2007－EMB－6 (6)
2007／12／4

動的再構成可能な組込みシステム向け分散オブジェクトシステムの開発

谷田貝純†，早川栄一‡
† 拓殖大学大学院工学研究科 〒193-0985 八王子市館町 815-1
‡ 拓殖大学工学部 〒193-0985 八王子市館町 815-1
E-Mail：† epsilon@os.cs.takushoku-u.ac.jp，‡ hayakawa@cs.takushoku-u.ac.jp

本稿では多様な環境を持つ組込みシステムに対して，その環境に適した機能と規模に動的再構成することが可能な分散オブジェクトシステムの開発について述べる．組込みシステムでは資源が制限された環境で様々な要求に対応する必要がある．そこで，システムの軽量化と機能の構成を再構成可能とする機構の開発を行った．さらに，ネットワーク構成の変化に対応するために再構成を動的に行うことが可能な分散オブジェクトシステムを開発した．

Development of a Dynamic Reconfigurable Distributed Object System for Embedded Systems

Jun YATAGAI†，Eiichi HAYAKAWA‡
† Graduate School of Engineering，Takushoku University 815-1 Tatemachi，Hachioji City，Tokyo，193-0985 Japan
‡ Faculty of Engineering，Takushoku university 815-1 Tatemachi，Hachioji City，Tokyo，193-0985 Japan
E-Mail：† epsilon@os.cs.takushoku-u.ac.jp，‡ hayakawa@cs.takushoku-u.ac.jp

This paper describes the development of a dynamic reconfigurable distributed object system that has function and scale suitable for various embedded systems.

It is necessary to cope with various demands in the embedded systems where resources were restricted. The lightweighting of the system and development of the machanism that makes structure of function reconfigurable were peformed. The system is able to dynamically reconfigure ORB structure for adapt to a change of the network structure.

1．はじめに

近年，情報家電やセンサネットワークなど組込みシステムがネットワークに接続され連携することで機能を実現するシステムが増加している．このような中でネットワークアプリケーションの開発効率の向上が注目され，それを満たす方法の一つとして分散オブジェクトシステムの導入が行なわれている．分散オブジェクトシステムはアプリケーションの開発者からネットワークプログラムを隠蔽し，オブジェクト指向の手続きとして通信することができる．これにより，資源の再利用が可能となり生産性の向上に繋がる．

しかし，従来の組込みシステム向け分散オブジェクトシステムは企業システム向けのものと比較して多くの機能を削除しているので，様々な機能要求に対応できないという問題がある．さらに，機能要求はネットワーク構成の変更により稼動後にも変化する．

そこで，本研究では分散オブジェクトシステムの規模と機能の割当てを設計することが可能であり，システムの稼動後も同様に環境の変化に応じた分散オブジェクト環境を構築できるシステムを開発する．

2．問題分析

組込みシステムにおける分散オブジェクトシステムの導入に伴う問題を次に示す．

（1）組込みシステムの制限

組込みシステムの資源は汎用的なコンピュータと比較して制限されている．多くの組込みシステムは，RAM，ROM の容量が少なく，NTT ドコモが提供する携帯電話のアプリケーション実行環境ではプログラムの保存領域とメモリ容量の合計が 1024KB 以下である必要がある．この

ような制限から，組込みシステムには資源の消費が大きなミドルウェアの導入は困難である．

（2）組込みシステムが持つ環境の多様性

組込みシステムの性能は様々であり，上述の携帯電話上の実行領域のように資源が制限された環境から，情報家電のように比較的に性能が高いものがある．同じ CPU アーキテクチャを備えたシステムでも，MB 単位のメモリ領域を利用でき，組込み Linux による豊富なシステムサービスを利用できるものから，RTOS を含めて 100KB 単位のものまで多種多様である．組込みシステムの資源はコストに大きく影響を与えることから，導入可能なシステム規模の境界は環境によって異なる．

