マルチメディア通信と分散処理　55－1
（１９９２．５．１５）

ソフトウェアモジュールの再利用化を考慮したOSI応用層ソフトウェアの実装方法

小花 貞夫　　杉山 敬三　　鈴木 健二
国際電信電話(株)　研究所

概要

近年、OSI(開放型システム間相互接続)の標準化が進捗し、FTAM(ファイル転送、アクセスと管理)、MHS(メッセージ通信処理システム)、ディレクトリをはじめ、OSI管理、RDA(遠隔データベースアクセス)やTP(トランザクション処理)など多くの応用層プロトコル標準が整備されつつある。OSI通信を利用する応用業務が高度化、多様化するにつれ、ひとつの応用業務で特定の単一応用層プロトコルを利用したり、また、複数の異なる応用層プロトコルを同時に利用するなど、さまざまな応用層プロトコルの利用形態に対応する応用層ソフトウェアの実装が必要となる。本稿では、このような応用層ソフトウェアを効率的に実装するため、ASE(応用サービス要素)の機能を再利用可能なソフトウェアモジュールとして実現し、それらを組み合せてさまざまな形態で利用される応用層ソフトウェアを体系的に開発可能とする手法を示した。具体的には、ASEソフトウェアモジュールの再利用化の方式として、ASEのソフトウェアモジュールを応用コンテクスト毎に異なる組合せで階層的に積み上げるスタック方式とALS(応用層構造)におけるSACF(単一アソシエーション制御機能)の機能をソフトウェアモジュールとして明示的に導入したSACF方式の2つの方式を提案し、それぞれ、1)ASEの組合せや依存関係の制御方法、2)ASEによるアソシエーションの共有方法、3)モジュール間の機能分担、および4)モジュール間インタフェースが異なることを示した。それぞれの方式を比較･評価し、スタック方式は、ひとつの応用層ソフトウェアに単一応用層プロトコルをサポートさせることを主目的とする場合や各ASEモジュール内部の作りに自由度をもたせ開発者がそれぞれ独自の方法により作成可能する場合に有効であり、また、SACF方式は、ひとつの応用層ソフトウェアに複数応用層プロトコルを同時にサポートさせることを主目的とする場合やASEモジュールに全く変更を加えずに再利用を図る場合に有効であることを示した。

Implementation Methodologies of OSI Application Layer Software by assembling Reusable Software Components of ASEs (Application Service Elements)

Sadao OBANA, Keizo SUGIYAMA and Kenji SUZUKI
KDD R & D Laboratories

Abstract

Standardization activities on Open Systems Interconnection (OSI) has been much progressed. As application of OSI communication become sophisticated, application layer software needs to handle single application layer protocol or multiple application layer protocols at a same time. Therefore, it is necessary to establish the implementation methodologies of application layer software in reusable manner. This paper discussed methodologies for the efficient implementation of OSI application layer software by assembling reusable software components of ASEs(Application Service Elements). To the methodologies, we have proposed the Stack approach and the SACF approach. In the Stack approach, application layer software is realized by the stack of different combination of ASE modules in each application context, and in the SACF approach, the software module of SACF(Single Association Control Function) is introduced, which dynamically controls the coordination of ASE modules depending on each application context. These approaches are different in 1)the control of coordination of ASEs, 2)the sharing of an association by ASEs, 3)the allocation of functions on each software module and 4)the module interfaces. The Stack approach is effective for implementing application layer software which supports single application layer protocol, and for enabling the introduction of the implementor's strategy to the design of each ASE module. On the other hand, the SACF approach is effective for implementing application layer software which supports multiple application layer protocols at a same time, and for realizing the reuse of ASE software modules without any modification.

1. はじめに

近年、OSI(開放型システム間相互接続)の標準化が進捗し、既に標準となっているFTAM(ファイル転送、アクセスと管理)、MHS(メッセージ通信処理システム)、ディレクトリに加え、OSI管理、RDA(遠隔データベースアクセス)やTP(トランザクション処理)など多くの応用層プロトコルの標準が整備されつつある。今後、応用業務が高度化、多様化するにつれ、ひとつの応用業務で特定の単一応用層プロトコルを利用したり、また、複数の異なる応用層プロトコルを同時に利用するなど、さまざまな形態で応用層プロトコルを利用することが予想される。このため、応用業務で必要となる応用層プロトコルをサポートする応用層ソフトウェアを効率的に実装することが重要な課題である。応用層ソフトウェアの実装については、これまでいくつか報告されているが[1][2]、応用業務からの応用層プロトコルの利用形態に応じたソフトウェアの体系的な実装方法を明確化したものはない。

