ロボットミドルウエア

安藤慶昭
国立研究開発法人産業技術総合研究所
ロボットイノベーション研究センター
ロボットソフトウェアプラットフォーム研究チーム長

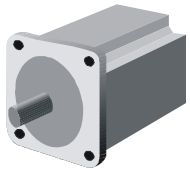
概要

- ロボットとソフトウェア
- ロボットミドルウエアとは？
- RTミドルウエア：OpenRTM-aist
- 様々なミドルウエア/プラットフォーム
- 終わりに

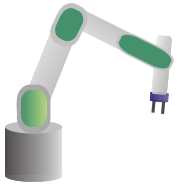
ロボットを構成する要素

- センサ
  - エンコーダ
  - 力・磁気・加速度など
- アクチュエータ
  - モータなど
- 機構（メカ）

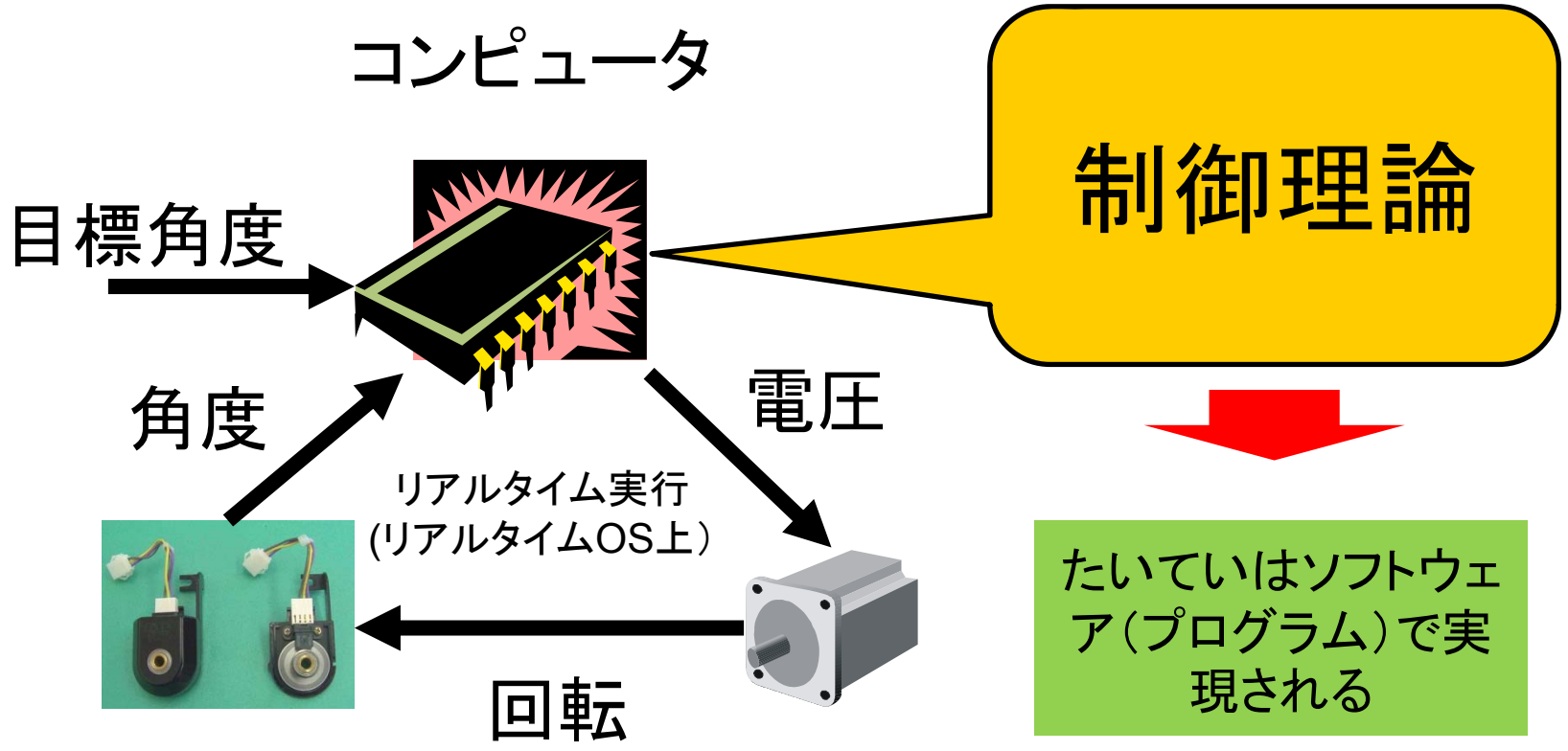
  ＋
- コンピュータ
  - ソフトウエア

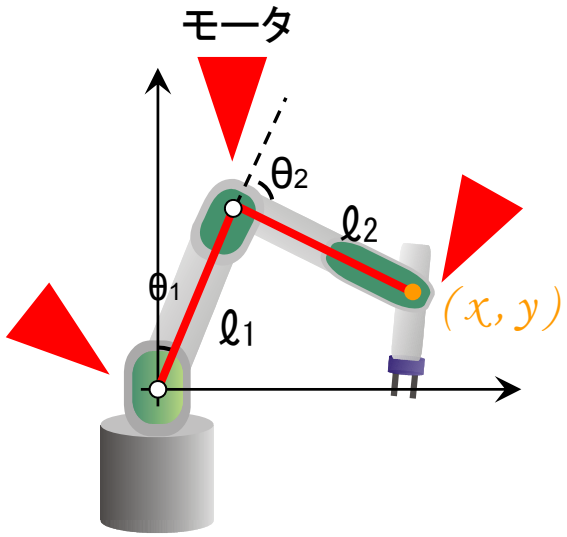





フィードバック制御

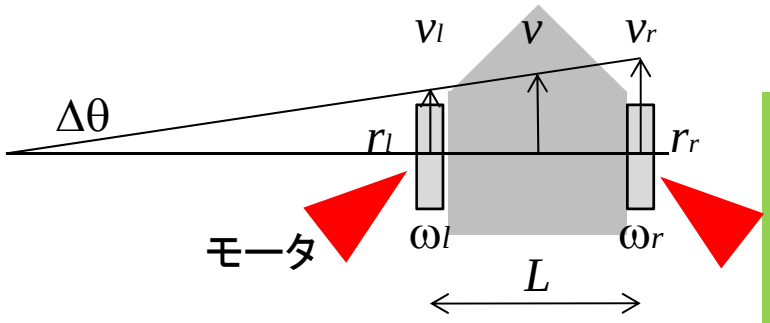

コンピュータ
制御理論
目標角度
角度
電圧
リアルタイム実行
(リアルタイムOS上)
回転
たいていはソフトウェア(プログラム)で実現される
ロータリーエンコーダ
(角度を読むセンサ)
モータ

運動学



ロボットアームの運動学

運動学（kinematics）とは位置の移動とその時間変化を対応させて考える学問であり、数理的な手法であり、物理学の原理を基礎としていない。一般的に物体の状態は、移動と回転の運動学の両方を兼ね合わせて記述する。

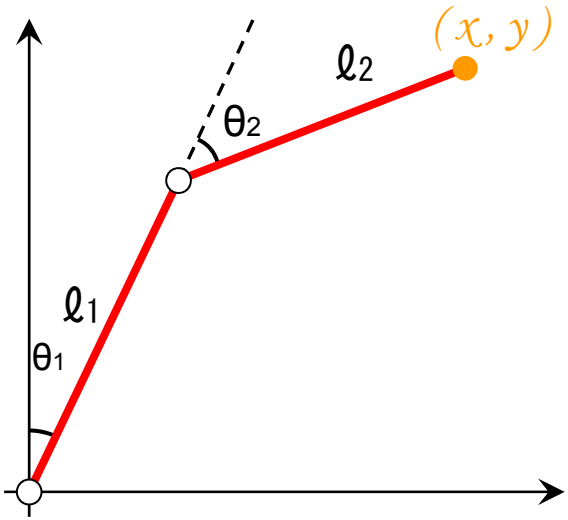


移動ロボットの運動学

ロボットの機構（メカ）の運動を
数式で（プログラムとして）記述する

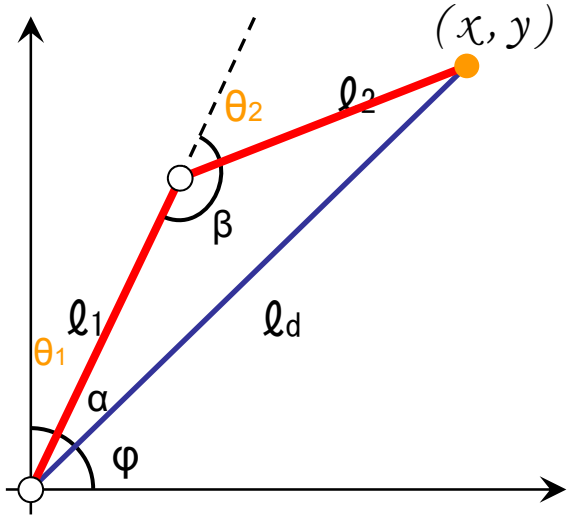
順運動学・逆運動学(2自由度の場合)

- 順運動学



2つの関節の角度がわかっているとき、XY座標を求めることができる

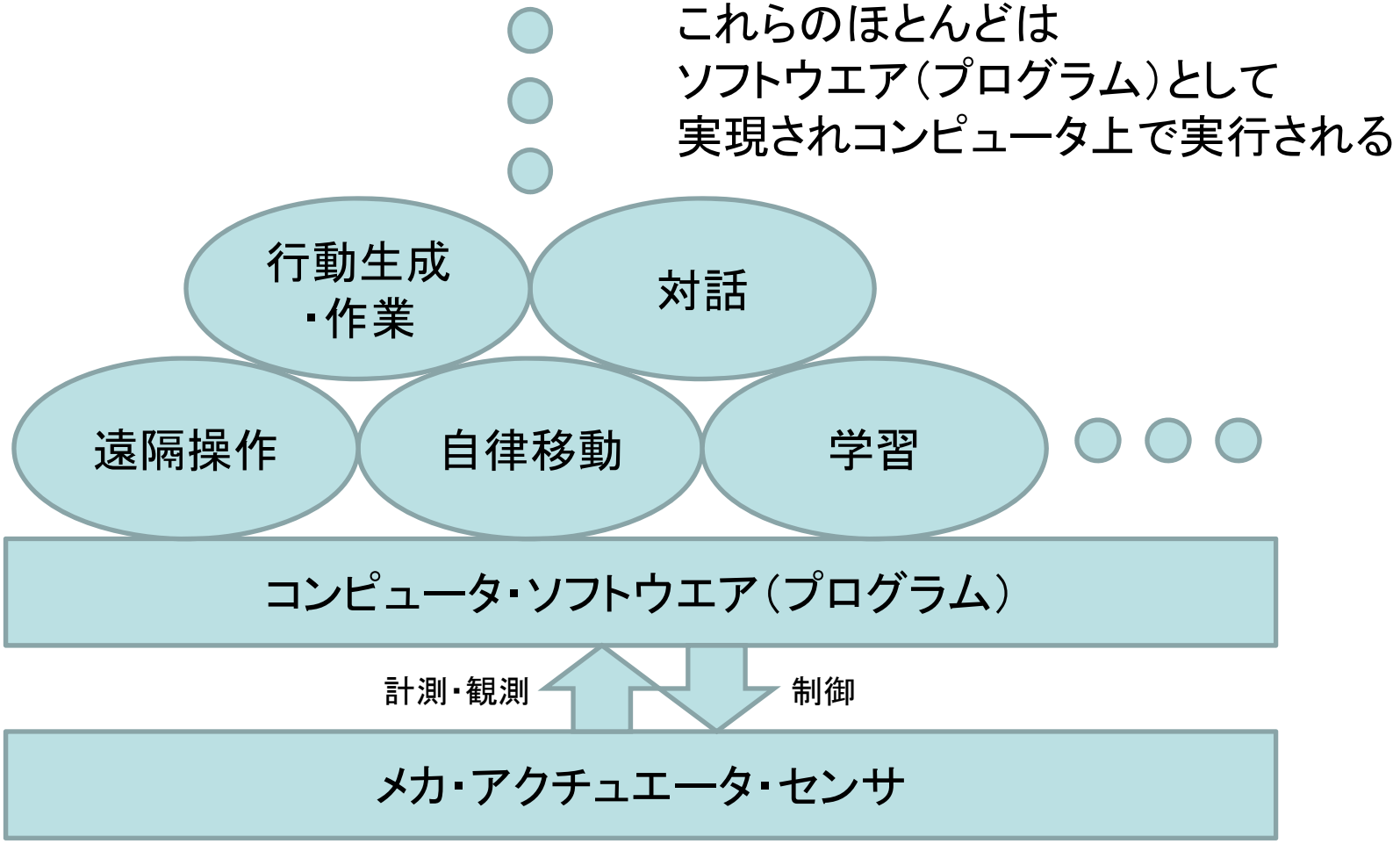
- 逆運動学



XY座標がわかっているとき、関節の角度を求めることができる

注:どの場合も腕の長さははじめから与えられている

ロボットを構成する要素

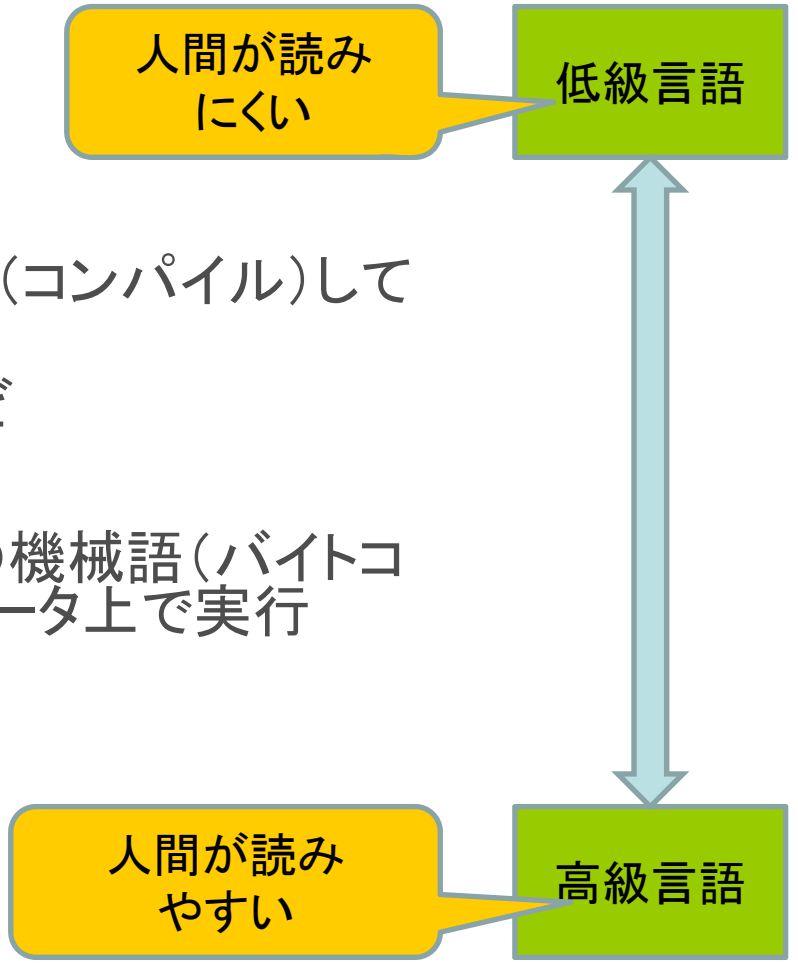


プログラム

- プログラムとは？
  - コンピュータにやらせたいことを書いた手順書
    - アルゴリズム
  - コンピュータは機械語（マシン語）しか理解できない
- プログラミング言語
  - プログラムはプログラミング言語で書く
  - プログラミング言語にはいろいろな種類がある
  - それぞれ長所・短所がある（適材適所）

プログラミング言語

- 機械語
- アセンブリ言語
- コンパイル型言語
  - プログラムをすべて機械語に翻訳(コンパイル)してから実行
  - C、C++、FORTRAN、Pascal など
- 中間言語型
  - プログラムを仮想コンピュータ用の機械語(バイトコード)に翻訳してから仮想コンピュータ上で実行
  - Java、.NET(C#など)
- インタプリタ型
  - プログラムを一行ずつ実行
  - Ruby、Python、Perl、BASIC



プログラミングパラダイム(枠組)

- オブジェクト指向プログラミング
- 関数型プログラミング
- ジェネリックプログラミング
- データ指向プログラミング
- 構造化プログラミング - 非構造化プログラミング
- 命令型プログラミング - 宣言型プログラミング
- メッセージ送信プログラミング - 命令型プログラミング
- 手続き型プログラミング - 関数型プログラミング
- 値レベルプログラミング - 関数レベルプログラミング
- 逐次実行型プログラミング - イベント駆動型プログラミング
- スカラプログラミング - ベクトルプログラミング
- クラスベースプログラミング - プロトタイプベースプログラミング - Mixin
- 制約プログラミング - 論理型プログラミング
- コンポーネント指向プログラミング
- アスペクト指向プログラミング
- パイプラインプログラミング
- 課題指向プログラミング
- リフレクティブプログラミング
- データフロープログラミング (スプレッドシート)
- ポリシーベースプログラミング
- ツリープログラミング
- 註釈プログラミング
- 属性指向プログラミング
- コンセプト指向プログラミング

