組込みシステム向けCORBAの開発と評価

伊賀徳寿[†] 中本幸一[†] 奥山嘉昭[†]
佐藤直樹[†] 檜原弘樹[††]

企業向けシステムでは，システムで提供する機能をオブジェクトとしてモデル化し，相互接続性を保証したCORBAにより実装するシステム構築が広まってきている．オブジェクト指向設計による生産性向上とソフトウェアの下位層の隠蔽による移植性の向上を目的として，組込みシステムから構成されるネットワークシステムでもCORBAは有効であると期待されている．しかし，CORBAは企業システム向けに設計されており，ハードウェアリソースやソフトウェアリソースの制約の厳しい組込みシステムでは適用が難しいところがある．本論文では，企業システムとは直接には接続しない組込みシステムネットワークに特化した機能を有する組込みシステム向けCORBA（Embedded CORBA）の設計思想，仕様，実装とその評価について述べる．最近，情報家電をはじめとする組込みシステムでは，機器の提供する機能の追加拡張を行う機能が求められている．Embedded CORBAは，このためにオブジェクトの動的追加機能を具備している．

Development and Evaluation of CORBA Suitable for Embedded System Network

Norihisa Iga,[†] Yukikazu Nakamoto,[†] Yoshiaki Okuyama,[†]
Naoki Sato[†] and Hiroki Hihara[††]

In enterprise system, CORBA becomes popular, in which the functionalities the enterprise systems are modeled as objects and the objects are implemented. CORBA is also expected to be effective in embedded system because high productivity of object-oriented design and high portability due to hiding lower software layers. Applying CORBA to the embedded systems is difficult because CORBA is basically designed for the enterprise systems and not suitable for the embedded system whose resources are very restricted in hardware and software. Embedded CORBA is suitable for embedded system networks, which are not directly connected to the enterprise systems. We describe the design philosophies, the specification and design and the implementation and evaluation of Embedded CORBA. Recently, extending functionalities in the embedded systems are required. Embedded CORBA also provides the functionalities to add objects dynamically.

1. はじめに

企業向けシステムでは，システムで提供する機能やサービスをオブジェクトとしてモデル化し，相互接続性を保証した高位の通信インタフェースで接続する分散システムによる構築が広まってきている．このソフトウェアプラットフォームとしてCORBA（Common Object Request Broker Architecture）が標準化され，利用されてきている．機器に組み込まれる組込みシステムでもネットワーク化が進みつつある．情報家電や自動車，工業用ネットワークがその一例としてあげられる．組込みシステムでのネットワーク（以下，組込みシステムネットワークと呼ぶ）においても，CORBAの適用が有望であるとされている．その理由として以下が考えられる．第1に，組込みシステムネットワークでは通信プロトコルが多彩であり，これを隠蔽したシステム構築が望まれている．下位プロトコルの設計変更が上位層に影響を与えないようにするためである．CORBAはまさしくこの機能を提供している．第2に，組込みシステムネットワークを構成する組込みシステムのソフトウェア開発でも高生産性が要求されており，これを支援する設計方法としてオブジェクト指向設計が注目されつつある．オブジェクト指向設計で設計されたオブジェクトをCORBA上に実装することにより，効率的なシステム構築が可能になる．第3に，CORBAが提供する相互接続機能により，企業

† NEC ネットワーク開発研究本部
NEC Network Development Laboratories
†† NEC 東芝スペースシステム（株）
NEC TOSHIBA Space Systems, Ltd.

システムとの接続が必要な場合にその接続が容易になると考えられる．

組込みシステムは，PC などに比べて CPU 性能が低い，搭載されるメモリ容量が小さい，2 次記憶がないなどのハードウェアリソースでの制約が厳しい[1]．これにともなって，ソフトウェアリソースに関しても，OS の小型化，プログラムやデータの小型化などの制約がある．利用可能なリソースが限定されたシステム向け CORBA として，Minimum CORBA[2),3)] が定義されている．Minimum CORBA は CORBA との相互接続性を維持しつつ，リソースが限定されたシステム向けの CORBA の機能をサブセット化したものである．しかし，Minimum CORBA は企業システムとの接続性を考慮しているため，仕様上プログラムサイズ，データサイズが大きくなるところがあり，リソース制約の厳しい組込みシステムには実装が難しいところがある．したがって，Minimum CORBA は企業システムとは直接には接続しないような組込みシステムネットワークでは，必ずしも適切であるとはいえない．

一方，組込みシステム向けの CORBA の研究は主にリアルタイムシステムの領域において行われている．Gokhale らは分散組込みリアルタイムシステム向けの CORBA の研究を行っている．文献 4) では，小型のメモリフットプリントの stub/skelton を生成する IDL コンパイラを開発し評価している．また，文献 5) では，マルチメディアシステムを対象として，CORBA の通信プロトコルにあたる IIOP のボトルネックを測定し，最適化を行っている．Lankes ら[6)] は，Scalable Coherent Interface と呼ばれる制御用ネットワーク向けの Real-time CORBA を開発し，リアルタイム性能を評価している．Chan ら[7)] は，IC カード内のアプリケーションプログラムをホストマシンからカードリーダ/ライタを通じて通信を行う場合に，CORBA を使って IC カード内のアプリケーションをオブジェクトと見なしてアクセスする手法を提案している．また，組込みシステム向け CORBA の製品としては，Objective Interface Systems 社の ORBExpress GT と IONA 社の Orbix/E などが知られている．ORBExpress GT[8)] は，CORBA2.5 仕様の機能と組込みシステム開発のための機能を提供している．一方，Orbix/E[9)] は，CORBA2.3 仕様のサブセットとなる機能を提供している．これらは，大規模な組込みシステム，もしくは IC カードのような限定された領域での研究であり，本論文で対象としている組込みシステムネットワーク向けの CORBA の研究はまだなされていない．

また，近年，情報家電をはじめとする組込みシステムでは，利用後に機器の提供するサービスの追加拡張を行う機能が求められている．これにともない，組込みシステム向けの CORBA においても，オブジェクトの追加機能が必要になってくる．

本論文では，企業システムとは直接には接続しない組込みシステムネットワークに特化した機能を有し，機器の提供するサービス拡張のためのオブジェクトの動的追加機能を具備した組込みシステム向け CORBA（Embedded CORBA）について述べる．2 章では Embedded CORBA の設計思想，仕様について述べる．3 章では，Embedded CORBA を μITRON 仕様☆のリアルタイム OS 上の実装，およびその性能評価に述べる．さらに，Embedded CORBA の衛星内ネットワークシステムの開発への適用を 4 章で報告する．

