RTコンポーネントのデプロイメント標準: DDC4RTC

○安藤慶昭 (産総研), ジョン・スンウク (韓国電子通信研究院),
ビグズ・ジェフ (産総研), 神徳徹雄 (産総研)

Deployment standard for RT-Component: DDC4RTC

○ Noriaki ANDO (AIST), Seung-Wook Jung, (ETRI),
Geoffrey BIGGS (AIST), Tetsuo KOTOKU (AIST)

Abstract—
New standard specification draft for dynamic deployment and configuration for RT-Components[1] (DDC4RTC)[2] has been approved in OMG[3] in June 2012. This paper, the draft specification, which is in the finalization stage, is introduced about its concept and technical feature with related other standards.

Key Words: Standardization, RT-Component, deployment and configuration

1. はじめに

コンポーネント指向システム開発において、コンポーネントの配置 (デプロイ) と設定 (コンフィギュレーション) いわゆる D&C (Deployment and Configuration) は、実用的なシステムを運用するうえで必要不可欠な機能である。

例えば、Javaにおいては OSGi (Open Services Gateway initiative)[4] によるデプロイメントフレームワークが広く利用されている。また、CORBA および CCM (CORBA Component Model)[5] にはデプロイメントに関する仕様が含まれている。

コンポーネント指向のロボットシステムを構成する際にこれらを利用して、システムのデプロイおよびコンフィギュレーションを行うことは可能である。しかしながら、ロボットシステムにおいてたびたび発生する、イベントに基づく動的システム構成の変更をこれら既存の D&C の仕組みにより実現しようとすると、状態とそれに関連付けられた多数のシステム構成を管理する必要に迫られる。こうした仕組みは汎用的に構成することが可能であり、ここに動的なシステムデプロイメントとコンフィギュレーションの標準化が必要であることが理解できる。

本稿では、新たに承認された RTC のための動的デプロイメントおよびコンフィギュレーションのための標準 DDC4RTC について、標準仕様を構成する 4 つのパートについて技術的な内容をモデルと共に概説する。

2. D&C (Deployment and Configuration)

コンポーネント指向開発の基本的な考え方は、アプリケーションを小さい再利用可能なコンポーネントに分割し、それらあるいは既存のコンポーネントを相互にメッセージをやり取りできるように接続・設定することでシステムを構成することである。コンポーネント指向システムでは、運用時に利用するコンポーネントを適切なノードに配置し、各種パラメータの設定、コンポーネント間の接続を確立したうえで、システムをスタートさせる。

2·1 既存の D&C 標準

Java においては OSGi (Open Services Gateway initiative)[4] とこれを基盤としたデプロイメントフレームワークが広く利用されている。統合開発環境 Eclipse のプラグイン管理機能は OSGi を利用したものである。モジュールは Bandle と呼ばれ、Bandle の実行基盤を OSGi フレームワークと呼ぶ。Bandle は複数のクラスファイルとリソースファイルをアーカイブとして一つにまとめた jar ファイルである。Apache ACE は OSGi 仕様を基盤とし、コンポーネントの依存関係の管理や、ノードへのデプロイメント機能などが実装されている。

かつて CCM (CORBA Component Model)[5] と呼ばれていた仕様は現在 CORBA の仕様の一部として含まれている。CCM 仕様にはデプロイメントに関する様々な記述方式やサービスインターフェースに関する標準仕様が規定されている。これを CORBA に依存しない形で一般化した DEPL (Deployment And Configuration Of Component-Based Distributed Applications)[6] は CORBA 標準と共に OMG において標準化されている。

CORBA に類似の分散オブジェクトミドルウエアである Ice (Internet Communication Engine)[7] ではアプリケーションサーバおよびデプロイメントの仕組みが備わっている。IceBox は共有オブジェクトのコンポーネントをロードしインスタンス化するアプリケーションサーバである。IceGrid はコンポーネントを適切なサーバに配置するための分散システムマネージャである。

2·2 RTC 標準

コンポーネント指向システムの基盤となるコンポーネントモデルは多数あるが、著者らは OMG で標準化されたロボット用コンポーネントモデルである RTC (Robotic Component Specification)[1] を前提とした。RTC 標準においては、コンポーネントの本体、コンポーネント間の相互作用の起点となるポート、コンポーネント内のビジネスロジックの能動的実行をつかさどる実行セマンティクスおよび、コンポーネントの動的システム構成変更に利用可能なメタ情報を取得するためのイントロスペクション (Introspection) 機能が規定されている。OMG RTC の実装としては、産総研の OpenRTM-

Table 1 OMG RTC standard compliant implementations.