プログラミング言語を学ぶ

- 文法を覚える
- コンパイル、実行の方法を覚える
  - 実際にプログラムを書いてみる
- 関数やライブラリの使い方を覚える
  - いっぺんに覚える必要はない、少しずつ
- デバッグの方法を覚える
- 人のプログラムを読む
  - 読むことは書くことより難しい！

いろいろな言語のHello World

C++

```
#include <cstdlib>
#include <iostream>
int main ()
{
  std::cout << "Hello, world!" << std::endl;
  return EXIT_SUCCESS;
}
```

いろいろな言語のHello World

Python

```
print "Hello, world!“
```

（Version 3以降は）

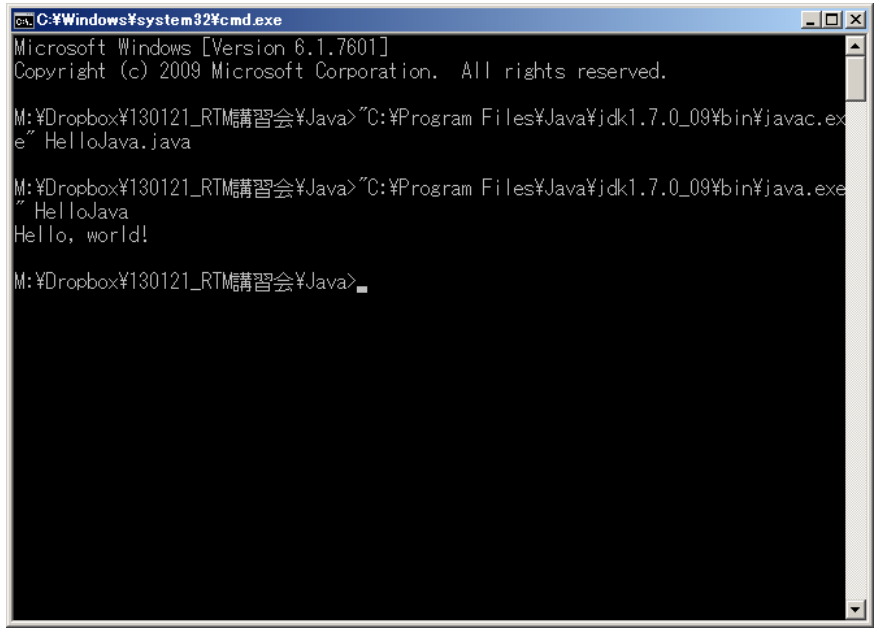
```
print(“Hello, world!”)
```

いろいろな言語のHello World

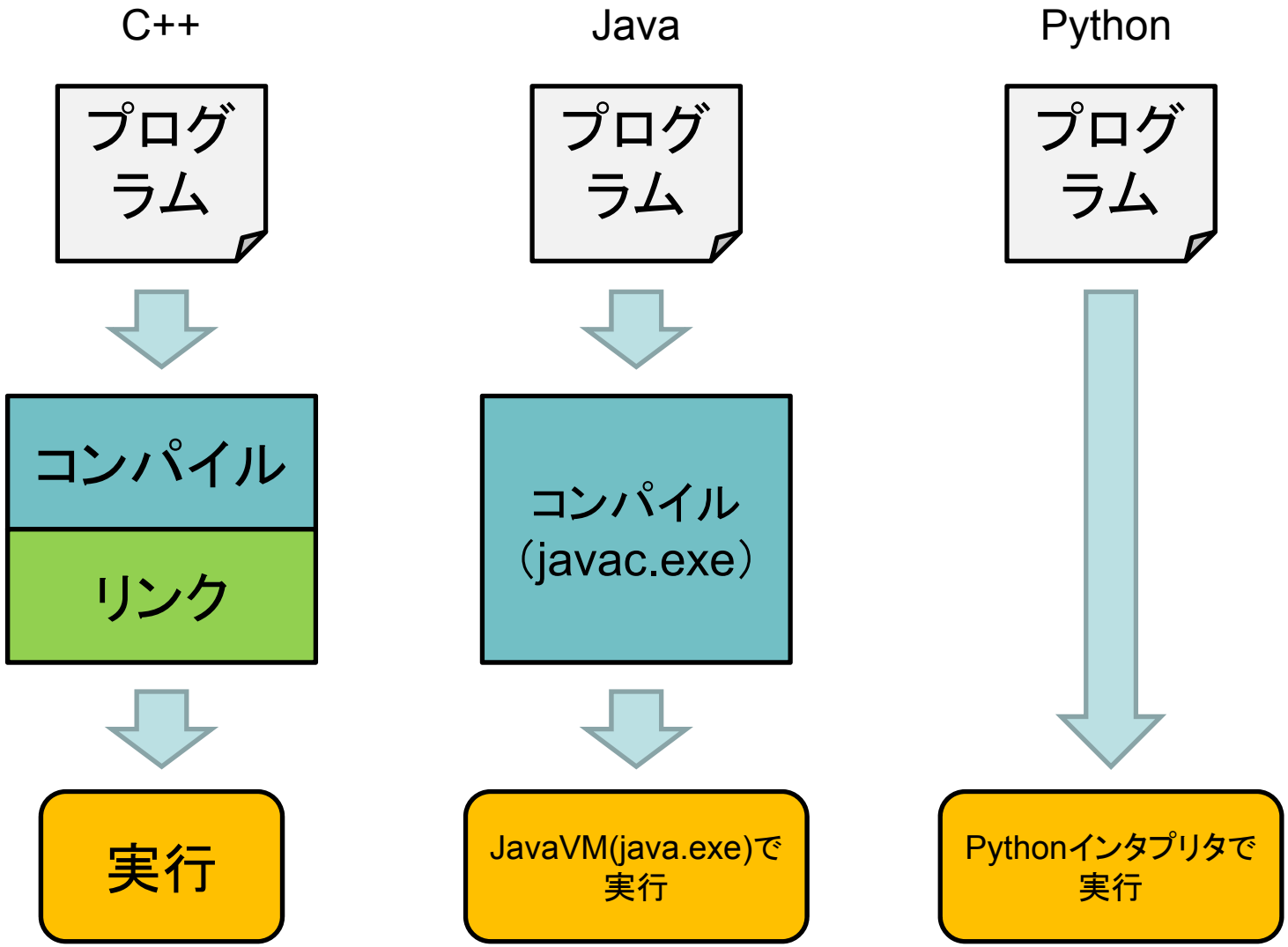
Java

```
import java.awt.*;
import java.awt.event.*;

public class HelloFrame extends Frame {
 HelloFrame(String title) {
  super(title);
 }
 public void paint(Graphics g) {
  super.paint(g);
  Insets ins = this.getInsets();
  g.drawString("Hello, World!", ins.left + 25, ins.top + 25);
 }
 public static void main(String[] args) {
  HelloFrame fr = new HelloFrame("Hello");

  fr.addWindowListener(
   new WindowAdapter() {
    public void windowClosing(WindowEvent e) {
     System.exit(0);
    }
  }
 );
 fr.setResizable(true);
 fr.setSize(500, 100);
 fr.setVisible(true);
 }
}
```

```
class HelloJava {
  public static void main(String[] args) {
  System.out.println(“Hello, world!");
  }
}
```



プログラミングの流れ


C++
Java
Python
プログラム
プログラム
プログラム
コンパイル
リンク
コンパイル
（javac.exe）
実行
JavaVM(java.exe)で実行
Pythonインタプリタで実行

プログラミング言語の特徴

	実行速度	リアルタイム実行	コンパイル	オブジェクト指向	手軽さ
C/C++	◎	◎	必要	△／○	△
Java	○	△	必要	◎	○
Python	×	×	不要	○	◎

	移植性	動的処理	GUIの作りやすさ	学習のしやすさ	大規模開発
C/C++	△	×	×	△／×	○／◎
Java	○	△	△	○	◎
Python	○	○	○	◎	△

用途・目的に応じて適切な
プログラミング言語を選択することが重要

オブジェクト指向プログラミング

オブジェクト指向プログラミング

- オブジェクト指向とは
  - カプセル化（≒モジュール化）
    - 処理の詳細とデータの隠蔽
  - インヘリタンス
    - 機能の継承や拡張
  - ポリモーフィズム
    - 多態性と共通化
  - ダイナミックバインディング
    - 型の違いによる処理の差別化

とりあえず

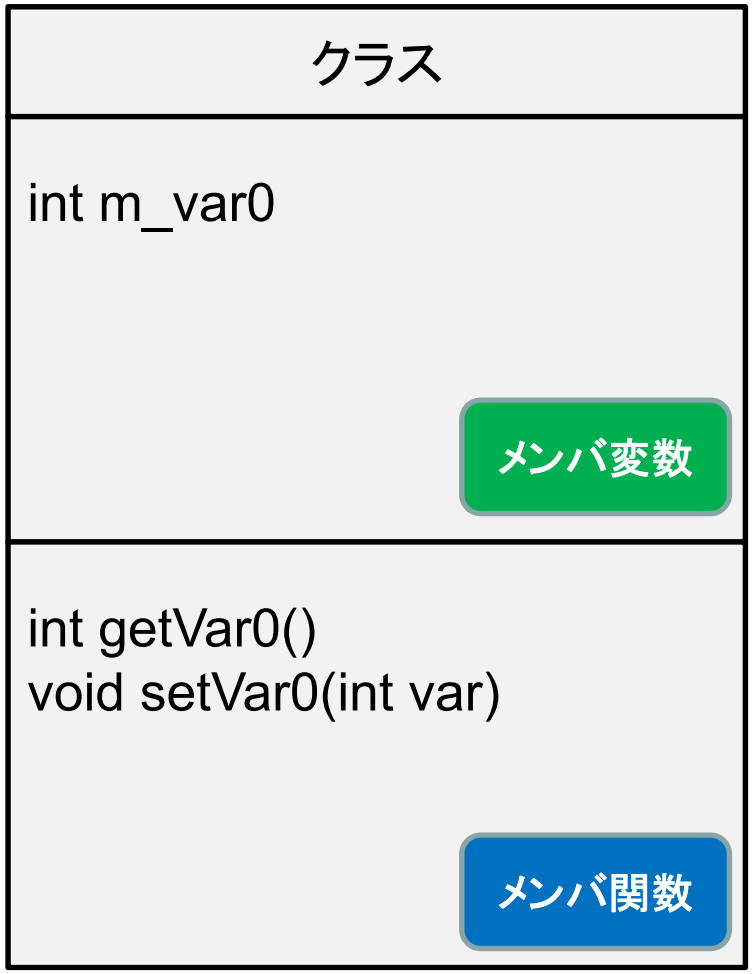
オブジェクト指向＝カプセル化

と覚えておけば十分

クラスの概念

クラス ＝ データ ＋ 処理

- メンバ変数
  - インスタンスと同時に生成される変数
  - オブジェクトの状態を保持
  - 通常、オブジェクトの外からは見えないようにする（隠蔽）
- メンバ関数
  - オブジェクトに作用を及ぼす
    - メンバ変数の値を変える
    - オブジェクトに処理をさせて結果を得る



例:なぜ変数を隠蔽するのか

- 実装の隠蔽
- 実装と振る舞いの分離
- 依存性の排除
- 隠蔽レベルの制御(C++の例)
  - public
  - protected
  - private

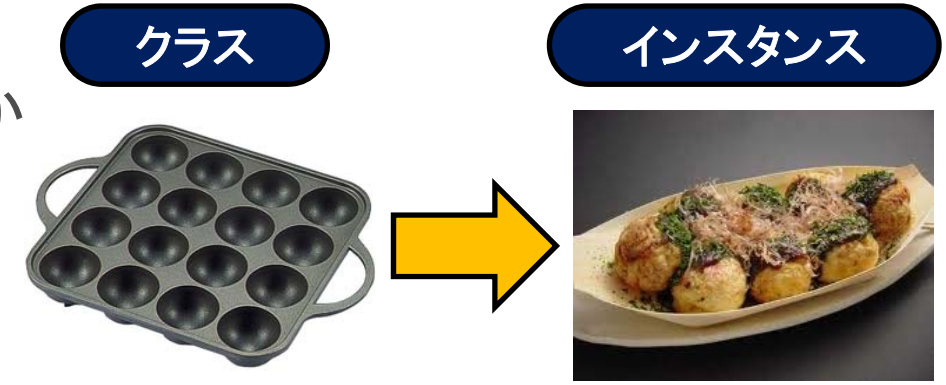
2つの setVar の例

```
void setVar0(int var)
{
  m_var0 = var;
}
```

```
void setVar0(int var)
{
  if (var > 100) { m_var0 = 100; }
  else if (var < 0) {m_var0 = 0; }
  else { m_var0 = var; }
}
```

クラスとインスタンス

- クラス
  - オブジェクトを作るための型
  - メンバ変数、メンバ関数から構成される
- インスタンス
  - オブジェクトとも呼ぶ
    - (厳密には両者は違うが…)
  - クラスを具現化したもの
  - クラス：たこ焼きの型
  - インスタンス：たこ焼き



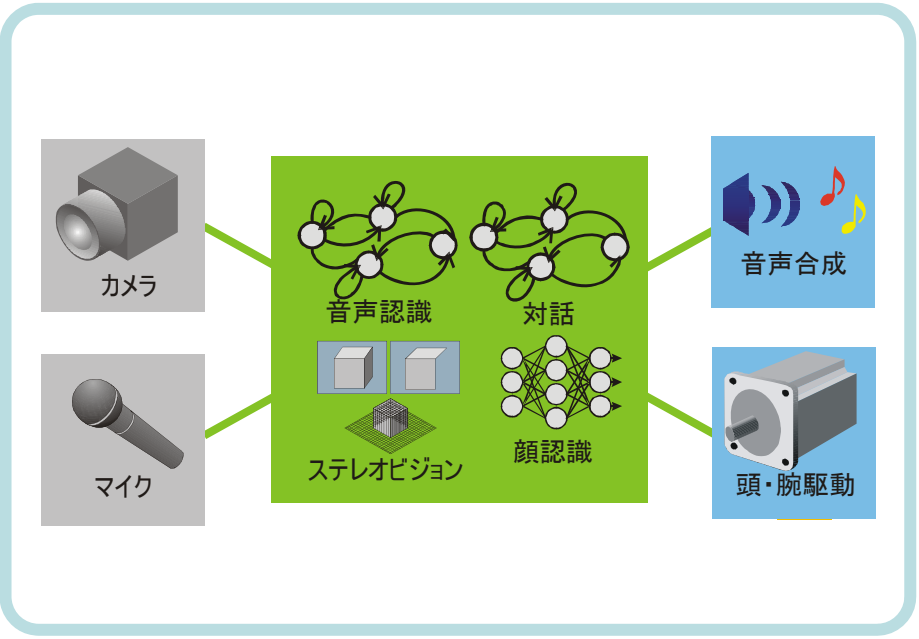
ロボットミドルウェアとは？

ロボットミドルウエアとは？

- ロボットシステム構築を効率化するための共通機能を提供する基盤ソフトウエア
  - 「ロボットOS」と呼ばれることもある
  - インターフェース・プロトコルの共通化、標準化
  - 例として
    - モジュール化・コンポーネント化フレームワークを提供
    - モジュール間の通信をサポート
    - パラメータの設定、配置、起動、モジュールの複合化（結合）機能を提供
    - 抽象化により、OSや言語間連携・相互運用を実現
- 2000年ごろから開発が活発化
  - 世界各国で様々なミドルウエアが開発・公開されている

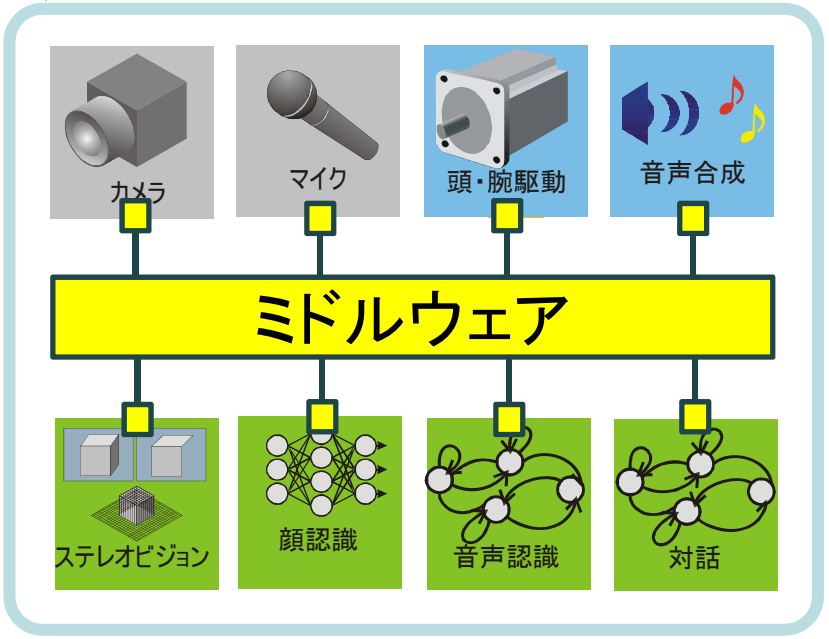
ロボットソフトウェア開発の方向

従来型開発



- ✓ 様々な機能を融合的に設計
- ✓ 実行時の効率は高いが、柔軟性に欠ける
- ✓ システムが複雑化してくると開発が困難に

コンポーネント指向開発



- ✓ 大規模複雑な機能の分割・統合
- ✓ 開発・保守効率化（機能の再利用等）
- ✓ システムの柔軟性向上

モジュール化のメリット

- 再利用性の向上
  - 同じコンポーネントをいろいろなシステムに使いまわせる
- 選択肢の多様化
  - 同じ機能を持つ複数のモジュールを試すことができる
- 柔軟性の向上
  - モジュール接続構成かえるだけで様々なシステムを構築できる
- 信頼性の向上
  - モジュール単位でテスト可能なため信頼性が向上する
- 堅牢性の向上
  - システムがモジュールで分割されているので、一つの問題が全体に波及しにくい

ミドルウエア、コンポーネント、etc…

- ミドルウエア
  - OSとアプリケーション層の中間に位置し、特定の用途に対して利便性、抽象化向上のために種々の機能を提供するソフトウエア
  - 例：RDBMS、ORB等。定義は結構曖昧
- 分散オブジェクト（ミドルウエア）
  - 分散環境において、リモートのオブジェクトに対して透過的アクセスを提供する仕組み
  - 例：DDS、CORBA、Ice、Java RMI、DCOM等
- コンポーネント
  - 再利用可能なソフトウエアの断片（例えばモジュール）であり、内部の詳細機能にアクセスするための（シンタクス・セマンティクスともにきちんと定義された）インターフェースセットをもち、外部に対してはそのインターフェースを介してある種の機能を提供するモジュール。
- CBSD（Component Based Software Development）
  - ソフトウエア・システムを構築する際の基本構成要素をコンポーネントとして構成するソフトウエア開発手法

様々なミドルウエア/プラットフォーム

- OpenRTM-aist
- ROS
- OROCOS
- OPRoS
- ORCA
- Microsoft Robotic Studio
- Player/Stage/Gazebo
- ORiN
- RSNP
- UNR Platform
- OPEN-R
- Open Robot Controller architecture

RT-Middleware
OpenRTM-aist

RTとは?

