マルチメディア通信と分散処理 33－5
（１９８７．５．２１）

オブジェクト指向型言語ｓｐｉｃｅＣによる通信ソフトウエアの開発

勝山光太郎　佐藤文明　中川路哲男　水野忠則
三菱電機（株）　情報電子研究所

情報通信システムの発展に伴い、通信処理機能を実装するソフトウエア（通信ソフトウエア）の開発事例が増大している。このために、我々は通信ソフトウエアの効率的開発を目的として、オブジェクト指向によるソフトウエアの開発を行なうこととした。オブジェクト指向型言語は、プログラムの開発においては効率のよい反面、その実行速度に問題のある場合が多い。そこで我々は、実行効率が要求される通信ソフトウエアに対しても利用可能なオブジェクト指向型言語ｓｐｉｃｅＣを開発した。本稿では、このｓｐｉｃｅＣの特徴について述べ、さらに本言語をＯＳＩのＦＴＡＭ，ＡＣＳＥ，およびプレゼンテーションのプロトコルの実装に使用したのでその報告を行なう。

Development of communication software using object oriented language "spiceC"

Kotaro Katuyama　Humiaki Satoh　Tetsuo Nakakawaji　Tadanori Mizuno

Information Systems and Electronics Development Laboratory
MITSUBISHI ELECTRIC CORPORATION
5-1-1　OFUNA　KAMAKURA-CITY KANAGAWA　247　JAPAN

For the purpose of developing communication software, we aimed at an object oriented language. Generally speaking, an object oriented language is very useful for programming, but it has a problem of execution speed. So we developed the object oriented language named "spiceC" being available for communication softwares which require execution speed. In this paper, we report the characteristics of spiceC, and the implementation of OSI protocols(FTAM,ACSE,Presentation) using spiceC.

１．はじめに

通信システムの発展に伴い、それを実現する通信ソフトウエアの開発規模が増大してきており、通信ソフトウエアを効率的に開発する必要が生じている。筆者らは、通信ソフトウエアの効率的開発という観点から、オブジェクト指向による通信ソフトウエアの開発を検討し、オブジェクト指向型言語［１］の特徴と、通信ソフトウエアに要求される実行速度を十分に考慮したオブジェクト指向型言語spiceCを開発した［２］。本稿では、このオブジェクト指向型言語spiceCについて報告し、更に本言語を用いて開放型システム間相互接続（以下，ＯＳＩと略す）のプロトコルのうちＦＴＡＭ(File Transfer Access and Management)、ＡＣＳＥ(Association Control Service Element)、およびプレゼンテーションの実装を行なったのでその報告を行なう。以下、第２章では通信システムとオブジェクト指向について、第３章ではオブジェクト指向型言語spiceCについて、第４章ではspiceCを利用した通信ソフトウエアの開発について、第５章では評価／考察について述べる。

２．通信システムとオブジェクト指向

オブジェクト指向では、一般に次のことが言われている。

（１）モジュールの高い独立性によるソフトウエアの部品化

（２）継承機能による開発ステップの縮小化

（３）オブジェクトを用いることによる人間の思考パターンに近いプログラム設計

これらのオブジェクト指向による設計の利点を通信システムの開発に利用する観点からとらえる。

通信システムの参照モデルによるモデル化は、オブジェクト指向的発想に基づいて行われている。つまり、機能毎の階層化を行ない、層毎の独立性をもたせ、層間にはサービスプリミティブというメッセージを定義し、それによってエンティティが動作するモデルとなっている。従って、通信システムを記述するために、オブジェクト指向の概念により整理を図り、仕様を記述することにより自然な形で仕様記述からプログラミングが可能となる。

また、各層が果たす機能を大別すると、層管理とプロトコル処理に分けられ、特に層管理に関わる処理においては、各層に類似の処理が多く、同種モジュールの流用をはかることが適切である。このソフトウエアの流用という点においても、オブジェクト指向型言語の利用は有効と考えられる。

このように、通信ソフトウエアの開発にもオブジェクト指向の利点を利用し開発効率をあげるアプローチをとることが有効と考えられる。

しかし、通信システムの実装用の言語という観点から、従来のオブジェクト指向型言語では、その実行速度に問題のある点や、専用マシンでの実行環境が必要といった点があり利用できるものが少ないのが現状である。

