SIP アプリケーションフレームワークの開発と適用

小高敏裕[†] 松塚貴英[†] 野村佳秀[†]
村上雅彦[†] 山本里枝子[†]

Session Initiation Protocol（SIP）は，二者間の P2P セッションをサーバを介して確立するためのプロトコルである．SIP アプリケーションの開発は，複数のエージェントとサーバ間のインタラクションを定義しなければならないために，一般的に HTTP を利用した Web アプリケーションの開発よりも複雑化する傾向がある．一方，SIP アプリケーションにおける開発環境は Web アプリケーションほど抽象化が進んでいないため，現状ではビジネスアプリケーションの開発においてもよりプロトコルレベルに近いレイヤの実装を開発者が行わなければならないという課題がある．本稿では，SIP におけるビジネスアプリケーションの効率化に焦点を当て，開発者がプロトコルレベルの開発に煩わされることなくビジネスロジックの開発に集中できるようにするフレームワーク技術についての提案を行う．

A Framework for SIP Application Development

Toshihiro Kodaka,[†] Takahide Matsutsuka,[†]
Yoshihide Nomura,[†] Masahiko Murakami[†]
and Rieko Yamamoto[†]

Session Initiation Protocol (SIP) is a protocol used to establish a point-to-point connection via a server. The development of SIP applications generally tends to be more complicated than that of web applications using HTTP because it needs to describe interactions between multiple agents and servers. On the other hand, the environment for the development of SIP applications has not been abstracted like that of web applications, so developers must implement programs that are close to protocol layer, for the development of the business applications. In this paper, we propose a framework technology that enables developers to focus on the development of the business logic, not bothered with protocol-level programming.

1. はじめに

SIP は，IP ネットワークを利用する二者間の通話等のセッションを確立することを目的としたプロトコルであり，セッション確立後の通信をエージェント間で直接行えるようにすることで，サーバへの負荷を抑えることを特徴としている．SIP プロトコルの詳細は RFC3261[1)] で定義されるが，SIP は HTTP と似たプロトコルであり，ヘッダ部とボディ部に分かれたテキスト形式で記述された SIP リクエストおよび SIP レスポンスを，エンティティ間で送受信することで通信が行われる．SIP を利用してサービスを行うサーバ上に配備されるアプリケーションを SIP アプリケーションと呼ぶが，その開発方法としては，CPL，SIP CGI，JAIN SIP Servlet[3)]（SIP Servlet）等の方法が提案されている．SIP Servlet は，HTTP における Servlet と同様に，通信に関する部分を Servlet コンテナにより隠蔽し，受信する SIP メッセージに応じて処理を振り分けることで，SIP Servlet で規定されたインタフェースを持つ SIP アプリケーションを呼び出す仕組みである．SIP Servlet では，SIP サーバが SIP リクエストおよび SIP レスポンスを Java 言語のインスタンスとして SIP アプリケーションに渡すことができるため，SIP アプリケーションの開発者は SIP の詳細を RFC どおり厳密に記述しなくても，SIP Servlet が代替して処理を実行するという利点がある．

一方ビジネス面からは，SIP は音声等の従来の Web アプリケーションでは取扱いが難しい分野への利用が可能なため，特に Web アプリケーションとの融合といったより複雑なアプリケーションへの期待が高まっている．

しかしながら，SIP は HTTP と似た面もあるものの，純粋にリクエストを待ってレスポンスを返すだけ

† 株式会社富士通研究所
Fujitsu Laboratories LTD.

の HTTP と比較すると非常に複雑で，Web アプリケーションに慣れた開発者でも容易に使いこなせない面がある．Web アプリケーションとの連携も，IETF に基づく 3PCC（3rd Party Call Control）[5] に従うように，HTTP からの受信情報を基に SIP メッセージを 1 つ 1 つ組み立てる必要があり，開発者への負担が重い．

本稿では，今後増加が想定される，携帯電話等を利用した音声サービスと連携する Web アプリケーションの開発を中心に，その開発の効率化を実現するフレームワーク（SIP フレームワーク）について提案する．

2 章で SIP と HTTP の対比を行い，3 章で SIP Servlet の概要と課題について述べる．4 章で SIP フレームワークのアーキテクチャについての概要を述べた後に，5 章でその構成と動作の詳細を記述する．6 章で，本技術適用時のソースコード例を提示し，そのソースコードによる効率化と，より複雑なアプリケーション上における機能の十分性やコンポーネント部品の保守性について評価する．7 章で関連する技術に関する考察を行い，最後に 8 章でまとめを行う．

2. HTTP および SIP における開発環境

HTTP では当初，動的に出力を変化させるアプリケーションの多くは CGI（Common Gateway Interface）により OS の標準入出力を介してプロセスを呼び出すことで構成されていた．アプリケーションサーバの登場によって，コンテナが HTTP リクエストを一括して管理することにより，パフォーマンスの向上だけでなくアプリケーション側におけるセッション管理が容易になる等の，開発効率化の効果をもたらした．現在では，**図 1** の上図に示す Apache Struts [4] のようなフレームワークのように，MVC（Model View Controller）アーキテクチャにより HTTP リクエストの振り分けを行い，ビジネスロジックとビュー層を分離する開発が主流になっている．

