# 4F-3 CORBA アプリケーションの Web サービス化における透過性の実現

日本アイ・ビー・エム(株) ソフトウェア開発研究所
森 和夫，渡邊 将一郎

要旨： 本論文では，SOA アプローチに従って CORBA アプリケーションを Web サービス化して J2EE システムに組み込む際の問題点について詳説し，その解決策として CORBA Proxy を提案する．CORBA Proxy は，内部的に CORBA アプリケーションをラップする一方，Web サービスの規約に沿ったインターフェースを提供する．筆者らは CORBA Proxy を生成する CORBA Proxy Code Generator ツールを開発した．ツールを使用して Web サービス化した CORBA アプリケーションを，実際のシステムである BPEL プロセスから実行した際のパフォーマンス測定結果を示し，本方式の有効性について論じる．

## 1. はじめに

CORBA は分散環境におけるオブジェクト通信の基盤技術で，プログラミング言語に依存しないオープンで成熟した技術であり，数多くのシステムで CORBA が利用されている．一方，近年のオンデマンド・システムでは，Web サービスとしてコンポーネント化された複数の既存アプリケーションをつなぎ合わせて，新たなアプリケーションを構築する SOA (Service Oriented Architecture)アプローチが用いられつつある．特に J2EE の世界では，Web サービスに関する標準化が加速度的に進んでおり，SOA 実装に J2EE が積極的に利用されている．

稼動中の CORBA アプリケーションも，Web サービス化することで，既存コードを変更することなくオンデマンド・システムに組み込むことが可能である．しかし，CORBA と Web サービスおよび Web サービスを実装するプログラミング言語，それぞれの仕様が規定するインターフェースの違いのために Web サービス化の作業が困難となることがある．これらの課題を解決するために筆者らは CORBA Proxy を考案し，これにより Web サービスから CORBA アプリケーションへの透過性を実現した．

## 2. Web サービス化における問題

### 2.1. J2EE と CORBA の仕様相違による問題点

IDL ファイルを入力として Web サービス化を行うとする．CORBA では struct 等を用いてユーザ独自のデータ型を宣言し，通信オブジェクトとして使用する．CORBA の struct データ型は Java ではクラスにマッピングされ，idlj を実行することで Java コードが生成される．この通信オブジェクトは WSDL 生成時には Web サービスの呼び出し元と呼び出し先の間で複合データ型(complex data type)にマッピングされる．ここで注目すべき箇所は複合データ型宣言の name 属性で，この名称が小文字で始まる場合があることである．この複合データ型は IDL で独自定義された struct データ型からマッピングされたものである．このようなデータ型名を小文字で始める作法は，C/C++言語等で実装される CORBA では珍しくはない．

他方，J2EE では JSR (Java Specification Request)として WSDL を含む Web サービスに関する様々な標準化が行われており，WSDL 文書の XML 定義名から Java クラス名へのマッピングの際に Java クラス名の先頭文字での英小文字使用を禁止している．この他，J2EE と CORBA の仕様相違により発生する問題として Java のメソッド名に相当する CORBA のオペレーション名がある．C/C++言語で実装される CORBA の場合，英大文字で始まる名称が多いのに対し，JSR では英小文字で始まる名称でなければならないという規約がある．

### 2.2. Java IDL の仕様による問題点

Java IDL を利用して CORBA アプリケーションを呼び出した際，ランタイム時に度々発生する例外として BAD_PARAM 例外がある．この例外は CORBA アプリケーションのオペレーションの引数に null 値または null ポインタが渡されることによって発生するもので，Java IDL の仕様で定められている．実際の CORBA アプリケーションでは，CORBA の複合データ型は，そのフィールドとしてさらに複合データ型を保持するようなネスト構造なっている場合が多い．こうした複雑な複合データ型を送信する際にランタイム時に発生する例外を防ぐために，Web サービス化した CORBA アプリケーションを呼び出すクライアント・プログラムにおいて，全てのフィールドに値を詰めておく必要があるが，これは開発者にとっては負担が大きく，開発コストの面からも問題点と言える．

