4.1.　W3C　SOAP 仕様に準拠

SOAP 仕様 1.1 及び 1.2 に準拠し、他の SOAP 実装製品等との相互接続性を可能な限り確保する。

1.4.2.　複数ＯＳ、複数言語をサポート

既存の他の SOAP 実装は、java や perl 言語に特化したものが多いが、既存業務システムはC やC++言語で記述されたものが多く、効率よく Web サービスを構築するためには、C やC++言語での開発環境においても SOAP を利用可能としなければならない。また、Linux を含む UNIX 系 OS だけでなく、Windows 系 OS においても利用可能とし、特定の言語や OS に依存しない、かつそれらの間で完全に透過的なアプリケーション間通信が保証されなければならない。

1.4.3.　セキュリティの確保

ビジネスアプリケーションへ Web サービスの適用を考慮した場合、少なくとも中間ネットワークや中間ノードでのメッセージ改竄・なりすまし等に代表されるセキュリティーリスクからシステムを保護するためのセキュリティ機能が必要である。ビジネスアプリケーションへの SOAP 適用においては,公開鍵暗号方式を用いた、高度な暗号化及び認証機能が必要不可欠である。

一般的に、インターネットでは SSL のようなトランスポート層の暗号化技術を用いることによりセキュリティを確保する場合が多いが、OpenSOAP による SOAP-SOAP サーバ間でのメッセージ転送の実現には、メッセージ自体の暗号化や認証が必要となるため、SOAP メッセージの任意の特定部分を認証・暗号化できるライブラリを開発することによりこれを解決する。

1.4.4.　非同期処理(メッセージング)機能

実システムにおいては、RPC のような同期型通信ばかりではなく、いわゆる「バッチ処理」や「ファイル渡し」のような非同期型通信も多く採用されている。

特に、既存システムの SI(System Integration)には非同期通信が必須となるため、

リクエストメッセージのキューイングやレスポンスメッセージのスプーリング機能を実装する必要がある。

1.4.5. トランザクション機能

異なるサービス間でデータの整合性を保たなければならない Web サービスの構築を容易とするため、個々のクライアントやサービスソフトウエアにおけるデータの整合性管理ではなく、ミドルウェアアーキテクチャの基本機能として、トランザクション処理を支援するためのフレームワークを開発する。

1.4.6. メッセージルーティング

ファイアウォールの構築、SOAP メッセージ処理の分散化、サーバ管理の一元化などを実現するため、OpenSOAP ノード間でのメッセージ転送機能を実現する。また、転送されたメッセージのトレースやデバッグやトレースのためのメッセージ仕様及びそのためのツール群も開発対象として実装を進める。

図 1.4.2 に OpenSOAP における、メッセージ転送のイメージを示す。OpenSOAP サーバが稼動するホストと、OpenSOAP サービスが稼動するホストの分離を可能とし、また、OpenSOAP サーバと OpenSOAP サーバ間のメッセージの受け渡しも、SOAP 仕様に基づく HTTP バインディングによらない、他のトランスポートプロトコルによるメッセージ転送も可能とする構造とする。

1.4.7. 多国語対応

実システムで問題となる漢字コードなど、処理系によって異なる多国語の扱いについても、Web サービスをベースとするアプリケーション開発が、より容易となるよう、API （ライブラリ）にワイドキャラクタ文字列やマルチバイト文字列間の相互変換のための関数群を揃えることにより、多国語に対応する。

1.4.8. 拡張性・プラグインモジュール化

本プロジェクトの終了以後も、機能拡張、バグフィックスが継続して行わなければならない。保守性を向上させるためにも、開発するミドルウエアはモジュール化設計により、用途に合わせた様々なバリエーション、システム構成を構築可能な構造とする。

1.4.9.　既存ＩＴ資産の有効活用

SOAP と似た目的を持つ既存の分散オブジェクトアーキテクチャ技術にCORBA やＤＣＯＭがある。既存のＩＴ資産の有効活用をはかるため、CORBA やＤＣＯＭで開発されたアプリケーションと本プロジェクトで開発するミドルウエアとの相互接続性を実現する。