SIP についても HTTP を後追いするような進化をたどっており，当初は単一の SIP メッセージを処理する CPL（Call Processing Language）や SIP CGI によりアプリケーションが記述されてきたが，2003 年に SIP Servlet の仕様が固まったことで，本仕様に準拠する Interstage SIPnet Application Container [6] を代表とするアプリケーションサーバの提供が始まっている（図 1 の下図）．しかし，SIP においては，HTTP Servlet におけるビュー層が存在しないことから，MVC アーキテクチャに基づくフレームワーク技術は提案されていない．本稿では，MVC におけるビュー層をセッションの遷移に置き換えたフレームワークアーキテクチャを提案する．なお，本稿においては，ある SIP または Web のセッションから他のセッションを呼び出す動作を「遷移」と呼ぶこととする．

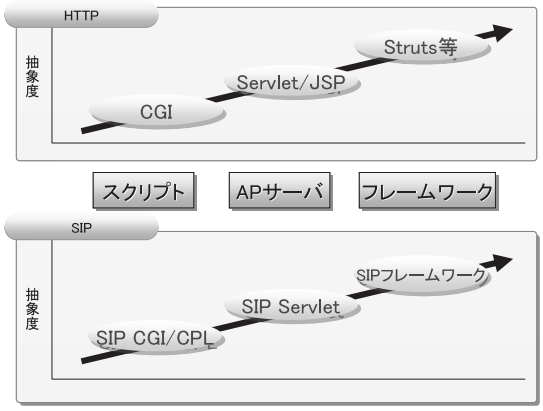

図 1 HTTP および SIP における開発環境の変遷
Fig. 1 Change of the development environment for HTTP and SIP.

3. SIP Servlet の概要と課題

3.1 SIP Servlet の構成

SIP Servlet は大きく分けて 2 つの要素から構成される．1 つはサーバとしての運用環境であり，外部からの SIP リクエスト/SIP レスポンスの送受信を行い，そのメッセージを抽象化，カプセル化して SIP アプリケーションに通知する．その際，セッション管理等の下位レイヤの共通処理を RFC3261 に従って行えるような API を提供している．2 つ目の要素は開発環境であり，SIP Servlet API に基づくアプリケーションを開発者が作成すれば，定義ファイルを利用して自在に SIP アプリケーションをカスタマイズすることができるようになっている．

このような SIP Servlet の動作は，RFC3261 に準拠するように HTTP Servlet をベースに改良することで仕様が決定されている．このため，HTTP Servlet の開発経験者には比較的習得が容易な構成となっており，特に SIP メソッドに応じた Java メソッドの呼び出し方法については HTTP Servlet と同一の仕様になっている．たとえば，「REGISTER」SIP メソッドを SIP サーバが受信した場合，呼び出される Java メソッドは doRegister() となる．

3.2 SIP Servlet の課題

SIP Servlet を利用した開発は，HTTP を前提とした開発と比較して，以下に示す課題のために処理の複雑さを招き，開発を困難なものとしている．

(1) 通信相手の増加による処理の複雑さ

SIP と HTTP の間のプロトコル仕様における根本的な違いから，SIP Servlet には HTTP Servlet に対するいくつかの拡張がなされている．1 つは，SIP サーバが他のエンティティ（SIP サーバ，ユーザエージェント等 SIP を利用して通信を行う機器の総称）に対してリクエストを送信する機能が付加されたことである．HTTP ではサーバはリクエストを待ち，受信したリクエストに対して作成したレスポンスを発信元に返すという比較的単純な作業となるが，SIP の場合にはリクエストを受信した際に，別のリクエストを他のエンティティに送信し直すというような処理が発生する．このため，SIP Servlet ではリクエストやレスポンスを送信するための send() メソッドが追加されている．このメソッドは送信先等のヘッダ情報が揃っていなかったり，二重送信となったりするような条件では例外を投げることが定義されており，取扱いが難しい処理の 1 つとなっている．

(2) セッション管理の複雑さ

セッション管理は SIP Servlet で拡張され，SIP セッションと SIP アプリケーションセッションの 2 種類が用意されている．SIP セッションは HTTP におけるセッションとほぼ等価な位置付けであり，二者間の関係を表している．しかし，SIP は上述のようにサーバがリクエストを他のエンティティに送信したり，HTTP と連携する仕組みを実現したりするために，より複雑なセッション体系を必要とする．この仕組みが SIP アプリケーションセッションであり，これがアプリケーション全体を取り扱うために必須の要素となっている．

(3) Web アプリケーション連携の複雑さ

HTTP と連携する 3PCC アプリケーションでは，連携のために HTTP の情報を SIP Servlet に受け渡したうえで，SIP アプリケーションセッションを開始する必要がある．この情報の受け渡しには，Servlet としての共通の機構であるリクエストディスパッチャが利用できるが，SIP アプリケーションセッションに必要な情報をリクエストディスパッチャにコピーしたり，SIP アプリケーションを呼び出した後に SIP リクエストの作成，送信等を行ったりする処理を，各開発者がプログラムとして記述する必要があり，プログラムが複雑になる．また，3PCC に関しては IETF [5] により複数の SIP アプリケーションセッションの確立方法が定義されており，どの方法を使ってどのように実装するかは開発者依存となっている．