### 2.3. EJB と CORBA の仕様相違による問題点

CORBA のオペレーション呼び出し時のパラメータ・タイプには in, out, inout の 3 種類がある．in はオペレーションを呼び出すクライアントからサーバへ一方向にのみ渡される引数で，EJB のメソッド呼び出しの引数と同等のものである．一方，out と inout は CORBA 特有の受け渡し方法で，out はサーバからクライアントへパラメータが渡され，inout はクライアントとサーバ間で双方向に渡される．

通常の Java プログラミング言語で実装する CORBA クライアントの場合には Holder クラスが機能するが，J2EE 環境の EJB ではこれは機能しない．EJB の仕様では，引数として受信したオブジェクトを EJB サーバ側で変更したとしても，その変更をクライアント側に引数として返すことはできない．したがって，J2EE 環境において out や inout タイプを引数として持つ CORBA オペレーションのインターフェースは，元のリターン値の他に，out と inout 型の引数もリターン値として返すように，インターフェースを修正する必要がある．

## 3. CORBA Proxy

これらの問題点を解決するため，筆者らは CORBA Proxy を提案する．CORBA Proxy はそれ自体が Web サービス化されるコンポーネントであり，J2EE の規約を遵守したインターフェースを提供する．つまり，英大文字で始まる名称のクラスと，英小文字で始まる名称のメソッドのインターフェースを提供し，その一方で CORBA クライアントとして動作することでバックエンドの CORBA アプリケーションの代理オブジェクトとして振舞い，CORBA アプリケーションとの整合性に関わる処理はこの Proxy コンポーネントがその一切を引き受ける．

図 1 に CORBA Proxy の概略図を示す．CORBA Proxy

は実行時に英大文字で始まる Java クラスと英小文字で始まる名称の Java クラスのオブジェクト間の双方向変換を行い，null フィールドについては変換時にオブジェクトを詰める Object Converter 機能を提供する．この機能により, Web サービス･クライアント･プログラムからは J2EE 規約に準拠したインターフェースで CORBA アプリケーションを呼び出すことができ，さらに呼び出し時の引数への値のセットも必要最低限で済ますことができる．

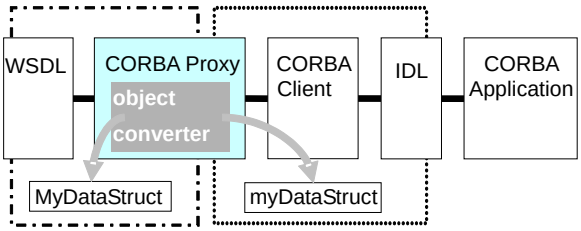


図 1. CORBA Proxy

## 4. CORBA Proxy Code Generator

筆者らは CORBA Proxy を生成するツール CORBA Proxy Code Generator を開発した．ツールは Eclipse のプラグインとして実装した．このツールは IDL ファイルのみを入力として CORBA Proxy を Façade Bean として生成する. ハイ･パフォーマンスを達成するため, Façade Bean は Stateless Session EJB として生成し, かつ内部では ORB オブジェクトをキャッシングする機構を備えている．

## 5. CORBA Proxy の評価

筆者らは既存の CORBA アプリケーションを Web サービス化して，BPEL (Business Process Execution Language)プロセスからそれらのサービスを呼び出すアプリケーションを開発するプロジェクトに携わった．その際，実際の CORBA アプリケーションに接続した環境で，開発した BPEL プロセスのパフォーマンス測定を行った．

パフォーマンス測定は，3 つの引数を持ち，リターン値を返す CORBA オペレーションを呼び出す BPEL プロセスを使用して行った．引数とリターン値共にネスト構造を含む複合データ型をインターフェースとして持ち，ネストの深さは最大で 3 階層となっていた．測定では BPEL プロセスを 300 回連続で実行し，各ステップで処理に要した経過時間を計測した．