本稿では、応用層機能のフレームワークを規定したALS(応用層構造)[3]におけるASE(応用サービス要素)の機能を、再利用可能なソフトウェアモジュールとして実現し、それらを組み合せて単一の応用層プロトコルをサポートしたり、また、複数の異なる応用層プロトコルを同時にサポートするなど、さまざまな形態で利用する応用層ソフトウェアを体系的に開発可能とする手法を論じる。具体的には、プレゼンテーション層以下の下位層と応用層とにおける構造上の相違を明確にし、応用層におけるASE機能のソフトウェアモジュール化の必要性を示す。ついで、応用業務からの応用層プロトコルの利用形態に応じたASE機能のソフトウェアモジュール化の方式として、スタック方式とSACF方式の2つの方式を提案する。さらに、両方式を比較・評価し、それぞれの適用領域を示す。

2. OSIにおける下位層と応用層の構造上の相違

OSIにおいては、プレゼンテーション層以下の層(以後、下位層と呼ぶ)と応用層は、次に示すように、構造上異なる特徴を持つ(表1)。

2.1 下位層における構造上の特徴

下位層では、各層の機能は、層毎にひかたまりの機能として規定され、それぞれ独立したプロトコルマシン(PM)と呼ばれる論理的機械により遂行される。n層のPMは常にn-1層のPMの上位にあるというように、各層のPM間の上位/下位の依存関係は不変である。各層には、独立のコネクションが存在し、上位/下位層のPMとのサービスプリミティブの交換を通して制御(確立/終了)される。また、n層のPMの状態遷移は、すべて上位または下位のPMからのサービスプリミティブ(ユーザデータとしてn層のPDUを含む)の受信により起こる。

2.2 応用層における構造上の特徴

応用層の機能は、ALS(応用層構造)[3]と呼ばれる論理モデルに基づき、さらに細分化された機能要素の集合として規定される(図1)。ALSによると、応用層は、ひとつ以上のアソシエーションを通して、相手システムの応用層と応用プロトコルデータ要素(APDU)を交換する。各アソシエーションの通信機能は、SAO(単一アソシエーションオブジェクト)が遂行する。SAOは、複数のASE(応用サービス要素)と呼ぶ基本機能とそれらの依存関係を制御するSACF(単一アソシエーション制御機能)からなる。ASEには、ACSE(アソシエーション制御サービス要素)、ROSE(遠隔操作サービス要素)、CCR(コミットメント、同時性と回復制御)ASE、FTAM ASE、RDA ASEなどがあり、それぞれのASE毎にプロトコルマシン(PM)が規定される。SACFの機能、つまり含まれるASEの組み合せや依存関係の制御機能は、応用コンテクストにより決定され、応用層プロトコルの種類(例えば、ファイル転送やメッセージ通信など)毎に異なる。また、複数のアソシエーションにまたがる機能は、MACF(複数アソシエーション制御機能)が遂行することとしている。

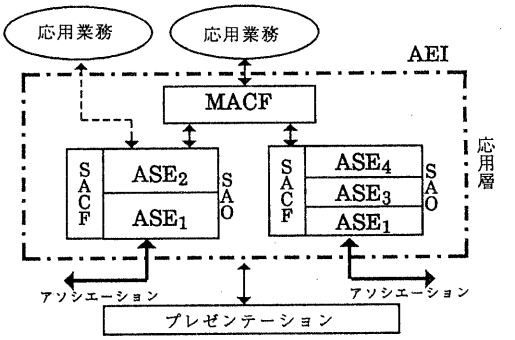


図1 ALS(応用層構造)

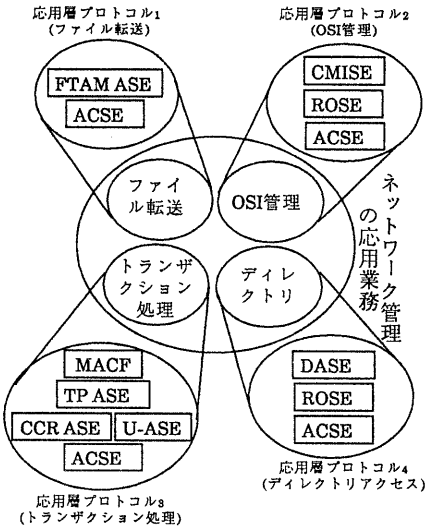
応用層では、一般に、下位層のように予め決められたASEの上位/下位の関係は存在しないが、特定の応用コンテクストに着目すれば、上位/下位の依存関係が存在すると考えられる。(ここで、ASE_iが提供するサービスをASE_kが使用する時、ASE_kはASE_iの上位のASEと呼ぶこととする)この依存関係は、下位層の場合とは異なり、上位/下位にそれぞれ複数のASEが存在する場合がある。例えば、ファイル転送の場合、FTAM ASEは、アソシエーションの確立/終了時に、ACSEとサービスプリミティブの交換を行うが、データ転送中は、直接プレゼンテーション層とサービスプリミティブの交換を行なう。つまり、下位のASE(または層)として、ACSEおよびプレゼンテーション層と依存関係にある。応用層には、下位層のコネクションに相当する概念としてアソシエーションがあるが、SAO内のASEがひとつのアソシエーションを共有する。ま

