2004－DPS－120　(9)
2004／11／5

異種分散コンポーネントを対象にした Plug and Play 環境の提案

名倉 正剛 *　　高田 眞吾 †　　土居 範久 ‡
* 慶應義塾大学大学院理工学研究科　† 慶應義塾大学理工学部　‡ 中央大学理工学部

概要

分散コンポーネントを利用する場合，クライアントはまずレジストリを検索することによって，コンポーネントが存在する場所を把握しないとならない．しかし “Plug and Play” の環境を考えた場合，サーバやクライアントはどのネットワークに接続されるかを事前に特定できないので，コンポーネントをレジストリにあらかじめ登録しておくことは現実的ではない．本研究では，クライアントが必要とするコンポーネントを利用時に動的に発見できるような，“Plug and Play” 環境を提案する．これはさらに，コンポーネントが必要な他の異種分散コンポーネントをネットワーク上で発見し，自動的に統合することを可能にする．

A Plug-and-Play Environment for Distributed Heterogeneous Components

Masataka NAGURA**　　Shingo TAKADA††　　Norihisa DOI‡‡
** Graduate School of Science and Technology, Keio University
†† Faculty of Science and Technology, Keio University
‡‡ Faculty of Science and Engineering, Chuo University

Abstract

When a client wants to use a component in a distributed environment, the client needs to first obtain the location of the component by searching registries. But in a ”Plug-and-Play” environment, the servers (or components) and clients cannot specify in advance which network they will be connected to. Therefore, it is not possible to register components in registries in advance for such environments. In this paper, we propose a ”Plug-and-Play” environment where clients can find required components dynamically. We also consider cases where a component searches for other components that it may need. Such components may be heterogeneous and are integrated.

1　はじめに

インターネットの発達に伴って，分散コンポーネント技術を利用したシステムが普及している．コンポーネントを利用する場合，クライアントはまず必要とするコンポーネントが存在する場所を把握するためにレジストリを検索する．このためにレジストリには，コンポーネントに関する情報をあらかじめ登録しておく．しかし，コンポーネントの存在するサーバやそれを利用するクライアントをネットワークに接続することによって，自動的にクライアントから利用可能になるような “Plug and Play” の環境を考えると，どのようなコンポーネントがネットワークに参加するのかをあらかじめ特定できない．従って，このような環境ではコンポーネントに関する情報をレジストリにあらかじめ登録しておくことは現実的ではない．そこで，このような環境でクライアントが必要とするコンポーネントを利用時に探索できるような機構が必要になる．

本研究では，コンポーネントの “Plug and Play” を実現するために，クライアントが必要とするコンポーネントを提供するサーバを利用時に探索できる環境を提供する．この環境を利用することで，各サーバ自身も同様に他のサーバを探索できる．従って，メッセージや返戻値

などの形式が互いに異なる複数の異種分散コンポーネントをネットワーク上で発見し自動的に統合することで，新しいコンポーネントを提供することを可能にする．

本論文ではまず，分散コンポーネントシステムで Plug and Play を実現する場合の問題点を分析する．その後，異種分散コンポーネントを対象にした Plug and Play 環境の提案を行なう．

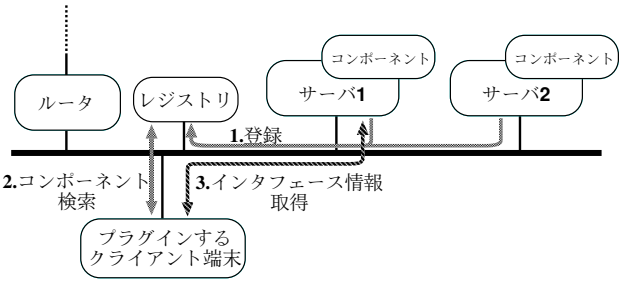


図 1: 分散コンポーネント利用までの手順

2 分散コンポーネントシステムとコンポーネント発見

コンポーネントは，外部からインタフェースを介してアクセスを行なうことのできる，独立して動作するソフトウェアのことである [1]．近年では，ネットワークを介して通信を行うソフトウェアに対してコンポーネント技術を適用した，分散コンポーネント技術 [2][3][4] が普及しつつある．一般的に，分散コンポーネントシステムは，従来のクライアントサーバシステム実現のための一つの手法として利用されている．そのためクライアントとサーバによって構成され，コンポーネントはサーバ上に配置される．そしてクライアントはネットワークを介してコンポーネントを呼び出す．なお本研究で提案する Plug and Play 環境は利用モデルを限定しないが，本論文ではこのようなクライアントとサーバから構成される利用モデルを前提として説明する．

