マルチメディア通信と分散処理　46－9
（１９９０．　７．　１２）

オブジェクト指向言語による ＯＳＩ通信ソフトウェア開発の評価

勝山　光太郎　　中川路　哲男　　宮内　直人　　楠　和浩　　水野　忠則
三菱電機株式会社　　情報電子研究所

情報処理システムの発展に伴い，異種システム間の接続に対する要求が高まってきており，ＯＳＩ（開放型システム間相互接続）の標準化が行われてきた．情報処理との接点である応用層についても，標準化が進展し，各地で実装が試みられている．また，通信ソフトウェア開発においても，生産性向上が重要視されるようになり，オブジェクト指向言語による開発が注目を集めている．我々は，オブジェクト指向言語superCによるＯＳＩソフトウェアの構築を提案し，実際にいくつかのソフトウェアを作成してきた．本稿では，オブジェクト指向言語を利用したＯＳＩ通信ソフトウェアの開発について，構成法，開発法，性能を評価，考察した結果を報告する．

Estimation of OSI software development using an object oriented language

Kotaro Katsuyama, Tetsuo Nakakawaji, Naoto Miyauchi,
Kazuhiro Kusunoki, Tadanori Mizuno

Information Systems and Electronics Development Laboratory
MITSUBISHI ELECTRIC CORPORATION
5-1-1 Ofuna Kamakura-City 247 JAPAN

Due to the progression of information systems, requiremnts for interoperability between heterogeneous systems are raised. So OSI has been standardized. The standardization of the Application layer is in progress, and several systems are implemented. Productivity has become an important factor in the development of communication software, so that object oriented approach is watched. We developed an object oriented language superC, and developed OSI software using superC. This paper presents estimation and consideration of structure, development method, and performance of OSI software which we developed using superC.

1．はじめに

情報処理システムの発展に伴い，異種システム間の接続に対する要求が高まってきている．このため，ＯＳＩ（開放型システム間相互接続）の標準化がＩＳＯとＣＣＩＴＴで行われてきた．情報処理との接点である応用層についても，標準化が進展し，実装が試みられている．

従来，通信ソフトウェアは，実時間性が要求されることから，Ｃ言語やアセンブラによって性能を重視したコーディングがなされてきた．しかしながら，マイクロプロセッサの性能向上や，メモリ実装容量の拡大により，近年は，生産性向上が重要視されるようになり，オブジェクト指向言語による開発が注目を集めている．

オブジェクト指向言語によるソフトウェア開発の特徴としては，以下のことがよくいわれている[1].

①情報隠ぺいとデータ抽象化により，信頼性が向上する．

②動的束縛は，既存のプログラムを変更することなく，新しいオブジェクトのクラスの追加を許すので柔軟性が向上する．

③継承機能と動的束縛の組合せによって，プログラムの再利用が可能となる．

オブジェクト指向言語としては，smalltalk[2]などがあるが，広く通信ソフトウェアの開発に適用されているＣ言語との親和性から，Ｃ言語にオブジェクト指向の拡張を加えたものが利用可能と考えられる．Ｃ言語にオブジェクト指向を加えた言語として，Objective-C[3],C++[4]などが存在する．

これらの言語を利用した場合，最終的に目的とする性能がでない時，最適化の道がとざされてしまうことも考えられるため，我々が自由に中身を変えて行ける言語として，Ｃ言語にオブジェクト指向の拡張を加えたsuperCを開発した[5].

我々は，このオブジェクト指向言語superCによるＯＳＩソフトウェアの構築を提案し，実際にいくつかのソフトウェアを作成してきた[6,7]．本稿では，オブジェクト指向による通信ソフトウェアの構成法，および作成したソフトウェアに対する評価を行った結果を報告する．

以下，２章では，通信モデルとオブジェクト指向についてのべる．３章では，通信ソフトウェアの構成法について従来の方法とオブジェクト指向言語による方法の対応について述べる．４章では，我々の実際の開発事例について報告し，５章で，評価，考察をおこなった結果を報告する．

2．通信モデルとオブジェクト指向

ISO/IEC SC21/WG6では，応用層のモデルを標準化している[8].