３．オブジェクト指向型言語spiceC

３．１　spiceCの設計方針

上で述べたように、通信システムでは実行速度が要求されるために、通信システムの実装に適用可能な、十分な実行性能をもち、更にオブジェクト指向の利点を生かせるオブジェクト指向型言語を開発する必要がある。

このため、開発する言語の設計方針を次の様に設定した。

①オブジェクト指向の特徴をもつ

②通信システムの実装用の言語として十分な実行性能をもつ
③移植性やすでにＣ言語で開発されているプログラムとの親和性を考慮し、Ｃ言語の拡張とする。

３．２　spiceCの主な機能と特徴

上で述べた設計方針に基づき、spiceCには次の機能／特徴を持たせた。

①の設計方針から

（１）継承機能

オブジェクトが定義されているクラスからインスタンスを生成する際に、インスタンスは、上位クラス（スーパークラス）の性質を継承する。

（２）情報隠蔽

オブジェクトの内部状態を外部に見せない。外部からオブジェクトに対して作用するためにはメッセージを送る。

（３）メソッドの動的束縛、静的束縛

spiceCでは、オブジェクトに対して送られたメッセージに反応するメソッドを実行時に検索する動的束縛と、コンパイル時にメソッドを決定できる静的束縛の機能を提供している。

動的束縛の構文

オブジェクト名＜メソッド名：引数，・・・＞

静的束縛の構文

クラス名：：オブジェクト名＜メソッド名：引数，・・・＞

静的なメソッドの束縛は、メッセージを送る先のオブジェクトが既知であれば、行えるので、多くの場合に静的なメッソドの束縛が可能となる。

②の設計方針から

（４）実行速度

静的束縛を利用すれば、コンパイル時にメソッドの解決が行われるので、Ｃの関数と同等の実行速度となる。

（５）Ｃとの混合記述

spiceCは、Ｃとの混合記述が可能である。実行速度の問題等で、どうしても関数を使いたいような場合に、特別な書き方をしなくても、関数の定義をクラス定義と混合して行なうことを可能とした。

③の設計方針から

（６）分割コンパイル

開発時には通常、コンパイル・リンク・実行が頻繁に繰り返されるため、分割コンパイルは必要不可欠なものであり、ｓｃは分割コンパイル可能とした。

メソッドの解決をすべて実行時に行う場合は、コンパイルの際に他のクラスの情報が不要であり、クラス間での依存性に注意を払わなくてよい。これに対して静的なメソッドの束縛を用いる場合には、クラス間に複雑な依存性が存在するために、一部の変更が他のクラスに影響し、複数のファイルをコンパイルする必要が生じる。

その対策としてｓｃは、クラス定義の部分をコンパイル毎にクラスに関する情報を

クラスファイルとして出力している。ｓｃは、コンパイル中に静的なメソッド束縛形式のメッセージパッシング文を見つけると、それに対応したクラスファイルを読み込みメソッドの解決を行って、Ｃのソースコードを生成する。また、他のクラスに影響を与えるような変更が生じた時のみクラスファイルを更新するようにしており、変更の影響をクラスファイルの作成時刻で知ることができ、コンパイルを最小範囲で済ませることを可能とした。

（７）Ｃ言語のスーパーセット

ｓｐｉｃｅＣの言語仕様は、システム記述言語として広く用いられているＣ言語を包含する形としている。このため、ｓｐｉｃｅＣによる記述は、Ｃ言語の長所を損なうことなくオブジェクト指向のプログラムを作成することが可能となっている。

（８）移植性の高さ

ｓｐｉｃｅＣのプリコンパイラの生成コードはＣで記述されている。更に、プリコンパイラ自身もＣで記述されている。従って、Ｃコンパイラがありさえすればよく、マシンにもＯＳにも依存せず、開発環境から実行環境への移植も容易に行なうことができる。