ネットワーク上の分散コンポーネントをクライアントから利用するために，まず必要とするコンポーネントを提供するサーバを発見しなければならない．このために一般的な分散コンポーネントフレームワークでは，コンポーネントの存在するサーバを登録しておくレジストリ [5][6] を，組織や会社などのネットワークの単位ごとに用意している．次に，クライアントは外部からアクセスを行うために，インタフェースに関する情報を入手する．一般的な分散コンポーネントフレームワークでは，クライアントはインタフェースに関する情報を，レジストリやコンポーネントを提供するサーバから入手する (図 1)．

このように，クライアントからコンポーネントを利用するためには，コンポーネントを提供するサーバや，それに関する情報を提供するレジストリを発見することが必要になる．

3 分散コンポーネントシステムの Plug and Play

近年インターネットの発達により，ユーザが端末をいろいろな場所に移動させて，その場所のネットワークに接続するような利用方法が増大してきた．従来は，ネットワークとそれに接続される端末の構成はある程度静的に決定されていたが，ユーザの利用形態が多様化したため，動的に変更されうる環境での利用を考慮する必要が生じてきている．

本章では，色々な端末が様々な場所のネットワークに動的に接続したり切断されたりする “Plug and Play” の環境において，分散コンポーネントシステムを利用する場合の問題点を考える．

3.1 サーバ発見の問題

移動体端末を分散コンポーネントシステムのクライアントとして利用した場合，クライアントは移動先のネットワークからサーバに存在するコンポーネントを利用する．また，家電品などの組み込み機器向けにサーバを組み込んだ場合，サーバは様々な場所のネットワークに接続し，コンポーネントを動作させることになる．このような時には，以下のような問題が発生する．

クライアントがプラグインする場合

クライアントは，プラグインしたネットワークでレジストリを検索するために，どこにレジストリが存在しているかを知っていなければならない．しかし，プラグインするネットワークを特定できないので，これは不可能である．従って，コンポーネントが存在するサーバの場所を発見することは困難である．

サーバがプラグインする場合

分散コンポーネントを提供するサーバは，自分自身をクライアントから発見できるようにする必要があるため，レジストリに登録する．接続したネット

ワークでレジストリを発見することは，クライアントの場合と同様にできない．それに加えて，プラグインしたサーバがレジストリの内容を変更すること自体，レジストリの管理者にとって望ましくない．このため，レジストリへの登録は困難である．

Plug and Play 環境ではプラグイン端末はプラグイン先のネットワークの構成を把握できないため，レジストリを発見できないことが問題になる．これを解決するために，プラグイン先のネットワークで簡単にレジストリを発見する方法が必要である．

しかしサーバがプラグインする場合を考えると，レジストリを発見できたとしても，ネットワークへの接続や切断が頻繁に行なわれる環境で，その都度レジストリへ登録や削除を行なうのは困難である．従って，コンポーネントが存在する場所を管理する特別な機構を設けずに，利用したいコンポーネントを探索することのできる方法が必要になる．

3.2 既存のサーバ探索手法

汎用的な探索プロトコル [8][9] では，クライアントがサーバを探索する方法を定義している．探索を行ないたいクライアントは，マルチキャストでネットワーク内のサーバ群にリクエストを送信する．該当するサーバは，リクエストに対し応答を返す (図 2).

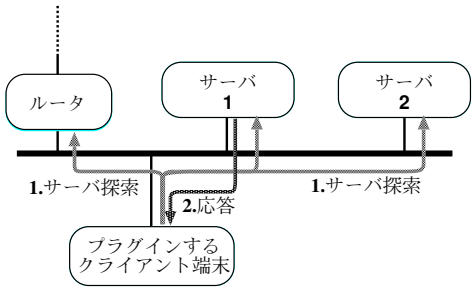


図 2: サーバ探索プロトコルの手順

分散コンポーネント技術の一つである Web Services にも，同様の方法でクライアントからコンポーネントを提供するサーバを発見するためのドラフトが提案されている [7]．これは，Web Services を利用するためのプロトコルである SOAP (Simple Object Access Protocol) を探索に利用している．しかし SOAP は HTTP を利用していてマルチキャスト通信に向かないため，実現可能性は低いと考えられる．

3.3 Plug and Play とコンポーネント統合

独立して動作するコンポーネントを組み合わせることによって，さらに新しいコンポーネントを生成することができる [10].