た、ASEによっては、PMの状態と他ASEからのサービスプリミティブ受信のみでは状態遷移が決定されず、例えば、ROSEやCCR ASEなどは、A-ASSOCIATEなどのアソシエーション制御サービスプリミティブを明示的に交換しないが、アソシエーション状態に依存した状態遷移を行う。

3. 応用層ソフトウェア実装における ソフトウェアモジュール再利用化の必要性

応用層の機能を構成する要素のうち、特に、SAOは独立のPMが規定されるさまざまなASEの組合せにより実現され、しかも、要求される応用層プロトコルの種類によりその組合せが異なる。例えば、図2は、ネットワーク管理という応用業務を遂行するために、OSI管理、ファイル転送、トランザクション処理およびディレクトリなど4種類の応用層プロトコルを複合的に使用する場合を示す。OSI管理は管理対象を監視、制御する操作や通知を行うため、ファイル転送はログ情報やプログラムの遠隔ローディングのため、また、トランザクション処理は複数の管理システム間で処理の一貫性を保つため、さらに、ディレクトリは被管理システムや管理対象のアドレス情報等を得るネームサーバアクセスのために利用することを想定している。各応用層プロトコルは、異なるASEの組合せにより実現され、例えば、ファイル転送ではACSEとFTAM ASEを、また、OSI管理ではACSE、ROSEおよびCMISEを用いる。ACSEやROSEなどは、各種の応用層プロトコルで共通に使用される。

このため、各種の応用業務で必要となる応用層プロトコルをサポートする応用層ソフトウェアの実装では、一般に各層のPMに対応するソフトウェアをモジュール化する下位層ソフトウェアの場合[1][4]と異なり、さらに応用層内部のASEも再利用可能なソフトウェアモジュールとして実現することが、応用層ソフトウェアの効率的開発に必要となる。



DASE : Directory Access Service Element
(readASE, searchASEおよびmodifyASEを含む)

図2 応用層プロトコルとASEの組合せ例

4. ソフトウェアモジュール再利用化に対する 2つの方式

ASEを再利用可能なソフトウェアモジュールとして応用層ソフトウェアを実装する方式として、スタック方式とSACF方式の2つの方式を提案する。これらは以下の特徴を持つ(図3)。

(1)スタック方式の特徴

ALSにおけるSAO内のSACF機能を実現するソフトウェアを明示的に導入せず、ASEのソフトウェアモジュールを、応用層プロトコルの種類(すなわち応用コンテクスト)毎に、異なる組合せで階層的(スタック)に積み上げる。

(2)SACF方式の特徴

SACFの機能を実現するソフトウェアモジュールを明示的に導入し、応用業務プログラムから要求される応用層プロトコルの種類に応じて、ASEのソフ

表1 下位層(プレゼンテーション層以下)と応用層における構造上の特徴

比較項目	下位層(プレゼンテーション層以下)	応用層
プロトコルマシン(PM)	層単位	ASE単位
上位/下位PM間の依存関係	不変 (n層PMは常にn-1層PMの上位)	応用コンテクスト毎にASEの組合せと依存関係が異なる
依存関係にあるPM数	上位/下位にそれぞれ1つ存在	上位/下位に複数存在可能
上位/下位PM間のコネクション(またはアソシエーション)の共有	なし (層毎に独立のコネクションが存在)	SAO内のすべてのASEがひとつのアソシエーションを共有
PMの状態遷移	上位/下位の層からのサービスプリミティブ受信により決定	ASEによっては、PMの状態と上位/下位ASEからのサービスプリミティブ受信のみでは決定されない

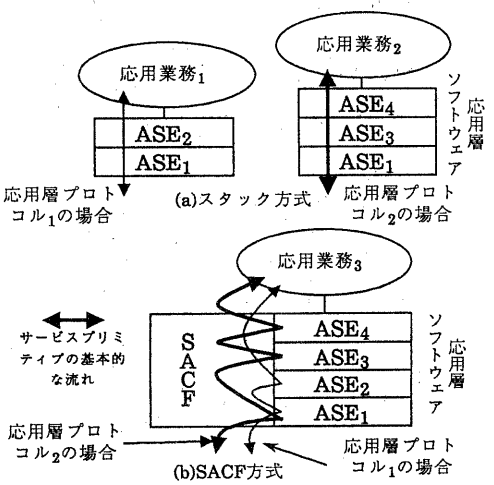


図3　スタック方式とSACF方式の特徴的相違

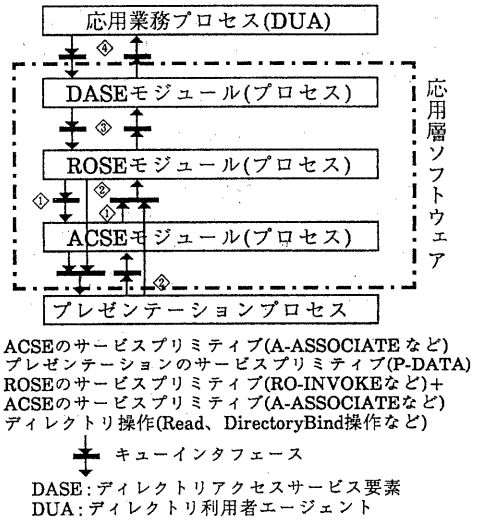


図4　スタック方式による応用層ソフトウェアの構成例(ディレクトリアクセスの場合)