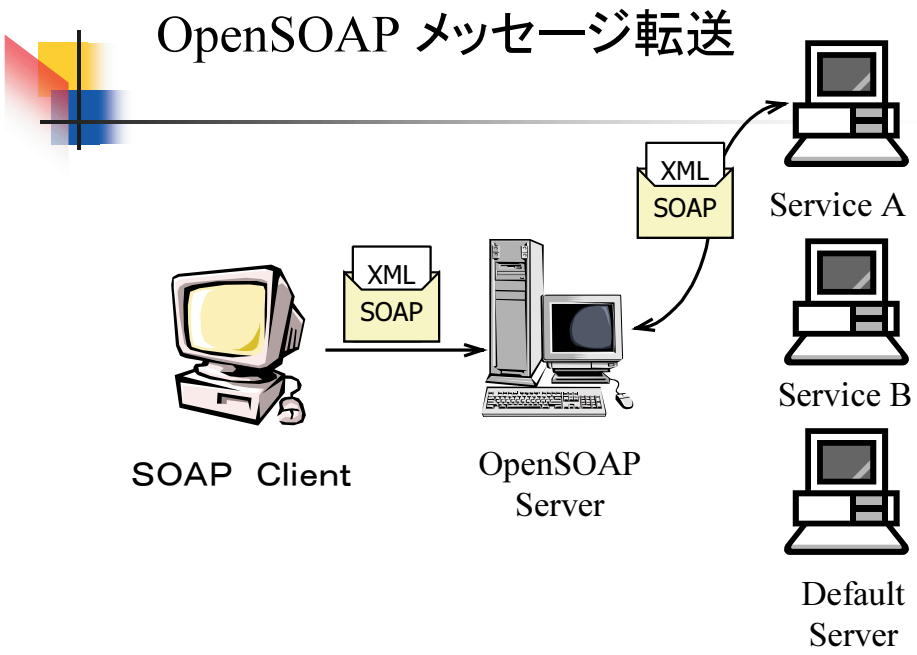

図 1.4.2 OpenSOAP メッセージ転送概念図

1.4.10. デバッグ・トレース機能のサポート

メッセージルーティングで、SOAP メッセージをサーバ・サーバ間でリレーしながら、サービス提供者に転送する場合、そのメッセージが何らかの原因で配送できない場合、メッセージ送信者は、どこに原因があって、どのように対処すべきなのか調査する術が備わっていなければならない。このため、メッセージのトレースなど、デバッグや調査を行うための仕組みを実現する。

1.4.11. 高速化

既に大手ベンダなどが製品としている SOAP を実装したアプリケーションサーバなどは、ApcheProject が開発した SOAP をベースとしている。このため、インタプリタが実行するという java の特性により、メッセージ処理に関わるパフォーマンスが期待できないという問題を抱えていることがわかった。本プロジェクトでは、C 言語を中心に OpenSOAP のコア部分（OpenSOAP API 及び OpenSOAP サーバ）を開発することにより、高速な SOAP メッセージ処理を可能とすることを図る。

1.4.12. 開発ツール（ＡＰＩ）、ドキュメント、サンプルコードを含む、パッケージ・ソースコードの無償配布

本プロジェクトの成果物のうち、OpenSOAP のコア部分はパッケージととして、ソースコードやドキュメントを無償で公開し配布する。これは、OpenSOAP を Web サービスの標準として広く利用してもらうことによって、事実上の標準として広く認知させることを意図している。このため、OpenSOAP 開発にあたっては、必然的に複数名のプログラマが並行的にコーディング作業をおこなうため、その記述の統一性を図るためにコーディングルールを作成し、またソースコード中のコメントについても、第三者が OpenSOAP を理解しようとする際の重要な意味を持つものと考え、その記述についても配慮を行う。