Plug and Play 環境では，サーバが動的に追加・削除される．あるコンポーネントの動作するサーバがネットワークに追加された時，そのネットワークに存在する別のサーバが提供するコンポーネントと組み合わせることにより，新しい別のコンポーネントを生成することができる可能性がある．

前節に挙げたようなクライアントがサーバを探索する方法を用いることによって，各サーバ自身も同様に他のサーバを探索できる．従って各サーバは，統合して動作することのできる他のコンポーネントを発見し，統合することが可能である．

4 Plug and Play 環境の提案

本研究では分散コンポーネントに対して Plug and Play を実現するための環境を提案する．まず，必要とするコンポーネントを提供するサーバを，クライアントが利用時に探索できる環境を提供する．そして，各サーバも同様に他のサーバを探索することによって，分散コンポーネントをネットワーク上で発見し自動的に統合することができる方法を提案する．

主に以下の 2 つの部分について，提案を行なう．

- コンポーネントを発見する部分
  必要とするコンポーネントの存在するサーバを探索するための方法を提案する．
- コンポーネントを統合する部分
  探索したコンポーネントを統合して新しいコンポーネントを生成する方法を提案する．

まず全体の動作について述べた後に，それぞれの部分について詳細に述べる．

4.1 概要

本研究で提案する Plug and Play 環境では，分散コンポーネントシステムのクライアントやサーバがプラグインした時に，他の必要となるコンポーネントを提供するサーバを発見する．そしてさらにサーバがプラグインした時には，提供するコンポーネントと組み合わせて動作できる別のコンポーネントがネットワーク内に存在する場合には，自動的に統合して新しいコンポーネントを提供する．

現在，EJB や CORBA や Web Services 等の異なった様々な形式の分散コンポーネント技術が普及しつつある．本機構ではこのようなメッセージや返戻値などの形式が互いに異なる複数の異種コンポーネントを自動的に発見して統合することを実現する．

4.2 サーバを発見する機構

提案する環境では，クライアントがコンポーネントを利用する時や，統合しようとするコンポーネントが別のコンポーネントを呼び出す時は，形式の異なる他の種類のコンポーネントを呼び出すことができるようにする．従って，サーバを発見する機構を，コンポーネントが実装されているアーキテクチャに依存しない機構にする必要がある．

Web Services のためのサービス探索機構である WS-Discovery [7] は，利用するプロトコル自体に SOAP を用いており，実装されるコンポーネントアーキテクチャの利用する特定のプロトコルに依存してしまっている．また，本機構は汎用的なサービス探索プロトコル [8][9] を拡張することによっても実現可能だが，拡張する部分が多過ぎ，結果として非常に冗長性の高いものになってしまう．そこで独自の機構を提案する．

コンポーネントを提供するサーバを発見できるようにするために，以下の情報をやり取りする．

(A) コンポーネントの存在を示す広告

(B) 探索のための要求／応答

(C) パラメータの要求／応答

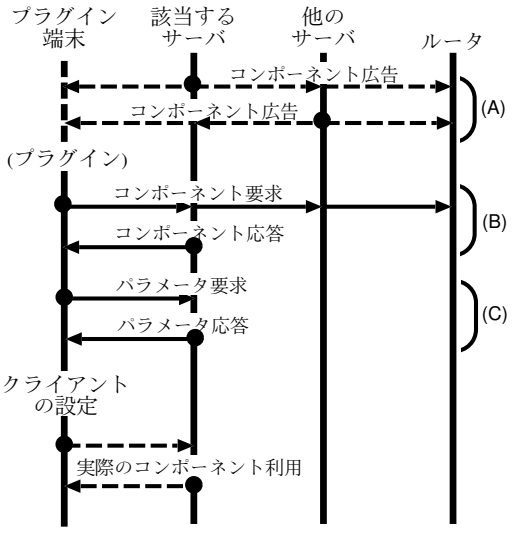


図 3: コンポーネントの探索と利用

コンポーネントを呼び出すためには，実装されているコンポーネントのアーキテクチャに依存した情報が必要になる．そこでそれらを，コンポーネントの探索とは別個のメッセージ (パラメータの要求／応答) として設計する．そして探索の結果，クライアントから呼び出すことのできる形式のコンポーネントだった場合にのみパラメータを要求する．

