トランザクション処理アプリケーション開発手法に関する検討

4G－5

竹内 宏典[1)]　菅沼 毅[1)]　西之上 実[2)]

1)　NTT情報システム本部　2)NTTデータ通信（株）

1．はじめに

日本電信電話株式会社（NTT）及びエヌ・ティ・ティ・データ通信株式会社（NTT DATA）は、公募による国内外のコンピュータ・ベンダ５社*1と共同で、異機種コンピュータでシステムを統合するための共通アプリケーション・インタフェースである「マルチベンダ・インテグレーション・アーキテクチャ（MIA）」を開発した。[1]

MIAでは、特にトランザクション処理アプリケーション（以降単にAPという）を記述するための専用言語として「構造化トランザクション定義言語（STDL）」を規定している。このSTDLによって記述されたAPは、従来APに比べ、APの構造が異なっている。このため、MIAのAP設計は、従来のAP設計法ではなく、STDLに適したAP設計法での構築が必要となる。

本稿では、MIAでAPを設計する際の設計法を提案する。さらに、その提案に従ってAPを試作し、従来の設計法で作成されたAPと比較して、工期の削減及び生産性の向上に有効であることを報告する。

2．MIAのSTDLの特徴とAP構造

STDLは、仮想化レベルの高いAPIを規定した高級言語であり、通信アクセスやユーザ・アクセス等のトランザクションにかかわる処理を記述するモジュールと実際の業務処理（データベース・アクセスやデータの演算等）を記述するモジュールに分割したAP構造を規定している。これによって、トランザクション処理APの生産性やポータビリティの向上を可能とする。[2]

分割されたモジュールはタスク定義とプロセシング・プロシジャ、プレゼンテーション・プロシジャの３つからなり、それぞれ以下のような機能を持っている。なお、各モジュール間で受け渡されるデータ形式は、STDLによって予め定義する必要がある。[3]

タスク定義 … トランザクション範囲と大まかな処理フローを制御する。STDLで記述する。

プロセシング・プロシジャ … データベース処理、計算等の詳細な業務処理を行う。COBOL、C等の標準プログラミング言語で記述する。

プレゼンテーション・プロシジャ … 端末とオペレータとの入出力を制御する。各ベンダが個々に提供するフォーム管理システムのフォーム定義を利用する。

MIAのAP構造を図１に示す

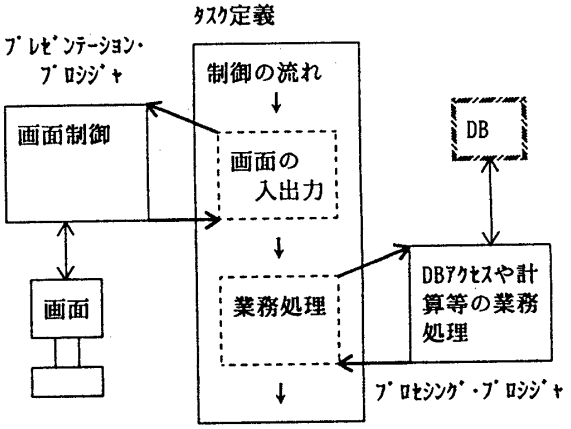


図１：MIAのAP構造

3．従来APとの比較

従来のAPは単一のプログラミング言語で、制御と処理の明確な区別をすることなく、ひとかたまりとしてAPをコーディングしていた。このため、「制御の流れを把握しにくい」とか、「モジュールの部品化が困難」、「保守性が悪い」といったような問題点があった。しかしながらMIAのAPでは、１つのAPを３つのモジュールに分離し、しかもそれぞれの記述言語が異なっているため、各モジュールを非常に独立した形で作成することができる。このため、従来APの問題点は解決され、AP設計者は、MIA－APを設計する過程において、無意識の内に、モジュール強度が高く、モジュール間の結合度の低い、保守性の良いAPを設計することができる。

4．MIA－AP設計法

ここでは、MIAのAPに適した設計法として次の２点に注目する。

（1）データ中心設計

２章でも述べたように、MIA－APで、タスク定義とタスク定義、タスク定義とプロシジャ間で扱われる全データは、全て呼び出し元のSTDL中で宣言しなければならない。つまり、AP設計段階の早い時期に各モジュール間の全てのデータの流れを洗い出し、入出力関係やデータ型を厳密に規定する必要がある。このような、AP構造の特性を考えると、システム設計のアプローチとしてデータ中心設計が非常に効果的であるといえる。

（2）APの分業体制

トランザクション処理を設計するには、システムの処理の流れを理解し、トランザクション範囲の設定や画面制御処理の設計、業務処理の設計等、さまざまな設計が

Development Method of Transaction-Processing Application.
Hironori TAKEUCHI 1),Takeshi SUGANUMA 1) and Minoru NISHINOUE 2)
1)NTT Information Systems Headquaters
2)NTT DATA COMMUNICATION SYSTEMS CORPORATION