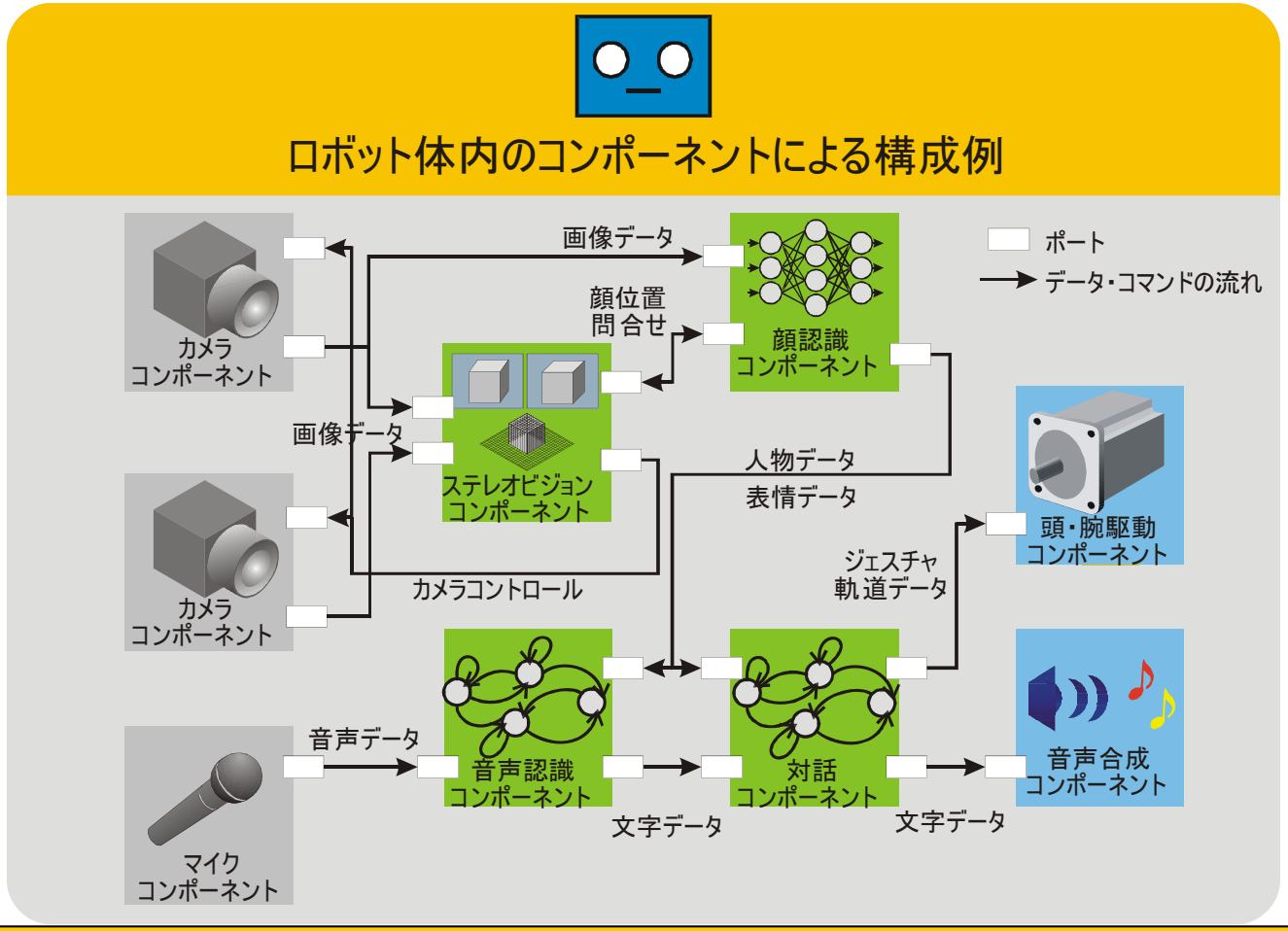
- RT = Robot Technology　cf. IT
  - ≠Real-time
  - 単体のロボットだけでなく、さまざまなロボット技術に基づく機能要素をも含む (センサ、アクチュエータ, 制御スキーム、アルゴリズム、etc.…)

産総研版RTミドルウエア

OpenRTM-aist

- RT-Middleware（RTM)
  - RT要素のインテグレーションのためのミドルウエア
- RT-Component（RTC)
  - RT-Middlewareにおけるソフトウエアの基本単位

ロボットシステムの構造



多くの機能要素（コンポーネント）から構成される

RTミドルウエアプロジェクト

NEDO 21世紀ロボットチャレンジプログラム（2002-2004年度）
「ロボット機能発現のために必要な要素技術開発」

- RT分野のアプリケーション全体に広く共通的に使われる機能およびRT要素の部品化（モジュール化）の研究開発
- 分散オブジェクト指向システムのミドルウェアであるCORBAをベースとして行う。
- RT要素の分類を行い、モジュール化の形態、必要な機能、課題、インタフェース仕様などを明確にする。

１４年度成果報告書より

従来のシステムでは…

Joystick

Joystick
software

Robot Arm
Control software

Robot Arm

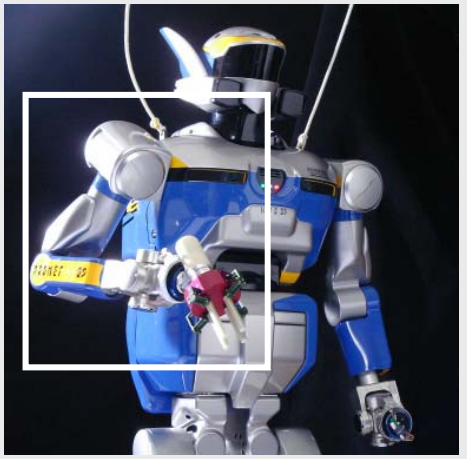
互換性のあるインターフェース同士は接続可能

従来のシステムでは…

Joystick

Joystick
software

Humanoid's Arm
Control software



Humanoid's Arm

Robot Arm
Control software

Robot Arm

ロボットによって、インターフェースは色々
互換性が無ければつながらない

RTミドルウエアでは…

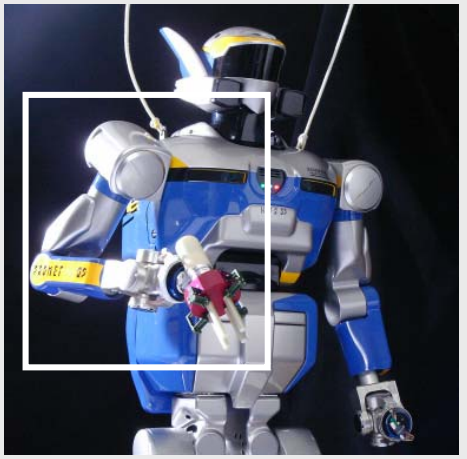
RTミドルウエアは別々に作られたソフトウエアモジュール同士を繋ぐための共通インターフェースを提供する

Joystick

Joystick software

compatible arm interfaces

Arm A Control software



Humanoid's Arm

Arm B Control software

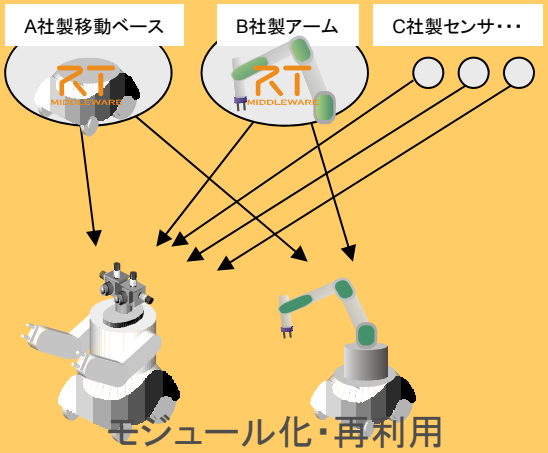
Robot Arm

ソフトウエアの再利用性の向上
RTシステム構築が容易になる

RTミドルウエアの目的

モジュール化による問題解決

コストの問題

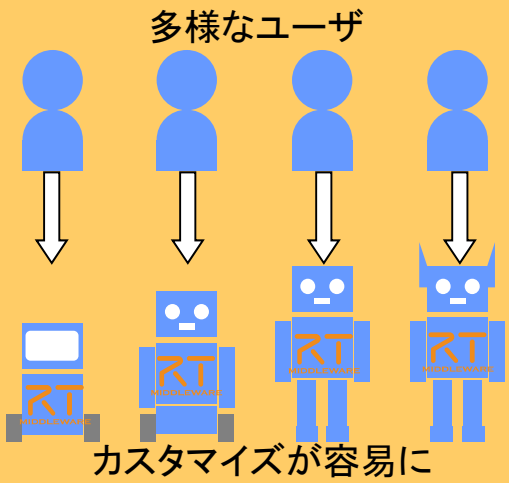


ロボットの低コスト化

技術の問題

最新技術を利用可能

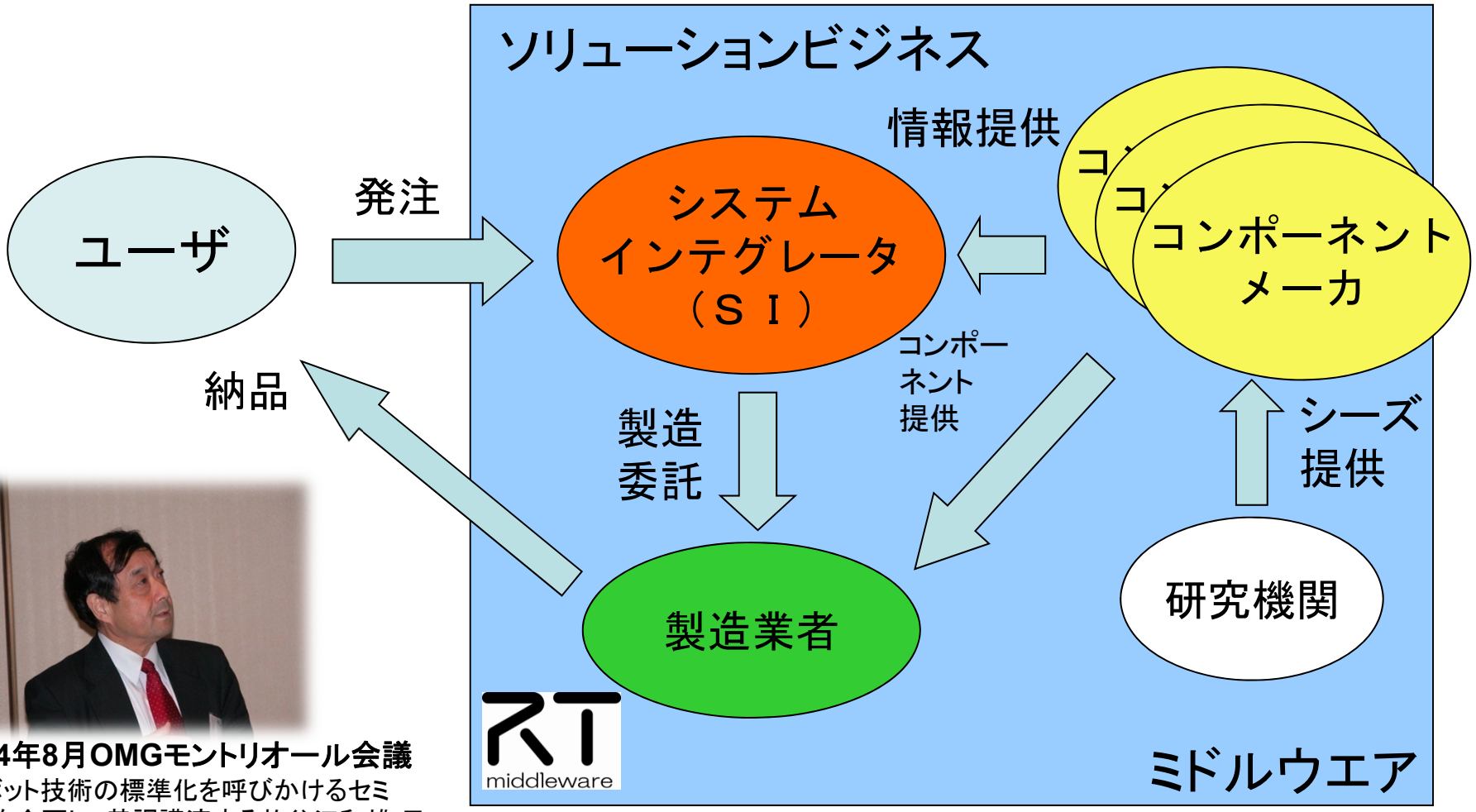
ニーズの問題



多様なニーズに対応

ロボットシステムインテグレーションによるイノベーション

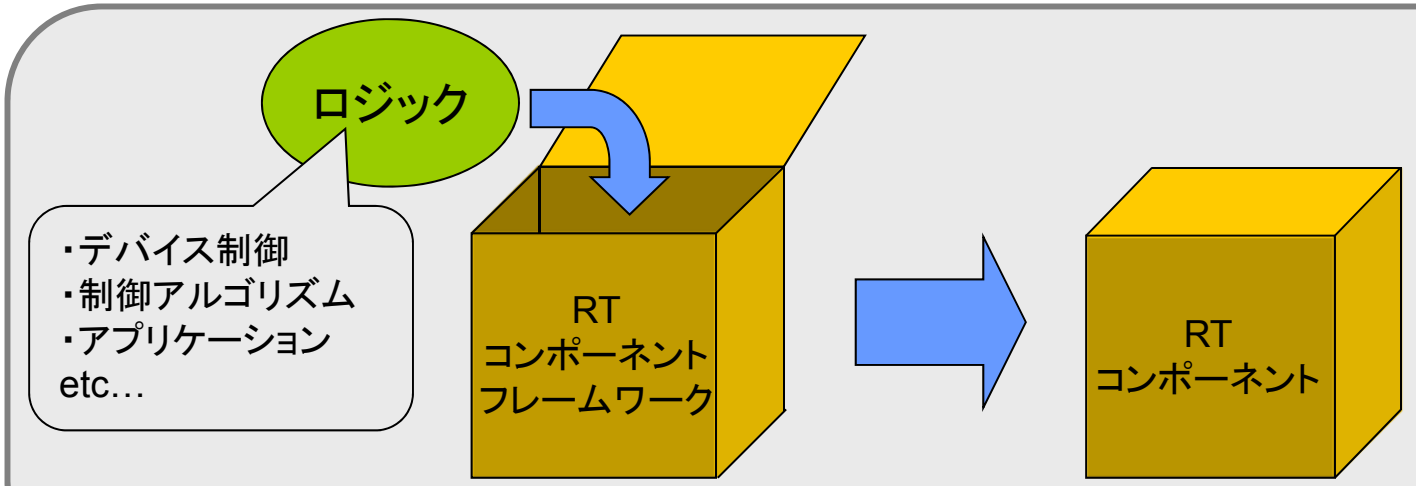
期待される未来のロボット産業のモデル



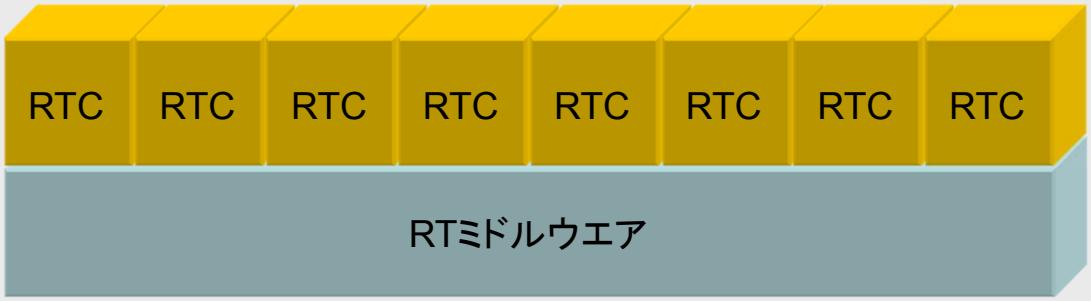
2004年8月OMGモントリオール会議
（ロボット技術の標準化を呼びかけるセミナーを企画し、基調講演する故谷江和雄 元産総研知能システム研究部門長、President of IEEE Robotics and Society ）