2. Embedded CORBA

筆者らは，Embedded CORBA と呼ぶ組込みシステム向けの CORBA 仕様を定義した．Embedded CORBA は，Object Managemnt Group（OMG）で標準化された CORBA 機能をベースとし，情報家電などのリソースの制約がきわめて厳しい機器を対象とした CORBA である．Embedded CORBA は Minimum CORBA 仕様に対して必要最小限の機能を実装しており， 最小限の相互接続性とポータビリティを保ちながら，コンパクト化および処理オーバヘッドの削減を行っている．さらに，運用中にオブジェクトを追加する機能，および Embedded CORBA の利用するリソースを制限する機能などの追加を行っている．なお，Real-Time CORBA[10)] が提供するリアルタイム機能に関しては，Embedded CORBA では考慮していない．

2.1 Minimum CORBA

Embedded CORBA のベースとした，Minimum CORBA について説明する．Minimum CORBA は，組込みシステムなどのリソースが限られた実行環境での使用を前提としている，機能を限定した CORBA のサブセットである．Minimum CORBA と CORBA の機能比較を**表 1** に示す．ただし，Object Request Broker（ORB）間の移植容易性と相互接続性を保つための機能は確保されている．ベースとなる CORBA は，CORBA2.2 である．

Minimum CORBA では，実行中にリソースを動的

☆ TRON は “The Real-time Operating system Nucleus” の省略語である．μITRON は “Micro Industrial TRON” の省略語である．

表 1 CORBA，Minimum CORBA，Embedded CORBA の比較
Table 1 Comparison of CORBA, Minimum CORBA and Embedded CORBA.

CORBA の機能	Minimum CORBA	Embedded CORBA
DII/DSI	無	無
Stub / Skeleton	全機能	同期通信のみサポート
Interface Repository	Repository ID と Typecode information のみを提供	Repository ID のみ提供
POA Manager	POA の生成と活性化のみ提供	POA の生成と活性化のみ提供
POA Adaptor Activator	無	無
POA Servant Manager	無	無
POA Policies and Lifecycle	動的処理に関連するポリシーはない	ポリシーを規定値に限定
Marshalling	全機能	C 言語の基本型とこれから構成される配列，構造体，共用体
Dynamic Any	無	無
GIOP/IIOP	全機能	Request，Reply と Close Connection のみを提供

DII: Dynamic Interface Invocation, DSI: Dynamic Schelton Interface
POA: Portable Object Adapter
GIOP: General Inter-ORB Protocol, IIOP: Internet Inter-ORB Protocol

に獲得する機能は含まれていない．これは，リソースを動的に獲得する機能は必要となるリソース量の予測ができず，システム設計が難しくなるためである．たとえば，Dynamic Invocation Interface（DII）ではインタフェース情報獲得時と，そのインタフェースに対応したリクエストの作成時に必要なメモリは，インタフェースごとに異なり，実装されているメモリが限られた組込みシステムでは，必要なメモリを予測することは困難である．

Minimum CORBA は相互接続性が確保されているため，CORBA と Minimum CORBA が混在した分散システムの構築がブリッジなしに容易に可能となる．つまり，リソースが十分確保されている機器については CORBA を使用し，リソースが比較的少ない機器については，Minimum CORBA を使用して分散システムを構築することが可能である．

2.2 設計方針

Embedded CORBA は，Minimum CORBA の仕様をもとに，比較的処理能力の低いプロセッサを搭載する，リソース制約の厳しい機器を対象としている．このために，以下に示す設計方針に従って設計および実装を行った．

P1 ローカルネットワークに適したコンパクト化
情報家電ネットワークや車載ネットワークをはじめとする組込みシステムネットワークは，企業向けネットワークシステムと直接接続されることはなく，ローカルなネットワークを構成していると考えることができる．想定するネットワークシステムを**図 1** に示す．ここでは，企業向けシステムとはブリッジを介するものと考えている．こうした組込みシステムネットワークの特性を利用して，Minimum CORBA の機能をさらに絞り込み，低処理能力や小容量のメモリの組込みシステムにも適用可能なように，コード，データのコンパクト化，処理の最適化を行う．

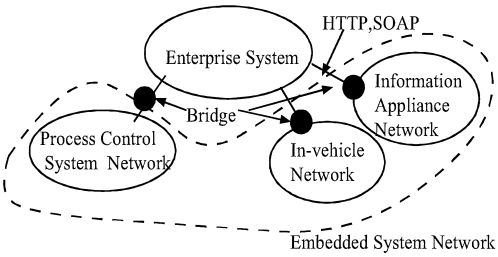

図 1 企業システムネットワークと組込みシステムネットワーク
Fig. 1 Enterprise system network and embedded system network.

P2 動的な拡張機能の提供
近年，組込みシステムにおいてもリリース後に機能を追加，変更するために，ソフトウェアの動的追加機能が要求されている．このために Embedded CORBA では，オブジェクトの追加，削除を行う機能を提供する．

P3 実行管理機能の追加
組込みシステムではハードウェアリソース，ソフトウェアリソースにおいて制約が大きい．こうした環境下で Embedded CORBA が利用可能となるように，Embedded CORBA の実行を管理する機能を追加する．

P4 Minimum CORBA への下位互換性
将来，組込みシステムのハードウェアが強化され，

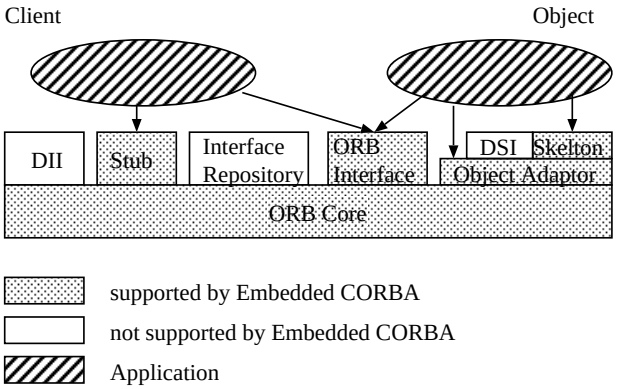

図 2 Embedded CORBA の構成
Fig. 2 Structure of Embedded CORBA.