（3）様々な機能要求

組込みシステムは利用される環境が様々であり，環境によって必要とされる機能が異なる．例えば，携帯電話との連携が必要であれば HTTP 通信への対応が必要となる．機能の削減が行われた組込みシステム向け分散オブジェクトシステムはこのような機能要求に対応することが困難である．

また，要求される機能は通信方式のサポートやセキュリティ機能など様々なものが考えられるが，すべてに対応しようとするとシステムは企業システム向け分散オブジェクトシステムのように大規模なものとなる．オープンソースである分散オブジェクトシステムの TAO[1]はリアルタイム仕様や QoS 機能など豊富な機能を持っているが，プログラムサイズは 37MB であり規模が大きい．これは上述の理由から組込みシステムでは利用が不可能である．

（4）システム構成の変化

ネットワークによるシステムはその稼動後にシステム構成の変更やアプリケーションの更新が必要となる．組込みシステムでは監視システムなどのように常時稼動が必要とされる場面があり，このような状況にもその変化に対応する必要がある．しかし，分散オブジェクト環境で構築されたシステムにおいて，稼動状態を維持したシステムの変化に対応することは分散オブジェクトシステムによるサポートが必要となる．

3．設計方針

上記の問題を解決するために，本システムの設計方針を次のように定める．

（1）ORB の軽量化

分散オブジェクトシステムの中核となる ORB(Object Request Broker)の軽量化を行う．軽量化の対象として RAM，ROM に影響するシステムのプログラムサイズ，フットプリントを削減する．

（2）柔軟な分散オブジェクトシステム

分散オブジェクトシステムを利用する開発者がターゲットとなる組込みシステムの環境に応じて，分散オブジェクトシステムの規模と機能の変更を可能とする．

（3）システムの変化に対応

システムの変化に伴うアプリケーションの更新や機能要求に対応するため，分散オブジェクトシステムはシステムの稼動を維持しながら要求に対応できるようにする．

4．全体構成

システムの全体構成を図 1 に示す．

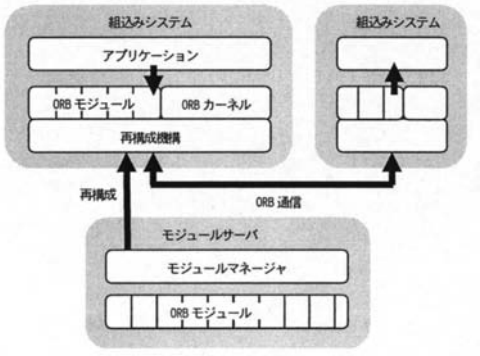

図 1　全体構成

組込みシステムは ORB を介して分散オブジェクト環境による連携を行うことができる．ORB は ORB カーネル(以下カーネル）と ORB モジュール（以下モジュール），再構成機構で構成されている．カーネルは ORB の基盤機能を提供し，モジュールは ORB の拡張機能を提供する．再構成機構は ORB 内のモジュールの管理を行う．ORB を基盤機能と拡張機能，管理機構に多層化することで，機能の拡張性と環境への特化が容易な構成としている．

モジュールマネージャはモジュールやその情報を管理する．再構成機構にモジュールを提供することで ORB の再構成を支援する．

5．設計

5．1．再構成機構

再構成機構は ORB 内のモジュール管理機構である．再構成機構はモジュールを取得し読み込むことでモジュールの機能を利用可能にする．再構成機構の構成を図 2 に示す．