SIP フレームワークの目的は，これらのプロトコルに近いレイヤの開発作業に対する開発者の負担をできるだけ小さいものとし，開発者がビジネスロジック部分の処理の開発に集中できるようにすることにある．

4. アーキテクチャ概要

4.1 設計方針

上述の課題を解決するために，以下の設計方針で開発の効率化を実現する．

(1) 通信相手の増加による処理の複雑さの解消

頻繁に利用される機能の雛形をフレームワークとして用意し，プロトコルレベルの処理を隠蔽・吸収することで，3.2 節 (1) の問題を解決する．

(2) セッション管理の複雑さの解消

開発者が記述するプログラムから SIP アプリケーションセッションの雛形を生成する機能を呼び出せるようにすることで，3.2 節 (2) の問題を解決する．

(3) Web アプリケーション連携の複雑さの解消

HTTP から SIP アプリケーションセッションを呼び出すためのライブラリおよび JSP カスタムタグを提供することで，3.2 節 (3) の問題を解決する．

4.2 全体アーキテクチャ概要

SIP フレームワークのアーキテクチャは**図 2** のようになる．SIP フレームワークはアプリケーションサーバ上にデプロイされる SIP Servlet のアプリケーション（SIP アプリケーション）として動作し，必要に応じてアクションと呼ばれる業務部品を呼び出す．アクションの単位は SIP（や HTTP）のメッセージ単位ではなく，ビジネスアプリケーションにおける再利用化を前提としたロジック単位としている．なお，アクションは SIP フレームワークで定義されたインタフェースを実装する Java クラスである．

4.3 SIP フレームワークにおける MVC アーキテクチャ

SIP フレームワークの全体アーキテクチャは MVC アーキテクチャを原型としている．通常 MVC アーキ

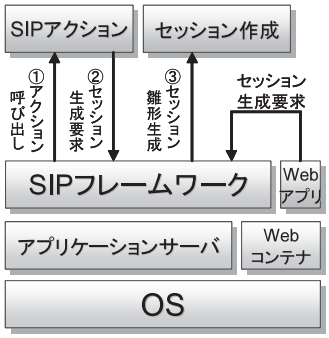

図 2 SIP フレームワークのアーキテクチャ
Fig. 2 The architecture of SIP framework.

テクチャにおいては，コントローラ（C）でメッセージを一括で管理し，そのメッセージの属性値等によって処理をモデル（M），ビュー（V）に振り分ける．このとき，モデル内にビジネスロジックを記述し，ビューに画面デザイン等のビュー層に関する処理を記述することで各処理を分離する．

HTTP では，Apache Struts [4] が MVC の代表的な例であるが，コントローラがモデルとして定義されるアクションを呼び出し，アクション内に記述されるビジネスロジックが呼び出すべき JSP ファイルをビューとして指定する．これにより，モデル内における画面出力項目を適切なビュー内で表示できるようになっている．Struts では，ビューは画面イメージを表すという考え方が一般的だが，モデルにおける処理結果に応じた遷移先画面ととらえることもできる．SIP フレームワークでも，この「遷移先」という考え方に基づき，ビュー層を別セッションへの遷移として定義する．

SIP フレームワークも Apache Struts と同様の構成となっており，SIP アプリケーションである SIP フレームワークがコントローラの役割を果たしてアクションを呼び出す．SIP においては画面に相当するレイヤが存在しないが，SIP フレームワークでは HTTP における遷移の概念をビューとして適用する．SIP における遷移とは，3PCC や B2BUA（Back To Back User Agent）等の SIP アプリケーションセッションの生成であり，アクション内でこれらの遷移が指定されると，生成された SIP アプリケーションセッションが開始されてその後の処理を継続する．

4.4 SIP フレームワークの動作

SIP フレームワークでは，SIP アプリケーションセッションを次々に遷移するようなアプリケーションを簡単に構築することが可能となっている．

処理の流れは，まずアプリケーションサーバが SIP メッセージを受信すると，SIP Servlet の仕様に基づき sip.xml 定義ファイルを参照して呼び出すべき SIP アプリケーションを決定する．SIP フレームワークを利用する場合，この SIP アプリケーションはつねに SIP フレームワークになる．次に，SIP フレームワークは SIP リクエストのメソッドの種類に応じた処理を実行し，sipframework.xml 定義ファイルを参照し呼び出すべきアクションを決定する（図 2 の ①）．たとえば SIP におけるログイン処理を行うためのメソッド REGISTER の場合には，ログインするユーザを特定，認証するための認証アクションと，そのユーザ情報をロケーションサーバに登録するためのロケーションサーバ操作アクションが呼び出される．また，呼び出される Java メソッドの返却値（図 2 の ②）に基づき，3PCC や B2BUA 等の SIP アプリケーションセッションを生成して遷移を行う（図 2 の ③）．遷移した場合，それ以降の SIP フレームワークの処理を破棄し，遷移先の SIP アプリケーション上の動作に切り替わる．