図１に応用層の構造を示す．

この図では，３つのアソシエーションが張られている状況を示している．

したがって，３つのＳＡＯが存在する．

アソシエーション確立毎にＳＡＯが生成される．これは，オブジェクト指向のタイプとインスタンスの概念に合致している．

このように，概念モデルの中にオブジェクト指向の要素を取り込んできている．

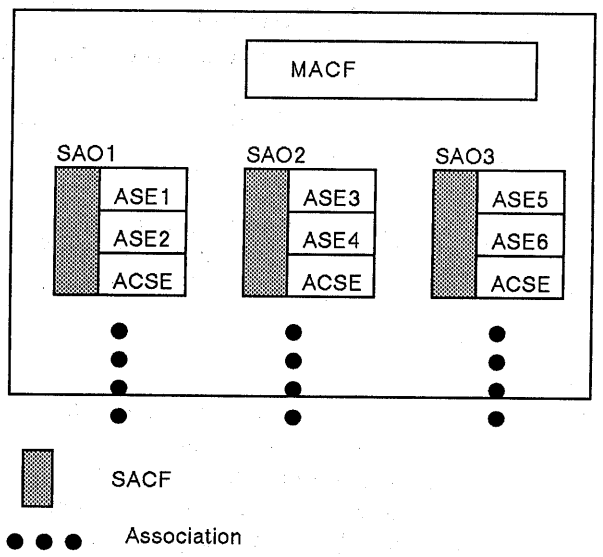

図１　応用層の構造

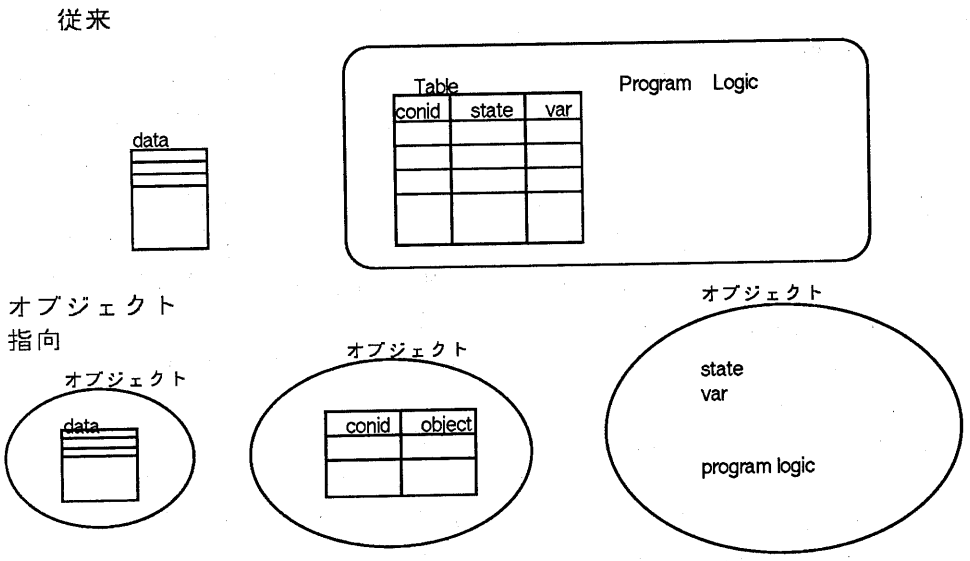

図２　構成要素の対応

３．通信ソフトウェア構成法

図２に従来のソフトウェア構造における要素と，我々の提案するオブジェクト指向による構成法における要素との対応を示す．

３．１　従来の構成法

Ｃ言語を利用して，通信ソフトウェアを開発する場合を考える．

（１）プロトコルデータ

確保されたメモリ領域にパラメタ値がセットされている．通常，コネクションを識別する識別子，プロトコルデータの型を識別する識別子，実際のデータから構成される．パラメタ値のセット等は，Ｃで記述された構造体に対する操作として行われる．