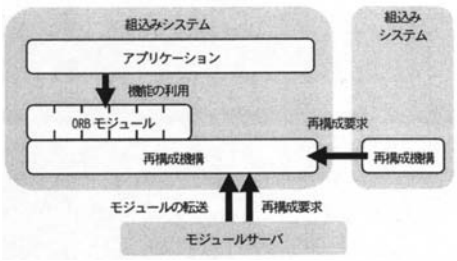

図2　再構成機構の構成

（1）モジュールの取得

再構成機構は通信によってモジュールを取得する．この取得方法は静的取得と動的取得の2種類がある．図3に各取得方法の仕組みを示す．

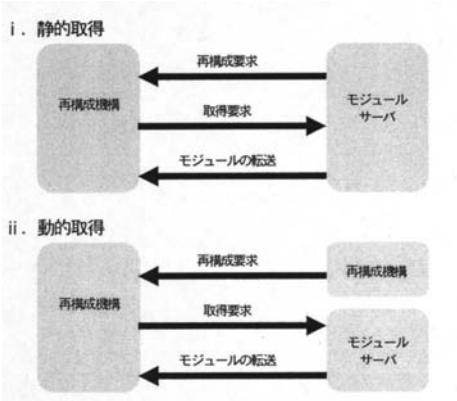

図3　モジュールの取得方法

静的取得は再構成要求がモジュールサーバから送信される方法である．これを受信した再構成機構はサーバに再構成要求で指定されたモジュールを転送するように取得要求を返信する．そして，再構成機構はモジュールサーバによって転送されるモジュールを受信する．これは開発者がモジュールサーバから組込みシステムを管理し，ORBの再構成を行う場合に利用される．

動的取得は再構成要求が別の組込みシステム上の再構成機構から送信される方法である．この要求は通信相手のORBが通信に必要なモジュールを所持していない場合に送信される．これを受信した再構成機構は静的取得と同様にモジュールサーバからモジュールを取得してそれを読み込む．

（2）モジュール例外による再構成

ORB通信時に通信相手が必要なモジュールを持っていない場合はモジュール例外として再構成要求を通信相手の再構成機構に送信する．これによりモジュールの動的取得が行なわれる．例外による再構成によって，新たに組込みシステムをネットワークに追加する場合や，他のネットワークとの接続時に自動的にモジュールを補完することができる．

（3）モジュールの衝突回避

各モジュールは同時の利用が不可能な機能や，特定のモジュールに依存する場合がある．再構成機構はこれらを回避する機構を持つ．各モジュールの情報はオブジェクトマネージャが管理し，オブジェクトマネージャからのモジュール受信時に，受信するモジュールと現在利用しているモジュールの情報を受取る，受信するモジュールが利用できない場合は受信処理を停止することで，各モジュールの衝突を防止する．

（4）モジュールの利用

再構成機構はモジュールをタイプ別に管理する．読み込まれたモジュールはカーネルが提供する機能を拡張する4タイプの機能と，それ以外のサービスに分類できる．四つのタイプは通信プロトコル定義，オブジェクト接続，機能呼出し，オブジェクト解放となる．

カーネルを拡張する機能はタイプ別のインタフェースによってカーネルと連携し，その機能はカーネルがアプリケーションに提供するAPIを通して利用することができる．サービスはサービス向けの汎用関数から利用することができる．

5．2．モジュールマネージャ

モジュールマネージャはモジュールを管理し，ORBの再構成を支援する．モジュールマネージャの構成を図4に示す．

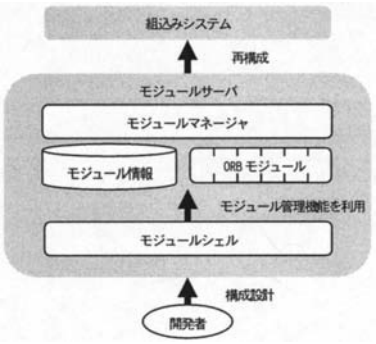

図4　モジュールマネージャの構成

（1）モジュールの管理

モジュールマネージャはモジュールを保持し，それらの情報をデータベースで管理している．これは多くのモジュールが保存される場合に管理を容易にするためである．主なモジュール情報として，衝突・依存モジュール，モジュ

ールグループがある．

衝突・依存モジュールの情報は ORB の再構成時に参照される．また，モジュールグループはモジュールを機能単位でまとめて扱うための情報で，後述するモジュールシェルでのモジュールの扱いを容易にするためのものである．

