4P－2

分散共同開発支援環境の開発(2)
－アーキテクチャ－

金谷延幸　大久保隆夫　堀田勇次　原裕貴　上原三八
(株)富士通研究所

1. はじめに

近年のインフラの進歩に伴い、高度な共同並行開発作業の効率向上を目標とする分散システムの構築が求められている。このようなシステムでは、業務の本質的な情報を格納するプロセス管理ツールや文書管理ツールなどの基盤ツールと、ユーザが直接作業を行うワードプロセッサ、CASE ツールなどのユーザアプリケーションを統合する必要がある。また、システムの動作は、プロジェクトの作業手順を表現する業務知識に基づき、その追加、変更、改善が容易に行える必要がある[1][2]。

本論論文では、分散共同開発支援システム CEE で用いられている、MMV(Model, Mediator, View)アーキテクチャの提案を行う。本アーキテクチャを用いることにより、さまざまなアプリケーションを異機種プラトフォーム上で統合し、システムの変更が容易な、分散共同作業支援システムの実現が可能となる。

2.分散共同開発システムの課題

一般に、分散共同開発支援システムは、以下の 3 つの論理を含む。

1. 業務論理
ユーザが実作業を行う際に必要となる本質的な情報と、作業手順と、作業内容
例:組織構成やドキュメントやスケジュールなどの共有情報、会議手順などの作業手順、会議開催日などの作業情報
2. アプリケーション論理
上記の作業に対してシステムが行うサービス内容
例: 会議開催作業の際に、システムは議題を表示と予定表の表示を行う。ユーザが、会議開催日を入力すると、会議作業情報の会議開催日に書き込む。
3. ユーザインタフェース論理
例:ウインドウやボタンの配置などのレイアウト、日程入力方法などのユーザとのインタラクション

従来手法では、これら 3 つの論理が混在して実装されており独立した変更が難しい。

さらに、複数の基盤ツールを統合する場合、例えば業務情報をデータベースから取り出す、スケジューラからスケジュールを取り出す、プロセス管理ツールから会議手順を取り出すといったコードが、アプリケーション論理中に散在する。従って、プログラムが複雑になり、変更が難しくなっている。

即ち、従来技術では分散共同開発支援システムの開発ができなかった。3 章では、これらの問題を解決する、MMV アーキテクチャについて説明する。

3.MMV(Model, Mediator, View)アーキテクチャの提案

この章では、MMV アーキテクチャの詳細について説明する。

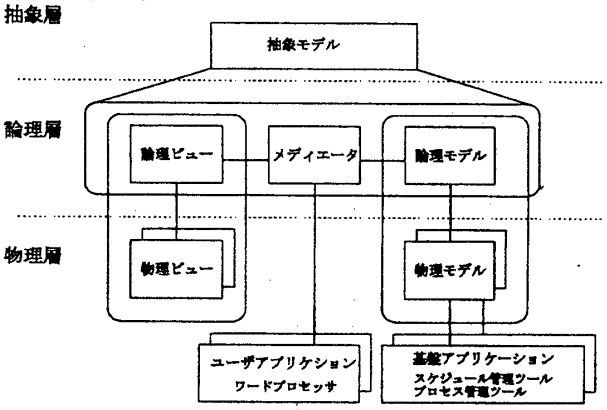

図 1. MMV アーキテクチャ

MMV アーキテクチャでは、まずユーザに提供する情報とサービスを設計し、これを抽象層と呼ぶ。次に、この抽象層で設計された仕様を、先に述べた 3 つの論理に分割し、対応する 3 つのクラス、モデル、メディエータ、ビューとして、設計し実現する。次に、3 つのクラスについて説明する。

3.1.モデル

モデルは、業務論理を実現するクラスである。企業情報やプロジェクトや個人、また扱っている作業や作業手順、ドキュメントなどの情報とその操作を実現するクラスなどがある。

A Development of Concurrent Engineering Environment(2), -Architecture-
Nobuyuki Kanaya, Takao Okubo,Yuuji Hotta, Hirotaka Hara, Sanya Uehara
Fujitsu Laboratories Ltd.
4-1-1 Kamikodanaka Nakahara-ku, Kawasaki 211-88, Japan

