SOA に基づくシステムのための共通プラットフォームの妥当性の検証

M2011MM076 若尾明日香
指導教員：野呂昌満

1 はじめに

Service-Oriented Architecture[1](以下, SOA) はビジネス環境の変化に対して柔軟に変更可能なシステムの開発技術である．エンタープライズ系システムに要求される非機能要求のひとつとしてビジネス環境の変化に対応する柔軟性が挙げられ，近年では多くのエンタープライズ系システムが SOA に基づいて開発されている．SOA に基づくシステムの開発では，各ベンダから提供されているミドルウェアを利用する．ミドルウェアを利用したシステムのアーキテクチャはミドルウェアに依存した構造となりがちであり，非機能特性を実現することが困難である．この問題を解決するために，SOA に基づくシステムのための共通プラットフォームとして，アプリケーションプラットフォームが提案されている．共通プラットフォームは，ミドルウェアをラッピングすることで，アプリケーションのアーキテクチャをミドルウェア独立にすることを目的としている．共通プラットフォームでは，SOA に基づくシステムの横断的関心事を分離したアスペクト指向アーキテクチャと，可変性を表したフィーチャモデルが定義されている．

本研究の目的は，SOA に基づくシステムのための共通プラットフォームの他ドメインにおける妥当性の検証である．他ドメインにおけるシステムを事例としてケーススタディをおこなうことで，共通プラットフォームの妥当性を考察する．

定義に用いた事例とは異なるドメインにおける共通プラットフォームのアーキテクチャを考察することで，共通プラットフォームとしての一般性について考察する．事例として，現在 OJL(On the Job Learning) として開発をおこなっている医療情報システムを用いる．医療情報システムは，共通プラットフォームの主な可変要因である OS，ミドルウェア，言語について，共通プラットフォームの定義に用いた事例における選択とは異なるので，検証に用いる事例として妥当であると考える．これら医療情報システムへの要求を整理し，要求を考慮しながら共通プラットフォームの可変性の選択をおこなう．選択した可変性を考慮した医療情報システムにおける共通プラットフォーム上において，可変性の選択が共通プラットフォームに与える影響について分析をおこなうことで，共通プラットフォームの一般性について考察する．

結果として，可変性による影響がアーキテクチャ上で吸収されていることを確認し，共通プラットフォームのアーキテクチャの妥当性を確認することができた．

2 SOA に基づくシステムのための共通プラットフォーム

SOA に基づくシステムは，各ベンダが提供するミドルウェアを利用して開発される．各ミドルウェアには想定された構造が存在することから，SOA に基づくシステムのアーキテクチャはミドルウェア依存となりがちである．また，ミドルウェア依存なアーキテクチャとなることで，非機能特性を考慮したアーキテクチャの実現が困難となる．これを解決するために，SOA に基づくシステムのための共通プラットフォームであるアプリケーションプラットフォームが提案されている．アプリケーションプラットフォームは，ミドルウェアをラッピングし，ミドルウェア独立なアプリケーションのアーキテクチャを実現する．

アプリケーションプラットフォームのアーキテクチャは，SOA に基づくシステムにおける横断的関心事を分離したアスペクト指向アーキテクチャである．また，異なるドメイン間でのアプリケーションプラットフォームの可変性が整理されており，可変性を表したフィーチャモデルが定義されている．

各ミドルウェアで提供されている API の構文や実現方法には差異がある．アプリケーションプラットフォームでは，ミドルウェアごとに特化したアダプタを定義することで，API の構文や実現方法の差異を，アダプタ内の実装で吸収することができる．アーキテクチャ上の変動部分にインタフェースを定義し，可変部品としてインタフェースを実装する各ミドルウェアに特化したアダプタを定義する．これにより，変動部分のコンポーネントの利用側は，異なるミドルウェアに対しても同一に扱うことが可能となる．

3 医療情報システムの概要

本研究では共通プラットフォームの妥当性を確認するための事例として医療情報システムを用いる．この医療情報システムは，診療所のスタッフがおこなう業務を支援することを目的としたシステムである．本システムの実現手段としてハードウェア構成とソフトウェア構成が指定されている．