（2）モジュールシェル

モジュールシェルは開発者に ORB のモジュール構成を変更するためのコマンドラインインタフェースを提供する．これを利用することで，開発者はコマンドから ORB のモジュール構成の参照や，モジュール構成の設計，モジュールの転送などの機能を実行することができる．

開発者は任意の組込みシステム内のモジュール構成を確認し，機能単位で構成の設計を行うことができる．続けてモジュールの転送を行うことで設計が適用される．これにより容易に ORB の機能を再構成することができる．

（3）転送機能

モジュールマネージャは再構成機構のモジュール取得方法に対応する転送処理を行う．静的取得時はモジュールシェルのコマンド実行時に再構成要求を送信し，衝突の問題がなければ実際にモジュールを転送する．また，動的取得時は再構成機構からの取得要求に対してモジュールを転送する処理を行う．

5．3．ORB カーネル

カーネルは ORB の基盤機能を提供する．カーネルの構成を図 5 に示す．

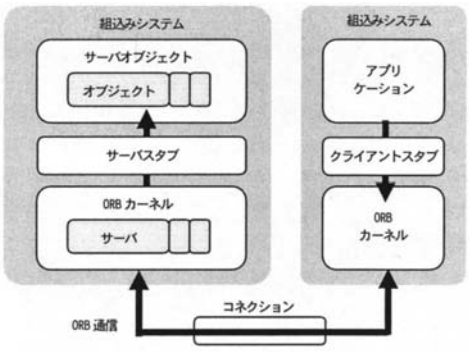

図 5　ORB カーネルの構成

アプリケーションはクライアントスタブをサーバオブジェクトのように利用することでサーバオブジェクトの機能を呼び出す．

（1）機能の削減

カーネルはオブジェクト間が行う通信を次の三つに制限することでシステムの軽量化を行っている．

・オブジェクトへの接続

・機能の呼出し

・オブジェクトの解放

（2）スタブの利用

アプリケーションは上述の通信をスタブを介して行う．スタブ生成の仕組みを図 6 に示す．

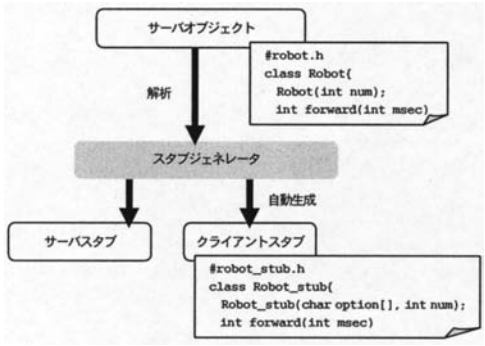

図 6　スタブ生成の仕組み

スタブはスタブジェネレータによって自動生成される．スタブジェネレータは機能を提供するサーバオブジェクトを解析してスタブを生成するので，開発者が IDL のようなインタフェース定義を記述する必要がない．

アプリケーションはクライアントスタブを利用することでサーバオブジェクトに近いインタフェースで機能を利用できる．

ORB によるエラーが発生した場合はスタブからアプリケーションに例外が返る．また，オブジェクトを引数として利用する場合はその参照を渡すことで処理する．

（3）リソースの制限

カーネルのフットプリントを環境に応じて変更可能とするためにリソースを制限する機構をもつ．制限するのは動的にメモリを消費する項目であり，オブジェクト数，コネクション数，モジュール数，サーバ数となる．

これらの最大作成可能数をシステムのプロファイル情報から起動時に設定する．また，システム起動時に値を直接に指定することができる．これによりフットプリントの調節が可能である．