Implementation	Vendor	Feature
OpenRTM-aist	AIST	C++, Python, Java
OpenRTM.NET	SEC	.NET (C#,VB,C++/CLI, F#, etc..)
miniRTC, microRTC	SEC	CAN and ZigBee based communication
RTMSafety	SEC/AIST	IEC61508 capable implementation
RTC CANOpen	SIT, CiA	Standardized in CiA (CAN in Automation)
OPRoS	ETRI	Developed in Korean national project
GostaiRTC	GOSTAI, THALES	C++ implementation on URBI

aist[8, 9] や ETRI の PRoS[10] など表 1 に示す多くの実装が存在する。

2·3 ロボットシステムにおける D&C

ロボットシステムでは、実世界を含めた事象の変化をトリガとして、システム構成を動的に変更することがしばしば求められる。例として、図 1 に示すように、サービスロボットが RoomA から RoomB へ移動した際、利用可能な環境側のセンサの数や種類が変化する。また、ユーザからのリクエストにより、適切なセンサを選択的に利用し、タスクを実行すると仮定する。コンポーネント指向で構成されたこのようなシステムにおいては、部屋の移動というイベントに伴い、ロボットが利用するセンサ群を入れ替える必要があり、またユーザの要求などによっても、ロボットと他のデバイスやセンサの接続関係を変更することが必要となる。

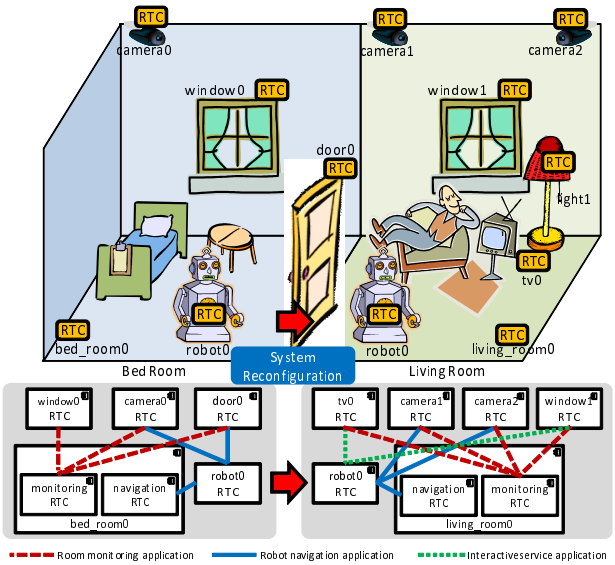

Fig.1 An example of RTC-based system required dynamic system reconfiguration.

従来のコンポーネント指向システムでは、ロボットシステムほど動的なシステム構成の変更を想定しておらず、これを実現する場合アプリケーション側において多数のコンポーネント制御しつつシステム構成を変更する必要がある。

3. DDC4RTC 標準仕様

上述した理由から、ロボットシステムのための動的な D&C 標準が必要であると考え、著者らは OMG の Robotics Domain Taskforce 内の Infrastructure Working Group において標準化活動を行ってきた。

3·1 仕様概要

DDC4RTC 仕様は、DEPL 標準仕様を継承して作成されている。DEPL は ComponentDataModel, ComponentManagementModel, TargetDataModel, Target ManagementModel, ExecutionDataModel, ExecutionManagementModel の 6 つのパッケージから構成されている。DDC4RTC はこのうち、ComponentDataModel, ComponentManegementModel, ExecutionDataModel, ExecutionManagementModel の 4 つを継承、拡張して動的特徴を付加した (図 2)。

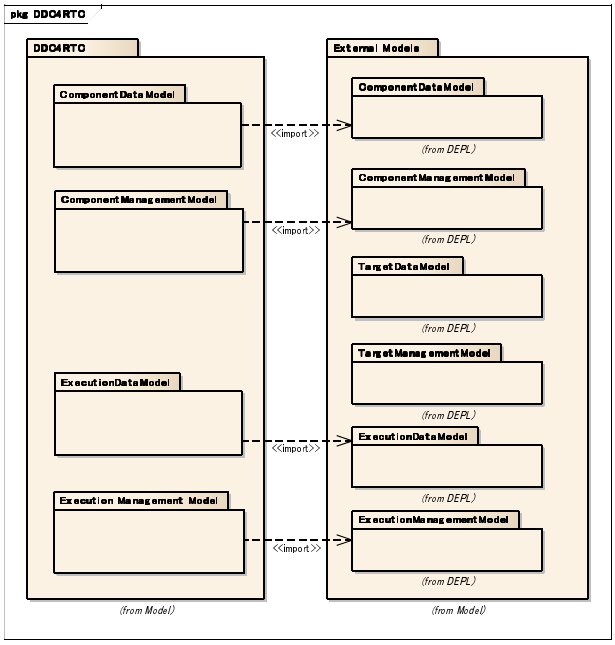

Fig.2 Pacakge Diagram of DDC4RTC specification.