ミドルウエア：SI、コンポーネントメーカ、製造業者間の
交流を促進する情報基盤

RTミドルウエアとRTコンポーネント



ロジックを箱（フレームワーク）に入れたもの＝RTコンポーネント（RTC）



RTCの実行環境（OSのようなもの）＝RTミドルウエア（RTM）
※RTCはネットワーク上に分散可能

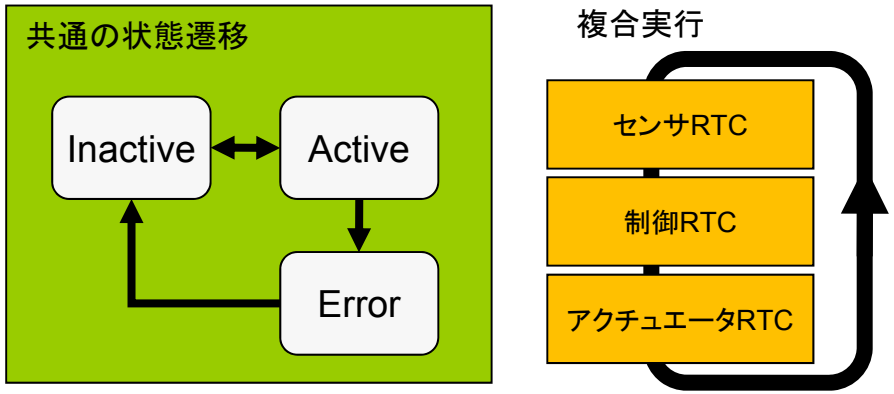
RTコンポーネント化のメリット

モジュール化のメリットに加えて

- ソフトウエアパターンを提供
  - ロボットに特有のソフトウエアパターンを提供することで、体系的なシステム構築が可能
- フレームワークの提供
  - フレームワークが提供されているので、コアのロジックに集中できる
- 分散ミドルウエア
  - ロボット体内LANやネットワークロボットなど、分散システムを容易に構築可能

RTコンポーネントの主な機能

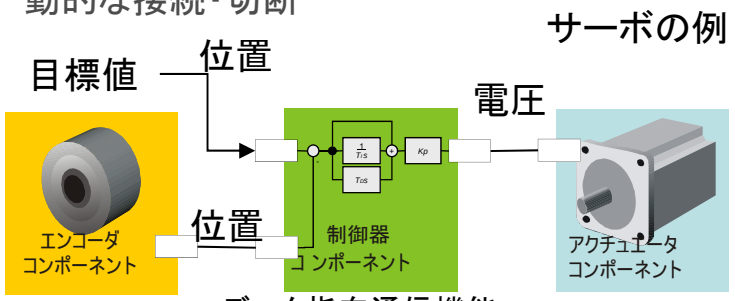
アクティビティ・実行コンテキスト



ライフサイクルの管理・コアロジックの実行

データポート

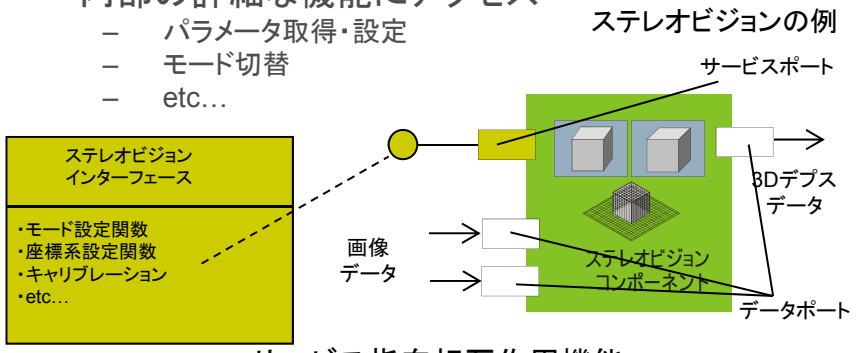
- データ指向ポート
- 連続的なデータの送受信
- 動的な接続・切断



データ指向通信機能

サービスポート

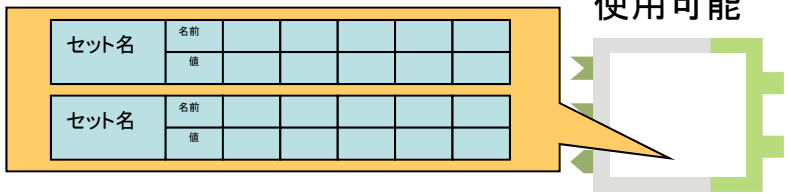
- 定義可能なインターフェースを持つ
- 内部の詳細な機能にアクセス
  - パラメータ取得・設定
  - モード切替
  - etc…



サービス指向相互作用機能

コンフィギュレーション

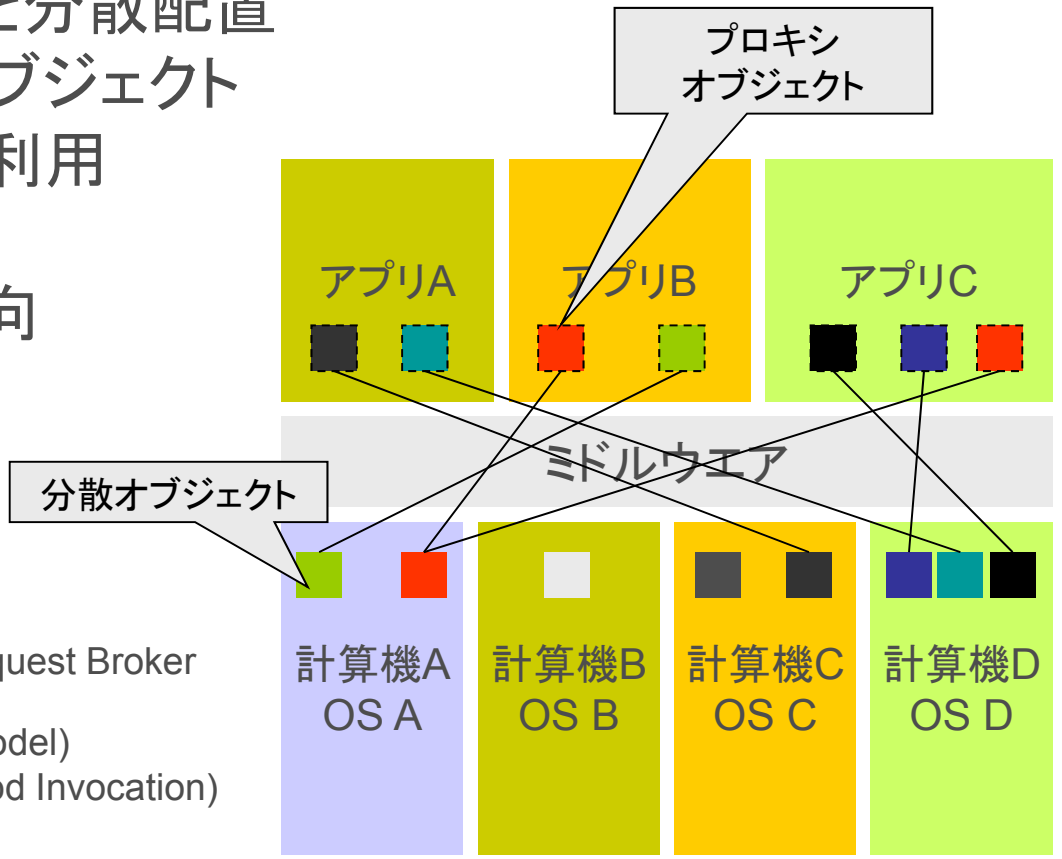
- パラメータを保持する仕組み
- いくつかのセットを保持可能
- 実行時に動的に変更可能



分散オブジェクトとは？

- システムの機能分割と分散配置
- ネットワーク透過なオブジェクト
- コンポーネント化と再利用

オブジェクト指向
＋
ネットワーク



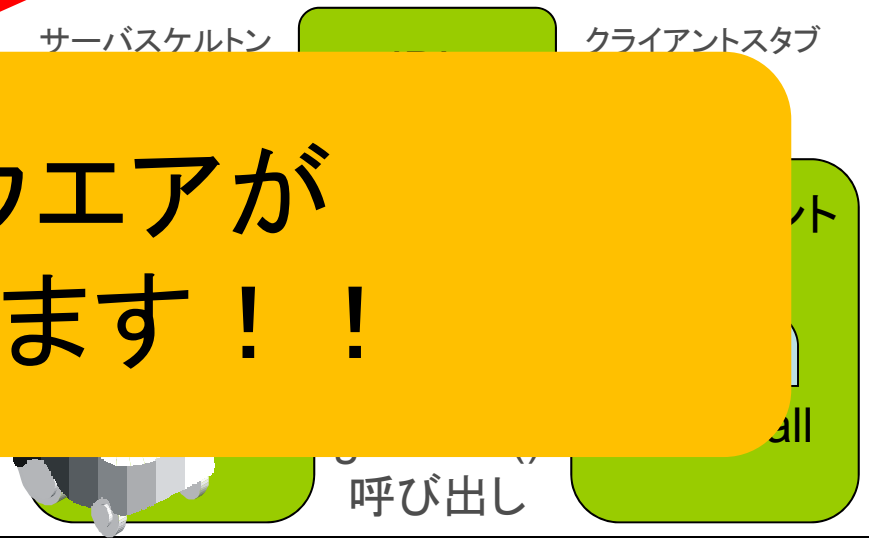
- 代表例
  - CORBA (Common Object Request Broker Architecture)
  - CCM (CORBA Component Model)
  - JavaRMI (Java Remote Method Invocation)
  - EJB (Enterprise Java Beans)
  - DCOM, HORB etc…

CORBAの例

本題にたどり着くまでが面倒

```
class MobileRobot_Impl
 : public virtual POA_MobileRobot,
   public virtual
     PortableServer::RefCountServant
```

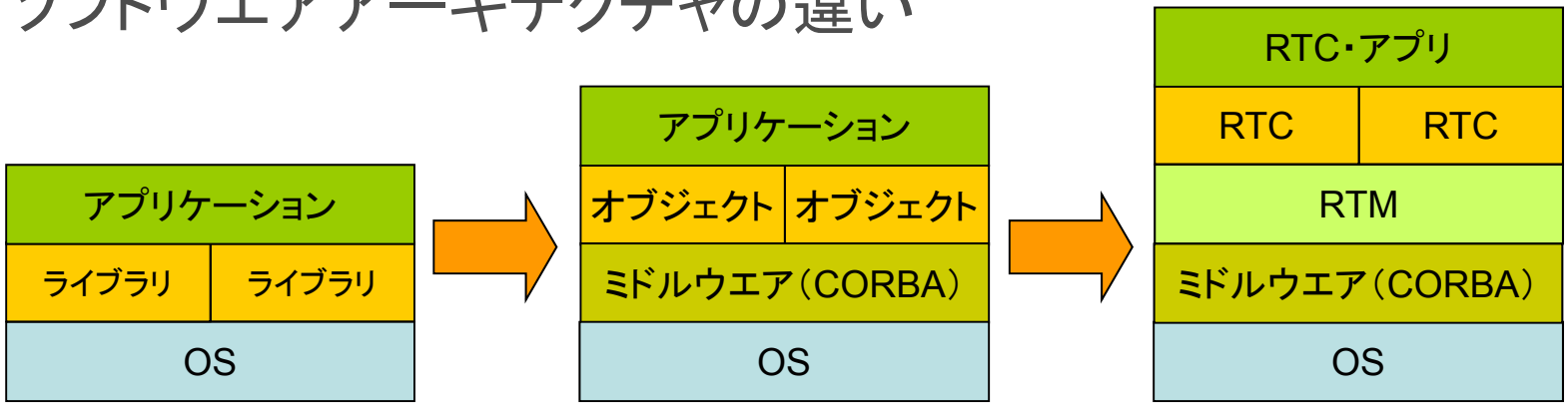
```
robot->gotoPos(pos);
```



RTミドルウエアが
全部面倒みます！！

RTM、RTCとは？

ソフトウエアアーキテクチャの違い



従来ソフトウエアから分散オブジェクトへ

- オブジェクト指向開発
- 言語・OSの壁を越えて利用できる
  - インターフェースをIDLで定義
  - 各言語へ自動変換
  - OS、アーキテクチャの違いを吸収
- ネットワーク透過に利用できる
  - 分散システムを容易に構築可能

分散オブジェクトからRTCへ

- インターフェースがきちんと決まっている
  - IDLで定義された標準インターフェース
  - 呼び出しに対する振る舞いが決まっている(OMG RTC 標準仕様)
  - 同じ部品として扱える
- コンポーネントのメタ情報を取得することができる
  - 動的な接続や構成の変更ができる
- ロボットシステムに特有な機能を提供
  - 後述