クライアントがコンポーネントを探索し，利用する場合のメッセージの流れを図 3 に示す．

サーバがプラグインした時，ネットワーク内に組み合わせて動作させることができる別のコンポーネントが存在する場合は，自動的に統合して新しいコンポーネントを提供する．既に統合されているコンポーネントと更に統合する場合は，どのコンポーネントと既に統合しているかについても探索する．これによって，それらの各コンポーネントの状態が変化した場合でも対応できるようにする．そのために，以下の情報をやり取りする．

(D) 構成しているコンポーネントを探索するための要求／応答

サーバがプラグインした際に，ネットワーク上の利用可能なコンポーネントを探索し統合する時のメッセージの流れを，図 4 に示す．

コンポーネントを削除する時は，削除することをネットワーク内に広告する必要がある．また，他のコンポーネントから利用されているコンポーネントを削除する時に，利用しているコンポーネントが存在するサーバは，代替となる別のコンポーネントを探す．その間はコン

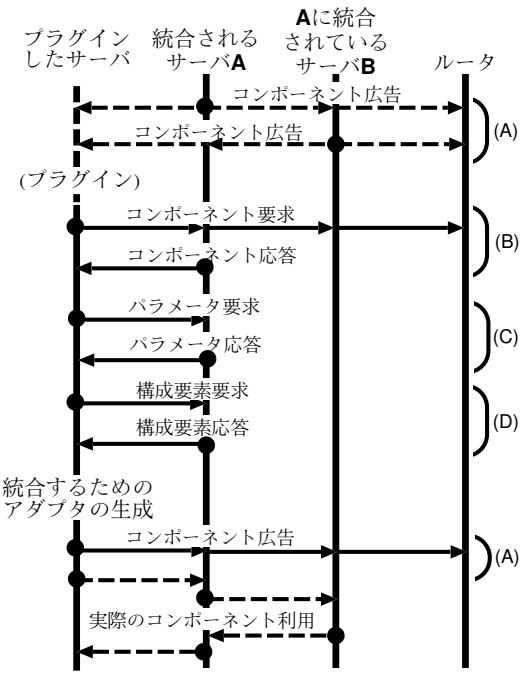
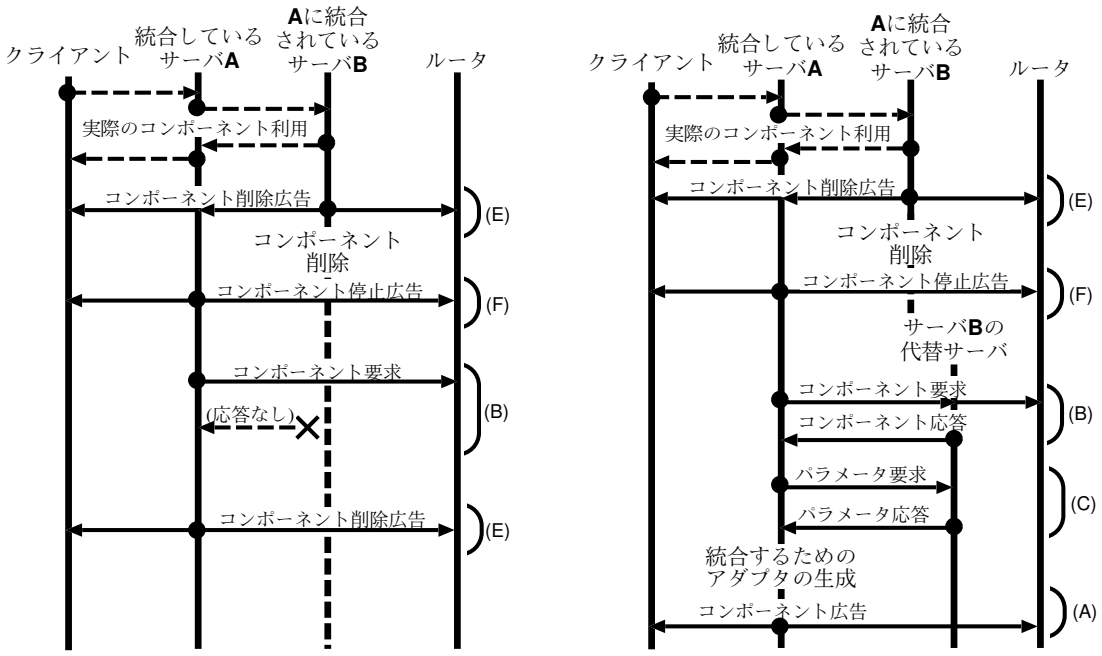