以下、拡張したそれぞれのパッケージの代表的な特徴について述べる。

3·2 Component Data Model

Component Data Model パッケージの主なクラス図を図 3 に示す。Component Data Model は、デプロイメントに必要なコンポーネントの関連情報やプロファイル、RT システムアセンブリについて記述するためのデータ構造である。これらは DEPL 仕様に基づいて定義されている。

RTC 仕様と DEPL 仕様で想定するポートの意味およびデータ構造が異なる。図 4 に示すように、DEPL では、

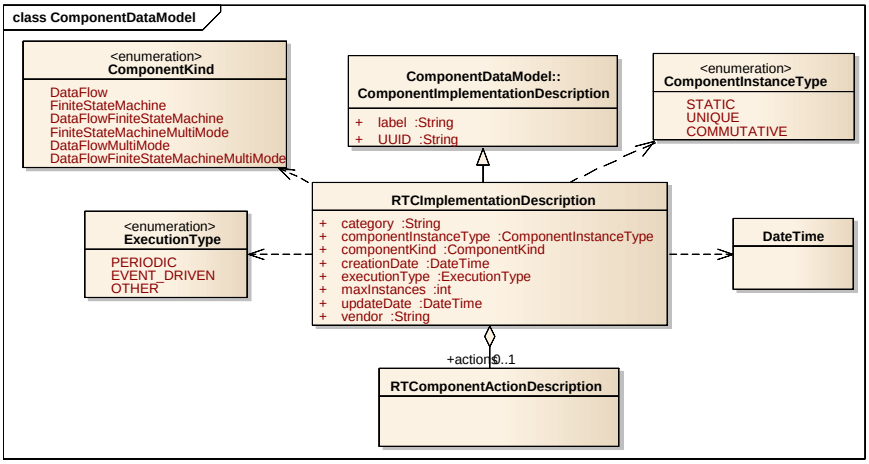

Fig.3 RTCImplementationDescription and its related classes.

ポートそれ自身が Provided/Required インターフェースの一種であり、接続時には相互のポート自身が直接接続される。一方 RTC のポートは UML で定義されているポートの概念に近く、ポートは接続を管理する一種のサービスであり、ポートに所属する Provided/Required インターフェースが接続される。

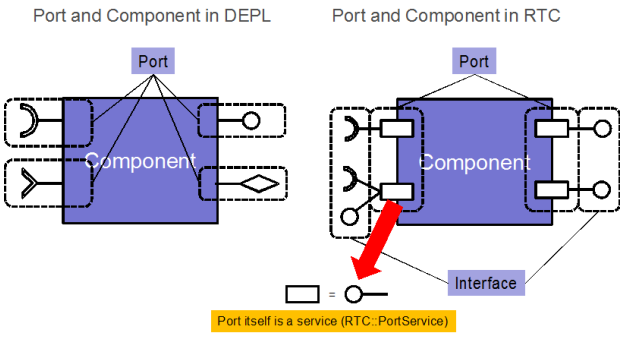

Fig.4 Port and component model difference between DEPL and RTC.

こうした違いを吸収するために、図 5 に示すように DEPL で定義されているポートの記述方式を、RTC のポートを記述できるよう拡張した。

3·3 Component Management Model

Component Management Model はコンポーネントの配置を管理するある種のサービスを提供する。RepositoryManager は RTC の格納、検索、取出しを行うためのインターフェースを提供する。RepositoryManager は RTC ベースのシステムの格納、検索、取出しを行うためのインターフェースも提供する。

DEPL RepositoryManager はより柔軟な検索方法を実現するために拡張された。もともとのサービスに対して、`search()` オペレーションが追加され、この関数に与えられるクエリ文字列は ISO/TC211 Geographic Information-filter encoding (ISO reference number: 19143) 標準に基づくものとなっている。