トウェアモジュールの組合せと依存関係を動的に制御する。

これらの方式では、以下に述べるように、a)ASEの組合せとその依存関係の制御方法、b)SAO内のASEによるアソシエーションの共有方法、c)各モジュールの機能分担およびd)モジュール間インタフェースが異なる。

4.1. スタック方式

4.1.1 ASEの組合せと依存関係の制御方法

ASEの組合せとASE間の依存関係の制御は、スタック状に積み上げられたASEモジュール群の固定的な組合せと階層における上下関係により実現される。つまり、依存関係にあるASEモジュール間に予め設けたキューや関数などの通信パスを通して、サービスプリミティブ情報をルーティングする。なお、プレゼンテーションモジュールからASEモジュールへのプリミティブのルーティングは、プレゼンテーションPDU(PPDU)のユーザデータに含まれるAPDUのプレゼンテーションコンテクスト識別子に従って、プレゼンテーションモジュールが該当するASEモジュールへ行う。

4.1.2 ASEによるアソシエーションの共有方法

ACSEやFTAMなどアソシエーション制御に直接関与するASEは、A-ASSOCIATEなどのアソシエーション制御サービスプリミティブを交換し、アソシエーションの状態管理を行う。また、本来アソシエーション制御に直接関与しないROSEやCCR ASEなどのモジュールでも、アソシエーション状態を把握する必要があり、これらのASEモジュールにアソシエーション制御サービスプリミティブを経由させる。

図4に、スタック方式によるディレクトリアクセスのための応用層ソフトウェアの構成例を示す。ここでは、各ASEモジュールを別プロセスとしたマルチプロセスのソフトウェア構成をとっている。

4.1.3 各ASEモジュールの機能

各ASEモジュールの機能を表2に示す。特に、ROSEやCCR ASEなどでは、標準で規定される機能に加え、アソシエーション制御サービスプリミティブの処理やそれに関連するAEタイトル、AEI識別子、プレゼンテーションコネクション端点(PCEP)、応用コンテクスト、プレゼンテーションコンテクスト、初期トークン設定、初期同期点通し番号などアソシテーション確立時のパラメタ値ならびに現在のトークン位置や同期点通し番号などの情報を含むアソシエーション情報を管理する。

表2　各ASEモジュールの機能

●他ASEモジュール/プレゼンテーション/応用業務プロセスとの入出力制御
●サービスプリミティブ/APDUの解析(ASN.1復号処理含)
●ASEの状態遷移
●サービスプリミティブ/APDUの生成(ASN.1符号化処理含)
●ASE固有情報管理
●同一ASEにおけるAPDUの連結/分離
●アソシエーション情報管理(当該ASEで必要な情報のみ)

また、FTAM ASEのように、同一ASEに関する複数APDU(F-DATA)をひとつのプレゼンテーションユーザデータ(P-DATA)へ連結する規則が定義されるASEの場合、当該モジュールで連結/分離の処理を行う。

4.1.4 モジュール間インタフェース

ASEモジュールは、基本的にASEの標準で規定されるサービスプリミティブ情報を他のASEモジュールと交換する。

4.2 SACF方式

4.2.1 ASEの組合せと依存関係の制御方法

複数応用コンテクストを同時にサポート可能とするため、SACFモジュールは、プログラムの外部から予め与えられる、1)SACFモジュールに結合されるASEモジュールとそれらを呼び出すサブルーチン名(関数名)との対応付け、2)応用コンテクスト毎に使用するASEモジュールとそれらの依存関係(ASEモジュールの呼出し順序)、3)ASEモジュールとそれが扱う抽象構文名との対応付けに関する情報(応用コンテクストデータと呼ぶ)(図5)を用いて、ASEモジュールの種類に依存しない汎用的なサービスプリミティブのルーティング機能[5]を提供する。

```
%SACF-LINKED-ASE
    LINKED-ASE={ACSE, ROSE, DASE}   /* SACFに結合
                                    されるASE */
%SACF-AP-CONTEXT
    OBJECT-ID={2-5-3-1}   /*ディレクトリアクセス
                応用コンテクストのオブジェクト識別子 */
    DEPENDENCY={DASE, DASE, USER}   /* ACSE,
                ROSEの上位がDASE、DASE
                の上位が応用業務を示す */
%SACF-ABSTRACT-SYNTAX
    ASE-NAME=ACSE
    OBJECT-ID={2-2-1-0-1}   /* ACSEのAPDU抽象構文
                                のオブジェクト識別子 */
```