RTミドルウエアによる分散システム

RTMにより、ネットワーク上に分散するRTCをOS・言語の壁を越えて接続することができる。

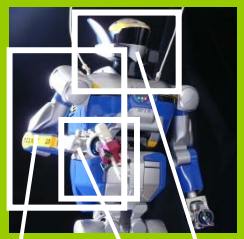
ロボットA

RTC | RTC
RTM
VxWorks

ロボットB

RTC | RTC
RTM
QNX

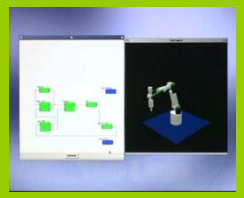
ロボットC



RTC | RTC | RTC
RTM
Real Time Linux

ネットワーク

Linux
RTM
RTC | RTC



アプリケーション

Windows
RTM
RTC | RTC

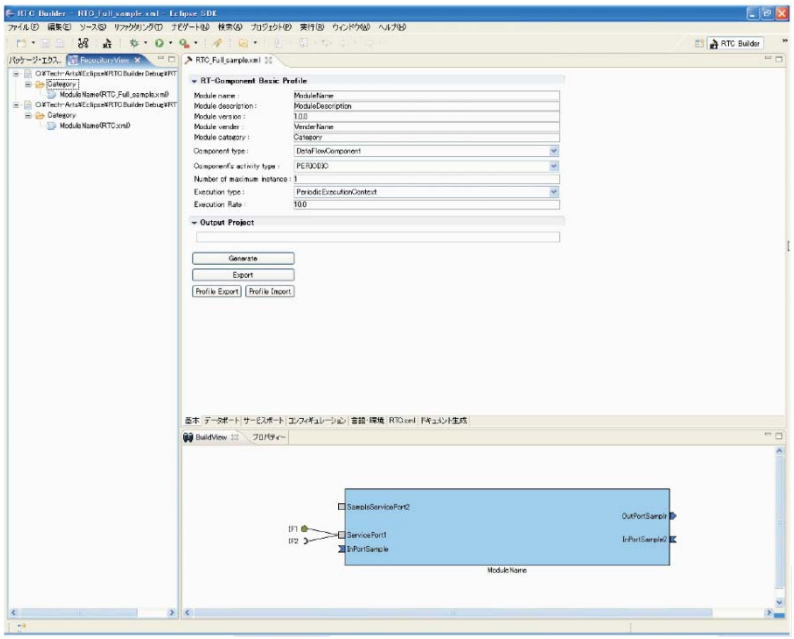
操作デバイス

uITRON
RTM
RTC | RTC

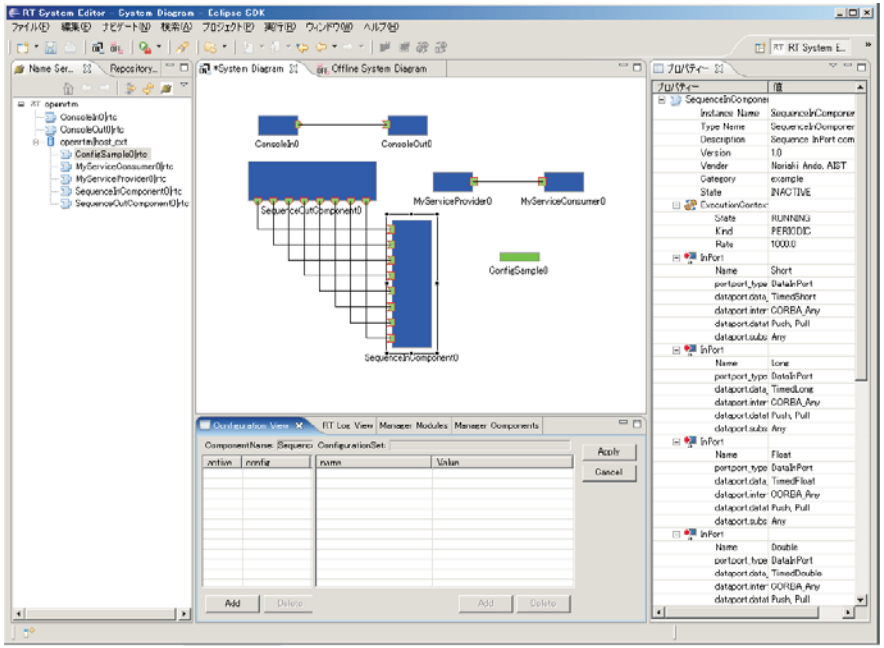
センサ

RTC同士の接続は、プログラム実行中に動的に行うことが出来る。

ツールチェーン



RTCBuilder
RTコンポーネント設計・コード生成



RTSystemEditor
RTCを組み合わせてシステムを設計

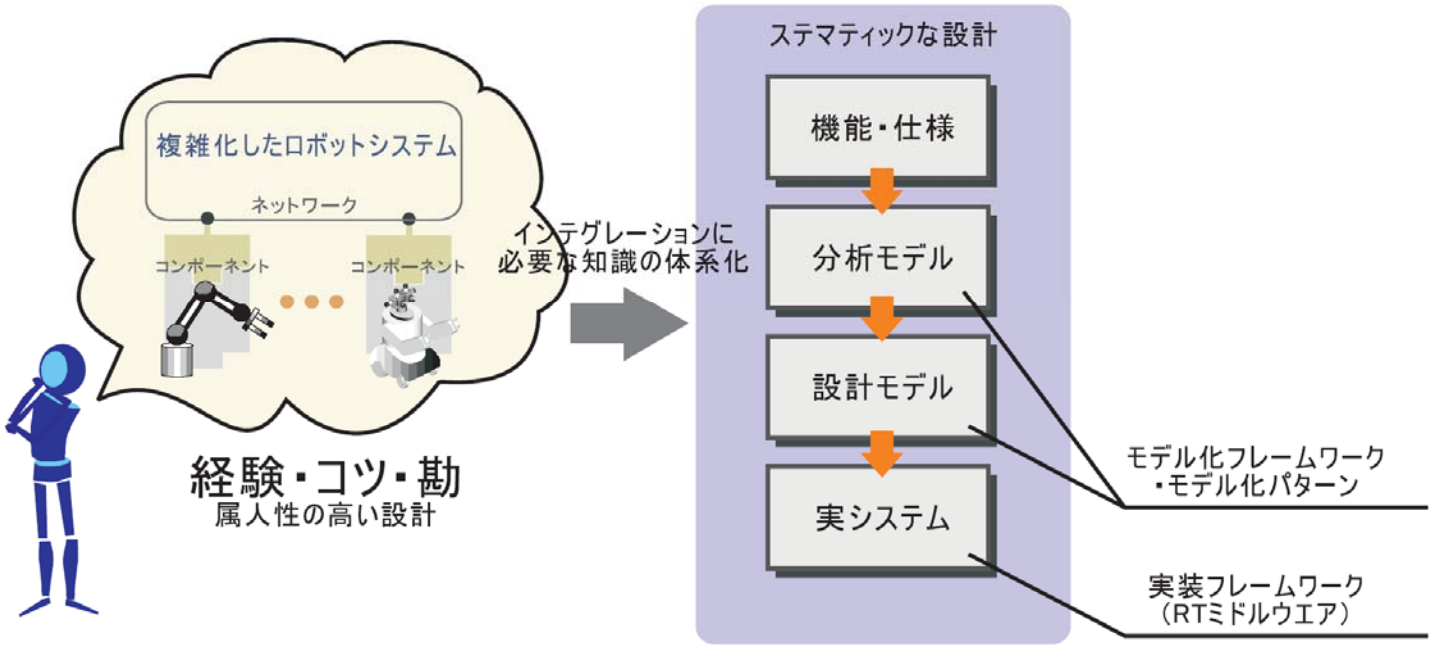
RTC・RTM統合開発環境の整備

RTC設計・実装・デバッグ、RTMによるインテグレーション・デバッグまでを
一貫して行うことができる統合開発環境をEclipse上に構築

体系的システム開発

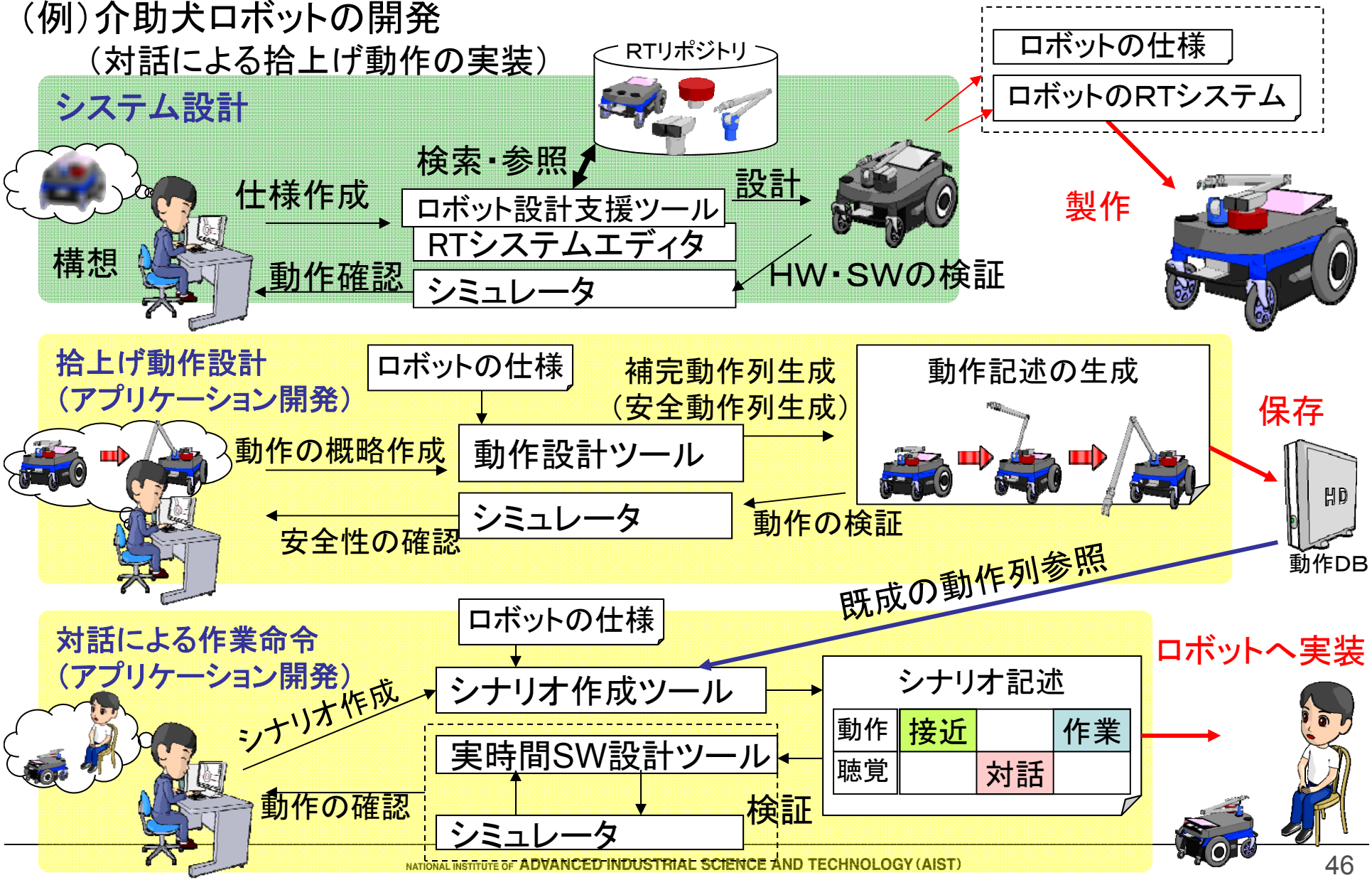
開発者の経験やノウハウに依存したロボットシステム開発

体系的システムデザイン: 分析、設計、実装の一連の流れ

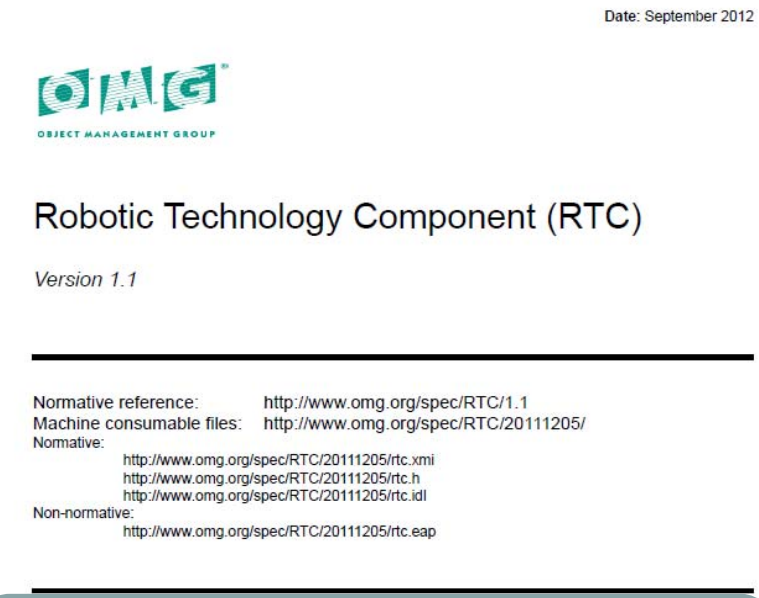


プラットフォーム概要

（例）介助犬ロボットの開発
（対話による拾上げ動作の実装）



RTミドルウェアの国際標準化



Date: September 2012

OMG
OBJECT MANAGEMENT GROUP

Robotic Technology Component (RTC)

Version 1.1

Normative reference: http://www.omg.org/spec/RTC/1.1
Machine consumable files: http://www.omg.org/spec/RTC/20111205/
Normative:
http://www.omg.org/spec/RTC/20111205/rtc.xmi
http://www.omg.org/spec/RTC/20111205/rtc.h
http://www.omg.org/spec/RTC/20111205/rtc.idl
Non-normative:
http://www.omg.org/spec/RTC/20111205/rtc.eap

OMG国際標準

標準化組織で手続きに沿って策定
→ 1組織では勝手に改変できない安心感
→ 多くの互換実装ができつつある
→ 競争と相互運用性が促進される

標準化履歴

- 2005年9月
  Request for Proposal 発行(標準化開始)
- 2006年9月
  OMGで承認、事実上の国際標準獲得
- 2008年4月
  OMG RTC標準仕様 ver.1.0公式リリース
- 2012年9月
  ver. 1.1改定
- 2015年9月
  FSM4RTC(FSM型RTCとデータポート標準) 採択

RTミドルウエア互換実装は10種類以上

名称	ベンダ	特徴	互換性
OpenRTM-aist	産総研	NEDO PJで開発。参照実装。	---
HRTM	ホンダ	アシモはHRTMへ移行中	◎
OpenRTM.NET	セック	.NET(C#,VB,C++/CLI, F#, etc..)	◎
RTM on Android	セック	Android版RTミドルウエア	◎
RTC-Lite	産総研	PIC, dsPIC上の実装	○
Mini/MicorRTC	SEC	NEDOオープンイノベーションPJで開発	○
RTMSafety	SEC/AIST	NEDO知能化PJで開発・機能安全認証取得	○
RTC CANOpen	SIT, CiA	CAN業界RTM標準	○
PALRO	富士ソフト	小型ヒューマノイドのためのC++ PSM 実装	×
OPRoS	ETRI	韓国国家プロジェクトでの実装	×
GostaiRTC	GOSTAI, THALES	ロボット言語上で動作するC++ PSM実装	×

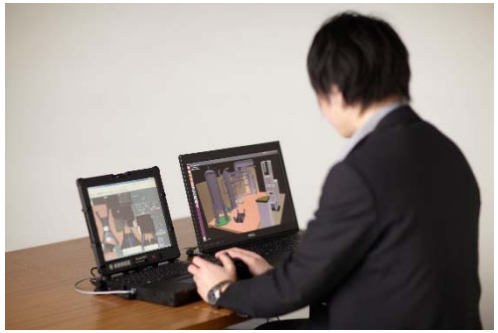
特定のベンダが撤退しても
ユーザは使い続けることが可能

実用例・製品化例

HRPシリーズ: 川田工業、AIST

S-ONE：SCHAFT

DAQ-Middleware: KEK/J-PARC
KEK: High Energy Accelerator Research Organization
J-PARC: Japan Proton Accelerator Research Complex



災害対応ロボット操縦シミュレータ:
NEDO/千葉工大

HIRO, NEXTAGE open: Kawada Robotics

RAPUDA：Life Robotics

ビュートローバーRTC/RTC-BT(VSTONE)

OROCHI（アールティ）

新日本電工他: Mobile SEM

ROS(Robot Operating System)

http://wiki.ros.org/

Open Source
Robotics Foundation

Willow Garage

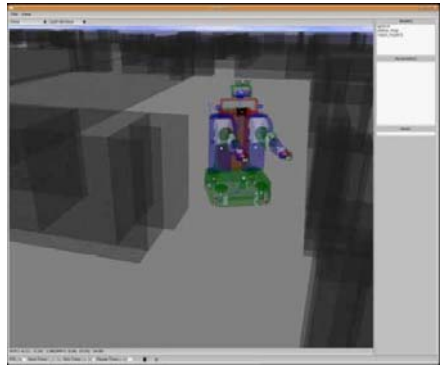
- Willow Garage
  - 2007年設立のロボットベンチャー（米、Menlo Park）
  - Scott Hassanが出資
    - googleの初期エンジンの作者の一人
  - Brian Gerkeyがソフトウエア部分のPL
    - Playerの作者の一人
  - ビジネスモデル
    - ソフト：ROS（無償）＋ハード：PR2 を販売
    - その他は不明、資金がたくさんあるので急いで利益を上げる必要はないとのこと
    - まずPR2を10台無償で大学などに配布予定

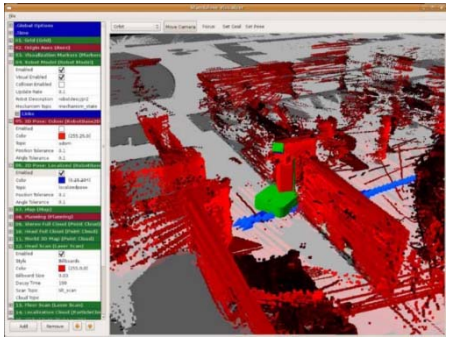
ROS（Robot Operating System）

- 目的：ロボット研究を効率的にする
- OSではない
  - Robot Operating System, Robot Open Source
- UNIX哲学
  - UNIX的なコンパクトなツール、ライブラリ群による効率的なインテグレーション環境を提供する。
  - モジュール性、柔軟性、再利用性、言語・OS非依存、オープンソース
- プラットフォームアーキテクチャ
  - 独自ミドルウエア（独自IDLからコード生成）
  - メッセージベース
  - pub/subモデル、RPCモデル

ROS（2）

- ライブラリ・ツール群が充実
  - シミュレータ：Gazeboベース
  - ナビゲーション：Player等
  - 座標変換・キャリブレーション
  - プランニング：TREX
  - マニピュレーション：OpenRAVE
  - ビジョン：OpenCV
  - 音声：ManyEars

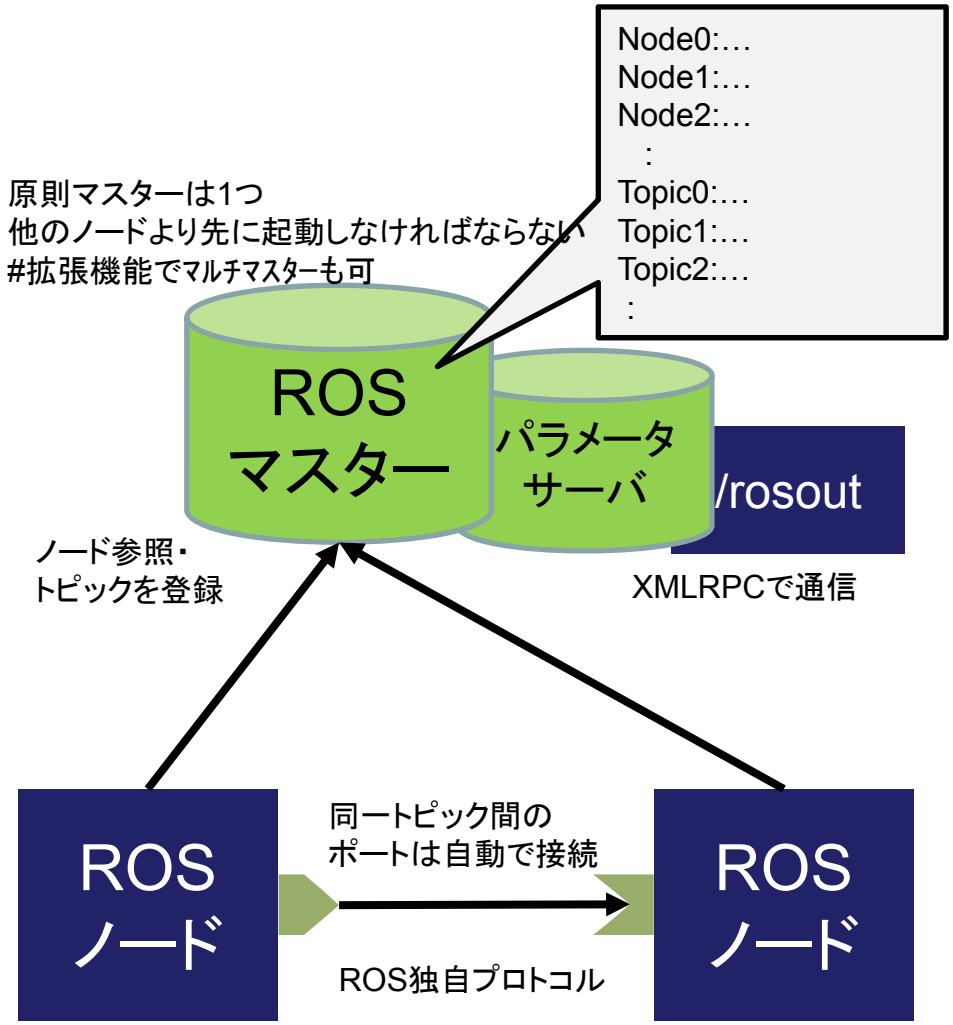




- 外部のオープンソースライブラリを取り込みライブラリ群として整備
- 最近、これまでとは大きくアーキテクチャの異なるミドルウェア“ROS2”をリリース。

ROS1の仕組み

- ノード：ROSのモジュール化単位
  - 1ノード=1プロセス
- マスター：ノードの参照やトピックを保持するネームサービス
  - システム全体で原則一つ
    →SPOF（Single Point of Failure）
  - 他のノードより先に起動しなければならない
- パラメータサーバ：ノードのパラメータを保持するデータベース
  - マスター内で動作、ノードからはXMLRPCでアクセス
- **rosout**: ノードに対してstdout, stderrのような役割を果たす