3·4 Execution Data Model

DEPL 仕様の Execution Data Model における主たる拡張は SupervisorFSM である。SupervisorFSM は RTC ベースのシステムを事前に定義された状態マシンに従って管理するアプリケーションの管理者である。SupervisorFSM モデルは UML State Machine (Behavior State Machine) に基づいている。

SupervisorFSMDescription は RTC ベースのシステムアプリケーションの振る舞いを有限状態マシンに基づいて定義する。各状態は、その状態の時にシステムがとる構成を記述するデプロイのプロファイルと関連付けられ記述される。また、状態間の遷移はそれを発生させるイベントと共に記述される (図 6)。

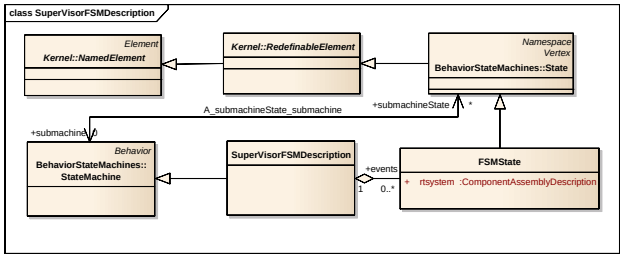

Fig.7 SupervisorFSMDescription and FSMState class diagrams.

3·5 Execution Management Model

DDC4RTC の Event Management Model は環境やユーザからのイベントを RTC ベースのシステムに通知したり、事前に与えられた条件に従ってイベントフィルタリングを行ったりするための、ある種のサービスを提供する。 イベント通知の気候は別の OMG 標準 Notification Service Specification[16] に準拠している。

DEPL の Execution Management Model に対する主な拡張は、ApplicationSupervisor である。DDC4RTC における SpplicationSupervisor は Notification Service Specification で 定義されている StructuredPushConsumer を継承する (Figure 8)。

ApplicationSupervisor は RTC ベースのシステムを構成するすべての RTC のライフサイクルを管理する。発生したシステムに関するイベントを受け取り、SupervisorFSMDescription によりあらかじめ定義された状態遷移に基づき状態を遷移させるとともに、遷移時には状態に関連付けられたシステムのデプロイとコンフィギュレーションを行う。ApplicationSupervisor は

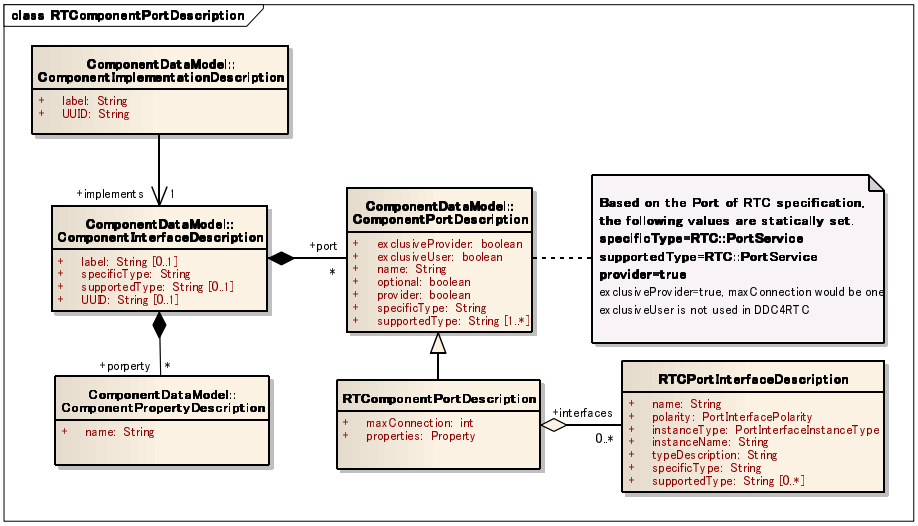

Fig.5 RTComponentPortDescription class diagram.