（２）プロトコルマシン

プログラムロジックとして，プロトコルの状態遷移表が実装される．

（３）コネクション管理テーブル

コネクション毎の，状態や，変数の管理をおこなうテーブルして存在する．

（４）状態および変数の扱い

コネクション管理テーブルで，コネクション毎に管理されている．

３．２　オブジェクト指向言語による構成法

オブジェクト指向言語を利用して，通信ソフトウェアを開発する場合を考える．

（１）プロトコルデータ

オブジェクトとして実現される．パラメタ値のセットは，メッセージによって行われる．コーディングする時，プロトコルデータの構造自身を気にする必要はない．

（２）プロトコルマシン

オブジェクトとして実現される．コネクション確立時に生成され，コネクション解放時に，消去される．

（３）コネクション管理テーブル

コネクションを管理しているオブジェクトの中に実体がとられる．コネクションとプロトコルマシンのオブジェクトの関係を保持するだけである．

（４）状態と変数の扱い

状態と変数は，プロトコルマシンのオブジェクトのインスタンス変数として存在する．

4．開発例

4．1　ソフトウェア構成

我々の提案する構成法にしたがって開発した事例を報告する．

図3にソフトウェア構成モデルをしめす．

この図では，ＦＴＡＭ（File Transfer, Access and Management），ＡＣＳＥ（Association Control Service Element），ＴＰ（Transaction Processing）を実装している例を示している．

ＦＴＡＭ管理オブジェクトは，３．２でのべたコネクション管理テーブル（ここでは，アソシエーション）をもつオブジェクトである．ＦＴＡＭプロトコルマシンオブジェクトは，アソシエーション確立毎に生成される．

ＴＰ－ＡＳＥ管理オブジェクトは，アソシエーションとＭＡＣＦオブジェクトの関連を管理しているオブジェクトである．ＭＡＣＦオブジェクトは，ダイアログとダイアログオブジェクトの管理をおこなう．ダイアログオブジェクトは，ダイアログ確立毎に生成され，ダイアログ内のプロトコルマシンを実現している．