Minimum CORBA が搭載可能となった場合に，アプリケーションソフトウェアの移行を考慮し，Minimum CORBA への下位互換性を確保する．

P5 処理オーバヘッドの削減
比較的処理能力の低いプロセッサを搭載した機器の負荷を軽減するために，データ通信量の削減，内部処理でのコピー回数の削減，および一度使用した情報の再利用を行う．

P6 C 言語マッピング
組込みシステム用アプリケーションのコーディング時に使用されるプログラミング言語は，C 言語およびアセンブラが大半を占めている[11]．こういった背景から，Embedded CORBA は，C 言語マッピングを採用する．

2.3 仕様と設計

本章では，先に述べた設計方針により決定された Embedded CORBA の機能仕様と設計の要点を述べる．以下では，まず CORBA の機能概要を紹介した後，Embedded CORBA の仕様とそのように仕様を定義した背景を説明する．**図 2** に Embedded CORBA の構成を，表 1 に Embedded CORBA と，CORBA，Minimum CORBA の機能差分をそれぞれ示す．

(1) ORB Core
CORBA では ORB Core はオブジェクトの基本的な表現方式と通信方式を提供する部分であり，ネットワーク間の相互接続性を保持するためのプロトコルである GIOP などの規定から構成される．Embedded CORBA では GIOP において使用できるメッセージタイプは Request，Reply，CloseConnection の 3 つに限定した（**表 2**）．これは以下の理由からなる．

- メッセージ CancelRequest は Stub/Skeleton では使用されないことから削除した．これは (2) で述べるように，Embedded CORBA では Stub/Skeleton は同期通信しか提供しないため，リクエストのキャンセルは不要であるからである．
- 組込みシステムネットワークでは，ネットワークに接続された機器がその機器に固有の機能を提供することが多く，機能分散型システムを構成している．したがって，機能を提供するオブジェクトの存在する機器は固定であると考えられるため，オブジェクトの場所を探す，あるいはオブジェクトの場所を確認するメッセージである LocateRequest，LocateReply は使用しない．このような Embedded CORBA を使用したクライアントおよびサーバの振舞いは次のようになる．もしクライアントがサーバにリクエストを送信する場合，クライアントはサーバとなるオブジェクトの存在確認（LocateRequest）をしないで，リク

表 2 Embedded CORBA における GIOP/IIOP 機能
Table 2 GIOP/IIOP functionalities in Embedded CORBA.

メッセージタイプ	送信元	クライアント側受信時動作	サーバ側受信時動作
Request	C	2	1
Reply	S	1	2
CancelRequest	-	2	2
LocateRequest	-	2	3
LocateReply	S,4	2	2
CloseConnection	S	1	2
MessageError	-	2	2

C: クライアント，S: サーバ，
1: メッセージ処理，2: メッセージ廃棄，
3: メッセージを廃棄し，UNKNOWN を LocateReply として送信する，
4: LocateRequest の応答としてつねに UNKNOWN を送信

エストをサーバに送信する（表 2 参照）．このとき，クライアントはサーバからのリクエストに対応するリプライに含まれる例外情報でサーバにリクエストが届けられたかどうか，また，正常終了か異常終了かを示す情報を受け取る．なお，サーバが LocateRequest を受信した場合，クライアントに UNKNOWN を LocateReply として返す．サーバとなる機器が一時的に停止している場合など，サーバとなる機器にリクエストが届けられない場合でも Embedded CORBA を使用したクライアントは，リプライとしてつねに例外情報を受け取ることを可能としており，クライアントプログラムでこの状況に対処することが可能である．

- 組込みシステムネットワークでは静的に構成要素が決定されており，正しいメッセージを発行するものと考えられることから，MessageError メッセージを削除した．

組込みシステムネットワークにおいては文字コードはあらかじめ固定できるものとし，送信可能な文字コードは ISO-8859-1（Latin-1）のみで十分と考えた．このためコードセット間の変換は不要となる．Embedded CORBA は企業向けネットワークシステムと直接接続しない方針で設計されており，上述のように，いくつかのメッセージタイプを削除し，文字コードを固定している．よって，他 CORBA を利用した企業向けネットワークシステムとの完全な相互接続を実現するためには，図 1 に示すように，他の CORBA と Embedded CORBA との差分を吸収するブリッジを使用する必要がある．

（2） Stub/Skeleton

Stub はオブジェクトに対する操作を呼び出す手段を提供し，Skeleton は操作に対応するメッセージを受け取り，対応する操作を呼び出す機能を提供する．Embedded CORBA では，Stub において同期通信のみをサポートした．これは組込みシステムネットワークにおいて，クライアント/サーバ間でのメッセージ通信で必ず応答が必要であると考えられるためである．クライアント/サーバ間の通信において，Embedded CORBA では，リクエストあるいはリプライの内容として，機器を制御するためのコマンド，また，そのコマンドに必要となるデータを想定している．このような通信では，コマンドあるいはデータが通信相手に届けられたかどうか確認できることが重要であると考えられる．つまり，クライアントからのリクエストを正常に受け取れたことをサーバはクライアントに通知する必要がある．このような信頼性を確保すべき通信には，組込みシステムネットワークでは非同期通信よりも同期通信が適切と考えられる．非同期通信により実現することも可能であるが，アプリケーションプログラムの処理が複雑となる傾向にあり，それにともなうプログラムサイズの肥大化が予想される．よって，リソースの少ない機器を対象にした組込みシステムネットワークには，同期通信が適切であると判断した．なお，応答を必要としない非同期通信の 1 つとしてストリーミングが考えられる．このような応用には Embedded CORBA を使用せず，UDP など下位の通信インタフェースを利用することを想定している．

マーシャリング/アンマーシャリング処理において高速な CPU を有しメモリ容量が大きいシステムでは，マーシャリング用に，あるサイズのメモリ領域を確保した後，順次マーシャリングするデータを格納し，このメモリ領域で不足する場合はさらに大きなメモリ領域を確保した後，先のメモリ領域からコピーしてマーシャリングを続けるという方法が単純である．しかし，この方法ではメモリコピーが発生しやすく，メモリ制約が厳しく，処理能力に低い CPU を有する組込みシステムでは適当でない．Embedded CORBA では，以下のような API を用意し，上記のようなことが起こらないようにした．以下の API で `XXX` には，`octet`，`boolean` などの型名が入る．