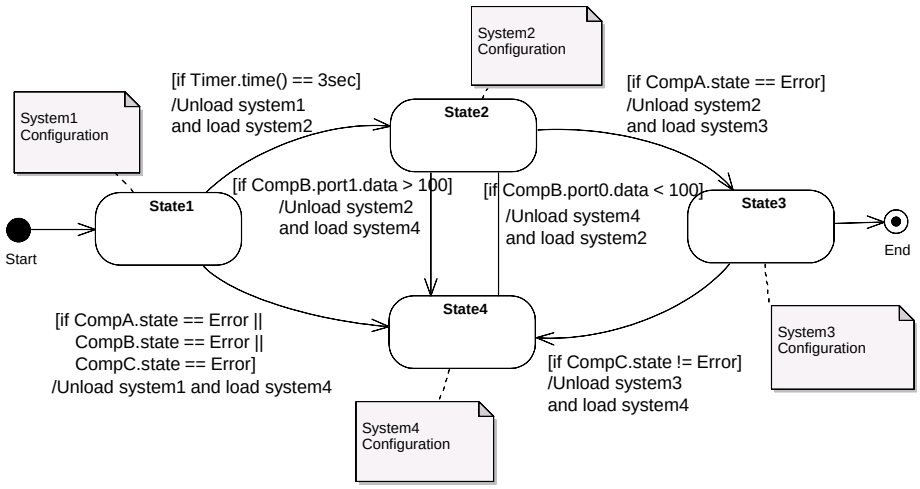

Fig.6 FSM of SupervisorFSM and systems associated with each state.

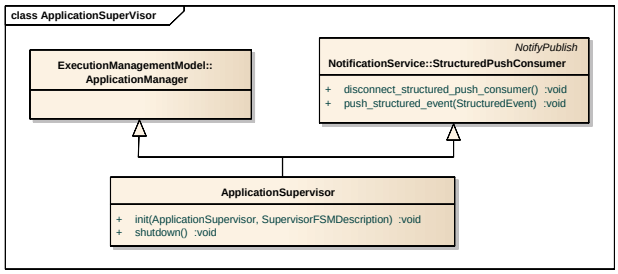

Fig.8 ApplicationSupervisor class diagram.

コンポーネント管理のために DEPL の ApplicationManager を継承しており、一方イベントハンドリングのために、Notification Service Specification の StructuredPushConsumer を継承している。

ApplicationSupervisor は SupervisorFSMDescription で定義された有限状態マシンに基づき動作する。初めに、親の ApplicationSupervisor および SupervisorFSMDescription 与えて初期化を行う。その後、ApplicationManager の `startLaunch()` オペレーションを呼ぶことにより、動作を開始する。動作の開始においては、ApplicationSupervisor が SupervisorFSMDescription に指定された初期状態に遷移し、指定されたデプロイとコンフィギュレーションを実行する。

一旦初期状態のデプロイが終了すると、ApplicationSupervisor は待ち状態に入る。イベントが通知されると起きてイベントの処理を行う。イベントを受け取ると現在の状態から次に遷移するとともに、現在の状態に関連付けられたシステム構成を破棄し、次の状態に関連付けられたシステム状態をデプロイおよびコンフィギュレーションを行ったあとにスタートさせる。遷移時に行われるプロセスを図 9 のシーケンス図に示す。

もし、状態遷移が有効ならば、ApplicationSupervisor は指定されたシステム構成のデプロイを実施しその状態へ遷移する。もし、遷移によって終了状態へ到達した場合、ApplicationSupervisor はシャットダウン処理を行う。RTC ベースシステム内からすべてのコンポーネントを削除し、コンポーネントの終了処理を初期化処理とは逆の順序で行う。

ApplicationSupervisor はイベントを処理するために必要なサービスである。もし、SupervisorFSMDescription に指定されていないイベントが発生した場合、ApplicationSupervisor はエラーで終了する。