4.5 Web アプリケーションとの連携

3PCC の場合，最も多い利用シーンは Web アプリケーションにおけるクリックをトリガとして二者間のセッションを確立するようなものだろう．このような状況を想定し，SIP フレームワークでは次の 2 種類の連携方法を用意している．

- HTTP Servlet から呼び出されるライブラリ形式
- JSP カスタムタグ形式

両者は同等の機能を提供するが，ライブラリ形式では SIP アプリケーションの処理結果を基に画面の出力を変化させることができる等，柔軟性に優れる．一方カスタムタグ形式では，Web アプリケーション開発で一般的な JSP を利用していれば，きわめて簡単に SIP アプリケーションへの遷移を行うことができるという特徴がある．

4.6 sipframework.xml 定義ファイルについて

SIP フレームワークでは，フレームワークとアクション群の間の静的な依存関係を断ち，外部から依存関係を注入できるように，XML で記述された定義ファイルを用いて，呼び出すべきアクションクラスを指定できる．このような外部からの依存関係の注入は DI（Dependency Injection）と呼ばれるが，Struts だけでなく，Spring Framework [8] や Seasar [9] 等で採用されている手法である．

この定義ファイルには，以下のような情報を記述することが可能である．

- アクションを呼び出すための条件
- アクションに対応する Java クラス名（フレームワークごとに定義する）
- 遷移先の SIP アプリケーションセッション種別
- 生成した SIP アプリケーションの振舞いを記述するクラス名（遷移の種別ごとに定義する）

4.7 制 限 事 項

SIP フレームワークでは，受信した SIP や HTTP のリクエストをトリガとしてアプリケーションが動作を開始する．また，クライアント–サーバモデルのため一定の時間内にリクエストの中継やレスポンスの返却を完了することが前提となっている．このため，タイマ等他のトリガで動作したり，SIMPLE 時に多数のエージェントに対して NOTIFY リクエストを送信したりするようなアプリケーションには適さない．

5. アーキテクチャ詳細

5.1 静的構成

SIP フレームワークは，**表 1** に示す 6 つのコアフレームワーク群を中心として，開発者が開発を行うためのインタフェース群，ライブラリ群から構成される．これらのコアフレームワーク群のうち，レジストラ，プロキシ，およびプレゼンスフレームワークは，対応する各 SIP メソッドにおけるデフォルトの処理を行いながら，ビジネスロジックに関する処理についてはアクションに委譲する．

アクションとしては**表 2** に示すとおり 4 種類が用意される．これらは，SIP フレームワークの一部として提供されるインタフェースを開発者が実装することで，それぞれ表 2 の呼び出し元に示す特定のコアフレームワークから呼び出される．

アクションから遷移が発生する場合には，遷移後の SIP アプリケーションセッションの属性を決めるためのビジネスロジック Java クラスであるフォワードクラスを，SIP フレームワークが呼び出す．生成される SIP アプリケーションセッションの種類に応じて，**表 3** に示すような 3 種類のフォワードクラスが用意される．各フォワードクラスは，SIP フレームワークが規定するインタフェースに従い，各アクションと同様に SIP アプリケーションの開発者が実装する．そのほか，Web システムとの連携を行うためのライブラリ，JSP 群とその他のヘルパクラス等，および 4.6 節で説明した定義ファイルが存在する．

以下の各節において，SIP フレームワークの動作例として，プロキシフレームワークの詳細を説明する．

表 1 コアフレームワーク種別
Table 1 List of core frameworks.

フレームワーク種別	説明
レジストラ FW (Framework)	REGISTER メソッドの処理を行う．
プロキシ FW	INVITE，CANCEL，BYE メソッドの処理を行う．
プレゼンス FW	SUBSCRIBE，MESSAGE 等 SIMPLE 関連の処理を行う．
サードパーティ FW	セッション生成後の振り分け処理を行う．
タイマ FW	タイマ発火後の処理の振り分けを行う．
認証 FW	SIP 処理全般に関わるダイジェスト認証による認証処理を行う．

表 2 アクション種別
Table 2 List of actions.

アクション名	呼び出し元
認証アクション	認証 FW
レジスタアクション	レジストラ FW，プロキシ FW，プレゼンス FW
プロキシアクション	プロキシ FW，サードパーティ FW，タイマ FW
SIMPLE アクション	プレゼンス FW

5.2 プロキシにおける動作

プロキシとは，ユーザエージェントクライアント（UAC，発呼元）から受信する SIP リクエストからユーザエージェントサーバ（UAS，発呼先）を探し出し，SIP リクエストを転送する，電話における交換機と同様の位置付けのサービスである．SIP フレームワークでは，プロキシフレームワークがこの処理を担当する．

図 3 はプロキシフレームワークを用いた際のシーケンス図であり，UAC が UAS に発呼した際の通信の流れを示している．プロキシフレームワークにおいては，プロキシに必要な SIP メッセージごとの処理を SIP フレームワークが行い（図 3 の ① 部分），ビジネスロジック部分のみを開発者がアクションとして記述する（図 3 の ② 部分）．開発者は，SIP メッセージを意識する必要がなく，認証，UAS の検索，通話セッションの確立・切断後の処理等を意識すればよいため，複雑性の排除が可能となる．

5.3 SIP アプリケーションセッションの生成

新規に SIP アプリケーションセッションを生成する際には，アクションにおけるプログラムにおいて，4.4 節で述べたように遷移先を返却値として設定すればよい．たとえば 5.2 節のプロキシの例では，プロキシサービスの呼び出し時や通話セッション確立時等に遷移先を決定することができる．SIP フレームワークは

表 3 遷移種別
Table 3 List of forwards.