- `MRSHL_index_XXX`：マーシャリング時に対象となる型を Common Data Representation（CDR）の形式に変換する際に，CDR でのデータアライメントを計算し，指定された型のデータが格納される送信バッファの先頭からインデックス値を返す．この API をマーシャリング対象のデータに対して順次適用することによって，マーシャリングに必要な送信バッファサイズをあらかじめ求めることができ，この大きさのメモリ領域を一度で確保することができる．これにより上述したような不必要なメモリコピーの発生とこれによるオーバヘッドをなくすことができる．
- `MRSHL_put_XXX`：指定された型のデータを CDR に変換し格納する．
- `MRSHL_get_XXX`：本 API を実行したマシンのエンディアンに合わせて，CDR 形式の受信バッファから指定された型のデータを取り出す．

次に Embedded CORBA で新たに追加した機能を説明する．これらは CORBA では仕様として記載されていないものである．

（1） リソース管理

Embedded CORBA が実行時に必要なリソースを使いすぎると機器内の他のプログラムが動作しなくなる場合がある．また，この逆も起こりうる．こうしたことを防ぐために，リソース数の上限を設定する機能を Embedded CORBA は提供する．管理可能なリソースは以下のとおりである．

- コネクション数：サーバタスクあたりの TCP コネクション数，クライアントで使用するコネクション数
- タスク数：サーバとなるタスク数を規定する．これには，(2) で述べる自動起動可能なタスク，動的登録可能なタスク，自動起動しないタスクが含まれる．
- POA 数，サーバント数
- メモリ量：通信バッファ，テンポラリメモリなど Embedded CORBA 実行中に動的に確保されるメモリの量

Embedded CORBA を使用する組込みシステムでは，使用する総リソース量をあらかじめ予測し，上述したコンフィグレーション情報の設定を行う．そして，Embedded CORBA の初期化 API である `ECORBA_init` で ORB Core に引き渡され，リソース利用時の上限監視を ORB Core は行う．

システム稼動中に予期せぬリソース消費が発生して，リソース不足となる場合，Embedded CORBA は必ず例外を返す．クライアントからのリクエストの処理中にサーバがメモリ不足となった場合でも，サーバは例外情報を含むリプライをクライアントに必ず返す．これは，サーバが緊急時リプライ用メモリ領域をあらかじめ確保しており，メモリ不足の場合でもクライアントへのリプライ送信を可能としているからである．また，実際にメモリ不足となった場合の対処方法として，クライアントまたはサーバとなるアプリケーションは，Embedded CORBA から渡されるリクエストあるいはリプライメッセージのメモリ領域の開放，また，オブジェクトリファレンスの開放（オブジェクト呼び出しの終了処理）を行うことでメモリ領域を新たに獲得可能となる．

(2) オブジェクト動的登録/自動起動

情報家電から構成されるネットワークなどでは，機器稼動中に機能の追加，削除が必要となる場合がある．このため，Embedded CORBA では稼働中の機器へのオブジェクトの登録/削除機能，および登録されたオブジェクトを自動起動（活性化）させる機能を提供している．

登録/削除可能な単位として，あるまとまった単位のオブジェクトを考える．これは，追加される機能は複数のオブジェクトで提供されるものと想定し，機能の追加，削除にともない，機能を提供する複数のオブジェクトの単位で登録，削除する手段を提供するのが望ましいと考えたからである．このオブジェクトとオブジェクトのポリシーを規定する POA の集まりを，オブジェクトスイーツと呼ぶことにする．なお，機器へのオブジェクトスイーツのダウンロード方法は多様であるため，Embedded CORBA では規定していない．ただし，オブジェクトスイーツのダウンロード時にサービスタスク管理テーブル（Service Task Management Table）にオブジェクトスイート識別子（Object Suite id）とオブジェクトスイート情報（Object Suite Info）が設定されるものとしている．サービスタスク管理テーブルの情報は後に述べる自動起動サーバが起動するオブジェクトを見い出すために利用される．オブジェクトスイーツ内の POA を生成し，オブジェクトを POA に登録させ，オブジェクトスイーツ内のオブジェクトを実行するタスクをサービスタスク（Service Task）と呼ぶ（**図 3**）．サービスタスクは，自サービスタスクのうちのオブジェクトへのリクエストを受信し，サービスタスクのコンテクストでオブジェクトへのリクエストを実行する．

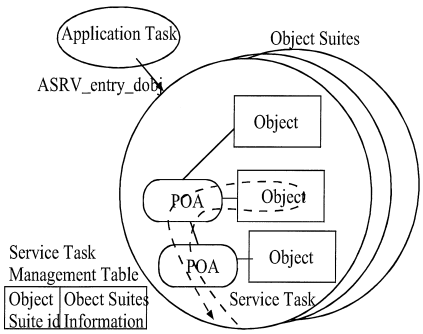

図 3 サービスタスクと登録オブジェクト
Fig. 3 Service Tasks and Registered Objects.

動的オブジェクトの管理のために以下の API を用意した．

- `ASRV_entry_dobj`：本関数によって，すでに稼働している CORBA 関連機能を用いたシステムに，サービスタスクが登録される．サービスタスクは対応するオブジェクトスイーツの POA の生成，オブジェクトの登録を行う．
- `ASRV_cancel_dobj_id`：関数 `ASRV_entry_dobj` によって登録されたサーバタスクのサービス停止と削除を行う．

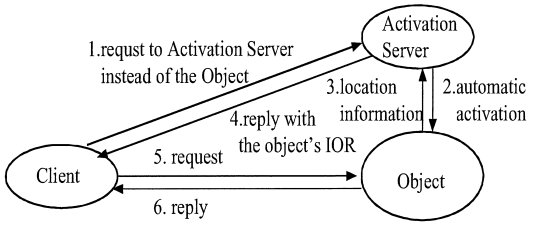

図 4 オブジェクトの自動起動
Fig. 4 Automatic activation of objects.