1.5. OpenSOAP アーキテクチャ

本プロジェクトが開発した、OpenSOAP のアークテクチャを図 1.5.1 に示す。

OpenSOAP は、大きく OpenSOAP クライアント、OpenSOAP サーバ、OpenSOAP サービスの３つのコンポーネントから構成される。

OpenSOAP クライアント

OpenSOAP クライアントは、リクエスト SOAP メッセージを作成し、トランスポートを通じ、OpenSOAP サーバへ送信する。また、OpenSOAP サーバ側から送信される、レスポンス SOAP メッセージを受信し、それを解析（パース）することにより、メッセージの送受信を行う。

OpenSOAP サーバ

OpenSOAP サーバは、OpenSOAP アーキテクチャの中核に位置し、この他にセッション管理、非同期通信のためのメッセージキューイング、サーバ・サーバ間でのメッセージ転送を受け持ち、OpenSOAP 独自の機能を実現及び利用するための処理を行う。OpenSOAP サーバでは、SOAP メッセージを受信するトランスポートインタフェースを OpenSOAP サーバから分離させ、プロトコル毎に個別のプロトコルインタフェースを用いることによって、例えば SMTP のような HTTP 以外のトランスポートプロトコルによるメッセージ交換も対応可能な構造とした。

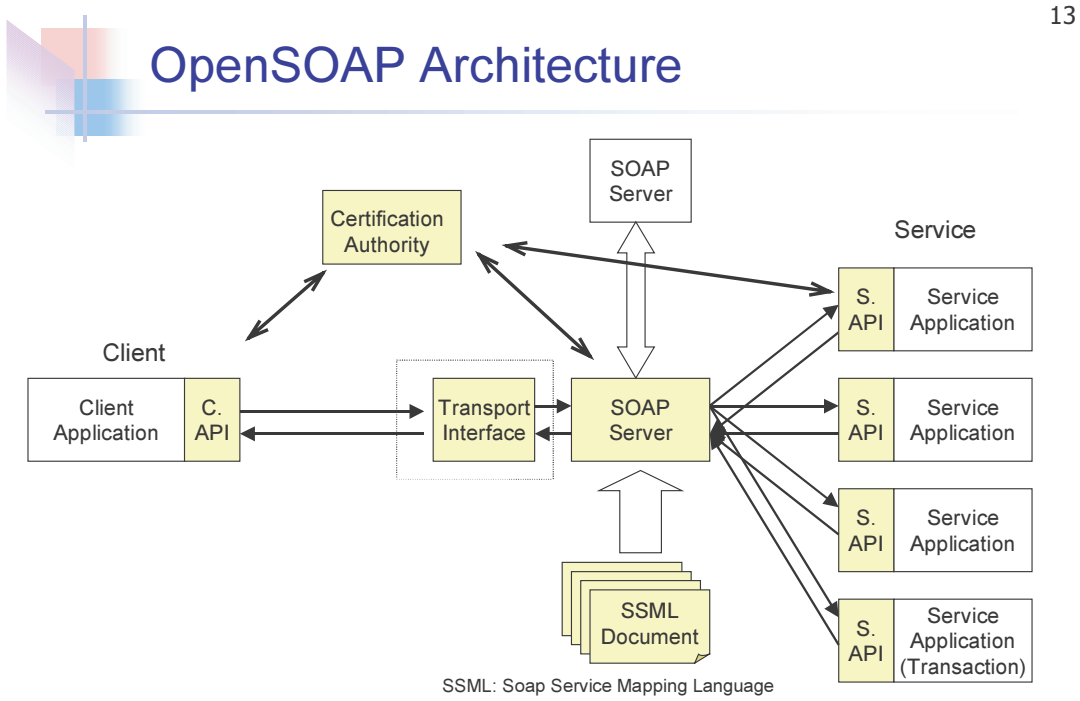

図 1.5.1 OpenSOAP アーキテクチャ

OpenSOAP サービス

OpenSOAP サービスは、OpenSOAP クライアントから送られたリクエストを処理し、その処理結果（レスポンスメッセージ）を OpenSOAP クライアントへ返す。