5．4．ORB モジュール

モジュールは ORB の拡張機能を提供する．これはカーネルと連携し様々な機能を提供することができる．モジュールの機能例を次に示す．

- HTTP 通信（通信プロトコルタイプ）
- 非同期呼出し（機能呼出しタイプ）
- セキュリティサービス（サービスタイプ）

6．実装

本システムは分散オブジェクトシステムを C++言語で記述し，オブジェクトマネージャを Java 言語で記述した．C++言語はオブジェクト指向言語であり，分散オブジェクトシステムへの親和性が高く，組込みシステムにおいて採用頻度も高い．オブジェクトマネージャは様々なサーバで容易に機能を提供できることから Java 言語を利用した．

（1）再構成機構

再構成機構の総行数は約 160 行である．システムの稼動中にモジュールを受信し，その機能を利用することが可能である．これにより，環境の変化に応じた機能の再構成が可能である．

（2）ORB カーネル

ORB カーネルの総行数は約 650 行である．基盤機能とリソースの制限を利用することができる．これにより，資源が制限された環境における分散オブジェクトシステムの導入が可能である．

（3）モジュールマネージャ

モジュールマネージャの総行数は約 1050 行である．データベースにはシステムへの組込みが可能な HSQLDB[2] を利用し，モジュールの情報を管理することができる．また，モジュールシェルから分散オブジェクトシステムの再構成を容易に行うことができる．

7．利用方法

本システムが提供する機能の利用方法を次に示す．

（1）開発手順

本システムを利用した開発手順を次に示す．

i. サーバオブジェクトの開発
ii. サーバオブジェクトを解析し二つのスタブを生成
iii. クライアントスタブを利用してアプリケーションを開発
iv. ORB を再構成

（2）モジュールシェルの利用

モジュールシェルを利用した画面とその手順を次に示す．

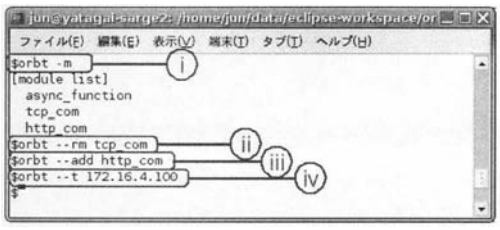

図 7　モジュールシェルの利用例

i. 現在利用できるモジュールの確認
ii. 設計にあるモジュールの削除
iii. 設計にモジュールの追加
iv. 指定のシステムへ設計を適用

分散オブジェクトシステムを利用する開発者はコマンドによって容易に機能の構成を設計し，再構成することができる．モジュールシェルが提供するコマンドを表 3 に示す．

表 3　モジュールシェルが提供するコマンド

コマンド	機能
orbt -g [IP]	指定システムの機能構成を参照
orbt -l	現在の設計を表示
orbt -m	利用可能なモジュールを表示
orbt -add [M]	指定モジュールの追加
orbt -rm [M]	指定モジュールの削除
orbt -t [IP]	モジュールの転送

（3）記述方法

本システムによる分散オブジェクト環境で，サーバオブジェクトとその機能を呼び出すアプリケーションの記述例を図 8 に示す．

i．サーバオブジェクト

```
class Robot{
  Robot(int num);
  int forward(int msec);
```

ii．アプリケーション

```
Robot_stub *robot;
robot = new Robot_stub( “async” , 100);
int result = robot->forward(2000);
```

図 8　サーバオブジェクトとアプリケーションの記述例

図 8 のアプリケーションの記述では，アプリケージョンがスタブを介して Robot オブジェクトに接続している．また，アプリケーションはスタブから非同期呼出し機能を用いて forward 関数を呼び出している．

8．評価

ORB カーネルの規模としてプログラムサイズとメモリ消費量を測定し，組込みシステム向けの軽量 CORBA 実装である Embedded CORBA[3]が公開している性能と比較した．この測定結果とその比較を表 4 に示す．

表 4　測定結果の比較

	本システム	Embedded CORBA
プログラムサイズ	23.6KB	24.9KB
メモリ消費量	13.0KB	N/A
API 数	4	14

Embedded CORBA のメモリ消費量に関する情報が不明であるので表では N/A とした．本システムはアプリケーションオブジェクトの最大数を 50 に設定し，測定環境として CPU が Pentium4 3.2GHz，メモリが 1GB，OS が linux である環境を用いた．