3.1 医療情報システムのハードウェア構成

医療情報システムを実現するハードウェア構成を図 1 に示す．オンライン請求 Network はレセプトをオンラインで提出するときに用いるネットワークである．外部システムからの悪影響を遮断するために，オンライン請求専用の環境を用意する．本研究におけるアプリケーションプラットフォームは，院内 LAN の範囲におけるサービス連携を実現する．患者の個人情報の漏洩を防ぐなどのセ

キュリティを実現するために，院内にアプリケーションサーバを配置し，院内専用のネットワーク環境上でシステムを稼働させる．

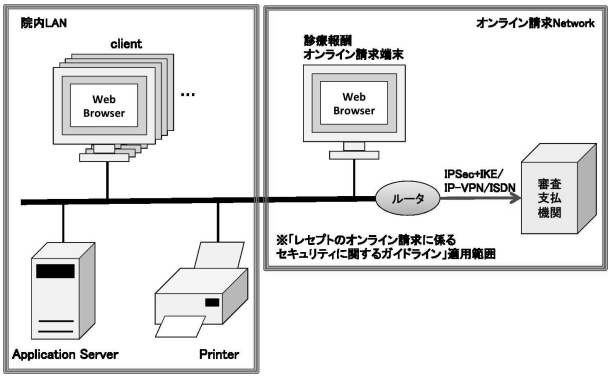

図 1 医療情報システムのハードウェア構成

3.2 医療情報システムのソフトウェア構成

本システムを実現するソフトウェア構成を図 2 に示す．クライアント側のブラウザとして IE，Firefox，Safari が指定されており，これらのブラウザ上で.NET Framework[2] で提供されている WebApplicationFramework である Silverlight が動作する．サーバ側の OS は WindowsServer であり，サーバソフトとして IIS を用いる．サーバ上では.NET Framework が動作する．

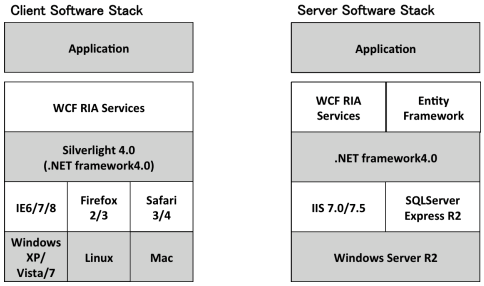

図 2 医療情報システムのソフトウェア構成

4 医療情報システムにおける共通プラットフォーム

4.1 共通プラットフォームの可変性の選択

3 章にて説明した医療情報システムへの要求を考慮し，共通プラットフォームにおいて定義されている可変性を選択し，フィーチャモデルの枝刈りをおこなう．フィーチャモデルの枝刈りをおこなった結果を図 3 に示す．

各可変性の選択理由を述べる．アプリケーションプラットフォームは院内 LAN の範囲においての実現であることから，アプリケーションプラットフォームの機能のうちの一つである Routing は必要ないと判断した．OS と言語については開発に対する要求から決定した．開発に用いるミドルウェア (WS Framework，DBMS，OR Mapper，シリアル化ライブラリ，Logger) は，3.2 章で示したソフトウェア構成に基づき，.NET Framework において提供さ