図 1.5.1 中、C.API 及び S.API は、各々クライアント用 API(Application Program Interface)及びサービス用 API を意味する。

C.API、S.API は、SOAP メッセージを作成、転送、パースするための関数ライブラリで、C 及び java 言語用を開発中である。これらには、多国語対応や公開鍵暗号方式を利用したセキュリティ機能、トランザクション機能を利用するための関数郡も含まれる。

Certification Authority は、OpenSOAP 用の認証局であり、Web サービスクライアント、OpenSOAP サーバ、サービスアプリケーションの公開鍵を配布するための Web サービスである。

また、トランザクション機能の実現のために、OpenSOAP サーバにトランザクション機能を持たせるのではなく、ひとつの Web サービスとして開発し実現を行った。

さらには、CORBA や DCOM など、他の分散アーキテクチャとのブリッジも、OpenSOAP クライアントと OpenSOAP サービス機能を組み合わせることにより実現している。

OpenSOAP サーバの内部構造については、クライアントからのリクエストメッセージを受け付けるトランスポートインタフェースを、OpenSOAP サーバ外部に切り離し、外部モジュールとしている。これは、現在の SOAP 仕様はトランスポートプロトコルとして HTTP を中心に議論されているが、将来の必要性に向けて、XMLP[4]などの新たな SOAP 用のトランスポートプロトコルの仕様が議論検討されていることを受けて、それをサポートすることが容易な構造にしてある。

これらはすべて、OpenSOAP サーバの機能を肥大化させるのを防ぐと同時に、モジュール化することにより、保守性と拡張性を高めることを狙いとしたものである。

1.5.1. メッセージ交換フロー

OpenSOAP アーキテクチャでの、基本的なメッセージ交換の流れは以下の手順で行われる。

[4] XMLP　XML Protocol

① SOAP メッセージを C.API を用い作成し、トランスポートインタフェースを介し OpenSOAP サーバへ送る。

② OpenSOAP サーバでは､送られてきたメッセージから OpenSOAP ヘッダを抽出し、予めサービス毎にサーバ及びサービスの挙動を規定した XML ドキュメントである SSML（Soap Service Mapping Language）ファイルに従い、クライアント側で指定したサービスソフトウエアを起動、または、メッセージルーティングを行う場合は、隣接する OpenSOAP サーバへメッセージを転送することにより、OpenSOAP サービスに SOAP メッセージを渡す。サービスが起動できない、メッセージの転送先が応答しない、メッセージの形式が不正であるなど、処理を実行できない場合のフォールトメッセージの作成・返送も、OpenSOAP サーバの役割である。

③ SOAP メッセージを S.API でパースし、リクエストを処理するサービスプログラムへ引数を渡す。サービスプログラムから返された戻り値は、S.API に含まれる C.API でレスポンスメッセージを構成し、リクエストメッセージが転送されてきたパス(フォワードパス)の逆（バックワードパス）をたどり、再び OpenSOAP クライアントへ戻される。