遷移種別	機能
フォワード B2BUA	B2BUA の生成（任意の SIP リクエスト）
フォワード 3PCC	3PCC の生成（サーバ以外の二者間のセッションの生成）
フォワードレスポンス	任意の SIP レスポンスの生成

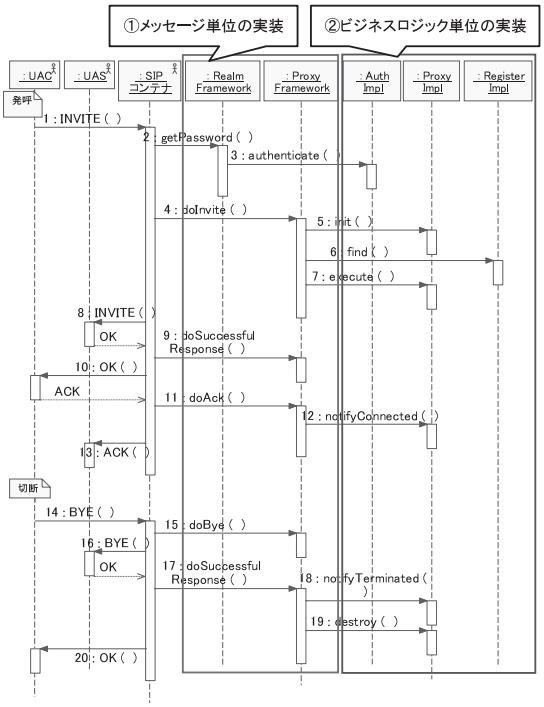

図 3 プロキシフレームワークの動作シーケンス
Fig. 3 A Sequence for proxy framework.

sipframework.xml ファイルを参照し，生成する SIP アプリケーションセッションが B2BUA なのか 3PCC なのかという SIP アプリケーションセッション種別と，その際に呼び出すべきフォワードクラス名を取得しそのインスタンスを作成する．たとえば 3PCC の場合，SIP フレームワークは SIP アプリケーションセッションや SIP リクエストのインスタンスを作成し，SIP リクエスト送信先等の 3PCC の属性を決定するため，上述のフォワードクラスのメソッドを呼び出す．この SIP アプリケーションセッションが 4.1 節 (2) で述べた雛形として機能し，さらにこの雛形がビジネスロジックを呼び出す．ビジネスロジック内にはセッションの生成や維持管理に関するコードを記述する必要はないため，SIP アプリケーションセッションの生成が容易となる．また，必要なビジネスロジックを適宜呼び出すアーキテクチャとすることで，柔軟な SIP アプリケーション開発を可能とする．

このように，3PCC や B2BUA 等の SIP アプリケーションセッションを容易かつ柔軟に生成することができ，開発者に対するセッション管理の複雑性を隠蔽することができる．

5.4 Web アプリケーションと SIP との連携

Web アプリケーションに対しては，4.5 節で示した 2 種類の連携方法を用意したが，簡便な方法として JSP カスタムタグによる SIP アプリケーションセッション生成について，動作を説明する．開発者は，このカスタムタグの属性として遷移先の SIP アプリケーションセッションの名称を JSP ファイル中に記述すると，JSP エンジンがこのカスタムタグを評価する際に，5.3 節と同様に SIP フレームワークが必要な SIP アプリケーションセッションを生成し，対応するビジネスロジック Java クラスを呼び出す．このとき，SIP フレームワークは HTTP リクエストの情報をこの Java クラスに渡すことができるため，HTTP リクエスト中の接続先に SIP リクエストを送信するような処理が簡単に行える．

これにより，3.2 節 (3) の課題である Web アプリケーション連携時の複雑さを排除している．

6. ケーススタディと評価

2 種類のケーススタディに基づき，SIP フレームワークに関して，① 複雑さの排除と生産性の向上の観点による開発効率化の効果，② 提供機能の実用性・十分性と保守の容易性の 2 点における評価を実施した．

6.1 ワークフロー承認システム

SIP フレームワークを適用したアプリケーションとして，**図 4** に示す Web による音声承認システムを開発した．このシステムでは，申請者が Web 画面から申請を行うと（図 4 の ①），承認者および VoiceXML サーバ間の通話セッションが確立される（図 4 の ②）．VoiceXML サーバが申請内容を音声合成により読み上げ，承認者が音声で承認結果を話すと，VoiceXML サーバが音声認識を行い，承認結果をアプリケーションサーバに書き込む（図 4 の ③）ことで，承認処理が完結する．さらにここでは，通話セッションの切断に基づき承認結果を申請者に通知するための SIP アプリケーションセッションへの遷移（図 4 の ④）を行う．