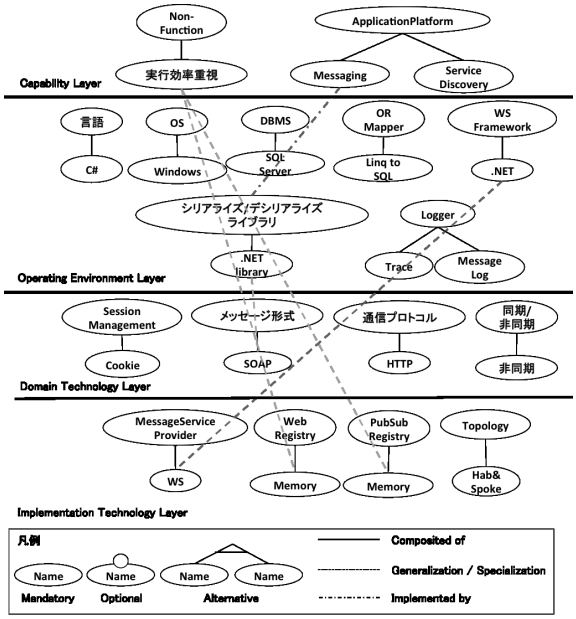

図 3 医療情報システムにおけるアプリケーションプラットフォームのフィーチャモデル

れているフレームワークやライブラリ，機能を選択した．メッセージングの実現については，ミドルウェアで提供されるメッセージング方法からメッセージ形式を SOAP，通信プロトコルを HTTP とする，また，クライアントがプロバイダの処理が完了するまで待つことによる資源効率性の低下を防ぐために非同期通信とする．院内で使うことから，アクセス頻度は低く，スレッド生成による資源効率への影響はないと考える．セッション管理の実現方法，Registry の実現方法は，時間効率性を考慮しそれぞれ選択をおこなった．Topology については，院内で使うことからアクセス頻度は低く時間効率性や資源効率性への悪影響はないと考え，ハブ型を選択した．．また，コンポーネントの分割については，C#ではインライン展開がサポートされておらず，実現には手書きでメソッドを記述する必要があり，保守性や，信頼性を損ねる可能性があると考え，オブジェクト指向による分割を選択した．アプリケーションプラットフォームのコンポーネントの配置については，院内にサーバを設置できる数に限りがあると考え，Web サービスと同一のアプリケーションサーバ上に配置することとする．MessageServiceProvider は Server 上に配置され，クライアントのマシン上で稼働する Silverlight アプリケーション上からリモートアクセスされることから，WS 化をおこなう．

4.2 医療情報における共通プラットフォームのアーキテクチャ

選択した可変性を考慮した医療情報システムにおけるアプリケーションプラットフォームのアーキテクチャを示す．図 4 に，医療情報システムにおけるアプリケーションプラットフォームの静的構造を，Views and Beyond の Module Viewrtype で示す．色のついたモジュールは，医療情報システムのドメインに特化したコンポーネントであるこ

とを表している．MessageServiceProvider インタフェースの実装は，WCF をラッピングするコンポーネントにより実現する．MessageComponent は，.NET Framework で提供されているスタブとしての役割を果たすチャネルにより実現する．ServiceConsumer は Silverlight，PrimitiveService は WCF により実現することから，それぞれのミドルウェアをラッピングするコンポーネントを用意し，MessageComponent インタフェースを実装する．

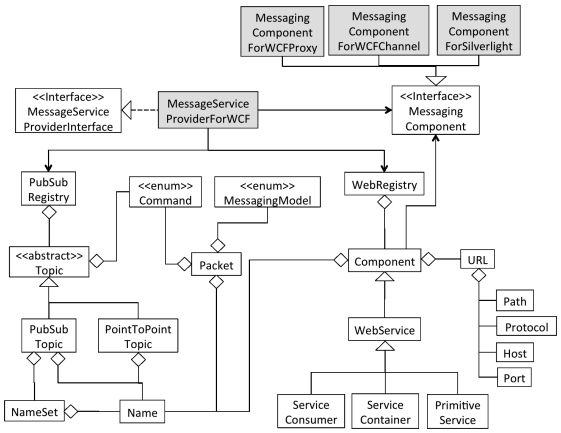

図 4 医療情報システムにおけるアプリケーションプラットフォームの静的構造

図 5 に，アプリケーションプラットフォームを実現する各コンポーネントの配置について，Views and Beyond の Allocation Viewrtype の Deployment style で示す．Silverlight により実現された WebApplication はクライアントの Web ブラウザ上で稼働する．Web サービスとアプリケーションプラットフォームのコンポーネントは ApplicationServer 上に配置され，クライアントの WebApplication は MessageServiceProvider にリモートアクセスをおこなう．

