1M−8

Conference Shellの実装と評価

赤羽喜治　山田達司　山藤哲嗣　玉置政一　菅野政孝
NTTデータ通信株式会社

1．はじめに

現在、我々は遠隔地間の共同作業アプリケーション（以下共同作業AP）に必要とされる共通機能ライブラリ「Conference Shell」を作成している。本稿では、Conference ShellをMS-DOSへ実装する際の問題点及び解決策を検討する。さらにAPへの適用の容易さと実行速度の評価を行なう。

2．Conference Shellについて

Conference Shellは共同作業APの生産性の向上と相互接続性の向上を目的としている。特徴として以下のものが挙げられる。

・共同作業APに必要とされる通信制御機能、DB制御機能（作業履歴を保存）などを提供する。
・様々な作業形態に応じて複数の装置（DB機能、通信回線等）間のデータの流れを変更する。(図1)

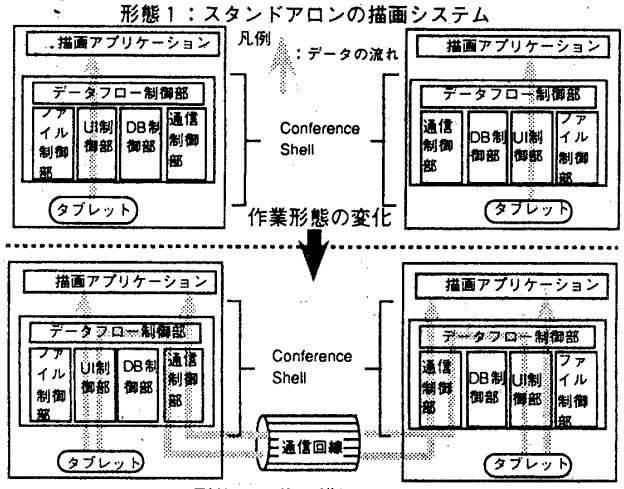

図1：Conference Shellの提供する作業形態例

3．MS-DOSへの実装

（1）課題

Conference ShellのMS-DOSへの実装にあたり、以下の2つの課題がある。

(a)処理要求の受付

APを構成する1モジュール（以下APサブモジュール）の実行中には、他のAPサブモジュールに対する処理要求を処理できない。

(b)制御部（DB制御部、通信制御部等）の並列実行

Conference Shellは描画中でも通信回線からのデータを処理する等、複数の装置からの処理要求を並列処理する。MS-DOSはマルチタスクをサポートしておらず、代替手段が必要となる。

（2）実装方式の検討

実装方式として、3つの方式が考えられる。Conference Shellの適用対象としてメインルーチンと、ユーザの操作によって呼び出されるAPサブモジュールから構成されるAPを考えた場合の各方式の違いを示す。（図2）

［方式1］APサブモジュールとConference Shellでループを作る。各APサブモジュールはConference Shellから処理要求（例：描画要求）を受け取る。

［方式2］メインルーチンとConference Shellでループを作る。Conference Shellは制御部（例：UI制御部：図1参照）から処理要求を得た後、メインルーチンに渡す。メインルーチンは処理要求に対応するAPサブモジュール（例：描画モジュール）を呼ぶ。処理が終わるとメインルーチンに戻る。

［方式3］簡易なマルチタスクモニターを作成する。APをタスクとして独立させる。各制御部からの処理要求に応じてメインルーチンは対応するAPサブモジュールを呼ぶ。

これらの方式を、課題に対する有効性、実装・移植・APへの適用等の容易さについて評価する。

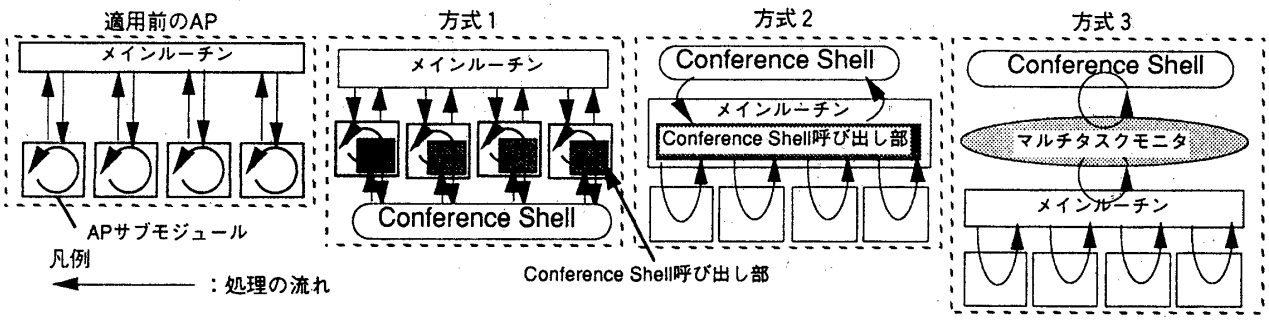

図2：実装方式の概要

An implementation and an evaluation of the comunication control system
Yoshiharu AKAHANE, Tatsushi YAMADA, Tetsuji SANTO, Masakazu TAMAKI, Masataka SUGANO
NTT DATA communications systems corporation