ROS1の通信

- マスター関係の通信はXMLRPCを使用
  - XMLRPCはSOAP同様に通信プロトコルとしてはHTTPを、メッセージフォーマットとしてはXMLを使用する
  - SOPAに比べて仕様が簡単（A4で2ページ）、単純で扱いやすい
- トピックベースの通信は独自のROS msg形式
  - .msg ファイル(≒IDL)で定義。ROS message description language
  - primitive型(+string, time, dulation)+配列（固定長・可変長）のみのシンプルな構成。構造体などはできない（はず）
  - TCPで通信するROSTCP、UDPのROSUDPがある
- サービスの通信もROS msg形式
  - サービス定義：.msg とほぼ同じ(引数、返り値)
  - 1サービス1オペレーション（1関数）

ROS messageは自前でIDLコンパイラ、シリアライザ・デシリアライザ、スタブ等を実装。結局のところCORBAと同じ

ROS1の前提

- 単一のロボット
- ワークステーションクラスの計算機パワー
- 非リアルタイム（リアルタイム性が必要な場合は特別なやり方で実施）
- 良好な通信環境（有線または広帯域な無線）
- 主に学術研究向け
- 何かを規定したり禁止したりすることのない最大の柔軟性
  - 例えば、main() 関数をラップしたりしない

http://design.ros2.org/articles/why_ros2.html

ROS1からROS2へ

ユースケース

- 複数のロボット
- 組込みCPU
- リアルタイム
- 理想的でない通信環境
- 製品向け使用
- あらかじめ規定されたパターンにのっとった構造化したシステム構成

新たな技術

- Zeroconf (avahi, bonjour, UPnP等)
- Protocol Buffers
- ZeroMQ (and the other MQs)
- Redis(次世代高速key-valueデータストア)
- WebSockets
- DDS (Data Distribution Service).

http://design.ros2.org/articles/why_ros2.html

ROS2

- 2015年6月ごろalphaリリース
- 通信部分は既存の標準に基づくミドルウェア：DDSで行うとした
  - ただし、DDSの複数の実装に対応
  - RTMが複数のCORBA実装に対応したように
- DDS依存部分はmiddleware interfaceとして隠ぺいすることとした
  - ちょうどRTMがCORBAを隠ぺいしたように
- コンポーネントモデルを導入することにした
- コードはWindowsでの使用も意識した形に

http://design.ros2.org/articles/ros_middleware_interface.html

Player/Stage and Gazebo

http://playerstage.sourceforge.net/

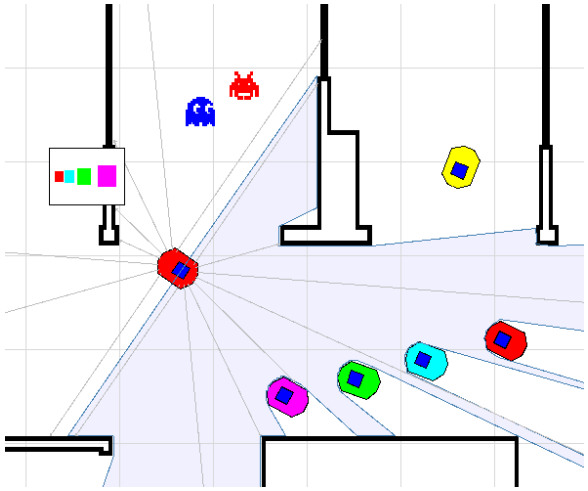
Player/Stage and Gazebo

- USC Robotics Lab
- Player
  - ロボット用ネットワークサーバ
  - ネットワーク経由でセンサ・アクチュエータにアクセスするためのインターフェースを提供
  - Playerは多くのロボットハードウエアをサポート
    - ex. ActivMedia Pioneer 2 family

“Player” は移動ロボット制御のための一連の共通インターフェースを提供するプラットフォーム

Player/Stage and Gazebo

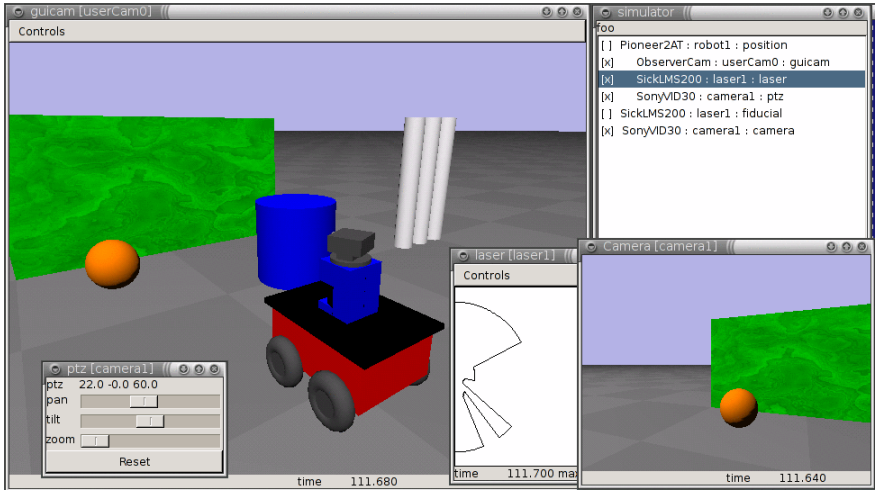
- Stage
  - 2次元のビットマップベースの環境において、移動ロボット、センサ、オブジェクト群をシミュレートする.
  - マルチエージェントシステムの研究のために設計されている.



“Stage” は2次元の移動ロボットシミュレーション環境を提供する.

Player/Stage and Gazebo

- Gazebo
  - 屋外環境のマルチロボットシミュレーション環境を提供.
  - 3次元環境のロボット、センサ、オブジェクト群のシミュレーション環境を提供する.
  - 現在はDARPAの支援を受けてOSRF†において開発を継続中



†OSRF: OpenSource Robotics Foundation

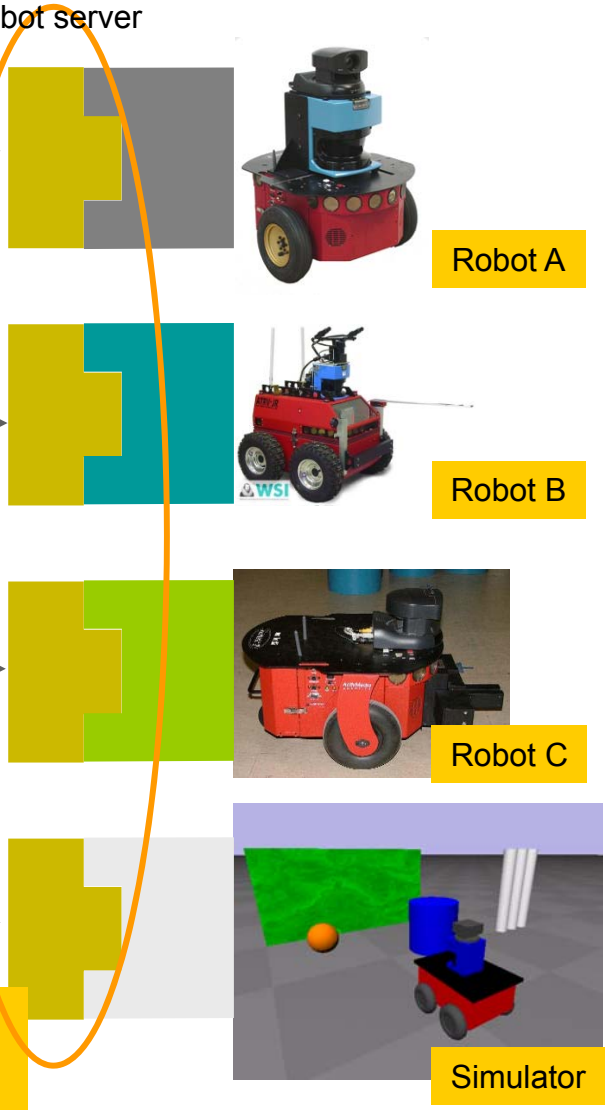
3次元移動ロボットシミュレーション環境を提供.

Player/Stage and Gazebo

- パスプランニング
- 衝突回避
- センサフュージョン
- マルチエージェント制御
- etc...

ユーザが開発するPlayerクライアント.

C, C++, Tcl, LISP, Java, Python
をサポート.
しかし、OOPではない.



ロボット・センサに対する
共通インターフェースを提供.

OROCOS

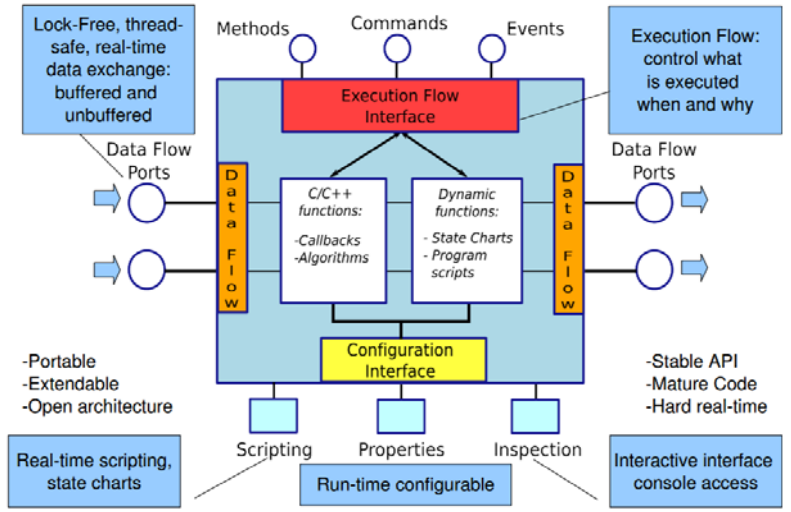
http://www.orocos.org/

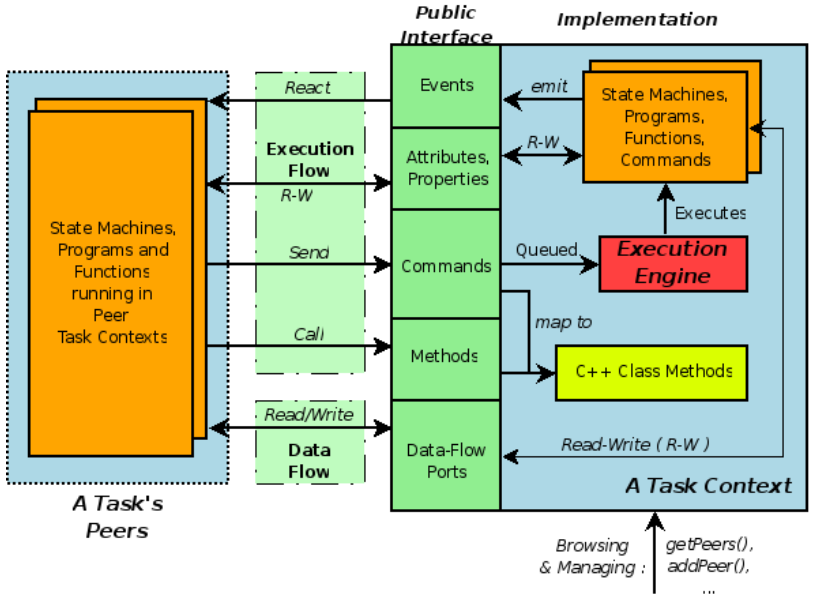
OROCOS

- EUプロジェクトで開発(2001-2003)
  - K.U.Leuven(ベルギー)
  - LAAS Toulouse(フランス)→ORCAへ
  - KTH Stockholm(スウェーデン)
- ハードリアルタイムのソフトウエアフレームワーク
- ロボットの制御に必要なライブラリ集（運動学、リアルタイム制御、etc…）
- 最近はコンポーネントベース開発のフレームワークも提供
- ツールによるモデルベース開発

OROCOS

- コンポーネントモデル
  - データフローポート
  - 種々のサービスインターフェース
  - コンフィギュレーション機能
  - コールバックベースのロジック実行フレームワーク





ほぼRTCと同じ（マネ？）

OPRoS

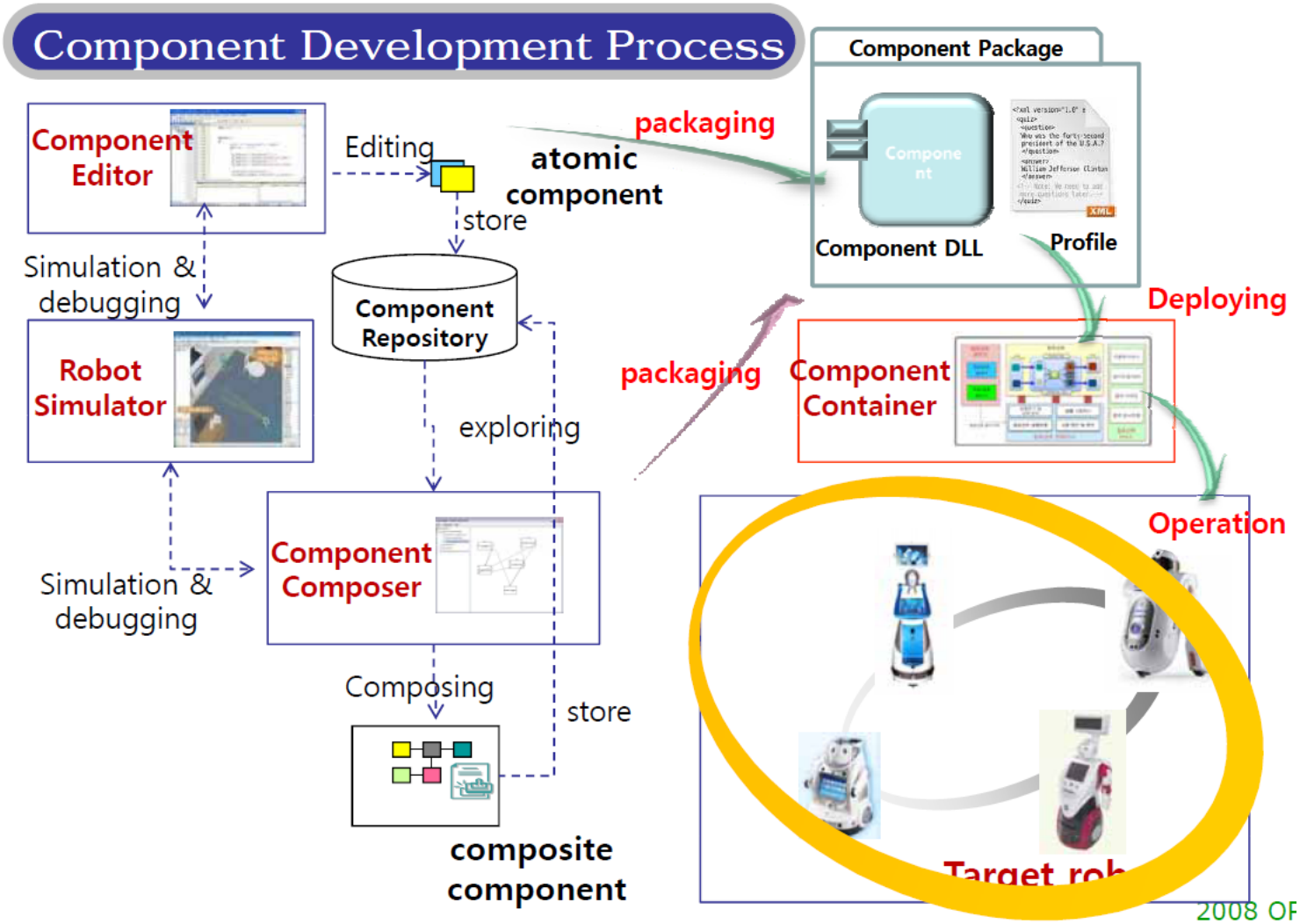
(The Open Platform for Robotic Services)

http://www.opros.or.kr/

OPRoS

- 韓国の国プロジェクトで開発されたロボット用プラットフォーム
  - ETRI(Electronics and Telecommunication Research Institute)
  - KIST, Kanwong Univ., etc…
- OMG RTC（ほぼ）準拠
- 通信ミドルウエアは独自（URC(Ubiquitous Robot Companion PJ)で開発したもの）
- ツールチェーンなども提供