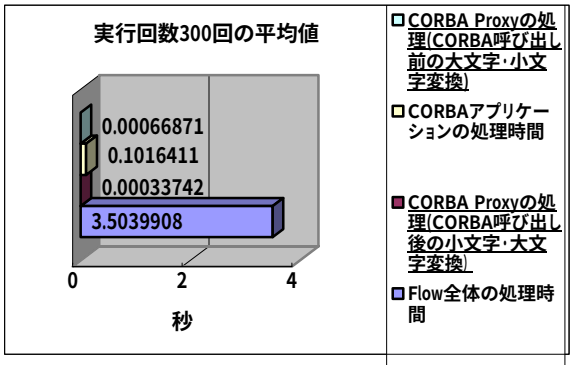


図 2. パフォーマンス測定結果

パフォーマンス測定結果は図 2 の通りである．グラフの横軸は 300 回の測定値の平均時間を秒で表している．CORBA Proxy によるオーバーヘッドは，グラフの上から 1 番目と 3 番目の処理時間の合計の約 0.001 秒となる．CORBA アプリケーションでの処理に 0.1 秒かかっている(グラフ上から 2 番目)のに対し，オーバーヘッドは CORBA 呼び出しの 1/100 程度であった．また今回パフォーマンス測定を行った CORBA オペレーションは比較的小さい処理のものであった．以上より CORBA Proxy は実用に耐えられるものであることが分かった．

## 6. 考察

CORBA Proxy について考察を行う．データ型定義名を英小文字で始める C/C++言語などのプログラミング言語で実装した CORBA アプリケーションでは，英大文字/英小文字名問題は高い確率で発生し得る. 一般的には, 稼動中の CORBA アプリケーションの書換えは困難であるため，CORBA Proxy の処理方式は有効と言える．ここで問題となってくるのは実行時のパフォーマンスである．処理時間の劣化の点から考えると，既存 CORBA アプリケーションはデータベース接続など，それ自体の処理時間が大きくなる傾向が強い．そのため CORBA Proxy 自体の処理は相対的に小さいものとなり，劣化への影響は少ないと考えられる．この考察はパフォーマンス測定結果からも妥当であると言える．

次に, 開発した CORBA Proxy Code Generator について考察を行う. 既存 CORBA アプリケーションの IDL ファイルで定義されているデータ型やオペレーションの数は膨大であるため，ユーザが CORBA Proxy の機構を備える代理オブジェクト･クラスや変換ユーティリティー･クラスをスクラッチから開発することは，開発工数の面からも負担が大きい．また，オブジェクト指向を適用可能な CORBA IDL では IDL を独自に拡張することが可能で，IDL の差分ごとに，あるいは IDL を変更する度に CORBA Proxy のコードを修正することもまた現実的ではない．その点，CORBA Proxy Code Generator ツールを利用する事で，ユーザは IDL ファイルのみを入力として用意すれば，Eclipse のユーザ･インターフェースから簡単に CORBA Proxy を生成することができ，開発効率を格段に向上できると言える．

## 7. おわりに

複合データ型名やオペレーション名の英大文字/英小文字問題は，CORBA アプリケーションに限らず，既存アプリケーションを Web サービス化する際に発生しうる問題と考えられる．なぜならば Web サービス･インターフェース(WSDL，XSD)で定義されている複合データ型が英小文字で始まる名称の場合は, 必ず大文字の Java クラスにマッピングされるからである．その際に，今回提案を行った Object Converter の考えを含んだ Proxy 方式を適用することが，一つの解決策になるものと考えられる．

## 謝辞

CORBA Proxy Code Generator の開発，および本稿執筆にあたり，多くの助言を頂きました日本アイ・ビー・エム インダストリアル ソリューションズ(株)の服部隆一郎氏，為広隆二氏，廣内正豪氏，髙梨尚美氏，日本アイ・ビー・エム(株)の川村建夫氏，岡野彰氏，浅香俊一氏にあらためて深謝いたします．