３．３　ｓｐｉｃｅＣの言語仕様

ｓｐｉｃｅＣの主な構文を図１に示す。

```
〈MessageStatement〉 :- 〈Object〉 〈MessageBlock〉
〈Object〉 :- 'self' | 'super' | 〈ObjectIdentifier〉 | '(' 〈MessageStatement〉 ')'
〈ObjectIdentifier〉 :- 〈ObjectName〉 | 〈ClassSpecifier〉 '::' 〈ObjectName〉
〈ClassSpecifier〉 :- 〈ExternIdentifier〉
〈ObjectName〉 :- 〈ObjectIdentifier〉
〈MessageBlock〉 :- '<' 〈MethodIdentifier〉 '>' | '<' 〈MethodIdentifier〉 ':' 〈ArgumentList〉 '>' |
                   '<' 〈MethodIdentifier〉 '>' 〈MessageBlock〉 |
                   '<' 〈MethodIdentifier〉 ':' 〈Argument〉 〈MessageBlock〉 '>'
〈MethodIdenfier〉 :- 〈Identifier〉
〈ArgumentList〉 :- 〈Argument〉 | 〈ArgumentList〉
〈Argument〉 :- 〈Identifier〉
〈ClassStatement〉 :- 〈ClassDefinition〉 〈InstanceDeclare〉 〈MethodDefinition〉
〈ClassDefinition〉 :- 'class' 〈ClassSpecifier〉 〈SuperClassSpecifier〉
〈SuperClassSpecifier〉 :- 'root' 〈ClassSpecifier〉
〈InstanceDeclare〉 :- '{''}' | '{' 〈Declarations〉 '}'
〈MethodDefinition〉 :- '{' 〈MethodDefinitionList〉 '}'
〈MethodDefinitionList〉 :- 〈MethodStatement〉 | 〈MethodStatement〉 〈MethodDefinitionList〉
〈ExternalClassExpression〉 :- 'externclass' 〈ClassSpecifier〉
〈ReferenceExpression〉 :- 'reference' 〈ReferenceFileName〉
〈ReferenceFileName〉 :- 〈Identifier〉
```