OPRoS

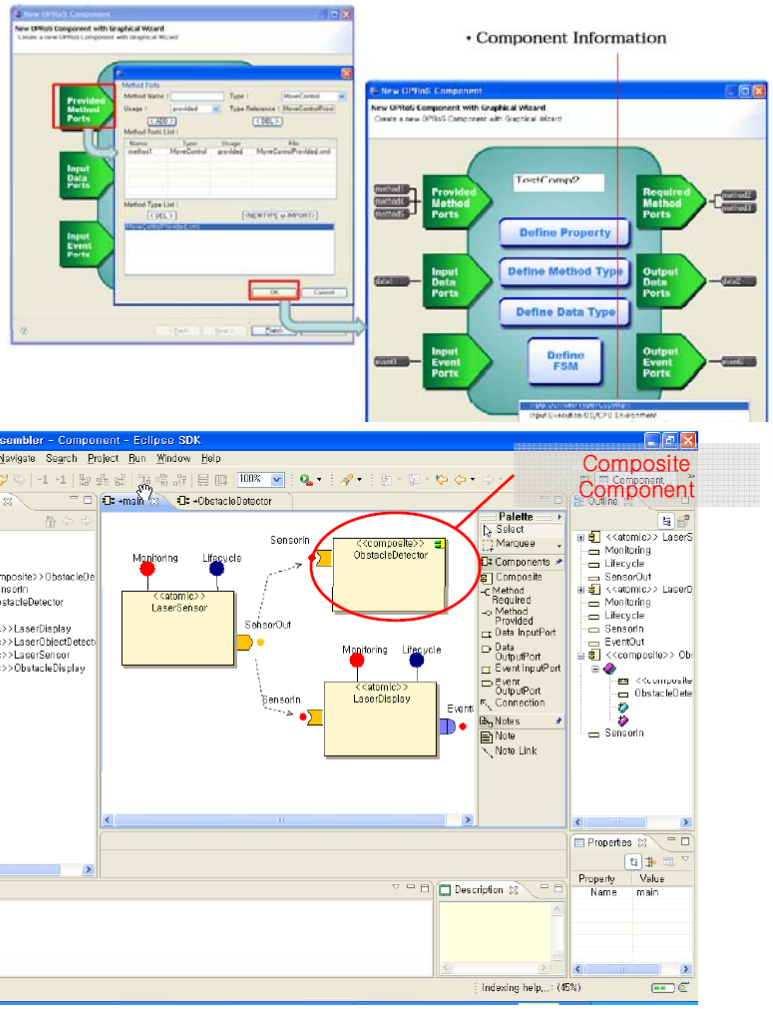


© ETRI, OMG Infra. WG, OPRoS Component Tools

OPRoS

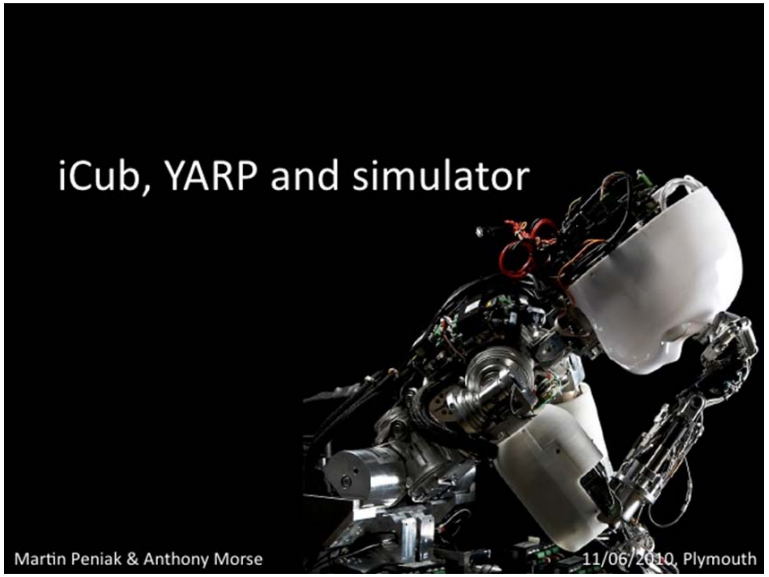
- コンポーネント開発
  - Component Editor
- システム構築
  - Component Composer

OpenRTM-aistに
よく似たツールチェーン



© ETRI, OMG Infra. WG, OPRoS Component Tools

YARP

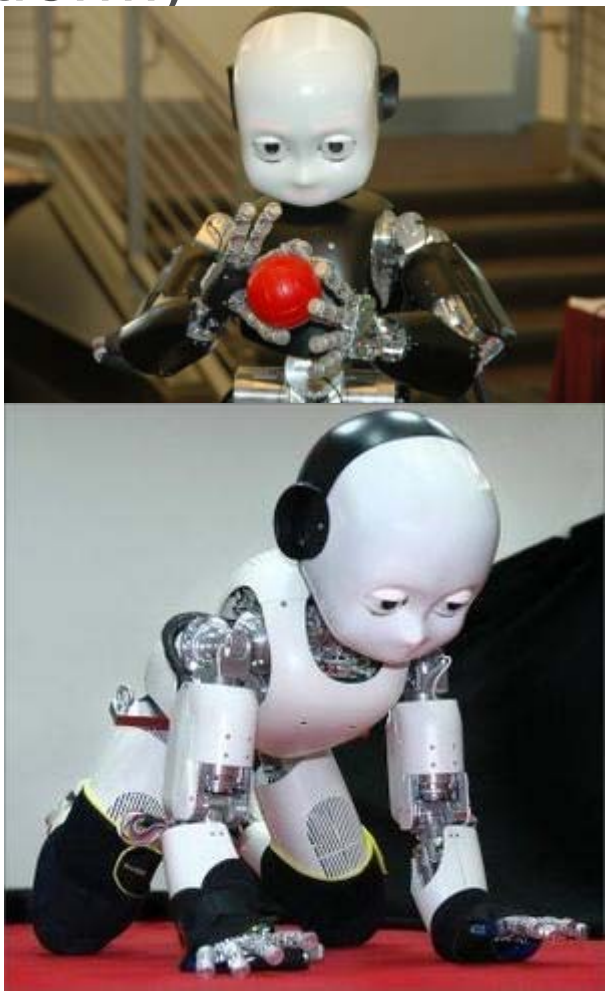


http://eris.liralab.it/yarp/

YARP

(Yet Another Robot Platform)

- IIT (Istituto Italiano di Tecnologia) で開発されたiCubのためのソフトウエアプラットフォーム
  - iCub: EUプロジェクト RobotCub,
  - 53DOFの赤ちゃんの様なヒューマノイド
- YARP



YARP

- コンポーネントフレームワークは無し
  - Mainから書き始める
  - 原則1プロセス1モジュール
- 多様な伝送方式のデータポートを提供
  - TCP, UDP, multicast
  - Carrier: 様々なマーシャリング、プロトコルを利用可能
- 簡単なRPCもある
- 独自のマーシャリング方式
- ノード間の利用にはネームサービスを利用
- CUIツール: yarp
  - 接続制御、モジュール制御

Microsoft Robotics Studio

http://www.microsoft.com/robotics/

Microsoft Robotics Studio

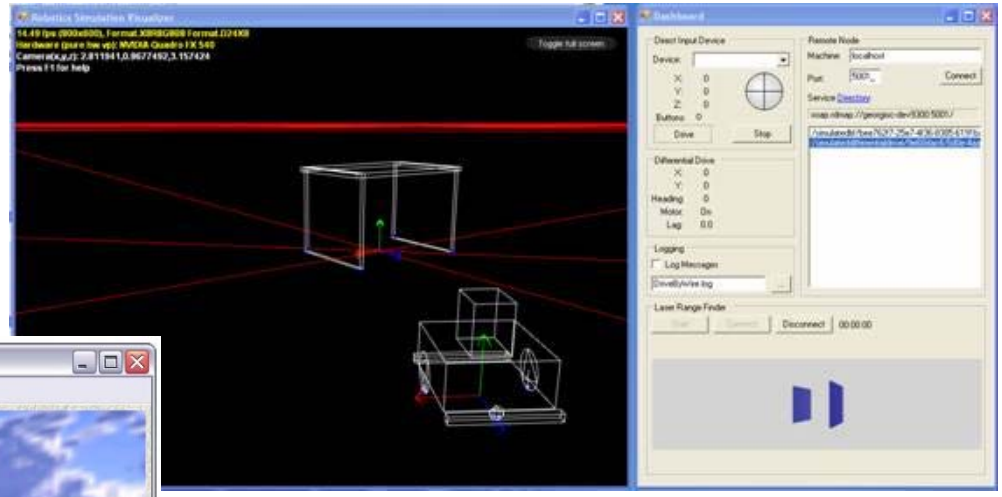
- 高レベルのビヘイビアやハードウエアのサービス志向の抽象化を提供
- 再利用性
- シミュレータエンジンを提供
  - AGEIA PhysXも使える
- ブラウザインターフェース
- Windowsのみで利用可能
  - ただし、ある種のハードウエアに対応
  - LEGO NXT, i-Robot, ActivMedia等

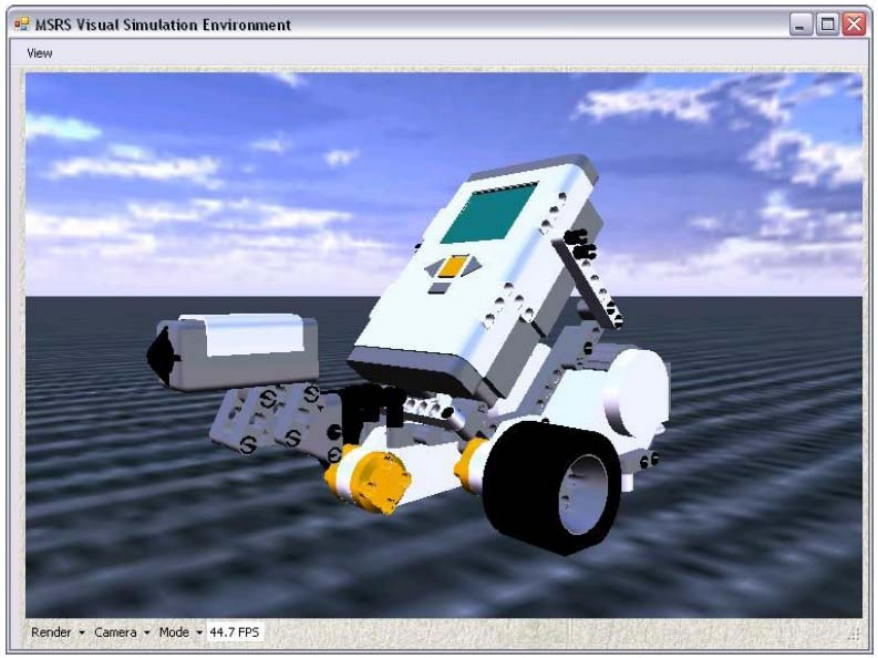
MSRSアーキテクチャ

DSSP：Decentralized Software Services Protocol

- DSSP: SOAPベースのプロトコルであり軽量なサービスモデルを提供する。
- DSSPはステート志向のメッセージオペレーションをサポート。
- アプリケーションはサービスの集合体として定義され、HTTPの拡張として提供される。
- サービスとは：アプリケーションのライフタイム中であれば、生成、操作、モニタリング、削除可能な軽量なエンティティ。
- サービス：ユニークなID、状態、振る舞い、コンテキストを持つ実体

MSRS シミュレータ




MSRS Visual Simulation Environment
View
Render
Camera
Mode
44.7 FPS

OPEN-R

OPEN-R

OPEN-R

- 開発元：SONY
- AIBO, SDR-[3,4]X, QURIO等の制御アーキテクチャ
- 目的
  - スケーラビリティ
  - ポータビリティ
  - インターオペラビリティ
    - 異なるタイプのロボットに対して同じインターフェースを提供
  - スタイル・フレキシビリティ
    - 車輪型、4足歩行型、2足歩行型…

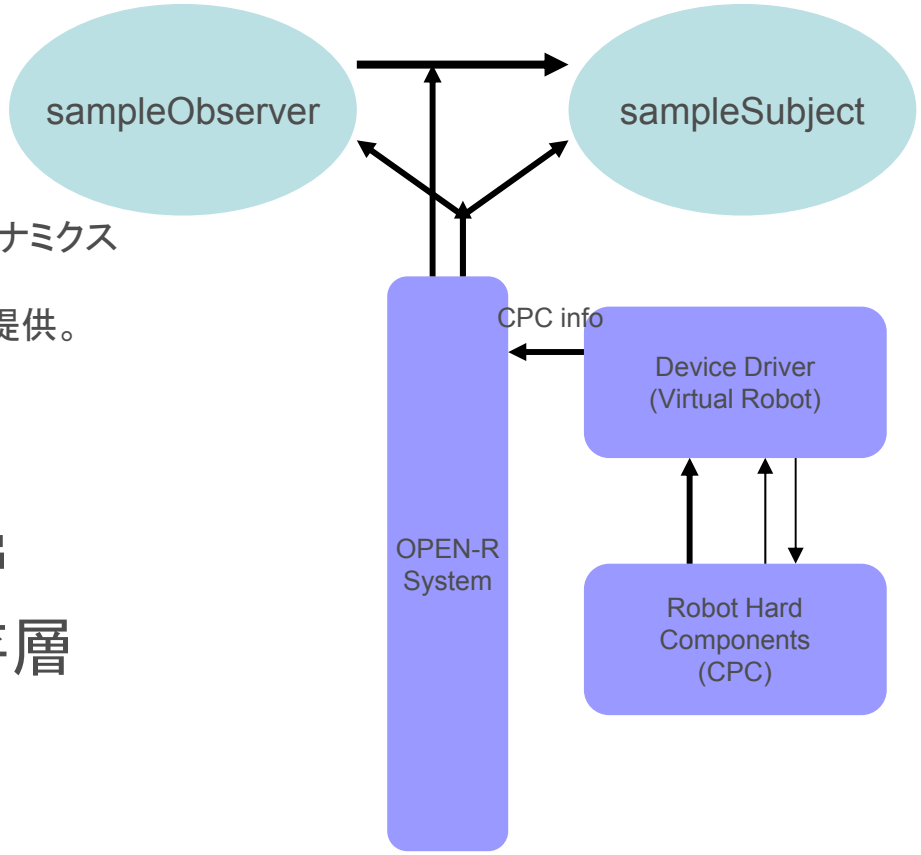
OPEN-Rアーキテクチャ

ハードウエアコンポーネント

- CPC (Configurable Physical Component)
  - 各コンポーネントは自身の機能、位置、ダイナミクスパラメータなどの情報を内蔵している。
  - ハードウエア自体がリフレクティブな機能を提供。

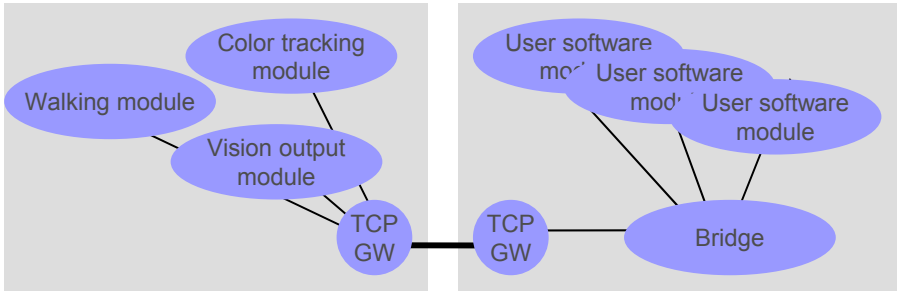
ソフトウエアコンポーネント

- コンフィギュレーション依存層
- コンフィギュレーション非依存層
- コンポーネント指向



OPEN-Rの特徴

- リモート処理
  - コードのポータビリティ
    - 組込システムとPC間
  - ネットワークを介したメッセージパッシング
  - ソフトウエアの再利用



- コンカレント開発環境
  - PCと組込み
- スケーラビリティ
- モジュール非依存性の確保
  - メッセージパッシング
- モジュラリティ
  - ハード＆ソフト

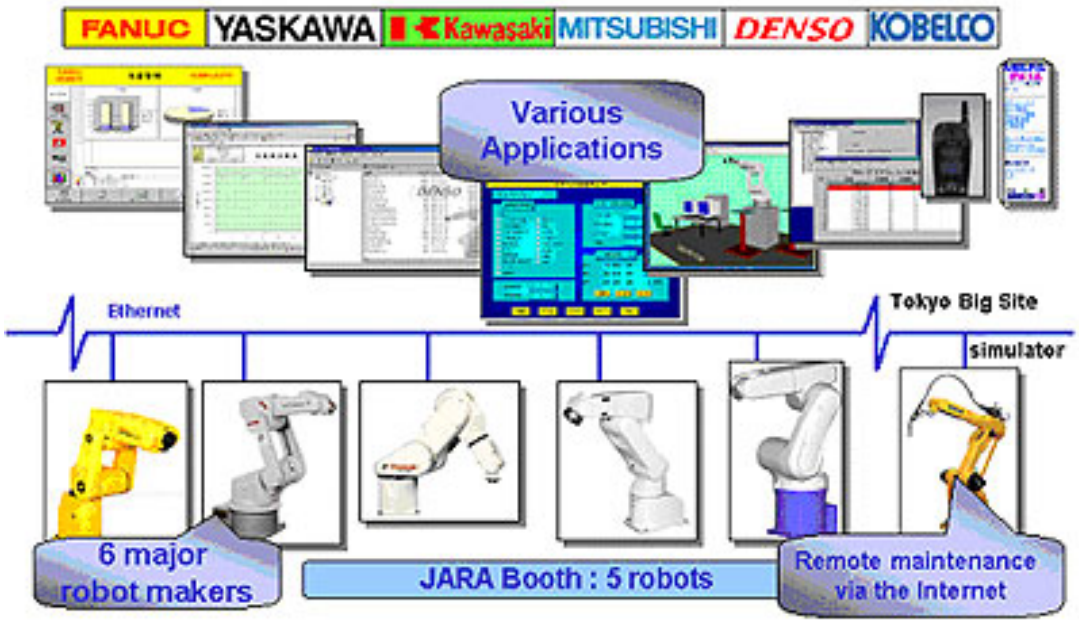
ORiN

http://www.orin.jp/

ORiN

- ORiN (Open Robot Interface for the Network)
  - 日本のFAロボット標準
  - FAロボットコントローラを抽象化
  - マルチベンダロボットシステムを容易に実現可能
  - メンバー:
    - ORiN consortium