次に動的に登録したオブジェクトに対する参照，アクセス方法を述べる．オブジェクトの参照は Interoperable Object Reference（IOR）を利用する．Embedded CORBA では，IOR の中で以下の実装依存情報を利用して，動的オブジェクトへのアクセスを行う．

- オブジェクトスイーツ識別子
- オブジェクト識別子
- 自動起動サーバ（後述）の IP アドレス，ポート番号

以下では，上記の IOR を使って動的に決定される IP アドレス，ポート番号を取得する仕組みを説明する（**図 4**）．オブジェクトスイーツ識別子，およびオブジェクト識別子は，設計時に，もしくはネーミングサーバで実行中に割り当てることができるものとしている．Embedded CORBA では IP アドレス，ポート番号を取得のためにオブジェクトの動的な登録，起動を管理する自動起動サーバ（Activation Server）を設けている．クライアントは，本来オブジェクトに対して出すリクエストと同じリクエストを自動起動サーバに対して出す（図 4 の 1，以下同図を参照するものとする）．自動起動サーバは，まずサービスタスク管理テーブルを使って対象のオブジェクトを起動し (2)，IP アドレス，ポート番号を自動起動サーバに返す (3)．次に，自動起動サーバはこの情報をもとに新たに IOR を生成する．自動起動サーバは，リプライとして生成された IOR を付与した “LOCATION_FORWARD” の状態を返す (4)．クライアントは自動起動サーバから返ってきた IOR を基にオブジェクトリファンレスを生成し，対象オブジェクトにリクエストを出す (5，6)．

(3) コネクション管理

クライアントとサーバ間の TCP コネクションの確立時のオーバヘッドを削減するための機能である．クライアントがあるサーバとの間で，リクエストを繰り返し発行する場合，最初のリクエスト発行時にソケットをオープンしてコネクションを確立し，2 回目以降のリクエストでは，最初のリクエスト発行時に確立されたコネクションを再利用してリクエストの発行を行う．Embedded CORBA のコネクションを保持することによる他プログラムへの影響を最小限にするために，コネクション情報を保持できる数はリソース管理で設定できるようにしている．コネクション確立の情報を保持する管理データは，双方向リンクリストで実装される．もし，その設定値以上のコネクション情報の作成が必要な場合は，利用時期が一番古く，現在リクエスト処理に使用されていないコネクション情報から破棄する．それと同時にソケットをクローズする．リソース管理で設定した値以上のコネクションを確立しなければならない場合，最も古いコネクションを廃棄し，新たにコネクションを確立する必要があり，オーバヘッドが発生する．このオーバヘッドを発生させないようにするために，コネクション数を適切にリソース管理に設定する必要がある．すなわち，設定するコネクションの数は，稼動時に予想される通信相手の数以上の数を設定しなければらない．これは，組込みシステムネットワークという閉じられたネットワーク内では比較的容易であると考えられる．

表 3 Embedded CORBA の API 比較
Table 3 Comparison of Embedded CORBA APIs.

Minimum CORBA	Embedded CORBA
40	14

3. 実装と評価

Embedded CORBA と Minimum CORBA の仕様の側面からの比較のために，API 数を比較した．その結果を**表 3** に示す☆．この表から分かるように API 数では Minimum CORBA の 1/2 以下であり，コンパクトな仕様になっているものと考えられる．なお，Embedded CORBA の API 仕様の概要を付録に示す．

Embedded CORBA を μITRON 仕様のリアルタイム OS と，その上で稼動する TCP/IP の上に実装した．使用したハードウェアは PC/AT 互換機（400 MHz CPU 搭載），使用したコンパイラは，GCC v2.7.2.1 である．実装規模を**表 4** に示す．Embedded CORBA のプログラムサイズは約 25 KB である．この大きさは，筆者らが開発した小型組込みシステム用 μITRON 仕様のリアルタイム OS [12] と同程度であり，組込み

☆ 両者とも，言語マッピング関係の API，および Embedded CORBA は 2 章で述べた機能拡張に関する API を除く．

システムネットワークを構成する機器にも適用可能な大きさであると考えられる．次に，実行時メモリ消費量を測定した．long 型データの引数と戻り値を持つオペレーションを実行させたときのクライアント側で必要なメモリ量を測定し，この値は約 4 KB であった．

Embedded CORBA の処理時のオーバヘッドとコネクション管理の有効性の調査を目的とした Embdded CORBA の性能測定を行った．送信データは 1 KB と 1 MB の 2 つの場合とした．Ethernet（10BASE-T）で接続された PC/AT 互換機 2 台に Embedded CORBA をインストールし，**図 5** に示すような測定点間の処理時間を計測した．クライアントとサーバとの通信を初めて行う場合，つまり，TCP コネクションの確立を行う通信を 10 回測定した値の平均値を 1 回目のリクエストの処理時間とした．そして，クライアントとサーバとの通信で 2 回目以降，つまり TCP コネクションの確立はすでに行われている状態での通信を 1,000 回測定した値の平均値を 2 回目のリクエスト処理時間とした．測定結果を**表 5** に示す．表 5 において，$t_{i,i+1}$ は測定区間 i，$i+1$ で要した時間 $(i = 1, 2, \ldots, 7)$，$t_{1,8} = \sum_{1=1}^{7} t_{i,i+1}$ はクライアントからサーバにリクエストを出してリプライを受信するまでの時間，$t_{1,4} = \sum_{1=1}^{3} t_{i,i+1}$，$t_{5,8} = \sum_{1=5}^{7} t_{i,i+1}$ はそれぞれリクエスト送信とリプライ送信の時間である．この表から以下のことを述べることができる．

- Embedded CORBA の処理オーバヘッドは，$(t_{1,2} + t_{3,4} + t_{5,6} + t_{7,8})/t_{1,8}$ で求められる．この値は，1 回目のリクエストでは全体の最悪値 50%を示すが，2 回目のリクエスト以降は全体の 10%未満であり，実用に耐えうる値であると考えられる．
- コネクション管理により，1 回目のリクエスト処理に比べて 2 回目のほうがはるかに高速に処理されている．特に，クライアント側のコネクション処理では 1/10 以下に減少している．これはコネクション管理が有効であることを示している．
- 送信データが増加した場合には，コネクション管理の有効性は減少してくる．これは処理オーバヘッドのうち，増大したデータのコピーオーバヘッドなどが支配的になるためと考えられる．

表 4 Embedded CORBA のサイズ
Table 4 Size of Embedded CORBA implementation.

機能 モジュール	コード ライン数	オブジェクト サイズ
ORB Core†	3.0 KL	5.5 Kbyte
マーシャリング	0.7 KL	1.2 Kbyte
GIOP	1.6 KL	3.7 Kbyte
POA	2.5 KL	5.8 Kbyte
コネクション管理	1.2 KL	3.3 KByte
オブジェクト自動起動	1.5 KL	3.7 Kbyte
システム依存	4.5 KL	1.7 KByte
合計	15.0 KL	24.9 KByte

†:マーシャリングと GIOP を除く．

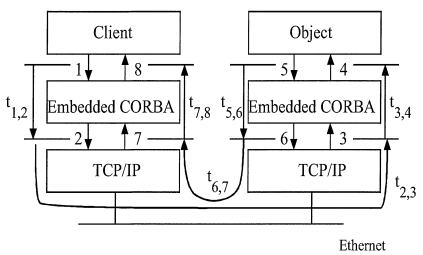

図 5 性能測定点
Fig. 5 Performance monitoring points.