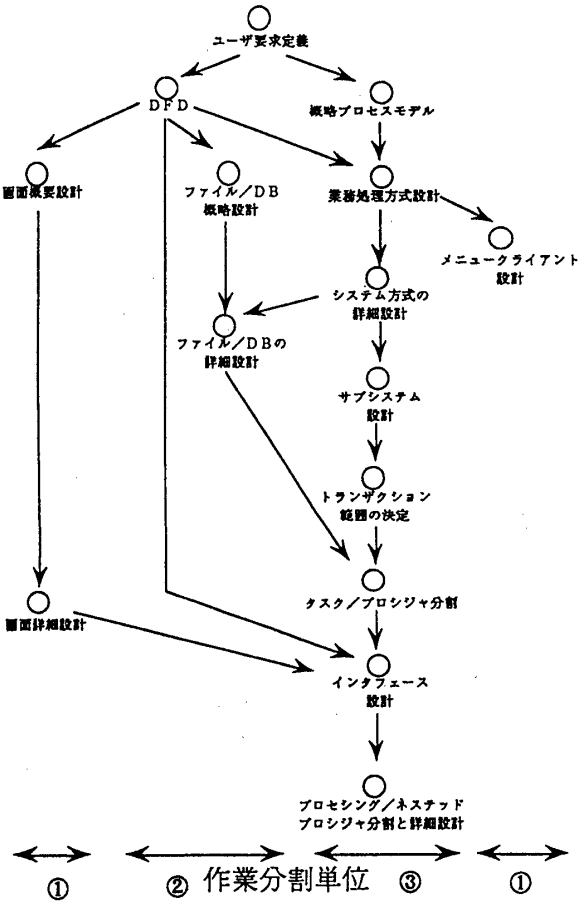
*1　日本アイ・ビー・エム株式会社、日本デジタル・イクイップメント株式会社、日本電気株式会社、株式会社日立製作所、富士通株式会社

必要となる。これらは、ＭＩＡ－ＡＰでいうとタスク定義、プレゼンテーション・プロシジャ、プロセシング・プロシジャの設計に相当する。そこで、ＭＩＡ－ＡＰで大規模システムを作成する場合は、次のような分業を行うことが有効である。

①タスク定義を作成するグループ
②業務処理（ＤＢ処理など）を作成するグループ
③プレゼンテーション処理を作成するグループ

これにより、タスク定義作成者のみがＳＴＤＬを習得すればよく、その中でトランザクションの設定やエラー処理を記述する。また、業務処理作成者は、トランザクション範囲などを意識せずＣ，ＣＯＢＯＬ等の言語で業務の実際の処理のみを記述するだけでよく、ＡＰの分業設計に非常に有効である。

今回のＡＰ試作は、データ中心設計の適用性、ＡＰ分担開発の評価を行うために図２に示す流れでＡＰの設計を行った。



①ﾌﾟﾚｾﾞﾝﾃｰｼｮﾝ ﾌﾟﾛｼｼﾞｬ担当　②ﾌﾟﾛｾｼﾝｸﾞ ﾌﾟﾛｼｼﾞｬ担当　③ ﾀｽｸ定義 担当

図２：ＡＰ設計の流れ

５．ＡＰの試作

試作ＡＰは、複雑なトランザクション処理（２フェーズ・コミット等）を含み、ワークステーションによるインタラクティブな処理を必要とする図３に示すモデルを作成した。

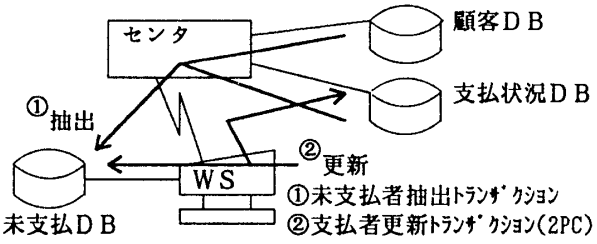


図３：試作ＡＰのモデル

６．評価

ＭＩＡの試作ＡＰの開発規模と生産性を下記に示す。

- 開発規模　…　総ステップは約５Ｋｓ
- 生産性　…　従来のＡＰ設計手法によるＡＰ作成の生産性を１とした場合、今回の生産性は約２となった。

図４に総ステップにおける各モジュールの占める割合を示す。なお、サンプルＡＰはワークステーション上の画面制御が多かったため、プレゼンテーション・プロシジャの割合が高くなっている。

①ソース全体の約２０％を占める全体のデータや定義部分をデータ中心設計により先に全て洗い出したため、ソースやデータの修正等が発生しても柔軟に対応することができた。
②分業体制を行うことによりスキル（あるいは専門）の差に応じたプログラムの設計を行うことができ、適材適所で全体の工期の削減ができた。なお、タスク定義部分はソース全体としての占める割合はわずかであるがＡＰの流れの制御やトランザクション設定など複雑な設計を行う必要があり、工期的には他の部分と同程度が必要であった。
③プレゼンテーション処理部分は、分業によって同一グループが全ての画面定義を作成したため、従来と比べＬｏｏｋ＆Ｆｅｅｌの統一に関する手戻りが少なかった。

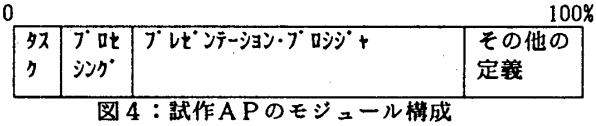


図４：試作ＡＰのモジュール構成

７．まとめ

今回提案したＭＩＡ－ＡＰ設計法は、試作ＡＰによる検証結果より生産性で２倍程度の効果が確認できた。今後、試作ＡＰの規模を大きくして、ＡＰ設計の分業体制と生産性等を評価し、ＭＩＡのトランザクション処理ＡＰの設計法を確立する。また、ＭＩＡ用のＣＡＳＥツールに関する要求条件を明らかにしていく。

参考文献

[１] Multivendor Integration Architecture,Version 1.0，第１編　概説，テクニカルリクワイアメント TR550001-1，ＮＴＴ，Ｊａｎ．，1991.
[２] 長谷川他（ＮＴＴ）　Ｒ＆Ｄレポート「マルチベンダ・システムの構築を容易にする技術を開発」ＮＴＴ技術ジャーナル　１９９１．５
[３] Multivendor Integration Architecture,Version 1.0，第２編　アプリケーション・プログラム・インタフェース仕様，テクニカルリクワイアメント TR550001-1，ＮＴＴ，Ｍａｒ．，1991.