ディスパッチオブジェクトは，データオブジェクトを該当するＡＳＥにディスパッチする．

4．2　プログラムの動作

アソシエーション確立を例に，プログラム動作の概略を図4に示す．番号は，図中の番号と対応している．

①アソシエーション確立要求のプロトコルデータが，オブジェクトとして生成され，ディスパッチオブジェクトに渡される．

②ディスパッチオブジェクトでは，モジュール識別子から判断し，ＡＣＳＥ管理オブジェクトにこのオブジェクトを渡す．

③ＡＣＳＥ管理オブジェクトでは，新しくＡＣＳＥアソシエーションオブジェクトを生成し，データオブジェクトを渡す．

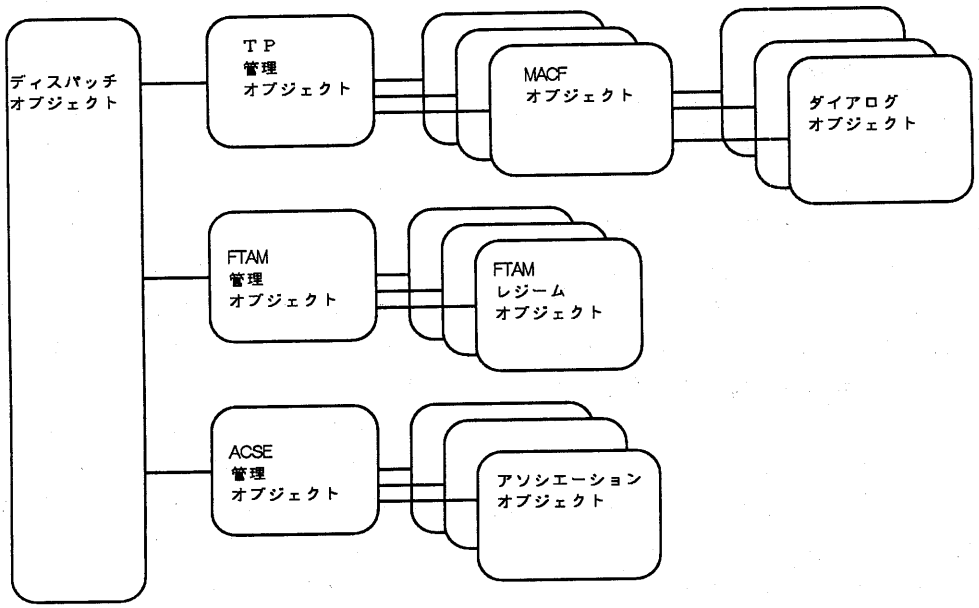

図３　ソフトウェア構成

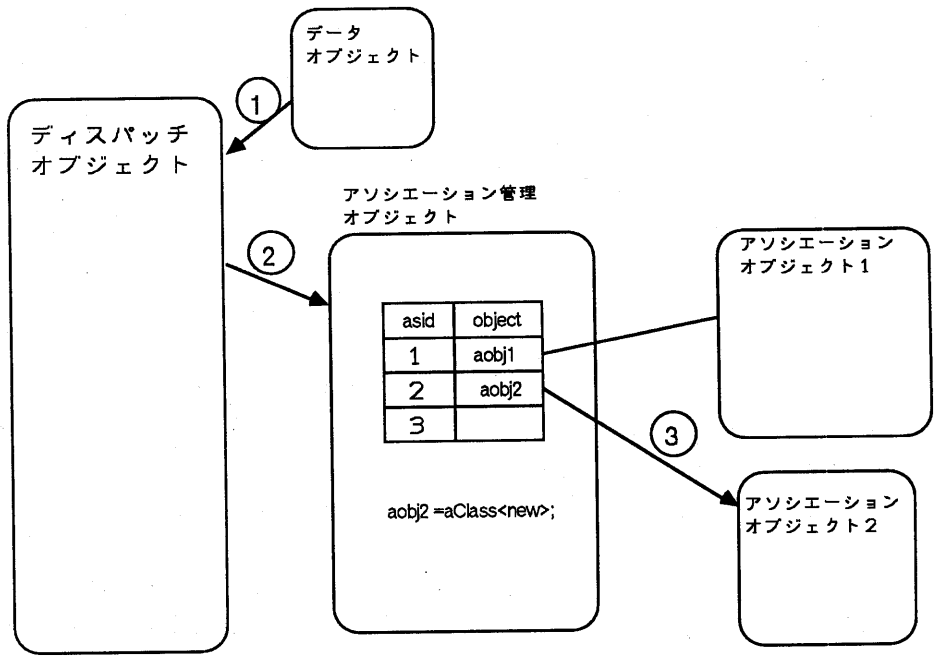

図4　プログラム動作

5．評価，考察

5．1　構成法

（1）オブジェクト指向言語による構成法と従来方法との違いは？

オブジェクト指向言語による構成法では，次の独立性が保たれる．

a)プロトコル間の独立性

複数のプロトコルを実装する場合，プロトコルを実現するモジュールを独立に作成し，それらを組み合わせてシステムを構築できる．

b)データ処理と状態遷移処理の独立性

プロトコル処理には，データ処理と状態遷移処理があるが，状態遷移処理の中で，データの構造を意識しなくてよい．

c)通信のインスタンスの独立性

プロトコルマシンをコネクション毎にオブジェクト（インスタンス）として生成することが可能で，コネクションをプロトコルマシンの中で意識しなくてよい．

以上の独立性により，オブジェクト指向による開発では，プロトコルマシンを構築しているモジュールの独立性が高まる．これによって，規格の変更や，仕様の拡張に対して柔軟に対処できる．

（2）オブジェクトをなににするか？

今回我々は，4．でのべたような構成としたが，プロトコルマシンオブジェクトのリファインメントとして，ステートをオブジェクトとするような方法も考えられる．

ステートテーブルの各ボックスを1つのオブジェクトとする．ステートの遷移は，次のステートオブジェクトを生成することでなされる．

こうすることで，プロトコルマシンオブジェクトのプログラムとしての分割が容易に可能となるが，状態遷移のたびに，オブジェクトの生成と消滅がおこるため，速度的に問題がでる可能性がある．