図 4: コンポーネントの探索と統合 (サーバプラグイン時)



a. 代替となるコンポーネントが見つからなかった時　b. 代替となるコンポーネントが見つかった時

図 5: コンポーネントの削除時の動作

ポーネントの機能が停止しているので，このこともネットワーク内に広告する必要がある．これらを踏まえて，以下の情報をやり取りする．

(E) コンポーネントの削除を示す広告

(F) コンポーネントの停止状態を示す広告

他のコンポーネントから呼び出されているコンポーネントを削除した時に，呼び出しているコンポーネントが代替となるコンポーネントを探す場合のメッセージのやり取りを，図 5 に示す．図 5-a に探すことができなかった場合を，図 5-b に探すことができた場合を示す．また，呼び出しているコンポーネントの状態遷移を，図 6 に示す．

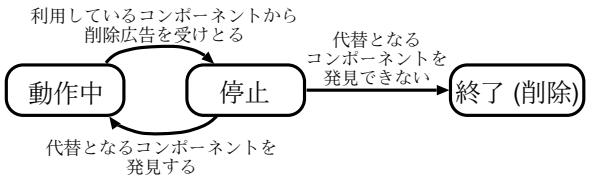


図 6: 統合しているコンポーネントの状態遷移

4.3 コンポーネントの統合

コンポーネントを統合する時には，以下の点が問題になる．

- どのコンポーネントどのコンポーネントが対応づけられるのか，呼び出す順番はどうなっているのか
- 引数や返戻値はどのように対応するのか

これらの情報は，コンポーネントを呼び出そうとする別のコンポーネントの開発者や提供者があらかじめ記述し，それらのコンポーネントが存在するサーバやクライアントに配置しておく．これにより，どのような種類のコンポーネントとどのように統合を行なえば良いのかを示すことができるので，異種コンポーネントを呼び出して利用するためのアダプタを自動的に生成して，統合することができるようになる．

5 事例研究

本研究の適用事例として，まず EJB で実装された logger コンポーネントが存在するネットワークに，logger クライアントがプラグインする場合を考える．logger コンポーネントは，クライアントからロギング処理に関するメッセージを受けとり，それをログとして記録するコンポーネントである．logger クライアントがプラグインする時には，クライアントからのコンポーネント探索に logger コンポーネントが応答することによって，クライアントから呼び出すことができるようになる (図 7).

次に，Web Services で実装されたコンポーネントが動作するサーバがプラグインする場合を考える．なおこの

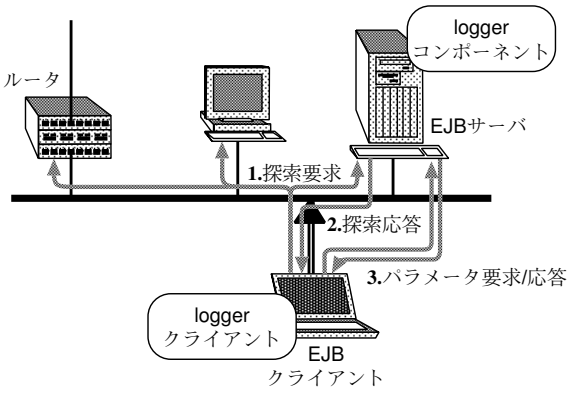


図 7: logger クライアントプラグイン時の動作

コンポーネントは，logger コンポーネントと統合することによって，詳細なログを記録することのできる別のコンポーネントを生成できる．プラグインするサーバには，あらかじめサーバ上で動作するコンポーネントが logger コンポーネントと統合して動作できることの記述が配置してある．そして，サーバがプラグインすると logger コンポーネントを探索する．それを発見すると，パラメータの探索を行ない，Web Services インタフェースを用いて logger コンポーネントを呼び出すことのできるアダプタを生成する．そしてそのアダプタと Web Services 上のコンポーネントを呼び出すことによって，それらを統合して動作することのできる新しいコンポーネントを生成する (図 8).