OpenSOAP におけるメッセージ交換の概念を図 1.5.2 の OpenSOAP 階層モデルに示す。

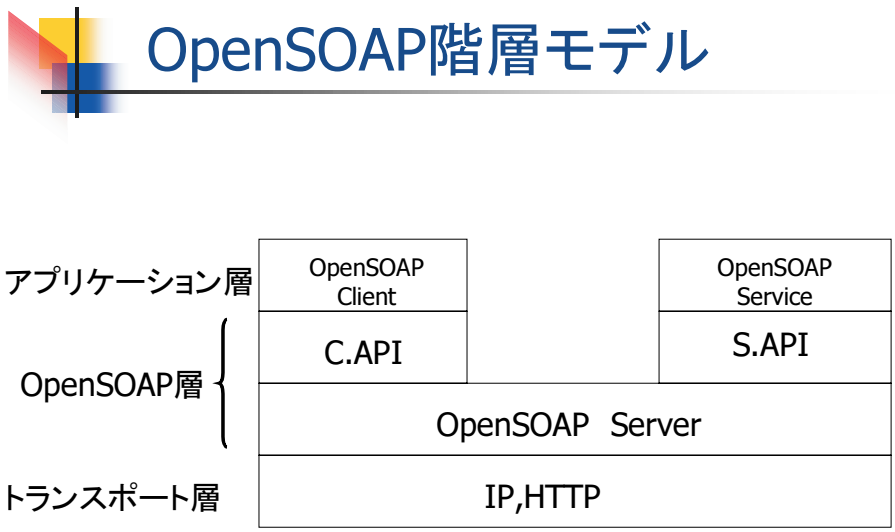

図 1.5.2 OpenSOAP 階層モデル

1.5.2. OpenSOAP ヘッダ

OpenSOAP では、非同期通信、サーバ・サーバ間メッセージ転送、トランザクション機能を実現するために、メッセージエンベロープのヘッダ領域に図 1.5.3 に示す例のような OpenSOAP ヘッダを埋め込むことにより、メッセージの管理情報を付加している。これは例えば、サーバ・サーバ間でのメッセージ転送の際に、メッセージのピンポン・ループの防止、メッセージ転送経路のトレースなどに用いられる。

OpenSOAP ヘッダは、他の SOAP 実装との接続で障害とならないよう、<opensoap-ext: …>と</opensoap-ext: …>タグで囲まれたブロックに埋め込み、OpenSOAP 以外の他の SOAP 実装ノードが誤って解釈したり、書き換えたりすることを防止している。

```
<opensoap-ext:opensoap-ext-block
    xmlns:opensoap-ext="http://opensoap.jp/ext/" mustUnderstand="1">
    <opensoap-ext:message_id>
      ServiceName.ServerName.RecieveDateTime
    </opensoap-ext:message_id>
    <opensoap-ext:from name="hoge" />
    <opensoap-ext:proccess type="async" count="second" ttl="200" />
    <opensoap-ext:received_path >
      <url>http://foo1.bar.com/trans_if.cgi</url>
      <time>2001-08-10T14:20:18+09:00</time>
    </opensoap-ext:received_path>
  </opensoap-ext:opensoap-ext-block>
```

図 1.5.3 OpenSOAP メッセージヘッダ(例)

1.6. 開発体制と開発計画

1.6.1. 開発体制

OpenSOAP コア部分の開発については、本サブプロジェクトが設計した OpenSOAP アーキテクチャをもとに、OpenSOAP メッセージ仕様、各コンポーネント間のインターフェース仕様を作成し、各々のサブテーマが担当部分を平行して開発を進めた。

例外を除き、毎週２回、各コア部分を担当するサブテーマの関係者が集まり定例ミーティングを行い、連絡調整を行った。その中で、仕様の見直し等を逐次行った。また、メーリングリストを活用し、開発中に発生した問題などを共有することにより、効率的な開発が行えるようにした。また、CVS[5]サーバを用いることにより、各開発者がコーディングした最新のソースコードを共有し、合わせてバージョン管理や履歴の記録も行った。

1.6.2. 開発計画

本プロジェクトは、OpenSOAP を開発すると同時に、それを用いたアプリケーションを開発することにより、OpenSOAP 及び Web サービスの有効性を評価する

