オブジェクト指向技術を用いたワークフローシステムの開発

4C－9

西川 大一　児玉 晴彦　伊賀 靖徳　清水 俊一郎　清 和由　小林 巧　村田 孝志
株式会社情報技術コンソーシアム

1.はじめに

オブジェクト指向技術の実践的な評価を行うためのパイロットアプリケーションとして、ワークフローシステムを開発した。本報告は、このシステムのオブジェクト指向技術を用いることにより実現した拡張機能について述べる。これにより、アプリケーションの拡張性に関するオブジェクト指向技術の有効性を示す。

2.ワークフローシステムの概要

ワークフロー[1]は仕事の流れを定義し、定義に従って仕事を遂行する仕組みである。仕事の単位であるアクティビティ[1]を直列または並列に接続したものをプロセス[1]と呼ぶ。アクティビティは担当者(Participant[1])に割り当てられ、担当者はその仕事を実行する。アクティビティで参照・更新される文書などの情報をフローデータ[1]と呼ぶ。

ワークフローシステムはプロセスの定義と管理を行い(定義機能)、定義に従ってプロセスを実行し(実行機能)、またその状態を監視する(監視機能)システムである。仕事(プロセス)の発注者がプロセスを起動すると、ワークフローシステムは担当者に作業指示し、担当者はフローデータを参照・更新する。ワークフローシステムと担当者が相互に作用しながらワークフローが遂行される。プロセスが終了すると、仕事の成果物が発注者に納入される。

"Development of Workflow System using Object-Oriented Technology",
D.Nishikawa, H.Kodama, Y.Iga, S.Shimizu, K.Sei, T.Kobayashi, T.Murata,
Information Technology Consortium

3.ワークフローシステムの実装

3.1 開発・実行環境

OMG CORBA[2]に準じた分散オブジェクト実行環境で動作するようワークフローシステムを実装した。サーバはJavaとC++言語で、クライアントはAppletとしてJava言語で作成した。

3.2 ソフトウエア構成

定義機能、実行機能、監視機能を実現する分散オブジェクトサーバ5本とそれらを利用するためのユーザインタフェースのクライアントツール3本より構成される。

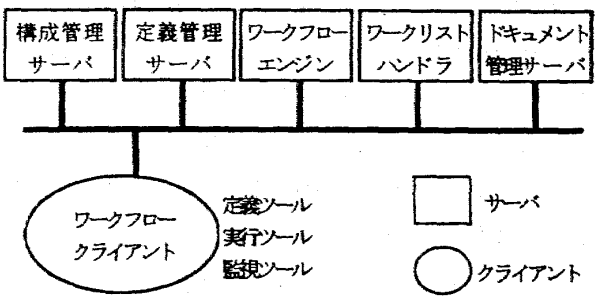

図1 システム構成図

■構成管理サーバ：サーバの構成を管理する。
■定義管理サーバ：プロセス定義を管理する。
■ワークフローエンジン：プロセス定義を解釈することによってプロセスインスタンスを生成する。このインスタンスはプロセス定義に従ってアクティビティを実行する。
■ワークリストハンドラ：担当者ごとの仕事リスト[1](作業すべき仕事の一覧)を管理する。
■ドキュメント管理サーバ：フローデータを管理する。
◆定義ツール：プロセスの定義を視覚的に行わせ、それを定義管理サーバに登録する。
◆実行ツール：プロセスの実行と仕事(アクティビティ)の処理を行う。
◆監視ツール：実行されているプロセスの監視・操作を行う。

4.ワークフローシステムの拡張機能

ここでは、システムのプログラムを変更することなく、システムの使用者が作成したプログラムをシステムが取り込む機能を拡張機能としている。

4.1 フローデータ画面の拡張機能

2節で述べたようにアクティビティの処理で担当者はフローデータを参照・更新する。この時のフローデータ画面は実行ツールで自動生成する。しかし、プロセスの定義者が独自の画面を使用したい場合が考えられる。そこで、プロセス定義者が作成した画面をフローデータ画面として使用する拡張機能を提供している。

この機能はデザインパターンの Factory Method パターン[3]と JavaBeans[4] (Java のコンポーネント)を使用して実現している。独自のフローデータ画面を使用するためには、まず、JavaBeans 形式で画面を作成する。これは市販されている Java 開発ツールを使用して作成する。次に、規定のディレクトリに作成した Bean を保存する。これで、定義ツールでアクティビティの情報として作成した Bean のクラス名で画面を指定できるようになる。

アクティビティの処理を行う時には、登録された画面のクラス名からインスタンスを生成する。これには実行時に結合するので動的リンクという(コンパイル時に結合するのは静的リンク)。動的リンクを使うことにより、システムのプログラムの変更やリコンパイルすることなく、プロセス定義者が作成した画面を簡単に使用することができる。

画面のインスタンスが生成されると、そのインスタンスのメソッドを検索してフローデータを設定する。アクティビティの処理が終了すると、インスタンスのメソッドを通して編集されたフローデータを取得して、サーバへ送信する。このインスタンスのメソッドを調べ、呼び出す機能は Java 言語で提供されている(Reflection[4])。

4.2 自動アクティビティの拡張機能

2節でアクティビティは担当者に割り当てられると述べたが、人の代わりにコンピュータなどが行うアクティビティもある。これを自動アクティビティ[1]と呼ぶ。自動アクティビティには、メール送信や印刷など汎用的なものの他にシステムを使用するユーザ固有のものが想定される。このユーザ固有の自動アクティビティを追加する拡張機能を提供している。

この機能は Template Method パターン[3]を使用して実装している。IDL で定義されている Node はアクティビティのテンプレートであるので、Node を継承して自動アクティビティを簡単に実装することができる。これにより、プロセスインスタンスは自動アクティビティをアクティビティとして認識して、自動アクティビティを実行することができる。

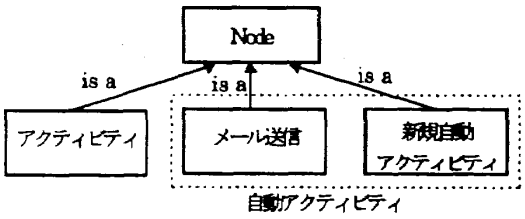

図 2 Node モデル図

5.おわりに

オブジェクト指向技術を用いることにより、柔軟で拡張性のあるシステムを構築できることを実証した。アプリケーションの拡張性においてオブジェクト指向設計技術は、特にデザインパターンは、非常に有効である。

参考文献

[1] WfMC (1996): *Workflow Management Coalition Terminology & Glossary*

[2] OMG (1991): *The Common Object Request Broker: Architecture and Specification*

[3] Gamma, E. et al (1995): *Design Patterns*, Addison-Wesley

[4] JavaSoft(1997): *JDK™1.1.4 Documentation*

この研究は、情報処理振興事業協会(IPA)の先進的情報処理技術の開発促進事業で実施されている「オブジェクト指向によるソフトウエア生産技術開発」プロジェクトの一環として実施している。