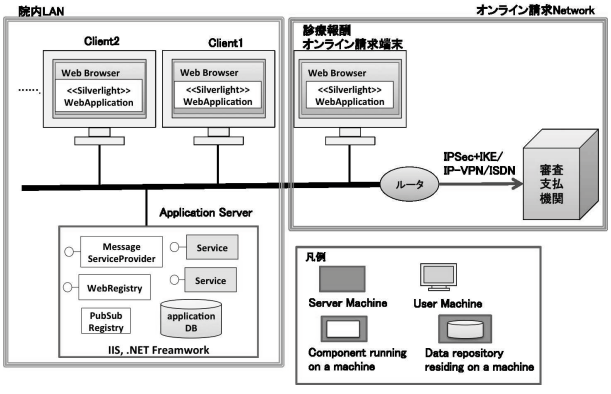

図 5 医療情報システムにおけるアプリケーションプラットフォームの配置

5 考察

5.1 共通プラットフォームの妥当性の考察

Web サービス，Web アプリケーション，アプリケーションプラットフォームを実現するコンポーネントはミドルウェアを用いて実現する．ミドルウェアによって前提とされているメッセージングの実現方法や用いるスタブが異なる，本研究では，ミドルウェアによって前提とされているメッセージングの実現方法や用いるスタブの違いを，アプリケーションプラットフォームのアーキテクチャ上でどのように吸収しているかについて分析することで，アプリケーションプラットフォームの妥当性を考察する．

5.1.1 ミドルウェアが前提とするメッセージングの実現方法

医療情報システムにおいて，アプリケーションプラットフォームのコンポーネントと，Web サービスを実現するミドルウェアには WCF，クライアント側の Web アプリケーションを実現するミドルウェアには Silverlight を用いる．本節では，WCF と Siverlight が前提とするメッセージングの実現方法を説明する．

WCF が提供するメッセージングの仕組みについて説明する．Web サービスは，ABC(Address，Binding，Contract) を定義したエンドポイントを公開する．Address はサービスを配置する URL，Binding はサービスへの通信方式，Contract は提供するサービスのインタフェースを意味する．Web サービスへのアクセスは，ABC を指定しスタブとなる Channel を作成し，Channel を通してメッセージを送信することで実現することができる．Channel の作成方法は，.NET の IDE である VisualStudio から自動生成された Proxy を用いて生成する方法と，ChannelFactory を用いて手動で ABC を指定し Channel オブジェクトを生成する方法の，2 通り存在する．WCF が提供するメッセージングの仕組みを図 6 に示す．

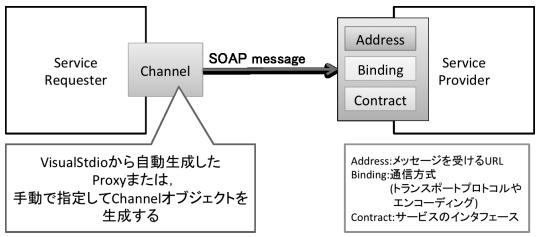

図 6 WCF が提供するメッセージングの仕組み

Silverlight により実現された Web アプリケーションへのメッセージングの仕組みについて説明する．Silverlight により実現されたアプリケーションは，クライアントの Web ブラウザ上で動作することを前提としており，URL を指定したアクセス方法は提供されていない．Silverlight により実現された Web アプリケーションへのメッセージングは，メッセージ送信元の情報を保持する OperationContext オブジェクトから CallbackChannel を生成し，この CallbackChannel を介すことでメッセージングをおこなう．CallbackChannel の生成時は，シリアル化やエンコーディングの指定をおこなうことはできない．また，Silverlight の Web アプリケーションにはあらかじめコールバック用のインタフェースを定義しておく必要がある．Silverlight が提供するメッセージングの仕組みを図 7 に示す．

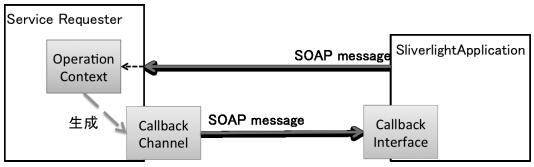

図 7 Silverlight が提供するメッセージングの仕組み

5.1.2 ミドルウェアの想定する構造を吸収する仕組みの分析

前節において説明したミドルウェアの提供するメッセージングの仕組みの違いは，アプリケーションプラットフォームのアーキテクチャ上ではアダプタにより吸収されている．

医療情報システムのアプリケーションプラットフォームにおいて，アダプタによりミドルウェアをラッピングしているのは，MessageServiceProvider の実現と MessageComponent の実現である．この 2 カ所について，ミドルウェアをどのようにラッピングしているかについて分析をおこなう．