[5] Concurrent Versions System　CVS はバージョン管理システムで、ソース・ファイルの変遷を記録するのに使用される。

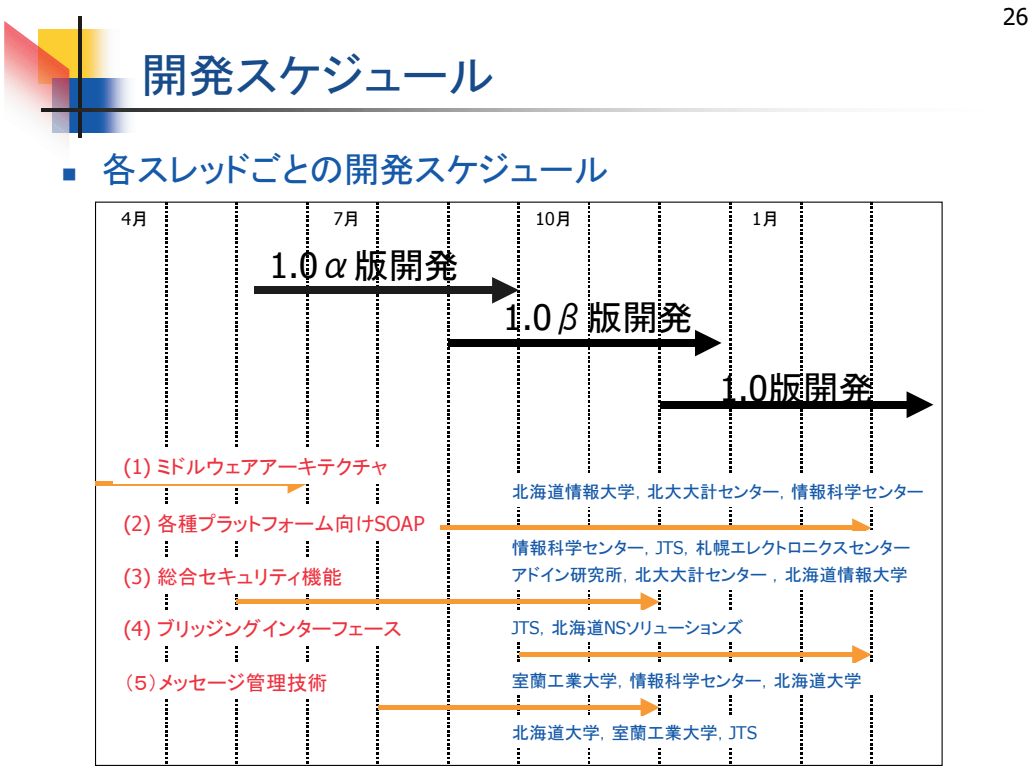

図 1.6.1 開発スケジュール

というミッションがある。しかしながら、OpenSOAP のコア部分のすべての機能の開発は、プロジェクト期間終了間近までかかる見込みであったため、OpenSOAP のコア部分の開発を待って評価・検証を行うのでは、評価・検証のための時間が十分に確保されないため、スレッド型の開発で開発を行うこととした。

図 1.6.1 にスレッド型開発による開発スケジュールを示す。また、図 1.6.2 に各 OpenSOAP コンポーネントと、開発を担当するサブテーマとの対応を示す。

OpenSOAP コンポーネント	サブテーマ
OpenSOAP API	各種プラットフォーム向け SOAP
	総合セキュリティ機能の実装
OpenSOAP サーバ	メッセージ管理技術の実装

図 1.6.2 OpenSOAP コンポーネントとサブテーマの対応

本プロジェクトでは、OpenSOAP1.0αバージョン、OpenSOAP1.0βバージョン、OpenSOAP1.0 の３つのバージョンを順次開発した。各々のバージョンの開発内容

	１．０α	１．０β	１．０
API	**単純型の一部**	**+Windows 対応**	**Java 用クラス**
トランスポート	**HTTP** **IPv6**		
OpenSOAP サーバ	**Linux のみ対応**	**+UNIX 対応**	**+Windows サポート**
非同期メッセージング		○	
メッセージルーティング	**同期のみ**	**+非同期転送**	
セキュリティ		**API**	**+認証局**
CORBA・DCOM ブリッジ			◎
サンプルコード **ドキュメント**	△ **（API リファレンス及び一部サンプルコード）**	○	◎

