ソフトウェア工学 124−15
（1999．10．22）

Javaによるサーバサイドアプリケーション開発

伊波 通晴†

アブストラクト — 生産性の向上と開発コストの低減を目標に，Java 言語を中心としたオブジェクト指向技術によるサーバサイドアプリケーション開発が現在進められている．再利用を意識した，これら技術の適用ポイントついて記述する．

Server-side Applications Development with Java

Iha, Michiharu

Abstract - In development of server-side applications, we can use Java and Object Oriented approaches to improve the productibity and the development cost. This paper describes the key point of using the Object Oriented technoroogy in consideration of reusability.

1. はじめに

Java によるシステム開発事例も徐々に増えており，開発言語/実行環境として主流となる可能性が高まっている．特に n 層型アプリケーション開発における，アプリケーション層(業務ロジック)を実装するサーバサイドでの Java 利用が進んでいる．

クライアントサイド Java としては性能面での問題がしばしば取り上げられるが，サーバサイドではほとんど問題にならず，次の利点が活かされる．

- 言語としての生産性の高さ
- オブジェクト指向分析・設計との相性の良さ
- マルチプラットフォーム

アプリケーション開発の効率を高め，開発コストを低減するためには，再利用が重要なキーワードになっている．これに対して、マルチプラットフォームという特性が，インターネットや WWW サーバに対応したシステム構築も視界に入れた場合，Java オブジェクトの再利用性は魅力である．

以下，Java をベースとしたシステム開発に，再利用という視点を加えたオブジェクト指向技術適用のポイントについて述べる．

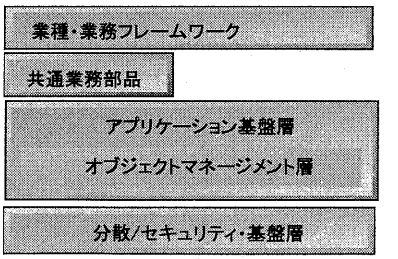
2. Java とアプリケーションフレームワーク

2.1. 再利用を意識した AP 開発基盤の整備

オブジェクト指向言語の Java の利用に加え，開発方法論も再利用を達成するためのポイントになる．再利用を進め，開発の効率化をはかり，開発コストを低減するための主役として，オブジェクト指向開発の導入をはかっている[1]．また，これまでプロジェクト独自の記述形式を取っていた各開発における設計書の記述形式を，標準の UML を使用するように統一することで，設計情報の再利用を促進している．

分析・設計工程では，アナリシスパターン[2]，デザインパターン[3]といった広く知られているパターンを積極的に利用することで効率と柔軟性の確保を目指している．

さらに、既存のパターンの利用だけではなく、プロトタイプ作りにおけるオブジェクト指向分析・設計を通して得られたドメインモデルから，ソフトウェアアーキテクチャに従ったフレームワークを構築し，アナリシス/デザインパターンを抽出するといった実システム開発の生産性をより向上させるための効率化作業も行っている．こういった作業工程の変化は，技術・方法論の進展はもちろんのことだが，受注型から提案型のシステム開発という流れにより，プロトタイプ作りに理解が得られやすくなったことも大きい．



フレームワーク作りはアプリケーション構築のための一貫したプログラミングインタフェースと構造を提供するように，体系的な文書化に重点を置いている．また、Java 言語の持つ interface 定義の利用，継承時の最適なモディファイアの選択，パッケージや protected な

† NEC クライアントサーバソフト技術研究所

どによるアクセス制限，およびReflection APIといったJavaの特色がパターンおよびコンポーネントオブジェクトの作成を支援してくれる．

OMGのドメインタスクフォースにおけるオブジェクトモデルへの取り組みの動向も注視しており，PDMなどでその成果が現れつつある．標準モデルに準拠することで各企業間での設計流用の可能性を秘めている．

2.2. Javaコンポーネントオブジェクト

分析・設計情報の再利用だけではなく，実装(ライブラリなど)の再利用も重要である．基盤技術(通信，トランザクション，DB アクセス，セキュリティなど)の違いを隠蔽してビジネスロジックの再利用性を高めることができるような，モデル/インタフェースを備えたJavaベースのサーバ側コンポーネント，具体的にはEnterprise BeanまたはCORBAオブジェクトの形態による流通部品としての拡充を今後行う予定である．これらを組み合わせることにより，サーバーサイドアプリケーションを迅速かつ容易に構築することができる．

コンポーネントオブジェクトの事例には，ドメインオブジェクトや共通オブジェクト，パターンを提供するIBM SanFrancisco[4]がある．

また再利用可能なJavaコンポーネントを新規に作成することを支援するための手順，ツール類の提供も検討課題であると認識している．

3. Javaとアプリケーション基盤

3.1. 分散オブジェクト基盤

分散オブジェクトの基盤としてCORBAがある．しかしCORBAでは，サーバ側コンポーネントのフレームワークを厳密に規定していないこともあり，トランザクション管理，データの永続化，セキュリティ管理などを行うコンテナが規定されたEnterprise JavaBeans(以下EJB)が注目されている．EJBコンテナの利用により，プログラミング作業を大幅に軽減できると期待されている．

また，宣言型プログラミングの面からもEJB利用のメリットがある．宣言型プログラミングは、アプリケーションの挙動を，コードを変更すること無く，属性の変更だけで実現する．ORB製品もデータベースアクセスに関する部分で，属性情報を利用したものが少なくないが，各社独自仕様であるため異なるベンダのサーバ上では，コンポーネントが上手く動作しない．一方EJBはDeploy Descriptorでこれを規定しているため再利用性が高い．

3.2. データベースアクセス基盤

データベースの検索結果として大量のデータが得られることが予想される場合，通信量や多数のインスタンスの生成といった問題を解決する必要がある．この場合，JavaでSerializeされるオブジェクトモデルを変更することも考慮している．

またオブジェクト指向DBの利用が可能ならば，オブジェクトをそのまま格納できて都合が良いが，既存資産や，あるアプリケーション分野においてはリレーショナルDBの利用が必然となる状況も多い．オブジェクトリレーショナルマッピング[5]という手法も使われる．

3.3. データリソース基盤 － XML

EDIやサプライチェーン管理といった企業間連携での標準形式としてXMLの適用が広く検討されている．以下の点でXMLを扱う言語としてJavaは適していると考えている．

- Unicodeを使用するXMLに対して，Javaは内部的にUnicodeで文字列を扱う．
- Javaに対応したXMLパーサーが多数存在する．
- DOMの標準バインディングorg.w3c.dom，およびSAXパーサーのorg.xml.sax.Parserといった Javaインタフェースが提供されている．

4. おわりに

Javaによるアプリケーション開発の基盤整備は急速に進められており，今後さらにJavaによるシステム事例は増大する方向である．

参考文献

[1] 石井慎一郎，他: *流通業店舗システムへのオブジェクト指向技術の適用*，情報処理，Jan. 1999.

[2] Fowler, M.(堀内一，他訳): *アナリシスパターン*，アジソンウェスレイジャパン(星雲社)，1998.

[3] Gamma, E. et al. *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley, 1995

[4] 日本IBM SFプロジェクト推進監訳: *IBM サンフランシスコ*，トッパン，1998

[5] Agarwal, S. et al.: *Architecting Object Applications for High Performance with Relational Databases*, Persistence Software Inc., 1998.