6.1.1 複雑さの排除

図 5 はこのアプリケーション上で使用される JSP ファイルであり，10 行目のカスタムタグによって遷移先が指定される．**図 6** は 3PCC フォワードクラスのソースコードであり，確立する二者の SIP URI や SIP リクエストのヘッダ情報のみを定義するプログラムとなっている．

このように，JSP ファイル上で SIP アプリケーションセッションの生成を要求し，これに応じて SIP フレームワークが生成した SIP アプリケーションセッションや SIP リクエストに対する属性をフォワードクラスに記述するだけで，3PCC を呼び出すことができ

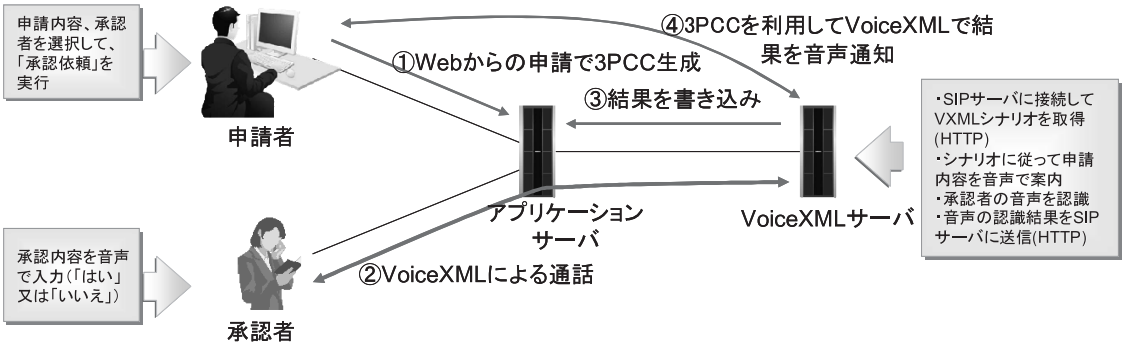

図 4 ワークフロー承認システムの構成概要
Fig. 4 Workflow authorization system.

```
<%@ page language="java" contentType="text/html; charset=Shift_JIS"
    pageEncoding="Shift_JIS"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">

<%@ taglib uri="/WEB-INF/struts-bean.tld" prefix="bean" %>
<%@ taglib uri="/WEB-INF/struts-html.tld" prefix="html" %>
<%@ taglib uri="/WEB-INF/sipframework.tld" prefix= "sip" %>

<html:html><head>
<sip:forward name="APPLY"/>
<title>申請受付完了</title></head><body>
<P>以下の内容で申請を受け付けました。</P>
リクエストID: <bean:write name="app" property="requestId" scope="request"/> <BR>
出張先: <bean:write name="app"  property="tripTo" scope="request"/> <BR>
金額: <bean:write name="app"  property="price" scope="request"/> <BR>
出張者: <bean:write name="app"  property="applier" scope="request"/>
(  <bean:write name="app"  property="applierAddress" scope="request"/> ) <BR>
承認者: <bean:write name="app"  property="certifierAddress" scope="request"/> <BR>
```

図 5 SIP と連携する JSP ソース例
Fig. 5 Source code for a JSP calling SIP.

```
public class ApplyForward2 extends GenericForward3pcc {
  public int requestId = 0;

  private ResourceBundle rb = ResourceBundle.getBundle("demo");

  public String getFrom(SipApplicationSession application,
      SipServletRequest request) {
    String sipAddressVxml = this.rb.getString("vxml.sipaddr");
    return sipAddressVxml; // VXMLサーバのSIPアドレス
  }

  public String getTo(SipApplicationSession application,
      SipServletRequest request) {
    HttpServletRequest httpReq = getHttpServletRequest(application);
    String key = httpReq.getParameter("certifierAddress"); // 承認者
    return this.rb.getString(key); // 承認者
  }

  public SipServletRequest getFirstRequest(
      SipApplicationSession application, SipServletRequest request) {
    HttpServletRequest httpReq = getHttpServletRequest(application);
    ApplicationForm af = (ApplicationForm) httpReq.getAttribute("app");
    int requestId = af.getRequestId();
    application.setAttribute("applicationForm", af);

    String ipSip = this.rb.getString("sip.ipaddr");
    request.setHeader("X-VXML-Document-URI", "http://" + ipSip
        + ":19000/sipframework/getScript.do?requestId=" + requestId);
    return request;
  }

  public SipServletRequest getSecondRequest(
      SipApplicationSession application,SipServletRequest request) {
    return super.getSecondRequest(application, request);
  }
}
```

図 6 フォワードクラスのソース例
Fig. 6 Source code for a forwarding class.