図 1.6.3 開発バージョンと実装機能

を図 1.6.3 に示す。

OpenSOAP1.0αバージョンは、プロジェクト内部評価向けの位置付けとして開発した。この段階では、OpenSOAP のアーキテクチャに基づき、基本的なメッセージの受け渡しを確認するために、プラットフォームも Linux のみを対象とした。

OpenSOAP1.0βバージョンは、OpenSOAP1.0αバージョンをベースに、非同期メッセージングやセキュリティ API, API の Windows 系 OS の対応やメッセージルーティング機能などの開発を行い、ここで OpenSOAP プロジェクトのホームページ（http://opensoap.jp）で公開し、一般の利用者もダウンロードして利用できるようにした。（OpenSOAP1.0βバージョンは、実際には一部機能実装の進捗の遅れから、β１バージョンとβ２バージョンバージョンの２種類をリリースしている。）

OpenSOAP1.0 バージョンは、トランザクション機能や OpenSOAP サーバにおけるサーバ署名の機能など、残りの機能を開発し、プロジェクト期間終了である平成１４年３月末までにリリースし公開した。

1.7. 使用許諾ライセンス

本プロジェクトは、SOAP をベースとする Web サービス構築用ミドルウエアを開発し、オープンソースとして配布することを主な目的としている。このため、本プロジェクトの成果物である、OpenSOAP パッケージは、ソースコードやドキュメントを含めすべて公開している。一般の利用者は、インターネットなどから入手し、その全部及び一部を自由に利用できるものとする必要がある。そのようなことから、OpenSOAP パッケージは、以下に示すようなライセンス表示を、OpenSOAP プロジェクトのホームページ、及びパッケージ中のドキュメントで表示している。これは Berkeley-based copyrights[6]と同様の条件である。

図 1.7.1 OpenSOAP の使用許諾ライセンス

[6]単純かつ制限の緩い非コピーレフトのフリーソフトウェアライセンス。

```
/*-------------------------------------------------
 * $RCSfile: OpenSOAP.c,v $
 *
 * See Copyright for the status of this software.
 *
 * The OpenSOAP Project
 * http://opensoap.jp/
 *-------------------------------------------------
 */
```

図 1.7.2 各ソースコードのヘッダ部分（例）

1.8.　OpenSOAP における暫定的制約事項

OpenSOAP が取り扱うメッセージに関する暫定的な制約事項を示す。

○１メッセージ/http セッション
○エンベロープサイズ　　Max 1024kByte
○ヘッダ
　Header name　Max。63 文字
○メッセージボディー
　method 名　　Max.　63 文字
　引数名　　Max.　63 文字
　型
　　string 型（Max. 64kbyte）
　　base64binary(Max. 1024kbyte)

これらの制約は、OpenSOAP アーキテクチャがサーバ・サーバ間のメッセージルーティング機構や、非同期メッセージング機能を有することに関し、他のＳＯＡＰ実装との相互運用性及び、安定的、効率的かつ安全に運用することを目的とし暫定的に定めたもので、今後の動向によって見直される可能性がある。

また画像や音声など大規模なデータ転送、いわゆるラージオブジェクトのハンドリングについては、W3C でも MIME エンコーディングやマルチパートなどの方

法によるファイル転送などが提案されているが、本プロジェクトでは、メッセージルーティングの支障となるため、この方法は採用していない。

1.9. 今後の課題

本プロジェクトでは、当初計画した機能の開発は概ね完了した。しかしながら、SOAP 仕様は標準化作業中であり、プロジェクト開始以後不定期に細かな仕様変更が W3C によりなされているが、適宜最新仕様をサポートする方針で開発を行った。OpenSOAP 開発している途中においても、随時、SOAP や XML 仕様など関連仕様などの変更が発生し、その度に本プロジェクトでも仕様細部の見直しを随時行い対応をしてきたが、今後もそのような対応が必要とされることが発生することが予想される。