図5 応用コンテクストデータの記述例

4.2.2 ASEによるアソシエーションの共有方法

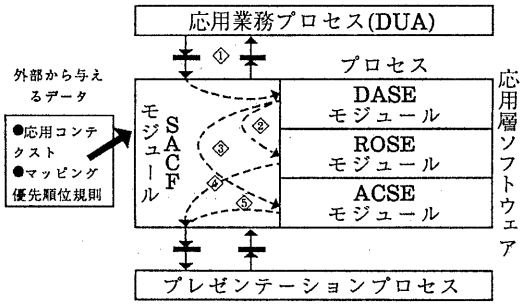
アソシエーション情報を各ASEモジュールで重複して管理せずに、SACFモジュールで一元的に管理し、ASEモジュールから参照や変更を可能とする。これにより、スタック方式のように、ROSEやCCR ASEなどへのアソシエーション制御サービスプリミティブのルーティングを行わずに、当該ASEでアソシエーション状態を把握できる。

図6は、SACF方式によるディレクトリアクセスのための応用層ソフトウェアの構成例を示す。ここでは、各種ASEモジュールをSACFモジュールのサブルーチン(関数)とし全体で単一プロセスとしている。また、図は、ユーザプロセスからRead操作を発行した場合のサービスプリミティブのルーティングを示す。

4.2.3 SACFモジュールとASEモジュールの機能

(1)SACFモジュールの機能

SACFモジュールの機能を表3に示す。各ASEモジュールから下位のASEモジュールやプレゼンテーションへのサービスプリミティブのルーティングは、要求/応答サービスプリミティブのパラメタとしてローカルに追加規定したASEの識別子に従って行う。一方、上位のASEや応用業務プロセスへのルーティングは、応用コンテクストデータにおけるASEモジュール間の依存関係や抽象構文/ASEモジュールの対応付けに従って行う。SACFモジュールは、プレゼンテーションから受信した指



①:ディレクトリ操作(Read、DirectoryBind操作など)
②:ROSEのサービスプリミティブ(RO-INVOKEなど)
③:ACSEのサービスプリミティブ(A-ASSOCIATEなど)
④:プレゼンテーションのサービスプリミティブ(P-DATA)
⑤:プレゼンテーションのサービスプリミティブ
(P-CONNECTなどのコネクション制御サービス)

キューインタフェース

図6 SACF方式による応用層ソフトウェアの構成例
(ディレクトリアクセスの場合)

表3 SACF/ASEモジュールの機能分担

モジュール	機能
SACF	●プレゼンテーション/応用業務プロセスとの入出力制御 ●応用コンテクストに従うサービスプリミティブのルーティング(外部データとしてASEの組合せと依存関係の情報を与える) ●異なるASEのAPDUの連結/分離 (外部データとしてマッピング優先順位規則の情報を与える) ●アソシエーション情報管理(一元管理)
ASE	●サービスプリミティブ/APDUの解析 (ASN.1復号処理含) ●ASEの状態遷移 ●サービスプリミティブ/APDUの生成 (ASN.1符号化処理含) ●ASE固有情報管理 ●同一ASEにおけるAPDUの連結/分離

示/確認のサービスプリミティブ(ユーザデータとしてASEのAPDUを含む)を該当ASEモジュールにルーティングするため、プレゼンテーションコンテクスト識別子と抽象構文(すなわちASEモジュールを示す)の対応付けを管理する。SACFで一元管理するアソシエーション情報は、後述するように特定のASEが生成/削除する。

また、TPなどの応用層プロトコルで、異なるASE のAPDU 、 例 えばTP-BEGIN-DIALOGUE-RI、C-BEGIN-RI、U-ASE(ユーザASE)-APDU、C-PREPARE-RIのAPDUを、単一のプレゼンテーションサービス(P-SYNC-MINOR)のユーザデータに連結するため、APDUの連結/分離処理をSACFモジュールで行なう。連結範囲の指示は、SAOのユーザ(応用業務やMACF)が明示的に行なう。連結/分離処理では、外部からデータとして与えられるマッピング優先順位規則に従って、サービスプリミティブの組替えやルーティングの待ち合わせを行う。

(2)ASEモジュールの機能

ASEモジュールの機能を表3に示す。SACFモジュールにおけるアソシエーション情報の生成(削除)は、アソシエーション確立(終了)を伴うサービスプリミティブを最初(最後)に受信したASEモジュールが行う。例えばファイル転送の場合、起動側では、FTAM ASEがF-INITIALIZE要求サービスプリミティブを受信した場合に、また、応答側では、ACSEがAARQ APDUを含むP-CONNECT指示サービスプリミティブを受信した場合にアソシエーション情報を生成する。

各ASEモジュールは、状態遷移などを実行するために必要なASE固有な管理情報(例えば、FTAMにおける機能単位、PMの状態、アクセスファイル名など)を管理する。この情報は、アソシエーションに関連する情報としてSACFモジュールで一元管理するアソシエーション情報に結合するが、SACFモジュールでは、各ASE固有情報のデータ構造を関知できないため、アソシエーション情報が削除される(アソシエーションが終了する)際に、SACFモジュールからの指示によりASEモジュールが削除する。