表１：MS-DOSへの実装方式の比較

	方式１	方式２	方式３
課題(a)	実行中でないAPサブモジュールから処理要求出来ない	APサブモジュールからの要求の処理は発生とほぼ同時	APサブモジュールからの要求の処理は発生とほぼ同時
課題(b)	常時処理可能	常時処理可能	常時処理可能
実装の容易さ	Conference Shellのみ作成	Conference Shellのみ作成	マルチタスクモニタの作成等が困難
移植の容易さ	OSに非依存	OSに非依存	OSや各機種に依存
APへの適用の容易さ	APサブモジュールへConference Shell呼び出し部を組み込むだけで容易	APサブモジュールが要求を処理する毎に終了するよう改造が必要	APをプロセスとして完結するよう改造が必要

・課題を解決するには方式３が一番望ましいが、実装に要求される条件を満たしていないため、課題をある程度解決し、かつ要求条件も満たしている方式２を採用する。

３．Conference Shellの評価

実装されたConference Shellの有効性を検討するため、以下の２点について評価する。

(a)APへの適用：一人用のAPにConference Shellを適用し、共同作業APに拡張することが容易かどうか。

(b)実行速度：Conference Shell上で動作するAPの実行速度はユーザ要求を満足するか。

（１）APへの適用

MS-DOS上で動作する描画ソフトウェアに対してConference Shellを適用した場合に対し、適用の容易さを評価した。

(A)改造内容

APにConference Shellを適用するには、Conference Shellの提供するAPIを使って、以下の２点の改造を行なう必要がある。

(a)入力インタフェースと処理部の分離：マウス読み取りルーチン等の入力インタフェースと、描画ルーチン等の処理部を分離し、Conference Shellを経由してデータを送るように改造する。（図３）

(b)制御フローの一本化：要求を処理する毎に終了するように各APサブモジュールを改造する。

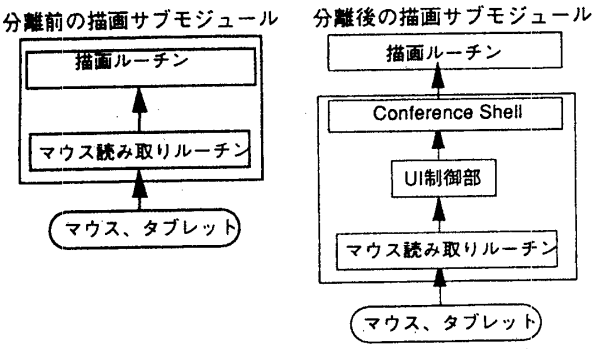

図３：入力インタフェースと処理部の分離

(B)改造量

Conference Shell適用のために、メニューの表示や機能の選択を行なう部分に対して２７％の改造を行なった。その他に、描画部に対して５％の改造を加えた。その結果、描画ソフトウェア全体の修正、追加は６％となった。AP全体を考えた場合、改造率は低く、適用は容易であるといえる。

（２）実行速度

Conference Shellを適用することによるAPの実行速度への影響を評価するため、Conference Shellによるオーバーヘッドを計測した。以下に形態毎の処理内容と１回当たりの実行時間を示す。

端末にはPC-9801(80486SX 16MHz)を用い、ISDN回線はシュミレータで代用した。

表２：Conference Shellの実行時間

	形態	実行時間(ms)
1	ローカル作業時	1.8
2	ISDN回線を介した共同作業（操作権：１人）（送信側）	5.7
3	ISDN回線を介した共同作業（操作権：2人）	6.1

表２で作業形態２、３の場合に実行時間が増加するのは、ISDN回線を利用することによる処理遅延であり、Conference Shellを適用することによるオーバーヘッドは作業形態１の場合と同程度である。

例えば、毎秒４０ポイントの描画処理を想定すると１ポイントの描画の実行時間は２５ミリ秒であり、Conference Shellを適用した場合オーバーヘッドは７．２％となる。

また、今回の描画ソフトウェアでは、いずれの作業形態においてもConference Shellの適用によって著しく操作感を損なうことはなかった。

４．まとめ

本稿では、MS-DOS上にConference Shellを実装する際の問題点と解決策を検討した。さらに実装されたConference Shellの評価を行なった。

今後、実装されたConference Shellの機能を拡張し、

(a)１対多、多対多の共同作業の実現

(b)LANへの拡張。ISDN回線との回線種別に依らないAPIの提供

の２点を実現していく。

［参考文献］

[1]山田他：「マルチメディアコンファレンスのためのConference Shellの概要」、情処全大H４秋、１９９２

[2]赤羽他：「マルチメディアコンファレンスのためのConference Shellの実現方式」、情処全大H４秋、１９９２