バグなどの不具合への対応はもちろんのこと、本格的なビジネス Web サービスに適用した場合の機能評価、パフォーマンス評価、安定性評価などは開発期間中に十分に行われていないため、今後は実環境への適用評価を行いつつ、さらなる機能の追加、安定稼働させるための改良を行っていく必要がある。

本プロジェクトで開発されたミドルウエアは、オープンソースとして一般に公開をしたため、今後は利用者からの苦情・要望等の反応が期待できる。このため、OpenSOAP コミュニティーの拡大を図りながら、Web サービスの容易な構築実現のために、開発を継続して行っていく予定である。

また、Web サービスインターフェースの記述に用いる WSDL[7]も標準化作業が開始されていることや、WSFL[8]などのように Web サービスのフローを記述する仕様も提案されているなど、今後の動向に対応した開発を行っていく必要がある。

---

[7] Web Services Description Language　Web サービスが提供する機能を記述するための、XML ベースの言語仕様の 1 つ。Web サービスによって実現される分散オブジェクト環境において、各 Web サービスオブジェクトがどのような機能を提供するのか、その機能を利用するために、どのようなパラメータを渡す必要があるかなどを標準的なデータ仕様で記述できるようにし、Web サービス同士のコミュニケーションを可能にする。

[8] Web Services Flow Language　Web サービスのフロー定義を行うための XML 言語。WSFL により複数の Web サービスを一連のサービスとしてまとめあげるといったことが可能になる。

2.　各種プラットフォーム向け SOAP 実装

2.1.　はじめに

SOAP は XML をパケットデータとして用いたサービスやオブジェクト、サーバにアクセスするためのワイヤプロトコルであり、そこで用いられる XML は、拡張可能なマークアップ言語であり、これまでにインターネット上で使用されていた HTML に、SGML の柔軟性とパワーを組み合わせるように、かつ、SGML より単純化されるように設計された。XML はいわゆるテキストデータのため、マシンアーキテクチャ等に依存しない形でデータを構築できる。そのため、これを用いた SOAP では、マシンアーキテクチャや OS に依存しないという特徴を持つことができる。また、XML は SGML より単純化されるように設計されているため、テキスト処理が可能なプログラミング言語ならば、どのようなものでも、XML 処理プログラムが SGML 処理プログラムよりも簡単に構築することが可能であり、それゆえ、プログラミング言語にも依存しないことになる。

以上のような特徴をもつため、SOAP を用いることにより、異なるプラットフォーム間での計算機資源の共有を実現できる。これにより、プログラムから利用可能なアプリケーション・コンポーネントである Web サービスの枠組が構築可能となった。そのため、現在ではさまざまな、SOAP の実装が行われているが、その多くが、Java 等の特定のプログラミング言語による実装のため、プログラミング言語に依存しないという特徴がいかされていない。また、プログラミング言語に依存しない実装も存在はするが、それは、特定 OS 上でしか、実行できないため、プラットフォーム非依存とはなっていない。これらの問題を解決するために、本サブテーマでは、まず Linux を含む UNIX 系 OS と Windows 系 OS で動作する C 言語による実装を行うことにした。UNIX 系 OS と Windows 系 OS を選んだ理由としては、現在稼働しているコンピュータの多く、特に、現在 SOAP を利用可能なものは、ほとんどが上記のカテゴリに含まれるためである。また、C 言語による実装を行う理由としては、ひとつには、C 言語は他のプログラミング言語との親和性が高いこと、また、現在では、C 言語はほとんどのプラットフォームで利用可能なこと、さらに、前述したプラットフォーム上での現在までに作成されたプログラムの多くは、C 言語での実装がなされており、そのため、過去の資産を利用可能となり、かつ、C 言語のプログラマは、Java 等の他のプログラミング言語をおぼえる必要無く利用可能となること、以上三つの理由による。

本サブテーマにおける各プログラミング言語から利用可能な SOAP 実装を OpenSOAP API と呼ぶ。C 言語における OpenSOAP API は図 2.1.1 に示すように三