る．ここで重要なことは，開発者は SIP アプリケーションセッションの生成や UAC，UAS からの SIP レスポンスに関するコードを記述する必要がまったくないことである．これらの処理はすべて SIP フレームワークが行うことで，開発者が SIP のプロトコルレベルのプログラミングを行う必要がなく，「どこに接続するのか」「そのときにリクエストにどのような情報を付加するのか」というビジネスロジックの処理に専念することができることを示している．

図 4 において ④ の，通話セッションの切断をトリガとした SIP アプリケーションセッションへの遷移は，SIP Servlet のみを利用して開発する場合には，ある特定の SIP レスポンスをトリガとしなければならず，より複雑な SIP アプリケーションではその文脈情報の解析も複雑化し，大きなコスト・工数を要する．しかし，SIP フレームワークを利用すると通話セッションが終了したタイミングで呼び出されるアクションの返却値のみで，簡単に SIP アプリケーションセッション生成の制御を行うことができる．このため，通信相手の増加による SIP 特有の複雑さを排除し，開発を単純化，効率化することができる．

6.1.2 生産性の向上

本アプリケーションでは比較のために，SIP フレームワークを利用せずに SIP Servlet のみを利用したプログラミングもあわせて実施した．SIP Servlet では UAC や UAS から受信する SIP メッセージごとに処理を記述しなければならず，そのための呼び出しごとの処理のために多くの行数を割く必要があるため，SIP アプリケーション関連部分のコード量は 178 行である．一方 SIP フレームワークでは，どのような SIP アプリケーションセッションを作成するかにコードが集中しており，コード量は JSP や定義ファイルを含めても 56 行となり，33%に削減されている．コード量においても SIP フレームワークは大きな効果を期待できる．

6.2 自動再接続サービス

CSBNA（Call Schedule on Busy or No Answer）は電話をした相手が話中だった場合に，その相手が通話可能な状態になった際に自動的に呼接続するサービスである．この SIP アプリケーションでは，SIP フレームワークにより話中を示すエラーレスポンスに対

応するアクションが呼び出され，アクション内では受信したエラーレスポンスを判別し，Web を使って発呼予約を簡単に行えるようにするための SIP レスポンスを発呼者に返却する．接続相手の通話状態（プレゼンス）が話中から待ち受けに変化すると，NOTIFY メソッドが SIP サーバに通知されるので，SIP サーバがこの NOTIFY メソッドを受信した時点で 3PCC が生成され，相手との通話が可能となる．

6.2.1 機能の実用性・十分性

SIP フレームワークでは，エラーレスポンスや NOTIFY メソッドをハンドリングし，対応するアクションを呼び出すことができるため，開発者はビジネスロジック部分の開発に専念することができる．本アプリケーションでは，Web を介した文書送信処理を行うために独自形式の IM 送信を行う設計とした．SIP フレームワークでは，原則として SIP メッセージの詳細に関する考慮は不要で，デフォルトの値を設定するが，必要に応じて送信する各 SIP メッセージで SIP ヘッダ，SIP ボディを含めたメッセージの詳細を柔軟に定義することもできるため，Web を介した文書の送信のような複雑な処理も問題なく記述することができる．

このように SIP フレームワークは，SIP メッセージを直接操作することもできるよう十分な拡張性を有する設計となっているため，大規模な SIP アプリケーションの開発時においても有効である．他のアプリケーションにおいても，これまでのケーススタディで見てきたような HTTP または SIP のリクエストをトリガとしている事例であれば有効性は高い．

6.2.2 保守の容易性

この SIP アプリケーションでは，後から文書送信処理を行う機能追加を行う等，インクリメンタルな開発手法を採用した．このような場合，SIP Servlet のみを利用して開発を行うと，SIP メッセージ受信時の場合分けが数多く発生し，保守性を損なうことが多い．しかし SIP フレームワークを適用することで，ビジネスロジックに対しての機能追加が行えるため，機能追加をすべきビジネスロジックの場所を容易に特定でき，開発効率は向上した．

7. 関連する研究・活動

7.1 JAIN SIP Servlet

SIP Servlet は SIP フレームワークと同様に SIP プロトコルを取り扱うフレームワーク技術を採用している．つまり，両者はともに定義ファイルに沿って SIP リクエストを Java プログラムへ振り分ける．レイヤとして見ると，SIP フレームワークは SIP Servlet の上位レイヤで動作する．SIP Servlet が SIP プロトコル寄りで SIP メッセージと Servlet の呼び出しが 1 対 1 で対応するのに対し，SIP フレームワークは複数の SIP メッセージに対して 1 個のアクションが呼び出される点が異なっている．

7.2 Apache Struts

Web システムにおいて，MVC を実装する代表的なアプリケーションフレームワークである．SIP フレームワークはこの Struts の概念を基に SIP 向けに実装しなおしたものである．SIP フレームワークは，ビジネス用途ごとに複数のアクションを定義している点や複数の SIP セッションを一括して扱う点，またビュー層を SIP アプリケーションセッション間の遷移としてとらえ，別の SIP アプリケーションセッションを容易に呼び出せる点で，Struts とは異なっている．