（3）継承機能の利用

継承機能は，データクラスの設計に利用した．データオブジェクトに共通の処理を親クラスに定義することによって，コーディング量の削減がはかれる．

5．2　オブジェクトとメモリ空間

同一プロセス内であれば，オブジェクトは問題なくモジュール間でやりとり可能である．

メモリ空間の異なるプロセスどうしオブジェクトをやりとりしようとする場合，データのコピーが必要となるが，このコピーの回数を減らすことが，処理速度の向上のポイントである．そこで，我々は，オブジェクト内に階層的メモリ領域の確保を許すことによって，これを実現した．つまり，オブジェクト内の，インスタンス変数領域に，メモリ領域を示すポインタを格納し，インスタンス変数領域との間に階層をもたせることにした．

これにより，インスタンス領域にデータを展開する必要性がなくなり，オブジェクト生成後，ポインタをつけかえる処理を行なうだけでよくなる．

ただし，オブジェクトを消去するときに，インスタンス領域の解放だけでなく，データ領域の解放もあわせておこなう必要がある．

プロセス間でデータの受渡しにコピーがおきないようにするためには，共有メモリを利用できるような動作環境が望ましい．

5．3　性能

（1）メモリサイズ

ＡＣＳＥプログラムサイズをISODE[9]のものと，我々のものを比較してみる．ISODE（the ISO Development Environment）は，UNIX上でＯＳＩ製品の普及をはかるためにソースレベルで配布されている通信ソフトウェアである．

ＡＣＳＥモジュールのステップ数については，我々のものが約４７００ステップで，ISODEもほぼ同じであった．

（2）実行速度

実行速度に関しては，ここでは，静的分析によって評価する．

実際のプログラムの中で，どの程度動的束縛と静的束縛が利用されているかを調べてみた．

表１に各クラス単位毎に集計した値を示す．

表１　静的束縛と動的束縛

クラス	総行数	静的	動的
ACSE管理クラス	372	8	23
ﾃﾞｰﾀｸﾗｽ	2506	25	36
ﾌﾟﾛﾄｺﾙﾏｼﾝｸﾗｽ	1878	176	142
合計	4756	209	201

この表からもわかるように，実際に，メッセージを利用してコーディングしている部分は，少なく，オブジェクト指向言語を利用しているが，コーディングはＣ言語を書いているのと変わらないことになる．

静的束縛は，Ｃ言語の関数に変換されるため，Ｃ言語による記述と実行速度上遜色ない．

オブジェクト指向言語で記述することによって，オーバヘッドとしては，①オブジェクト生成時のメモリの獲得，②動的束縛を利用した場合のメソッド探索のための時間がある．

動的束縛の箇所を実行時にどれだけ通るかを調べる必要がある．

5．4　開発方法

（1）デバッグ

ｓｕｐｅｒＣを利用して，プログラム開発を行うと，Ｃ言語のデバッガを利用する場合に，生成されたＣコードを見る必要がある．基本的には，生成されたＣコードとｓｕｐｅｒＣのコードは，ロジック部分ではほぼ１体１に対応している．

開発時のエラー発生箇所では，オブジェクト指向言語を利用していることに起因する誤りとして次のような誤りがある．

①動的束縛利用時，メッセージを探索した結果，該当メッセージがない．

②オブジェクトの解放を忘れる．

動的束縛時のエラーは，実行ライブラリからの

エラーメッセージによって解析する．

①の誤りについては，静的束縛を利用すると，コンパイル時にエラーが発見されるため，可能な限り，最終コーディングは静的束縛を利用することとした．

②の誤りについては，superCのデバッグオプションで，オブジェクトのトレースをとることによって解析できるようにした．

（2）プロトタイピング

継承機能と動的束縛を利用することによって，プロトタイピングが容易に行える．

初期の開発において，オブジェクトを何にするとよいかを試してみるような，試行錯誤的なプログラミングに有効である．

一旦，オブジェクトの構成が決まり，設計が確定した状況では，とりあえず動かしてみるという時に威力を発揮する．つまり，データオブジェクトのようなものは，細かいパラメタの設定を行わなくても，動作の確認だけはできる．パラメタの追加が，他のロジックに影響を与えないことから，この方法は有効である．

