Model 分散型 MVC を用いた 部品組合せによるアプリケーションの構築

6J−9

大島 満 * 原 辰次 †

* 日本アイ・ビー・エム (株) 東京基礎研究所 † 東京工業大学大学院 システム科学専攻

1 はじめに

部品をプロセス化し、これらを組み合わせて使用する分散アプリケーションは、負荷分散、拡張性などの点で有効である。このようなシステムを提供する方法として、粗粒度オブジェクトを部品としたデータ駆動方式に基づく方法[1][2]が研究されている(図. 1)。これは、Smalltalk 上のデータ変更伝搬メカニズムをネットワーク通信で実現し、プロセスとして実現されたオブジェクト間を結ぶものであり、高い拡張性と、オブジェクト間の結合をユーザーに解放することにより、自由なカスタマイズも実現している。反面、反面ネットワーク透過性が低く開発効率が悪い、計算機負荷が大きいなどの欠点も持っており、実用上の問題が多く残されていた。

本稿では、このような欠点を克服するため、クライアントサーバ型分散アプリケーション環境を実現するアーキテクチャ、**Model 分散型 MVC(Distributed-Model View Controller:DmVC)** を提案する。さらにこれらの考えに基づき分散 CAD 環境 CENet(CAD Environment on Netowrk) を開発した。また、このシステム上に制御系設計支援 CAD システムを構築したので報告する。

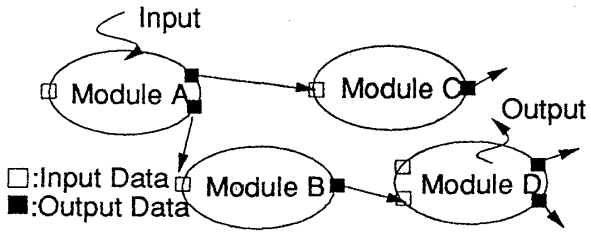

図 1: データ駆動型分散アプリケーション

2 Model 分散型 MVC

本稿が提案する DmVC は、クライアントサーバ型の分散処理を基礎にした MVC パラダイムの一つである

Construction of Distributed-Application based on Distributed-Model View Controller

Misturu Oshima * and Shinji Hara †

* IBM Research, Tokyo Research Laboratory † Dept. of System Science, Tokyo Institute of Technology

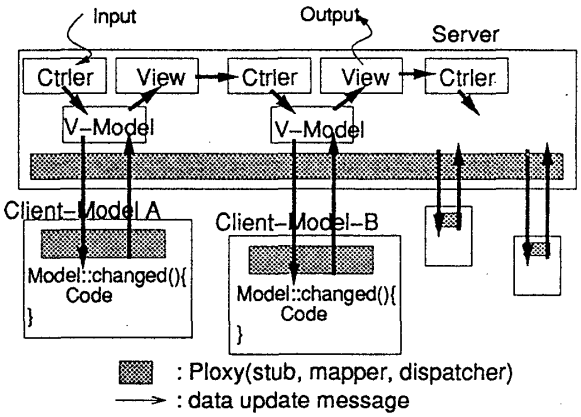

図 2: Model 分散型 MVC

(図 2)。Model は一つの独立したクライアントプロセスであり、View,Controller はサーバ側で生成されるオブジェクトである。インターフェイスを受け持つサーバと、実際の処理を受け持つ Model(クライアント) が複数集まって分散アプリケーションを構成する。

Model 自身はユーザーインターフェイスや他の Model との通信機能は持っておらず、サーバとの通信機能だけを持つ。インターフェイスや Model 間の通信や管理機構などはすべてサーバによって実現される。したがってプロセス間通信の管理が容易になり、かつ Model 自体も軽くなるので、分散効率の向上がはかれる。

クライアント (Model) が起動されると、サーバ側に V-Model 呼ばれる仮想的なモデルと、それに付随する View と Controller が生成される。DmVC では、View,Controller が一つの Model に属する形をとっており、Model は複数の View,Controller を持つことができる。View と Controller は Model から送られるプログラムで定義され、インタプリタによって生成/実行される。このプログラムは Model のプログラム自身とは独立に提供され、このプログラムを変更することによりクライアントのプログラムを変更/コンパイルをすることなくインターフェイスの変更が可能である。

V-Model はサーバ上の View,Controller とクライアント側の Model との橋渡しをうけもつオブジェクトで、自分自身に送られたメッセージを再送する機能がある。たとえば、クライアント側の Model から View に送ら