4. 応　　用

第 5 著者らのグループでは，Embedded CORBA を利用して人工衛星に搭載されるバス機器の統合ソフトウェアのプロトタイプを開発し，評価した[13),14)]．人工衛星に搭載される機器の概要を**図 6** に示す．この図において，データ処理系システム（Data Management System，DMS）は，各種データ処理を行う．姿勢制御システム（Attitude and Orbit Control System，AOCS）は外部からのイベントに対してリアルタイムで応答するリアルタイムシステムである．DMS，AOCS は人工衛星にとって必須の装置であり，バス機器と呼ばれる．この両者は別々の設計手法で開発されてきた．しかし，人工衛星システムも低コスト化，短

表 5 Embedded CORBA の性能
Table 5 Performance of Embedded CORBA implementation.

回数	送信 データ サイズ	$t_{1,8}$	$t_{1,4}$	$t_{5,8}$	$t_{1,2}$	$t_{3,4}$	$t_{5,6}$	$t_{7,8}$
1 回目	1 KB	17.6	12.9	4.58	8.10	0.30	0.10	0.29
2 回目	1 KB	9.48	5.11	4.55	0.18	0.29	0.09	0.27
1 回目	1 MB	8510	4190	4220	120	113	121	130
2 回目	1 MB	8510	4200	4210	113	113	119	129

単位はミリ秒．

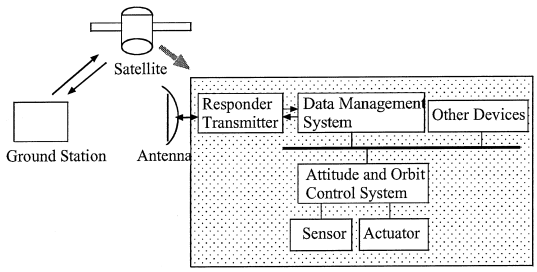

図 6 人工衛星内ネットワーク
Fig. 6 In-Satellite Network.

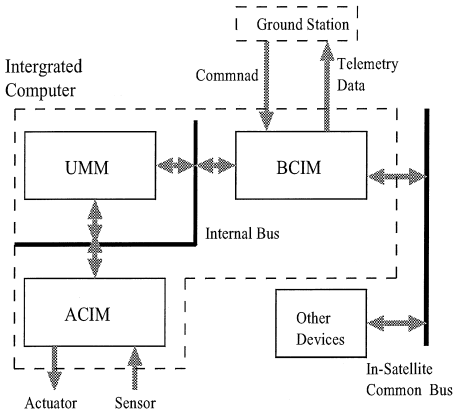

図 7 統合システムの構成
Fig. 7 Structure of integrated system.

納期化が要求されてきており，両者を統合したシステムの開発が必要となってきている．一方，人工衛星システムでも高性能な CPU を利用可能となってきた．こうした背景から，データ処理系システムと姿勢制御系システムを Unified Modeling Language（UML）を利用して統合化設計を行い，統合システムとして再設計を行うことになった（**図 7**）．ここでは統合システムのプロトタイプとして，統合システムに必要な機能をオブジェクトとして設計し，これをネットワークで接続された 3 台の PC の Embedded CORBA 上に実装した．Embedded CORBA 上に実装する目的は，UML による設計結果を素直な形態で実装すること，オブジェクト間通信により下位のプロトコルを隠蔽した実装を行うことである．

設計したオブジェクトは次のとおりである．

- Attitude Control Interface Module（ACIM）：統合システムにおいて姿勢制御のフロントエンドの役割を果たす．センサから姿勢データを受信し以下に述べる UMM に伝える．また，UMM からの制御データに基づき，アクチュエータに対して制御コマンドを送信する．ACIM では姿勢情報入手，姿勢制御といったメソッドを定義している．
- Bus Control Interface Module（BCIM）：統合システムにおいて地上からのデータとの入出力を司り，バス制御のフロントエンドの役割を果たす．地上からのコマンドを受信して UMM に伝え，UMM からのテレメトリデータを地上に送信する．また，衛星に搭載されている機器からのデータを UMM に伝え，UMM からのコマンドを衛星搭載機器に配信する．BCIM ではコマンド転送，テレメトリ転送といったメソッドを定義している．
- Unified Management Module（UMM）：ACIM，BCIM からデータ処理の統合化を司る処理を行う．すなわち，BCIM を通じて取得した搭載機器からのデータをもとに必要な制御コマンド列を生成し，コマンドのスケジュール情報を付与して BCIM に送出する．ACIM を通じて取得した姿勢情報をもとに姿勢計算処理を行い，ACIM に姿勢制御を依頼する．UMM では，姿勢計算，テレメトリデータ受け取り，といったメソッドを定義している．

本プロトタイプシステムでは，ACIM，BCIM，UMM で提供されるメソッドを呼び出すことによって，人工衛星に搭載されるバス機器の機能を実装した．すなわち，AOCS の機能が ACIM オブジェクトと UMM オブジェクトによって，DMS の機能が BCIM オブジェクトと UMM オブジェクトによって実現されている．

この実装と評価を通じて，次のような知見を得た．

- 比較的小規模ではあるが，実用システムのプロトタイプを Embedded CORBA 上に構築することができた．これにより，Embedded CORBA の機能の検証を行うことができた．
- システム設計を行っていくうえで，CORBA の提供する通信手段の抽象化は有効であった．従来バス機器では使用される CPU の性能が低いため，ソフトウェアから通信ハードウェアを直接アクセスする場合が多く，ソフトウェアとハードウェアが強く結合しており，ハードウェアとソフトウェアの並行設計ができない，ハードウェアの設計変更が上位ソフトウェアに影響を及ぼすなどの問題を引き起こしていた．統合ソフトウェアの設計では，CORBA の提供するオブジェクトのメソッド呼び出しにより通信ハードウェアの詳細を隠蔽することによって，両者間の結合を疎にした．この結果，上述の問題は解決した．次に実際に設計されたオブジェクトを Embdedded CORBA 上で実装し，Ethernet など通信ハードウェアの詳細を隠蔽して，抽象度の高いレベルでのメソッド呼び出しによって，バス機器の機能