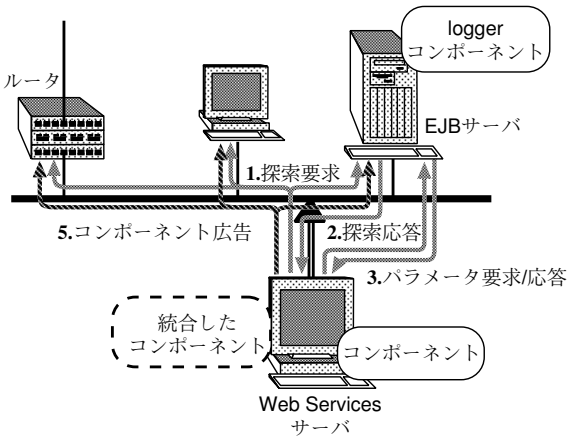


図 8: logger コンポーネントを統合する際の動作

6 まとめ

本研究では，異種分散コンポーネント対象に “Plug and Play” 環境を実現するための問題を分析し，実現に必要となるコンポーネントの探索機構を設計した．この機構では，異なった種類の分散コンポーネントを発見して呼び出すため，探索できるコンポーネントの実装アーキテクチャを特定しない．この環境を用いることによって，統合して動作できる異種分散コンポーネントを発見し，自動的に統合することが可能になる．

謝辞

本研究は，文部科学省科学技術振興調整費「環境情報獲得のための高信頼性ソフトウェアに関する研究」の支援による．

参考文献

[1] Alan Brown, “Large-Scale, Component-Based Development”, Prentice Hall PTR (2000).

[2] Linda G. DeMichiel, “Enterprise JavaBeans Specification, Version 2.1”, Sun microsystems, *http://java.sun.com/products/ejb/* (2000).

[3] Object Management Group, “Common Object Request Broker Architecture: Core Specification (3.0.3)”, OMG Document, *http://www.omg.org/technology/documents/formal/corba_2.htm* (2004).

[4] David Booth, Hugo Haas et al., “Web Services Architecture”, W3C Working Group Note, The World Wide Web Consortium, *http://www.w3.org/TR/2003/WD-ws-arch-20030808/* (2003).

[5] Object Management Group, “Naming Service Specification (1.2)”, OMG Document, *http://www.omg.org/docs/formal/02-09-02.pdf* (2002).

[6] Tom Bellwood, Luc Clement et al., “UDDI (3.0.1)”, UDDI Spec Technical Committee Specification, Organization for the Advancement of Structured Information Standards (OASIS), *http://uddi.org/pubs/uddi_v3.htm* (2003).

[7] John Beatty, Gopal Kakivaya et al., “Web Services Dynamic Discovery (WS-Discovery)”, Microsoft Corporation, *http://msdn.microsoft.com/ws/2004/02/discovery/* (2004).

[8] Yaron Y. Goland, Ting Cai et al., “Simple Service Discovery Protocol/1.0”, IETF Internet-draft $(expired)$, The Internet Engineering Task Force, *http://www.upnp.org/download/draft_cai_ssdp_v1_03.txt* (1999).

[9] Erik Guttman, Charles Perkins et al., “Service Location Protocol, Version 2”, RFC 2608, The Internet Engineering Task Force, *http://www.ietf.org/rfc/rfc2608.txt* (1999).

[10] Masataka Nagura, Tadashi Iijima, Shingo Takada and Norihisa Doi, “Automated Adapter Generation for Gluing Heterogeneous External Components”, Proc. of 14th International Conference on Software & Systems Engineering and their Applications (ICSSEA 2001), pp.1-8 (2001).