なお，Struts と SIP フレームワークとは排他関係にはないため，同時に利用することが可能である．

7.3 SIPHIA

SIPHIA [7] は，Web アプリケーションと SIP サーバを分離し，Web アプリケーション側から簡単なプログラムで 3PCC 等の SIP ライブラリを呼び出すことができる実行・開発環境である．SIPHIA では，呼び出しの容易性と，複数の環境・言語からの呼び出しを可能とする点を特徴としているが，その一方で SIP 部分の動作は固定となり，受信したリクエストやレスポンスを基に動作を変化させるようなカスタマイズが困難なために，複雑なアプリケーションの構築には向かない．SIP フレームワークは，SIP アプリケーションの内部にもカスタマイズ可能なホットスポットを用意し，より複雑なアプリケーションを容易に構築できるメリットがある．

7.4 SIP-HTTP 連携アーキテクチャ

Web システムと連携するための開発手法として SIP-HTTP 連携アーキテクチャ[2] が提案されており，ここでは Web と SIP のコンポーネント間でセッション情報を共有するためのフレームワークと，Web と SIP の開発を分離し効果的な分業を実現するアーキテクチャが示されている．SIP フレームワークでは，SIP と Web を分離することよりも，従来 Web のシステムのみを開発してきた開発者が簡単に SIP 機能を付加できることを主眼とし，Web のシステムで一般的でかつ開発者が使い慣れている MVC アーキテクチャを SIP に導入することを最大の特徴とする．

8. ま と め

本稿では，SIP プロトコルを導入するビジネスアプリケーション開発の効率化をターゲットとして，プロトコルレベルの開発からビジネスロジックレベルの開発を可能とする SIP フレームワークを提案した．SIP アプリケーション開発における課題である，① 通信相手の増加による複雑性，② セッション管理の複雑性，③ Web アプリケーションとの連携の複雑性を解決した．本技術により，開発者はアプリケーション内でどのようにパケットが送受信されるかを意識することなくビジネスロジック部分の開発に注力することが可能となる．2 つのケーススタディで約 67%の開発コード量の削減，および実用アプリケーションの機能要求に対する十分性の評価を得た．

今後としては，より大規模なシステムにおける本技術の適用検証を行う．また，さらなる開発効率の向上と汎用化のために，モデルをベースとした開発の自動化の検討を予定している．

参 考 文 献

1) RFC 3261, Session Initiation Protocol.
2) 相原 論：J2EE での SIP と HTTP 連携システムのアーキテクチャ，NS2004-24 (2004).
3) JAIN SIP Servlet. http://www.jcp.org/en/jsr/detail?id=116
4) Apache Software Foundation, Struts. http://struts.apache.org/
5) Internet Engineering Task Force: SIP WG, Transcoding Services Invocation in the Session Initiation Protocol (SIP) Using Third Party Call Control (3pcc).
6) Fujitsu Limited: Interstage SIPnet Application Container. http://interstage.fujitsu.com/jp/sipnet/
7) NEC: SIPHIA. http://www.sw.nec.co.jp/netsoft/SIPHIA/
8) Spring Framework. http://www.springframework.org/
9) Seasar Foundation: Seasar. http://www.seasar.org/

(平成 18 年 12 月 7 日受付)
(平成 19 年 3 月 1 日採録)

小高 敏裕

1993 年早稲田大学理工学部物理学科卒業，1995 年同理工学研究科修士課程修了．同年富士通株式会社に入社し，主に課金決済系システムや Web を中心としたアプリケーションの開発・構築に従事．2005 年より株式会社富士通研究所にて，HTTP，SIP 等のネットワークプロトコル上のアプリケーション開発のためのソフトウェアアーキテクチャや，テスティングフレームワーク等の研究開発を行う．

松塚 貴英（正会員）

1971 年生．1994 年東京工業大学電気電子工学科卒業，1996 年東京工業大学情報理工学研究科計算工学専攻修士課程修了．同年富士通株式会社に入社．現在，株式会社富士通研究所 IT コア研究所ソフトウェアイノベーション研究部所属．分散企業システム，Web アプリケーションフレームワーク，ソフトウェアアーキテクチャ等の研究，開発に従事する．2001〜2002 年，米 Carnegie Mellon University 客員研究員．

野村 佳秀（正会員）

1973 年生．1998 年青山学院大学大学院理工学研究科経営工学専攻博士前期課程修了．同年株式会社富士通研究所入社．以降，Web サービスセキュリティ技術，企業間商取引基盤，業務プロセス可視化技術等の研究開発に従事．修士（工学）．

村上 雅彦（正会員）

1992 年京都大学大学院工学研究科電気工学第二専攻修士課程修了．同年，株式会社富士通研究所入社．グループウェアの研究開発を経て，SIP をベースとしたコミュニケーションサービス等の研究開発に従事．

山本里枝子（正会員）

1983 年早稲田大学理工学部電子通信学科卒業．同年株式会社富士通研究所入社．現在，富士通研究所ソフトウェア&ソリューション研究所主席研究員，富士通株式会社コーポレート IT 推進本部主席部長．ソフトウェア工学の研究開発に従事．コンポーネント，ソフトウェアパターン，ソフトウェアテスティング，ビジネスプロセスモデリング，サービス指向開発等の技術開発と実践を担当．情報処理学会山下記念研究賞受賞．東京農工大学非常勤講師，早稲田大学非常勤講師他．IEEE 会員．