本システムは組込みシステム向け CORBA 製品と比較してプログラムサイズは小さく，メモリ消費量も組込みシステムでの利用が可能なサイズである．

9．応用事例

本システムをロボット上の組込みシステムに導入し，モータ制御による遠隔操作とタッチセンサ・可視光センサの値を取得するアプリケーションを開発した．組込みシステムにはシリコンリナックス社の CAT709[4]を用い，通信は本システムによる分散オブジェクト環境を利用している．組込みシステムの環境と構成を次に示す．

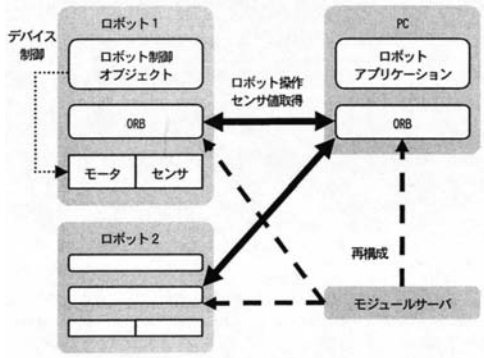

図 9　応用事例の構成

このシステムは分散オブジェクト環境を利用し，ロボット操作側の PC からロボット側に用意されたロボット制御オブジェクトの機能を利用することができる．

ロボット制御オブジェクトはモータ制御を行うロボットの移動用関数，タッチセンサ・可視光センサの値を取得しそれを返すセンサ値取得関数を持っている．移動関数はモータ制御ライブラリを呼び出し，前後の移動と左右の旋回を行うことができる．

また，このシステムで複数のロボットを制御する場合，非同期呼出しのモジュールを再構成機構によって追加することで，ロボットの動作終了まで PC が待機せずに別のロボットを操作するということが容易に可能になる．このように本システムを実際に組込みシステムに導入し，利用することが可能である．

10．関連研究

本研究と関連を持つ組込みシステム向け分散オブジェクトシステムの研究として次のものがある．

Minimum CORBA[5]を組込みシステム向けに軽量化した実装として Embedded CORBA がある．これはアプリケーションの動的な更新が可能だが，ORB 機能を拡張する機構を持たないので，様々な機能要求に対応できない．

機能の要求に柔軟に対応できる分散オブジェクトシステムとして Flexible micro-ORB[6]がある．これは ORB 機能を中間言語で管理し，動的コンパイラを利用することで様々な環境で ORB 機能を利用することができる．しかし，システムの規模が大きくなってしまうことや，機能を独自の言語で記述しなければならないという問題がある．

11．おわりに

本稿では様々な環境を持つ組込みシステムに対し，機能要求に柔軟に対応できる軽量分散オブジェクトシステムの開発について述べた．本研究では軽量な基盤システム，動的な再構成機構を持つ分散オブジェクトシステムの開発を行った．これにより，組込みシステムの制限や機能要求に対応した分散オブジェクト環境を構築し，アプリケーションを実行することができた．

今後の課題として，モジュールの開発を進める必要がある．

参考文献

[1] The ACE ORB：http://www.theaceorb.com/

[2] HSQLDB：http://hsqldb.org/

[3] 伊賀徳寿，“組込みシステム向け CORBA の開発と評価”
情報処理，Vol.44 No.SIG10 pp.164-175 (2003)

[4] シリコンリナックス CAT709
http://www.si-linux.co.jp/product/cat709/

[5] Minimum CORBA
http://www.omg.org/technology/documents/formal/minimum_CORBA.htm

[6] Frederic Ogel，“A Step Towards Ubiquitous Computing：an Efficient Flexible micro-ORB”，Proceedings of the 11th workshop on ACM SIGOPS European workshop: beyond the PC (2004)