の実装が可能であった．

今後，人工衛星用実システムにおいて MIL 規格など異なる通信プロトコルに実装する場合に，通信ハードウェアが隠蔽してあるので，ホストマシン上で開発したソフトウェアのターゲットマシンへの移植が容易になると考えられる．

Embedded CORBA で提供していないリアルタイム機能を，本プロトタイプでどのように実装したかを述べる．高い応答性能を要求されるリアルタイム処理は ACIM オブジェクトに隠蔽し実装した．具体的にはセンサ・アクチュエータからの割込みに対して応答する必要があることから，イベント駆動型の非同期タスクを ACIM オブジェクトに実装した．一方，BCIM オブジェクトでは，データ処理が主でありリアルタイム処理も比較的緩いものである．しかし，ネットワークデバイスとのインタフェースにはデッドライン要求が厳しく，高速に実行するデバイスドライバを実装した．

5. おわりに

本論文では，企業システムとは直接には接続しない組込みシステムネットワークに特化した機能を有し，機器の提供する機能拡張のためのオブジェクトの動的追加機能を具備した組込みシステム向け CORBA（Embedded CORBA）の設計思想，仕様，実装評価と応用について述べた．残された課題としては，以下があげられる．

組込みシステムで要求される機能として，リアルタイム機能がある．Embedded CORBA はリアルタイム機能を具備していない．CORBA 仕様には，リアルタイム機能を提供する Real-time CORBA [10] があるが，組込みシステムネットワークには大きすぎる仕様であると考えられる．このような応用に適したサブセット化が必要であろう．

次に，Embedded CORBA では C 言語マッピングのみ提供しているが，今後 Java 言語や C++などオブジェクト指向言語でオブジェクトが実装されることが考えられる．こうしたオブジェクトを利用可能とする機能強化があげられる．

現在，Embedded CORBA は，http://www.assoc.tron.org/itron/から取得可能である．

謝辞 Embedded CORBA は，1998 年度に情報処理振興事業協会が公募した “次世代デジタル応用基盤技術開発事業” における “情報家電のための分散ソフトウェアプラットフォームの構築” プロジェクトにおいて開発されたものである．

参考文献

1) 中本，高田，田丸：組込みシステム技術の現状と動向，情報処理，Vol.38, No.10, pp.871–878 (1997).
2) OMG: Minimum CORBA, CORBA v2.6.1, pp.23–1, 23–30 (2002).
3) 伊賀，佐藤，中本：組込みシステム向け分散オブジェクト環境，システム/制御/情報，システム制御情報学会，Vol.45, No.3, pp.18–25 (2001).
4) Gokhale, A. and Schmidt, D.: Techniques for Optimizing CORBA Middleware for Distributed Embedded Systems, *Proc. INFOCOM '99* (1999).
5) Gokhale, A. and Schmidt, D.: Optimizing a CORBA Internet Inter-ORB Protocol (IIOP) Engine for Minimal Footprint Embedded Multimedia Systems, *IEEE Journal on Selected Areas in Communications*, Vol.17, No.9, pp.1673–1706 (1999).
6) Lankes, S., Pfeiffer, M. and Bemmerl, T.: Design and implementation of a SCI-based real-time CORBA, *4th IEEE International Symposium on Object-Oriented Real-Time Distributed Computing*, pp.23–30 (2001).
7) Chan, A., Tse, F., Cao, J. and Leong, H.: Enabling Distributed Corba Access to Smart Card Applications, *IEEE Internet Computing*, Vol.6, No.3, pp.27–36 (2002).
8) Objective Interface Systems, http://www.ois.com/products/.
9) IONA Technologies, http://www.iona.com/products/orbix-e.htm.
10) OMG: Real-time CORBA, CORBA v2.6, pp.24–1, 24–56 (Dec. 2001).
11) （社）トロン協会：組込みシステムにおけるリアルタイム OS の利用動向と ITRON 仕様 OS に関するアンケート調査報告書 (2001).
12) Iga, N., Nakamoto, Y. and Monden, H.: Real-time Software Development System RTiplus, *12th TRON Project International Symposium*, pp.24–33 (1995).
13) 檜原，安達，田辺，齋藤，馬場，坂部，長瀬，森田，共田：UML による人工衛星バスシステムの統合化，オブジェクト指向最前線 2001，pp.143–150，近代科学社 (2001).
14) Nakamoto, Y., Iga, N. and Hihara, H.: Embedded CORBA Development and Its Applications to In-Satellite Network Software, *5th International Symposium on Object-oriented Real-time Distributed Computing*, pp.315–321, IEEE Press (2002).

表 6 Embedded CORBA と Minimum CORBA の API 一覧
Table 6 A list of Embedded and Minimum CORBA APIs.

		API 名	説明	Embedded CORBA にて実装
ORB	ORB	ORB_init	ORB オブジェクトを獲得する.	○
		object_to_string	指定された CORBA オブジェクトを，インターオペラビリティを考慮した IOR 形式の文字列に変換する.	○
		string_to_object	引数に指定された IOR 形式文字列を CORBA オブジェクトに変換する.	○
		get_service_information	サポートされているファシリティとサービスの情報を獲得する.	○
		list_initial_services	サービスのオブジェクト ID リストを獲得する.	
		resolve_initial_references	引数で指定されたサービスの CORBA オブジェクトを得る.	○
		run	オブジェクト実装は，本関数を呼び出し，クライアントからのリクエスト受信待ちループに入る．クライアントからのリクエストを受信した場合，適切なサーバントを呼び出す.	○
	Object	is_nil	CORBA オブジェクトの NULL チェックする.	*
		duplicate	CORBA オブジェクトの複製する.	
		release	指定された CORBA オブジェクトを解放する.	○
		is_equivalent	CORBA オブジェクトの等価チェックする.	
		hash	CORBA オブジェクトをハッシュ管理のための ID に関連付ける.	
		get_policy	引数で指定したポリシーオブジェクトの権限を獲得する.	
		get_domain_manager	自分自身が属しているドメインリストを得る.	
	Policy	copy	ポリシーオブジェクトのコピーを返す.	
		destroy	ポリシーオブジェクトを破棄する.	
		get_domain_policy	指定したオブジェクトに関連付けられた Domain Manager が管理するポリシーリストを獲得する	
IR	TypeCode	kind	TypeCode を獲得する.	
		id	リポジトリ ID を獲得する.	
		name	名前を獲得する.	
POA	POA Manager	activate	POAManager を活性化する.	○
	POA	createPOA	POA を作成する.	○
		find_POA	指定された名前の POA を検索する.	
		destroy	POA を削除する.	○
		create_lifespan_policy	POA の永続化ポリシーを作成する.	○
		create_id_uniqueness_policy	POA のサーバントに単一オブジェクト ID のみ割り当てることができるポリシーを作成する.	
		create_id_assignment_policy	POA のサーバントにユーザが定義したオブジェクト ID を割り当てることができるポリシーを作成する.	
		activate_object	引数に指定されたサーバントをアクティブオブジェクトマップに登録する.	
		activate_object_with_id	POA にオブジェクト ID を識別子としてサーバントを登録する.	○
		deactivate_object	POA に登録されているサーバントを削除する.	○
		create_reference	CORBA オブジェクトを作成する.	
		create_reference_with_id	CORBA オブジェクトを作成する.	
		servant_to_id	サーバントからオブジェクト ID を獲得する.	
		servant_to_reference	サーバントから CORBA オブジェクトを獲得する.	○
		reference_to_servant	CORBA オブジェクトからサーバントを獲得する.	
		reference_to_id	CORBA オブジェクトからオブジェクト ID を獲得する.	
		id_to_servant	オブジェクト ID からサーバントを獲得する.	
		id_to_reference	オブジェクト ID から CORBA オブジェクトを獲得する.	
	Current	get_POA	オブジェクトを実装している POA への参照を獲得する.	
		get_object_id	オブジェクトを識別可能なオブジェクト ID を獲得する.	