MessageServiceProvider を実現する静的構造を図 8 に示す．MessageServiceProvider インタフェースを実装する MessageServiceProvider コンポーネントは WCF を用いて自作する．WCF を用いたことから，受け取るメッセージ形式は SOAP である．

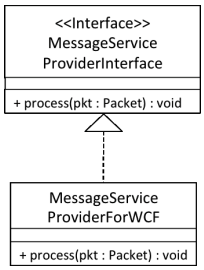

図 8 MessageServiceProvider を実現する静的構造

MessagingComponent を実現する静的構造を図 9 に示す．MessagingComponent インタフェースを実装する MessagingComponent として，MessagingComponentForWCFProxy，MessagingComponentForWCFChannel，MessagingComponentForSilverlight の 3 つを定義する．MessagingComponentForWCFProxy は，WCF により実現されたサービスへの VisualStudio により自動生成された Proxy を保持しており，この Proxy 用いてメッセージングをおこなう．MessagingComponentForWCFProxy は，WCF により実現されたサービスの URL を保持しており，Channel を生成することでメッセージングをおこなう．MessagingComponentForSilverlight は，Silverlight を用いて実現された Web アプリケーションの OperationContext を保持しており，OperationContext から CallbackChannel を生成することでメッセージングをおこなう．

図 4 において示した通り，WebRegisty は Component と Name を関連を保持している．WebRegistry は，Component の登録時に登録する Component に合わせて MessagingComponent を生成し，Component に保持させる．目的のコンポーネントへのメッセージングは，Component に保持される MessagingComponent を用いておこなう．

MessagingComponent インタフェースを実装する MessagingComponent という継承関係を用いることで，メッセージ送信側は Messaging 送信先の想定するメッセージング方法を意識せずに，メッセージングを実現することが可能となる．

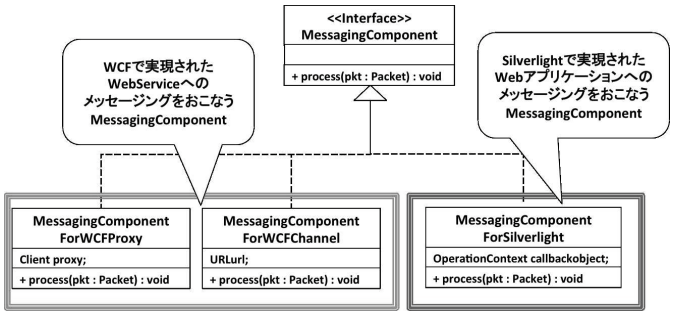

図 9 MessagingComponent の静的構造

5.2 アプローチの妥当性についての考察

本研究のアプローチの妥当性について考察をおこなう．本研究では，OJL として開発をおこなっている医療情報システムのドメインにおいて共通プラットフォームの一般性を考察した．医療情報システムでは共通プラットフォームの定義に用いた ATM 監視システムと実現に用いる OS やミドルウェアが異なり，共通プラットフォームにおける可変性の選択が異なる，本事例を用いることで，可変性のひとつであるシステムの実現に用いるミドルウェアの選択によって，共通プラットフォームのアーキテクチャへの影響はないことが確認できた．このことから，本研究に用いる事例として医療情報システムを用いたことが妥当であったと考える．

6 おわりに

本研究では，OJL として開発をおこなっている医療情報システムを用いて，提案されている SOA に基づく共通プラットフォームの妥当性の検証をおこなった．医療情報システムに求められるハードウェアとソフトウェアの構成要求から，共通プラットフォームで定義されている可変性を選択し，アーキテクチャ上でどのようにミドルウェアをラッピングしているか分析をおこなった．結果として，ミドルウェアの想定するメッセージングの実現方法の違いにもアーキテクチャを崩すことなく対応できたことから，共通プラットフォームが妥当であるといえる．

参考文献

[1] D. Georgakopoulos, and M. Papazoglou, *Service-Oriented Computing*, The MIT Press, 2009.

[2] Microsoft, "*.NET Framework*, " http://www.microsoft.com/ja-jp/net/, 2013.