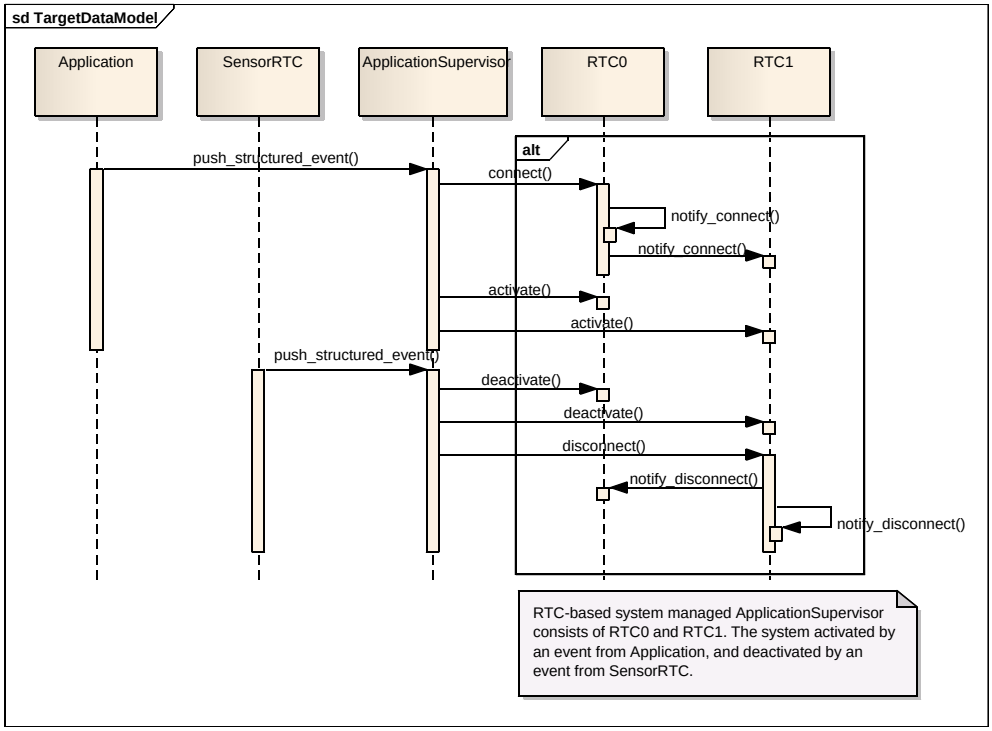

Fig.9 ApplicationSupervisor sequence diagram.

4. 結言

本稿では、現在 OMG において標準化された、動的デプロイメントおよびコンフィギュレーションの標準を紹介した。当標準仕様は他の OMG のコンポーネントモデル標準である RTC のための D&C 仕様であり、もう一つの OMG 標準である一般的な D&C 標準仕様 DEPL 仕様に基づいている。

ロボットシステムのための動的特徴のために、既存の DEPL 仕様を拡張し、システム記述方式としての SupervisorFSM、および動的システム再構成サービスとして ApplicationSupervisor を追加した。RTC 特有の属性やポートの構造を記述するための RTCImplementationDescription 等を追加した。既存の標準仕様を再利用し拡張することで、仕様の多くの部分を共有することができ、拡張部分を最小限にすることができた。

現在、FTF のステージであり最終レポートに向けて仕様の修正を行う予定である。したがって、最終仕様は現在の DDC4RTC Specification Beta1 から変更される可能性があることに注意されたい。

参考文献

[1] Robotic Technology Component Specification Ver. 1.1, OMG Specification formal/2012-09-01, http://www.omg.org/spec/RTC/1.1/

[2] Dynamic Deployment and Configuration for Robotic Technology Component (DDC4RTC), OMG Specification ptc/2012-08-33, http://www.omg.org/spec/DDC4RTC/1.0/Beta1/

[3] Objet Management Group, http://www.omg.org

[4] OSGi Release 5 specification, http://www.osgi.org/Specifications/

[5] Common Object Request Broker Architecture (CORBA) Specification, Version 3.1, OMG Specification formal/2008-01-04,07,08, http://www.omg.org/spec/CORBA/3.1/

[6] Deployment And Configuration Of Component-Based Distributed Applications (DEPL) Specification, OMG Specification formal/06-04-02, http://www.omg.org/spec/DEPL/

[7] Ice (Internet Communication Engine), ZeroC Inc., http://www.zeroc.com/

[8] Noriaki ANDO, Takashi SUEHIRO, Kosei KITAGAKI, Tetsuo KOTOKU, Woo-Keun Yoon, "RT-Middleware: Distributed Component Middleware for RT (Robot Technology)", 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS2005), pp.3555-3560, 2005.08

[9] OpenRTM-aist official site, http://openrtm.org

[10] OPRoS (Open Platform for Robotic Services), http://opros.or.kr/

[11] Common Object Request Broker Architecture (CORBA) Specification, Version 3.1, Part1: CORBA Interfaces, OMG Specification formal/2008-01-04, http://www.omg.org/spec/CORBA/3.1/

[12] Common Object Request Broker Architecture (CORBA) Specification, Version 3.1, Part2: CORBA Interoperability, OMG Specification formal/2008-01-07, http://www.omg.org/spec/CORBA/3.1/

[13] Common Object Request Broker Architecture (CORBA) Specification, Version 3.1, Part3: CORBA Component Model, OMG Specification formal/2008-01-08, http://www.omg.org/spec/CORBA/3.1/

[14] Apache ACE project, http://ace.apache.org/

[15] Unified Modeling Language version 2.4.1, OMG specification formal/2011-08-05, formal/2011-08-06, http://www.omg.org/spec/UML/

[16] Notification Service Specification, OMG Specification formal/04-10-11, http://www.omg.org/spec/NOT/