* 各 API 処理の最初でエラーチェックを行っているため，is_nil によるエラーチェックは不要である.

付　　録

付録では，Embedded CORBA と Minimum CORBA の API を比較する．Minimum CORBA 仕様でサポートされている API を**表 6** に示す．○印がついている API は Embedded CORBA でサポートされていることを示す．**表 7** には，Embedded CORBA で新たに拡張した API を示す．この中には，言語マッ

表 7 Embedded CORBA の追加 API 一覧
Table 7 A list of Embedded CORBA extention APIs.

		API 名	Embedded CORBA
初期化	リソース管理	ECORBA_init	組込み CORBA のリソースを設定する.
ORB	ORB	shutdown*	サーバ側アプリケーションに登録されたサーバントの関数内で，本関数を呼び出された場合，関数 run からリターンすることが可能となる.
言語マッピング	例外	CORBA_exception_init	例外情報を環境変数を初期化する.
		CORBA_exception_set*	例外情報を環境変数に設定する.
		CORBA_exception_id*	環境変数から例外識別子を獲得する.
		CORBA_exception_value*	追加例外情報を獲得する.
		CORBA_exception_free*	例外情報メモリ領域を解放する.
	メモリ	CORBA_malloc	メモリを獲得する.
		CORBA_free*	メモリを解放する.
	POA	PortableServer_POA_get_the_name*	POA の名前を獲得する.
		PortableServer_POA_get_the_parent*	親 POA を獲得する.
		PortableServer_POA_get_the_POAManager*	POAManager を獲得する.
Stub/Skeleton	マーシャリング	MRSHL_index_xxx	送信バッファ内インデックス値を獲得する．API 名の xxx の部分は，インデックス値を獲得型名が入る（例：long，float，など）.
		MRSHL_put_xxx	マーシャリングを行う．API 名の xxx の部分は，インデックス値を獲得型名が入る（例：long，float，など）.
		MRSHL_get_xxx	アンマーシャリングを行う．API 名の xxx の部分は，インデックス値を獲得型名が入る（例：long，float，など）.
	GIOP	GIOP_init_cli	クライアント側送受信を初期化する.
		GIOP_end_cli	クライアント側送受信を終了する.
		GIOP_create_reqbuf	リクエスト用送信バッファを作成する.
		GIOP_invoke_req	リクエスト送信/受信を行う.
		GIOP_create_repbuf	リプライ用送信バッファを作成する.
		GIOP_send_rep	リプライを送信する.
		GIOP_send_repexc	システム例外のリプライを送信する.
動的オブジェクト	登録/削除	ASRV_entry_dobj	オブジェクトを登録する.
		ASRV_cancel_dobj	ASRV_entry_dobj によって登録されたオブジェクトを削除する.

* の付与された API は CORBA 仕様のものを流用している.

ピングに関する API など，CORBA で規定されているものを流用しているものがある.

(平成 14 年 12 月 21 日受付)
(平成 15 年 2 月 18 日採録)

伊賀 徳寿

平成 4 年日本大学大学院理工学研究科電子工学専攻博士前期課程修了. 同年 NEC 入社. 現在ネットワーク開発研究本部モバイルサービス開発研究部主任. リアルタイムシステム，ソフトウェア開発環境，分散システム，モバイルシステムソフトウェアの研究開発に従事.

中本 幸一（正会員）

昭和 57 年大阪大学大学院基礎工学研究科前期課程修了. 同年 NEC 入社. 現在ネットワーク開発研究本部モバイルターミナル開発研究部長. リアルタイムシステム，モバイルシステムソフトウェア，ソフトウェア開発環境，分散システムの研究開発に従事. 平成 2〜3 年 Cornell 大学計算機科学科客員研究員. 平成 9 年大阪大学大学院博士課程入学. 平成 12 年単位取得退学. 博士（工学）. 平成 15 年より電気通信大学客員教授. 電子情報通信学会，日本ソフトウェア科学会，IEEE Computer Society 各会員.

奥山　嘉昭（正会員）

平成 10 年京都工芸繊維大学大学院前期課程修了．同年 NEC 入社．現在ネットワーク開発研究本部モバイルターミナル開発研究部所属．組込み TCP/IP，組込み CORBA，モバイルシステムソフトウェアの研究開発に従事．

佐藤　直樹（正会員）

昭和 63 年東京工業大学大学院修士課程情報科学専攻修了．同年 NEC 入社．現在ネットワーク開発研究本部モバイルターミナル開発研究部マネージャー．リアルタイムシステム，分散システム，携帯端末ソフトウェアの研究開発に従事．日本ソフトウェア科学会会員．

檜原　弘樹（正会員）

昭和 61 年大阪大学大学院工学研究科電子工学専攻前期課程修了，同年 NEC 入社．現在 NEC 東芝スペースシステム（株）機器開発本部搭載機器開発部マネージャー．人工衛星搭載用計算機，通信制御装置，画像処理装置，レーザー・レーダ等の開発に従事．電子情報通信学会，IEEE Aerospace and Electronic Systems Society/Computer Society 各会員．