図１　ｓｐｉｃｅＣの構文

４．通信ソフトウエアの開発への適用

４．１　開発したソフトウエアの全体構成

開発したソフトウエアの全体構成を図２に示す。

各プロトコル処理モジュールの構成は次節以降で述べる。

セションシミュレータ［３］は、セション層のサービスを模擬している。

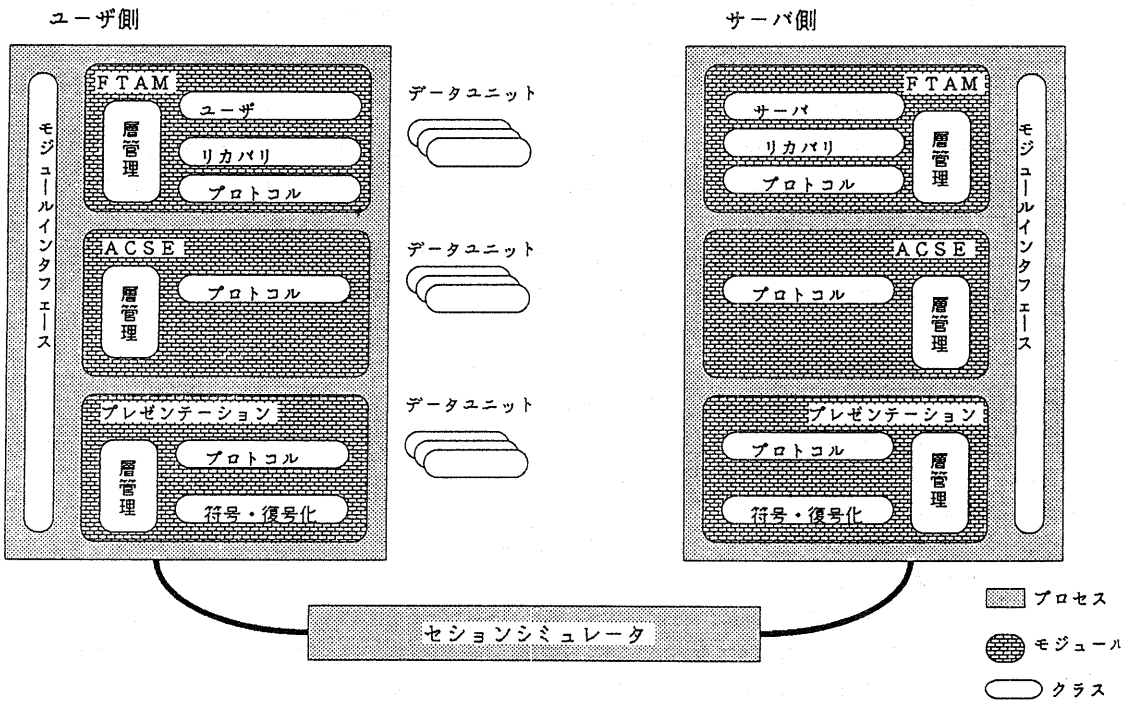

図２　ソフトウエアの全体構成

４．２　ＦＴＡＭの構成

ＦＴＡＭは、応用層に位置づけられファイル転送アクセスと管理を行なう。今回は簡単なファイル転送のみを実装した。

ＦＴＡＭは次のクラスから構成される。

（１）ユーザクラス

ユーザクラスは、ＦＴＡＭのユーザにＣ言語のライブラリインタフェースを提供するためのクラスである。

（２）リカバリクラス

リカバリクラスは、ユーザがサービスレベルとして高信頼ファイルサービスを要求している時に、リカバリ処理を実現するためのクラスであり、プロトコルクラスからエラー通知を受けそれがＦＴＡＭの範囲でリカバリの対象となるものであれば、ユーザクラスにはエラー発生を通知せずにリカバリ処理を試みる。

（３）層管理クラス

層管理クラスは、プロトコルクラスとリカバリクラスによって実現されるＦＴＡＭのプロトコルマシンのコネクションの管理を行なうクラスである。

（４）プロトコルクラス

プロトコルクラスは、ＦＴＡＭのプロトコルマシンを実現しているクラスである。上位層または下位層からのプリミティブの通知に対して状態を遷移させ、適当なプリミティブを上位層または下位層に対して通知する。

（５）サーバクラス

サーバクラスは、実ファイルシステムとＦＴＡＭで規定している仮想ファイルとのマッピングをとりながら実ファイルシステムにアクセスするクラスである。

（６）データユニットのクラス群

データユニットを扱うクラス群は、ＦＴＡＭのサービスプリミティブ及びプロトコルデータユニットに関するオブジェクトを生成するためのクラスである。

４．３　ＡＣＳＥの構成

ＡＣＳＥは、応用層で共通的に利用されるモジュールでアソシエーションの確立、解放を行うためのものである。

ＡＣＳＥは、次のクラスから構成される。

（１）層管理クラス

上下層とのインタフェースをもち、必要に応じてプロトコルクラスのオブジェクトを生成／消滅する。

（２）プロトコルクラス

コネクションの数だけ存在し、各アソシエーション対応のプロトコル処理を行なう。具体的には、状態をインスタンス変数として保持し、イベントに応じた処理をメソッドとして持つ。

（３）データユニットのクラス群

ＡＣＳＥのサービスプリミティブ及びプロトコルデータユニットの種類毎に存在し、パラメタの設定取り出しなどデータの形に依存した処理をメソッドとして持つ。プロトコルヘッダの付加はこれらのクラスのインスタンスの連鎖で表現される。

４．４　プレゼンテーションの構成

本モジュールは、プレゼンテーション層に位置づけられ、応用層からの抽象構文を、転送構文に変換するものである。

プレゼンテーションは、次のクラスから構成される。

（１）層管理クラス

上下層とのインタフェースをもち、必要に応じてコネクションクラスのオブジェクトを生成／消滅する。

（２）プロトコルクラス

コネクションの数だけ存在し、各コネクション対応のプロトコル処理を行なう。具体的には、状態をインスタンス変数として保持し、イベントに応じた処理をメソッドとして持つ。

（３）データユニットのクラス群

プレゼンテーションのサービスプリミティブ及びプロトコルデータユニットの種類毎に存在し、パラメタの設定取り出しなどデータの形に依存した処理をメソッドとして持つ。プロトコルヘッダの付加はこれらのクラスのインスタンスの連鎖で表現される。