5．5　オブジェクト指向言語に対する要求

（1）パーシステントオブジェクトの実現

パーシステントオブジェクトは，オブジェクトのもつ情報を不揮発なメモリ領域に保持するオブジェクトである．

通信ソフトウェアそのものを記述するのには，必要ないと思われるが，ディレクトリ情報を格納したり，管理情報を格納したりする場合，パーシステントオブジェクトによって，オブジェクト指向で統一的に，プログラミングすることが可能である．

（2）プロセスオブジェクトの実現

プロセスオブジェクトは，プロセスとして動作するオブジェクトである．

仕事をいくつものプロセスに分割しておこなうアプリケーションの開発に，有効と考えられる．

例えば，トランザクション処理のアプリケーションなどに有効と考える．

しかし，オブジェクトが，プロセスであるため，オブジェクトへのメッセージは，プロセス間通信の機構を利用することになる．そのため，オブジェクトの単位が細かすぎると，プロセス間通信のオーバーヘッドが多くなり，好ましくない．

（3）静的クラスの実現

オブジェクトの生成時に，あらかじめ確保されているメモリ領域から，インスタンス変数の領域を割当てることを宣言するクラスを静的クラスとよぶ．

メモリ割当を要求するシステムコールに時間のかかる計算機に，通信ソフトウェアを移植する場合に，この静的クラスを利用することによって，オブジェクト生成時のオーバーヘッドを削減できる．

この静的クラスについては，すでにsuperCの拡張として機能追加されており，以下のような仕様となっている．

```
sclass conreqClass *2 { --- }
```

sclass が静的クラスであることの宣言，conreqClassはクラス名，2はインスタンス変数の領域を2つ分確保することを示している．

6．おわりに

オブジェクト指向言語を利用したＯＳＩ通信ソフトウェアの開発について，構成法，開発法，性能について，評価，考察をおこなった結果を報告した．

実際に開発したプログラムを分析することによって，オブジェクト指向であることを意識してプログラミングする部分は少なく，基本的には，C言語のコーディングであることがいえる．

オブジェクト指向言語で懸念されている処理速度の問題は，それほど大きな問題ではなく，オブジェクト指向は，設計に対してよい枠組みを与えたり，概念モデルと実装モデルのマッピングが容易に行えるといった，オブジェクト指向設計によるプログラミングのみとおしの良さからくる生産

性向上が重要な要素であると考える．

今後は，実行時の動作解析と性能測定を行なうことによって，さらに評価検討を進める予定である．

＜参考文献＞

［１］Pascoe,G.A.:Elements of Object Oriented Programming, Byte, pp137-144(1986).

［２］Goldberg,A. and Robson, D.:Smalltalk-80:The Language and Its Implementation, Addison-Wesley(1986).

［３］Brad, J.C:Object Oriented Programming An Evolutional Approach, Addison-Wesley(1986).

［４］Stroustrap,B.:The C++Programming Language, Addison-Wesley(1986).

［５］勝山，佐藤，中川路，水野：通信ソフトウェア向けオブジェクト指向言語ｓｕｐｅｒＣ，情報処理学会論文誌，Vol.30, No.2, pp234-241（１９８９）．

［６］中川路，勝山，芥川，水野：国際標準仕様に準拠したファイル転送プロトコルの実現と評価，情報処理学会論文誌，Vol.29, No.11, pp1071-1078（１９８８）．

［７］楠，田中，中川路，勝山，水野：ＯＳＩトランザクション処理システムの試作，情報処理学会マルチメディア通信と分散処理研究会，４５－５ (1990).

［８］ISO:Information technology - Open Systems Interconnection - Application Layer Structure,ISO /IEC 9545(1989).

［９］Marshall T. Rose:Tehs ISO Development Environment:User's Manual,(1990).