れるメッセージは一旦 V-Model に送られ、View に再送される。したがって、Model と View,Controller 間の通信は MVC 間のメッセージパッシングパラダイムで行なわれ、通信を隠蔽することができる。実際のプロセス間通信に使用する、stub、mapper、dispatcher は、クライアントおよびサーバ内に用意された Ploxy 内で実現されており、クライアントのプログラマは意識する必要がない。

各 Model(クライアント) は、入出力用データを持っており、これらのデータをインターフェイスとしてデータ駆動的に動作する。クライアントのプログラマは、データ更新メッセージで起動されるメソッドを定義することができ、一般的にここにアプリケーションコードを記述する。入力データの更新メッセージが Controller から Model に送られると、定義したメソッドが起動される。メソッド内で出力データが変更された場合、これに依存する View に対してデータ変更のメッセージが自動的に送られる。メッセージを受けとった View はデータの表示や対応する処理を行なう。このときデータの依存関係は、暗に生成される DependencyField オブジェクトによって管理される。

一方、Model 自身は相互通信する機能を持たないため、他の Model に出力データの変更を知らせる機構が別に必要になる。このため DmVC では通常の View,Controller とは別に、ConnectView,ConnectController と呼ばれるオブジェクトを提供する。ConnectView は Model の Model データ変更を監視し、ConnectController は他の ConnectView からのデータ変更メッセージを Model に伝える機能を持つ。そして、別の Model に属する ConnectView と ConnectController を結合することによって、ある Model でおきた出力データの変更をこれらのオブジェクトを通して、他の Model へと伝えることができる。また、ConnectView と ConnectController 間のデータの結合は、ユーザーによって自由かつ動的に変更が可能であるので、ユーザーが望むアプリケーションを構築することができる。

3 CENet システム

本研究で開発した CENet システムは、CENetBrowser とクライアントライブラリからなり、分散アプリケーションの総合的環境を提供する。CENetBrowser は DmVC のサーバー部と、ユーザー支援環境を統合したもので、Model クライアントの起動／終了／結合や実際の処理に加えて、各 Model の状態の inspection などの機能を持つ (図 3)。ユーザーはサーバ上に現れる仮想 CAD システムを通して、アプリケーションを操作する。Model は C++ で開発されたクライアント用ライブラリ (CENetLib) を

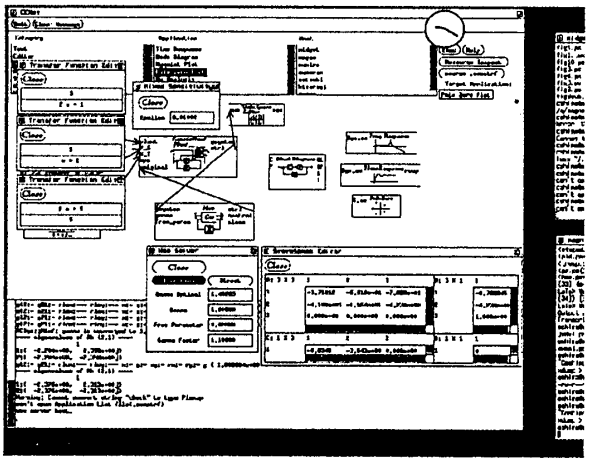

図 3: CENet システムと制御系 CAD

使用して作成するが、この時クライアントのプログラマはプロセス間通信や他のモデルの存在を意識する必要はない。CENetBrowser から起動された Model はサーバからのメッセージに応じて処理を行なう。アプリケーションの目的に応じて Model となるクライアント群を作成することにより、さまざまな種類の分散アプリケーションを開発できる。

4 おわりに

本稿では、ユーザーが自由にカスタマイズできる分散アプリケーションのための基本的アーキテクチャである、Model 分散型 MVC を提案した。これを利用することにより、ネットワーク透過性と開発効率の向上、負荷分散などが期待できる。このシステムを利用した制御系設計支援用の CAD システムがすでに実現されており、今後の研究／改良のために試用している。

参考文献

[1] 赤堀, 原 : オブジェクト指向に基づく制御系の解析・設計支援システム, 計測制御学会論文集,vol24,No. 5

[2] S.Hara, G.Yamamoto, et al. : A DISTRIBUTED COMPUTING ENVIRONMENT FOR CONTROL SYSTEM ANALYSIS AND DESIGN, Proc. of IEEE Symposium on CACSD'92, p47-54(1992)

[3] A. Goldberg: Smalltalk-80 - The Interactive Programming Environment, Addison-Wesley Publishing Company (1983).

[4] W.E.Stevens : UNIX NETWORK PROGRAMMING, Prentice-Hall International,Inc. (1991).