4.2.4　モジュール間インタフェース

SACFモジュール/ASEモジュール間の関数呼出しには、a)SACFモジュールがASEモジュールを呼出す関数に加えて、b)ASEモジュールがSACFモジュールのアソシエーション情報を生成/削除/参照/変更する関数、c)ASEモジュールがASE固有の管理情報を生成/削除する関数、d)ASEモジュールがプリミティブを送信する関数など約20種類の関数が必要となる。主な関数を図7に示す。また、a)およびd)の関数により交換されるサービスプリミティブは、各ASE標準で規定されるサービスプリミティブに加えて、SACFモジュールがルーティングするために必要なアソシエーション識別子、サービスタイプ(要求、指示など)、ASE識別子などの共通ヘッダ情報を含む。

```
unsigned int ACSEMain(SACFASSTBL*,PRIM*)
    機能：  SACFからASE(ACSE)の呼出(SACF→ASE)
    引数：  アソシエーションテーブルへのポインタ
            プリミティブへのポインタ

unsigned int ASSetAssTable(SACFINF*)
    機能：  アソシエーション情報の生成(ASE→SACF)
    引数：  設定する情報内容へのポインタ
    戻り値：  TRUE(登録実行)/FALSE(既登録)

unsigned char *ASGetAssInf(unsigned short,unsigned
        short*)
    機能：  アソシエーション情報取得(ASE→SACF)
    引数：  取得するアソシエーション情報の種類(コード),
            取得したアソシエーション情報長
    戻り値：  取得情報へのポインタ

void ASEntryASETable(void*)
    機能：  ASE固有情報のアソシエーション情報への結合
            (ASE→SACF)
    引数：  ASE固有情報へのポインタ

void ASWritePrimitive(PRIMLNG*)
    機能：  プリミティブ送信(ASE→SACF)
    引数：  送信プリミティブへのポインタ

                    •
                    •
```

図7　SACF/ASEモジュール間の主な関数

5.　スタック方式とSACF方式の比較検討

両方式の比較結果を表4にまとめる。

(1)ソフトウェアの再利用性

スタック方式では、ASEモジュールの組合せを変更して別の応用層ソフトウェアを実現する際に、既存ASEモジュールへの変更が必要な場合がある。例えば、CCR ASEとROSEを用いる応用(OSI管理にTPを導入した場合)と、ROSEを使用しCCR ASEを使用しない応用(CMIP)やROSEを使用しないでCCR ASEを使用する応用(TP)では、ROSEモジュールやCCR ASEモジュールへの上位/下位のASEモジュールからのプリミティブルーティングが異なり、モジュール間の通信パスの変更が必要となる。また、(2)で述べる複数応用コンテクストを同時サポートする際に、共有されるASEモジュールは、応用コンテクストに従って上位のASEモジュールとの依存関係を制御するように変更する必要がある。SACF方式では、ASEモジュールへの変更なしに、応用コンテクストデータ等の外部からSACFモジュールに与えるデータの変更と、SACFモジュールとASEモジュールのオブジェクトの再リンクで対応可能で、スタック方式よりソフトウェアの再利用性が高い。

(2)複数応用層プロトコルのサポート

複数の応用層プロトコルを同時にサポートするには、いづれの方式でも、単一の応用プロトコルに必要なASEモジュールの組合せからなるソフトウェアを応用層プロトコル毎に生成する方法が考えられるが、全体メモリサイズが大きくなる欠点がある。そこで、スタック方式では、図8(a)に示すように共通に使用されるASEモジュールを実行時に共有する形でスタックを構成する。SACF方式では、図8(b)に示すように、応用業務で必要とするすべての応用層プロトコルで使われるASEモジュールを単一のSACFモジュールにリンクし、応用コンテクストデータやマッピング優先順位データの追加を行うだけで実現できる。また、同一ユーザプログラムインタフェースから複数の応用プロトコルを利用可能である。

(3)ASEモジュール作成における自由度

スタック方式では、各ASEモジュールは外部インタフェースとして、基本的にASE標準で規定されるサービスプリミティブに従えば、内部の作りに自由度が大きい。一方、SACF方式では、サービスプリミティブに加えてSACFモジュールとの協調動作を行うための関数使用の制約等に従ってASEモジュールを作成する必要がある。