モデルはさらに、物理モデルと論理モデルからなる。論理モデルは業務論理に対応するクラスであり、物理モデルは基盤ツール毎に実装するクラスである。例えば、会議開催作業情報という論理クラスに対して、作業手順、スケジュールといった、別の基盤ツールに格納されている情報を、別の物理クラスとして実装する。この分離によって、メディエータからは、論理クラスが提供する統一したインタフェースで、複数の基盤上の情報にアクセスでき、また基盤ツールの変更を容易に行うことができる。

3.2.メディエータ

メディエータは、アプリケーション論理を実現するクラスである。メディエータはモデルから情報を取り出し、情報を変更し、操作を実行し、その結果をビューに反映する。メディエータ設計する上では、注意点として以下の2つがあげられる。

1. メディエータのみが他サービスを起動
   例:「会議開催サービス中に、日付入力を行う際に、予定表メディエータを生成して日付を入力させる」といった操作は、必ずメディエータだけが行うよう設計する。
2. モデルから取り出した情報の単純化
   例: 会議開催情報の中に参加者氏名を表示させる場合、会議モデル情報の関連付けされた個人情報モデルから氏名だけを取り出し、それを文字列としてビューに渡す。

これらの点を意識して設計することにより、業務論理やインターフェイス論理との分離を図ることができる。

3.3.ビュー

ビューは、ユーザインタフェース論理を実現するクラスである。メディエータから渡された情報を表示し、ユーザとのインタラクションを行う。

さらに、ビューは論理ビューと物理ビューとして実装する。物理ビューは実際のGUI部品、例えばウィンドウ、ボタン、ダイアログや、その物理的な配置を持つクラスである。論理ビューは、物理ビューの構成を把握し、メディエータから渡される情報を適切な物理ビューに渡し、GUIのイベントが起こった際にはメディエータのどのメソッドを呼ぶかを把握する。物理ビューによって、GUIの実装によらないメディエータとビュー間インタフェースを実装できる。

4.MVCアーキテクチャとの比較

MVC(モデル, ビュー, コントロール)アーキテクチャのモデルはMMVアーキテクチャのモデルとメディエータに、またMFCのビューはMMVの物理ビューに、コントロールは論理モデルの一部に対応する。

MVCアーキテクチャは、単にユーザインタフェース論理をアプリケーションから分離する為のアーキテクチャであり、業務論理、アプリケーション論理を分離することはできない。MMVアーキテクチャは、この3つの論理を分離が可能である。

5.実装

このMMVアーキテクチャを共同作業支援システムCEEに適用した。この結果、以下のような利点が得られた。

1. モデルの柔軟な変更
   例:プロセス管理ツールに格納されている作業手順の変更が容易に行える。
2. 新たなユーザサービスの追加が容易
   例:会議参加者の追加操作を組織一覧から入力するように変更することが、容易に行える。
3. 別のユーザインタフェースの構築が容易
   例:現在のCEEでは、JAVA、VisualC++とMFC、HTML、キャラクタコンソールといった様々なフレームワークに基づくユーザインタフェースを実装することが、メディエータやモデルの変更無しに行えた。

6.結論

本論文では、分散共同作業支援環境実現の課題について述べ、この課題を解決するMMVアーキテクチャの提案をおこなった。このアーキテクチャでは、システムを3つの論理に対する3つ(MMV)のクラス群に分割し、抽象度の高いインタフェースによって各論理間の関係を実現する。このアーキテクチャを用いることによって、従来のアーキテクチャでは難しい、システムの変更が可能となった。

今後の課題としては、業務モデル仕様記述から、モデル、メディエータ、ビュークラスを自動生成がある。このシステムは、抽象度の高い業務モデル記述からの自動生成に適しているので、この利点を生かし、業務モデル仕様記述からシステムの自動生成を実現したい。

参考文献

[1] 上原三八、その他：プロジェクトモデルに基づく分散共同開発支援の自動化,オブジェクト指向最前線(情報処理学会OO'96シンポジウム),朝倉書店,pp. 79-86 (1996).

[2] 大久保隆夫：分散並行開発支援環境の開発(1)、情報処理学会第53回全国大会