（４）符号／復号化クラス

指定された符号／復号化規則に従い、抽象構文／転送構文間の変換を行なう。

４．５　spiceCによるコーディング例

spiceCによるコーディング例を、ＡＣＳＥでのアソシエーション確立を例に示す。

```
acse_manClass   root   { struct  connect  {
                                  int   avail;
                                  object  conObj;
                                 ----
                                 }
{
    -----------
   if ( prmtvObj(get_primID) == A_ASSOCreq) {
       -------
       connect[i].conObj = (object)acse_conClass(new);
       connect[i].conObj(issue:prmtvObj);

}
 acse_conClass   root   { int   state;
                          ----
                         }
 {
    -----
      switch(state) {
         case  AC_STA1: -----
                        ------
```

５．評価／考察

オブジェクト指向型言語spiceCを用いてＯＳＩ高位層プロトコル（ＦＴＡＭ，ＡＣＳＥ，プレゼンテーション）を実装した。

spiceCによる開発の利点として次のようなことがあげられる。

（１）一般的にオブジェクト指向の得意とするプロトタイピングについてその効果を確認

するために、ＦＴＡＭ，　ＡＣＳＥ、プレゼンテーションのコネクション確立フェーズについてプロトタイピングを実施した。クラス構成の検討等、初期の設計時の確認を行なうのに有効であった。
（２）プロトコルデータ単位、サービスデータ単位をオブジェクトとし、データの構造情報を隠蔽したために、設定するパラメータの変更あるいは追加等が容易に行える。
（３）コネクション毎にプロトコル処理のインスタンスが生成されるというモデルと親和性のあるソフトウエア構造とすることができた。さらにこの構造は各層とも同じであり、流用可能となる部分がある。
（４）プログラムが次の３つの部分に明確に分けられ、読みやすくメンテナンスしやすいソースプログラムとなった。

・コネクションの管理　－－－　規格外
・コネクション内のプロトコル処理　－－－　規格そのもの
・データ構造　－－－　インプリメント依存

今後の課題として、次の点に関して更に検討を加えて行く必要がある。
（１）プロセス間通信方式
各プロトコル処理モジュールをまとめて一つのプロセスとしているために、モジュール間でのオブジェクトの受け渡しは可能であるが、各プロトコル処理モジュールをプロセスに分けたて、各プロセスのメモリ空間が異なる場合に、メッセージ形式を定めデータを受け渡す必要があり、メッセージ形式とオブジェクトの形式の変換のオーバーヘッドが生じる。オブジェクト自身をプロセス間通信で伝達する方式の検討が必要である。
（２）オブジェクト指向型言語spiceCの各機種への移植
現在、spiceCは、４．２ＢＳＤ（ｏｎ　ＶＡＸ），ＳｙｓｔｅｍＶ（ｏｎＭＥ１００００，ＭＸ３０００）での動作を確認している。さらに、他機種への移植も検討中である。
（３）統合的通信ソフトウエア開発環境
spiceCのデバッグ環境の充実、セションシミュレータの改良や、試験環境との統合を進めていく予定である。

６．おわりに
通信ソフトウエアの効率的開発を目的として、オブジェクト指向型言語spiceCを開発した。さらに、それを利用してＯＳＩの上位層プロトコルの実装を行なった。spiceCを利用することにより、プログラムの設計が容易となり、読みやすさ、保守性が向上し、開発効率があがった。
今後さらに、このspiceCを用いて他の応用層プロトコルの実装を行ない評価を進めてゆく予定である。

＜参考文献＞
[１] Daniel H, H.Ingalls : "Design Principles Brhind Smalltalk", BYTE, pp286-298, Aug 1981.
[２] 水野、佐藤、中川路、勝山：「オブジェクト指向型言語ｓｐｉｃｅＣの言語仕様」電子情報通信学会全国大会（１９８７．３）
[３] 勝山、中川路、水野：「ＯＳＩ高位層プロトコル実層検証用セションシミュレータ－機能」第３３回情報処理学会全国大会