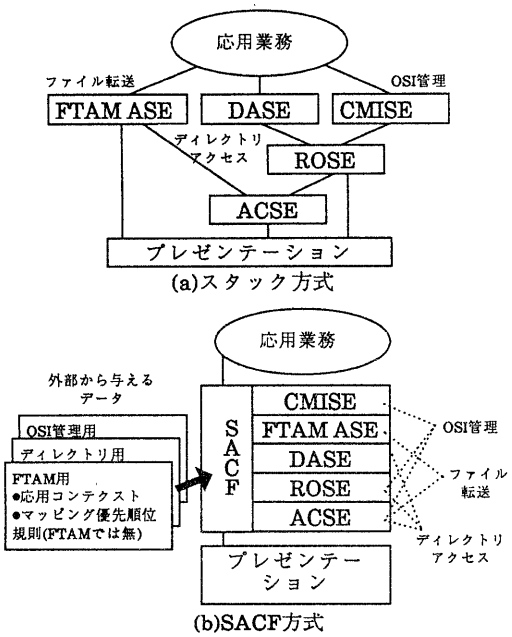


図8　複数応用層プロトコルのサポート

(4)ソフトウェア規模

スタック方式では、すべてのASEがアソシエーション制御サービスプリミティブの解析およびアソシエーション情報の管理を行う必要がある。SACF方式では、上記(3)で述べたインタフェース上の制約がある反面、アソシエーション情報の管理はSACFモジュールが一元的に行うため、各ASEで行う機能が軽減され、しかもASE間の依存関係を全く意識する必要がないため、ASEモジュールのソフトウェア規模が小さくなる。実際にスタック方式とSACF方式により作成したACSEとROSE(プレゼンテーションマッピング)のソフトウェア規模(C言語)を比較するとSACF方式の方が1.5~2.1Kステップ程度少ない。(表5)

表5　ソフトウェア規模の比較(Kステップ)

モジュール	スタック方式	SACF方式
ACSE	8.1*	6*
ROSE	7.1*	5.6*
SACF	-	5

注)*：ASN.1符号化/復号処理部分(約4Kステップ)を含む

この傾向は、他のASEの場合も同様と考えられる。しかしながら、単一の応用層プロトコルをサポートする場合、必要なASEモジュールの数は比較的少なく、例えば、ファイル転送では、ACSEとFTAM ASEのモジュールのみであるため、SACF方式ではSACFモジュール(約5Kステップ)も含む全体のソフトウェア規模が、スタック方式よりも大きくなる可能性が高い。逆に、複数の応用層プロトコルをサポートする場合には、ASEモジュールの数が多くなるため、SACF方式の方が全体のソフトウェアをコンパクトに実現できる。

6. 考察

(1)スタック方式、SACF方式のいづれの方式でも、応用業務が単一応用層プロトコルを利用する形態および複数応用層プトコルを利用する形態に対応する応用層ソフトウェアを実現できる。特に、スタック方式は、ASEモジュールの数が少ない単一応用層プロトコルのソフトウェアを実現する場合のソフト

表4　スタック方式とSACF方式の比較

比較項目	スタック方式	SACF方式
ソフトウェアの再利用性 (応用コンテクストの変更に対する柔軟性)	既存ASEモジュールへのプログラム変更が必要となる場合がある。(モジュール間通信パスの再定義や複数ASEに関するAPDU連結処理の変更など)	既存ASEモジュールへのプログラム変更はない。1)応用コンテクストデータおよびマッピング優先順位規則データの変更、2)SACFモジュールとASEモジュール(オブジェクト)の再リンクのみでよい。
複数応用層プロトコルサポート	必要となるASEモジュールをスタックで構成し、実行時のASEモジュールの共有を図る。共有されるASEモジュールは、応用コンテクストごとに他ASEモジュールとの依存関係を制御する必要がある。	必要となるASEモジュールをSACFモジュールにオブジェクトでリンクし、複数の応用コンテクストデータとマッピング優先順位規則データを追加定義するのみでよい。
ASEモジュール作成における自由度 (部品間インタフェースにおける制約)	基本的にASE標準で規定されるサービスプリミティブに従うことにより、ASEモジュール内部の作りに制約は無い。ただし、ROSEやCCR ASEなどには、アソシエーション制御サービスプリミティブの処理を追加する必要がある。	SACFモジュールとの協調動作で必要となる約20の関数の使用制約に従う必要がある。(SACFモジュールからASEモジュールの呼出し、アソシエーション情報の生成/削除/参照/変更、ASE固有の管理情報の生成/削除、プリミティブの送信などの関数)
ソフトウェア規模	SACF方式より、ASEモジュールのソフトウェア規模大きい。(全てのASEモジュールで重複してアソシエーション情報管理が必要である。ASEモジュール間の依存関係を意識する必要がある。)	スタック方式より、ASEモジュールのソフトウェア規模小さい。(ASEモジュールでのアソシエーション管理は必要ない。ASEモジュール間の依存関係も意識する必要がない。)

ウェア規模のコンパクト化ならびにASEモジュール内部の作りにおける自由度の大きさの点で有利である。一方、SACF方式は、多くのASEモジュールを必要とする複数応用層プロトコルサポートの容易性とソフトウェア規模のコンパクト化、ならびにASEモジュールに全く変更を加えずに再利用できる点で有効である。このため、パソコンなど一般に主記憶サイズや処理能力に制約のある計算機環境で、特定の単一応用層プロトコルをサポートする端末応用を実現する場合にはスタック方式により、また、高機能ワークステーションや汎用機などで複数の応用層プロトコルを同時に扱うような高度な応用業務を実現する場合には、SACF方式により応用層ソフトウェアを開発するのが肝要と考えられる。

(2)筆者らが、これまでに作成したFTAM[6][7]、ディレクトリ[8]の応用層ソフトウェアは、スタック方式によるアプローチとして、また、RDA[9]、TP[10]の応用層ソフトウェアは、SACF方式によるアプローチとして位置付けることができる。特に、本稿では、スタック方式におけるASEモジュールのスタックを実現する方法として、各ASEモジュールをプロセスとしたマルチプロセス構成を示したが、シングルタスクOS環境のパソコン上で動作するように、各ASEモジュールをサブルーチン(関数)としそれらを順に呼出すメインルーチンとともに全体で単一プロセス構成とした上記[7]のアプローチもスタック方式と考えられる。同様に、上位ASEモジュールから下位ASEモジュールを階層的に関数呼出しし全体で単一プロセス構成とするアプローチ[1]もスタック方式と考えられる。

(3)本稿では、ASEモジュール単位でのソフトウェアの再利用を述べたが、標準化の進捗に伴い修正や機能拡充(機能単位の追加など)などの変更が加えられる場合に、これらに対する影響を極小化するため、プリミティブの生成/解析、APDUの生成/解析、状態遷移など各ASEモジュール内部の機能もさらにモジュール化し、例えば、APDU定義の改善修正が状態遷移やサービスプリミティブの生成/解析に影響しないようにすることも肝要である。

(4)本稿では、SAO(単一アソシエーションオブジェクト)におけるASEのモジュール化に焦点をあてて論じたが、MACF機能のモジュール化についても、いづれの方式でもASEモジュールと同様に扱うことができる。つまり、スタック方式では、SAOのソフトウェアの上にMACFモジュールを積み上げ、また、SACF方式に場合には、SACFモジュールにMACFモジュールを結合する。

7. おわりに

本稿では、多様化する応用層ソフトウェアを効率的に実現するために、ASE(応用サービス要素)の機能を、再利用可能なソフトウェアモジュールとして実現し、それらを組み合せてさまざまな応用層プロトコルの利用形態に対応する応用層ソフトウェアを体系的に開発可能とするための手法を論じた。ASEソフトウェアモジュールの再利用化の方式として、スタック方式とSACF方式の2つの方式を提案し、それぞれ、1)ASEの組合せや依存関係の制御方法、2)ASEによるアソシエーションの共有方法、3)モジュール間の機能分担、および4)モジュール間インタフェースが異なることを示した。いづれの方式でも、応用業務が単一応用層プロトコルを利用する形態および複数応用層プトコルを利用する形態に対応する応用層ソフトウェアを実現できるが、特に、スタック方式は、ひとつの応用層ソフトウェアに単一応用層プロトコルをサポートさせることを主目的とする場合や各ASEモジュール内部の作りに自由度をもたせ開発者がそれぞれ独自の方法により作成可能とする場合に有効である。一方、SACF方式は、ひとつの応用層ソフトウェアに複数応用層プロトコルを同時にサポートさせることを主目的とする場合やASEモジュールに全く変更を加えずに再利用を図る場合に有効である。方式の差による処理の効率や多重度への影響については、OSの種類などソフトウェアの実行環境に依存し、今後の検討課題としたい。

最後に日頃御指導いただくKDD研究所 小野所長、浦野次長に感謝します。

参考文献

(1) Rose M. T., et al. : The ISO Development Environment : User's Manual, Version 7 (1991.7).
(2) Nakagawaji T., et al. : Object-Oriented Implementation of OSI Application Layer Protocols, Trans. on Comm. IEICE, Vol.E74, No.11 (1991.11).
(3) ISO / IEC 9545, Application Layer Structure (1989).
(4) 鈴木, 加藤, 浦野 : OSIトランスポートおよびセション・プロトコルの実装, 情処学会論文誌, Vol.29, No.12, pp.1180-1192 (1988.12).
(5) 杉山,小花,鈴木:OSI応用層プロトコルソフトウェアの部品化のための応用コンテクスト制御方式の提案,44回情処全大 (1992).
(6) 小花, 加藤, 鈴木 : OSIプレゼンテーション, ACSE, FTAMプロトコルの実装と評価, 情処学会論文誌, Vol.30, No.7 (1989.7).
(7) 小花, 堀内, 井戸上, 加藤, 鈴木, 三上, 野沢 : パソコン用FTAMソフトウェアの実装と評価, 信学全大 (1991.3).
(8) 小花, 西山, 鈴木 : リレーショナルアプローチによるOSIディレクトリのDIB(ディレクトリ情報ベース)の実装と評価, 情処学会論文誌, Vol.32, No.11, pp.1488-1497 (1991.11).
(9) 小花, 堀内, 杉山, 鈴木 : OSI RDAプロトコルソフトウェアにおけるトランザクション機能の拡充, 43回情処全大 (1991.10).
(10)杉山, 小花, 鈴木 : OSIトランザクション処理(TP)プロトコルソフトウェアの設計, 情処学会, DPS-47-11 (1990.9).