    (FANUC, YASUKAWA, Kawasaki, MITSUBISHI, DENSO, KOBELCO)
    - http://www.orin.jp/

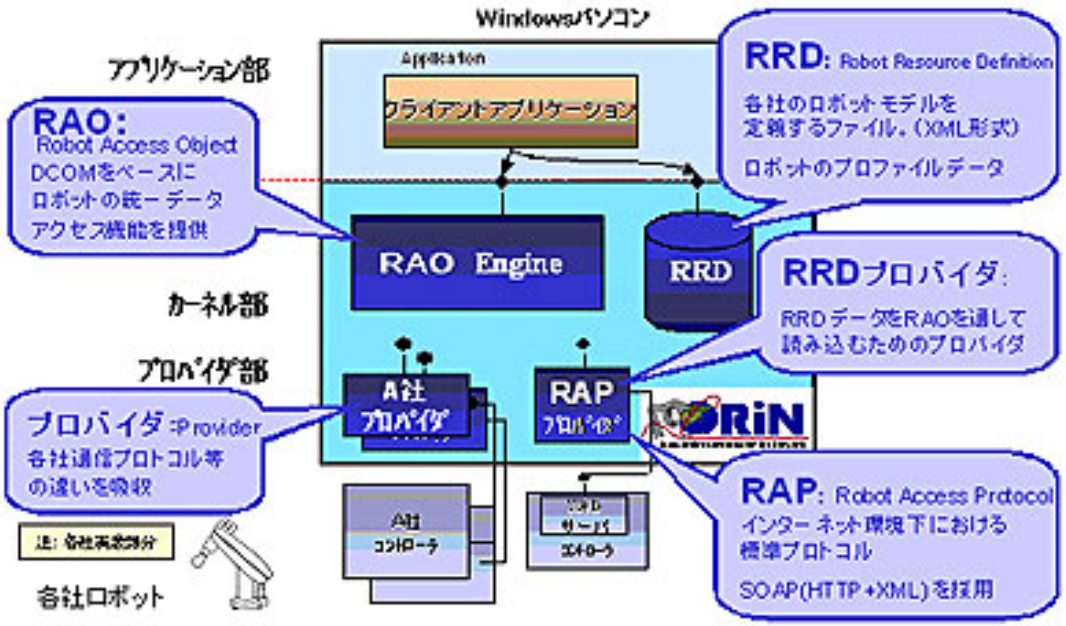


ORiN

- 設計ポリシ
  - 緩やかな標準化
    - 様々なタイプのロボット仕様を包含可能
  - 拡張性
    - ベンダ特有のオプションを定義可能
  - ネットワークプロトコルのモジュール化
    - 既存のロボットに適用可能
  - 実装と仕様の分離
    - OOP
- 実装ポリシ
  - デファクト標準
    - PC&Windows
  - Distributed object model (DCOM)
    - ネットワーク透過
    - 言語非依存
  - XML
    - ベンダ独自の仕様を記述するための標準フレームワーク
  - インターネット技術
    - HTTP, XML, SOAP

ORiNアーキテクチャ

- RAO (Robot Access Object)
  - ロボットコントローラに対する統一されたデータアクセシビリティを提供
- RRD (Robot Resource Definition)
  - ロボットプロファイルデータ
- RAP (Robot Access Protocol)
  - インターネットを介したアクセシビリティを提供



RSNP

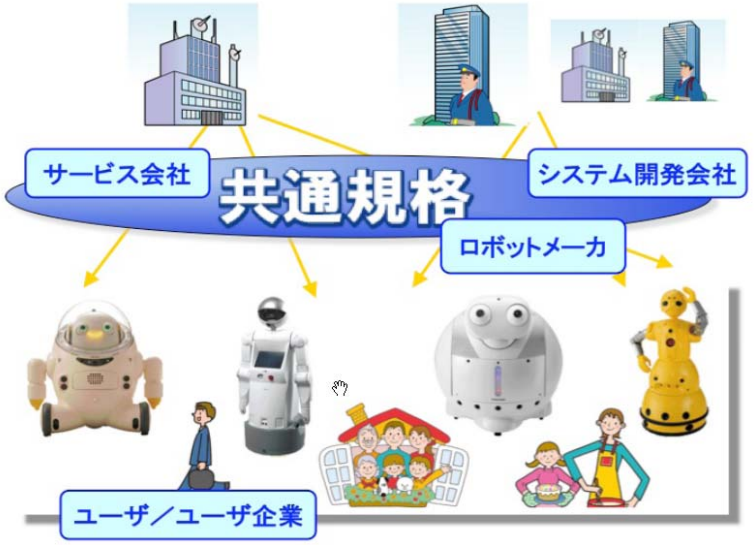
（Robot Service Network Protocol）

http://robotservices.org/

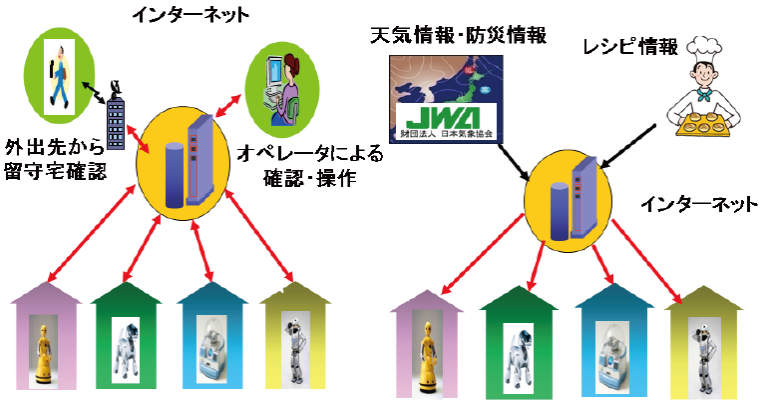
http://www.robotservices.org/wiki/jp/

RSi (Robot Service Initiative)

- RSi (Robot Service Initiative)が主導となって定めるロボット用インターネットサービスのプロトコル
  - RSi:2004年発足
  - 富士通、三菱重工、東芝、安川電機、日本気象協会等が加盟
- ロボットをプラットフォームとしたインターネットと連携した新たなビジネス創出を目指す



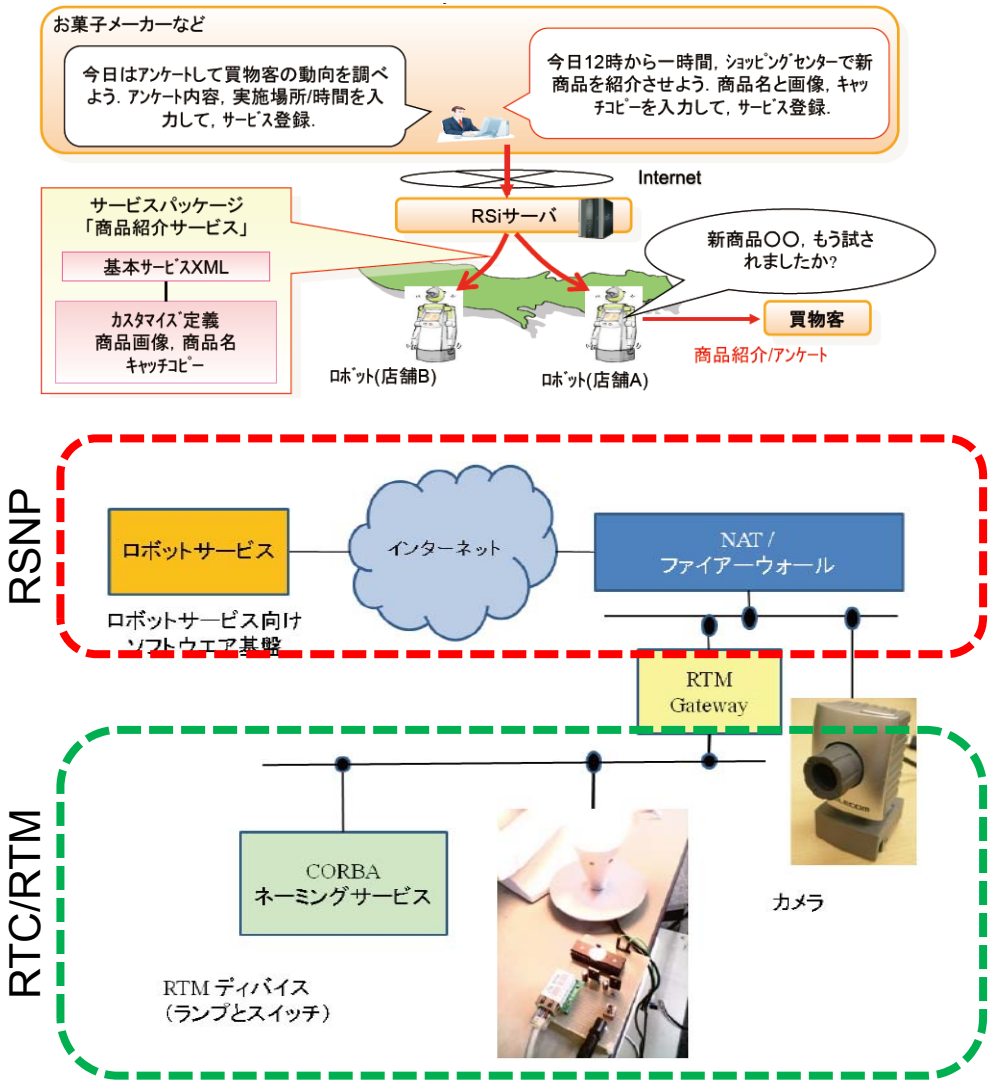
例1　遠隔操作によるサービス（見守りサービス等）　　例2　情報提供サービス



© 成田雅彦, 産業技術大学院大学
ROBOMEC2013 RTM講習会資料より

RSNP

- RSiサーバ等で提供するサービスを各地のロボットを介して提供
  - 天気予報、見守り、ロボットマップ、各種ロボット制御
- SOAPを利用
- 疑似Push機能を利用しFirewall越し通信を実現
- 主としてロボットとインターネット(クラウド)との連携に利用
  - cf. RTC



© 成田雅彦, 産業技術大学院大学 ROBOMEC2013 RTM講習会資料より

UNR Platform

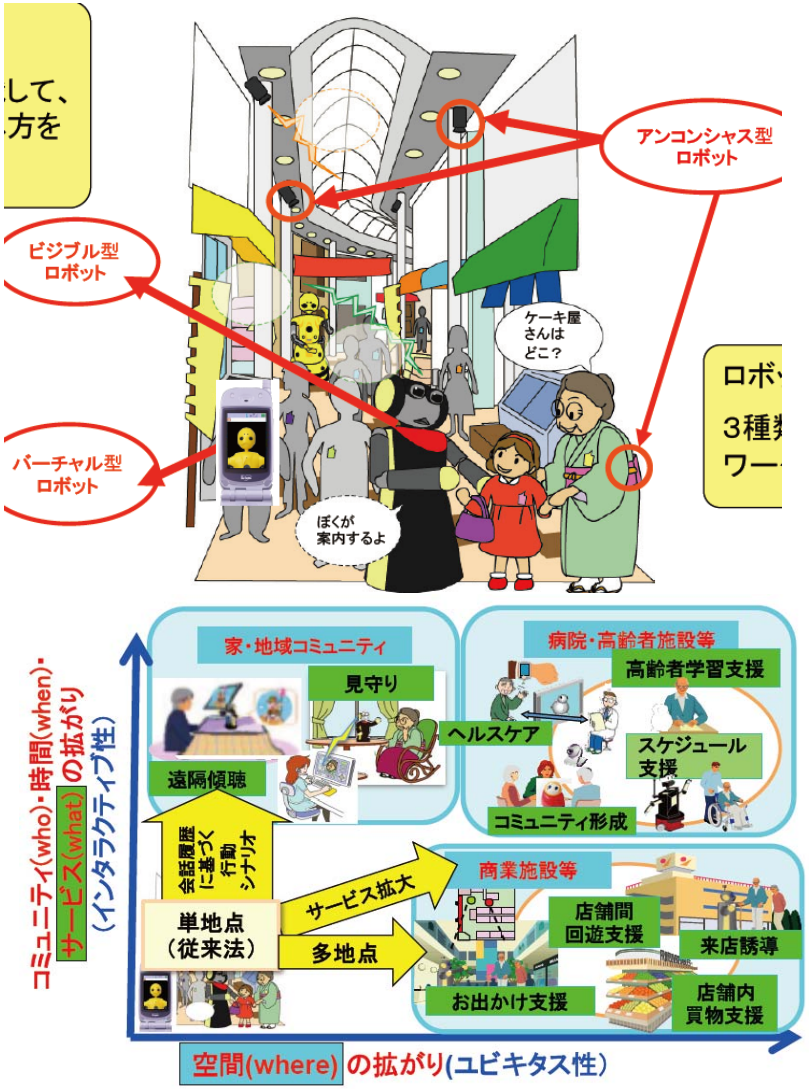
（Ubiquitous Network Robot Platform）

http://www.irc.atr.jp/std/UNR-Platform.html

ATR Advanced Telecommunications Research Institute International

UNR Platform

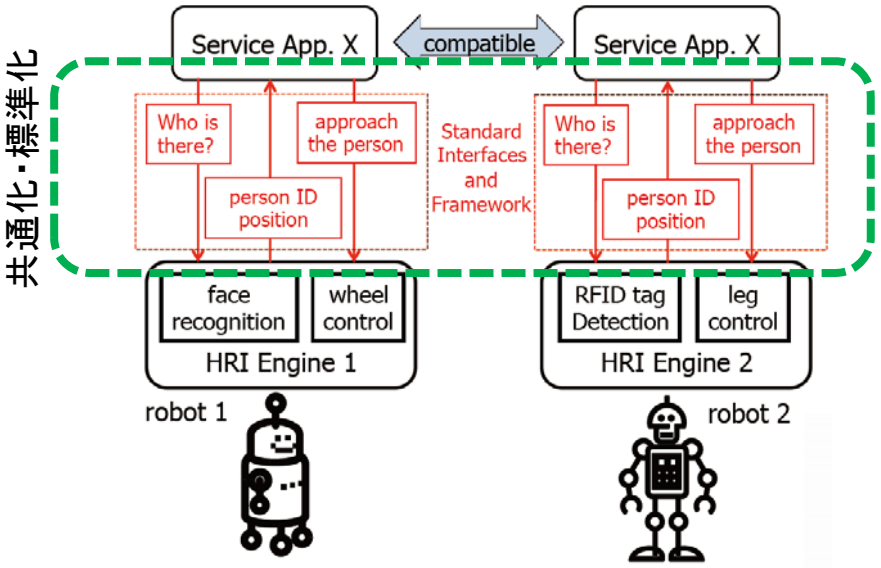
- 商業施設・病院・家などのさまざまな場所における 人々の活動を支援
- ロボット、スマートフォンアプリ、環境センサがネットワークを介して連携し、多地点で人々にサービスを提供することを目指す
- ARTにより開発、配布

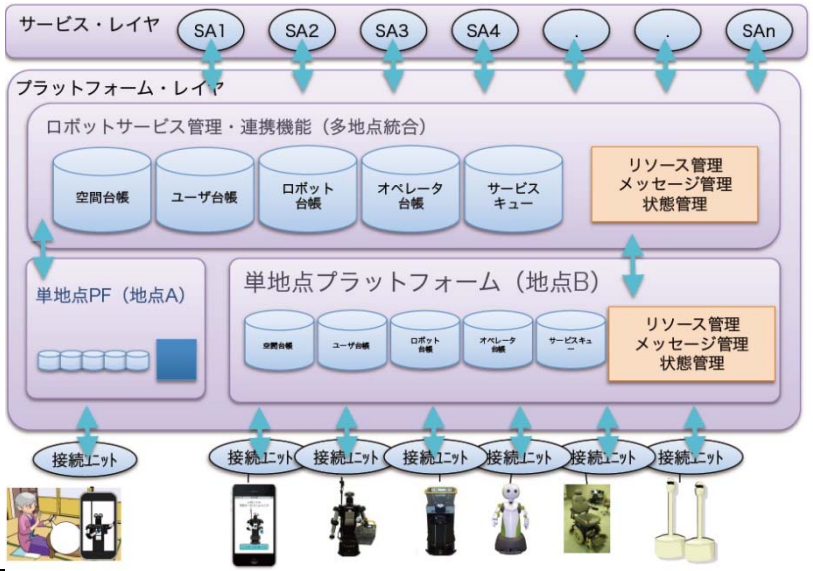


© 亀井剛次 ATR, CNR研究会2013

UNR Platform

- 一部はOMG RoIS (Robot Interaction Service)標準に準拠
- 個々のロボット仕様を気にせずアプリケーションを記述可能
- アプリと下位コンポーネントのデータのやり取りを仲介



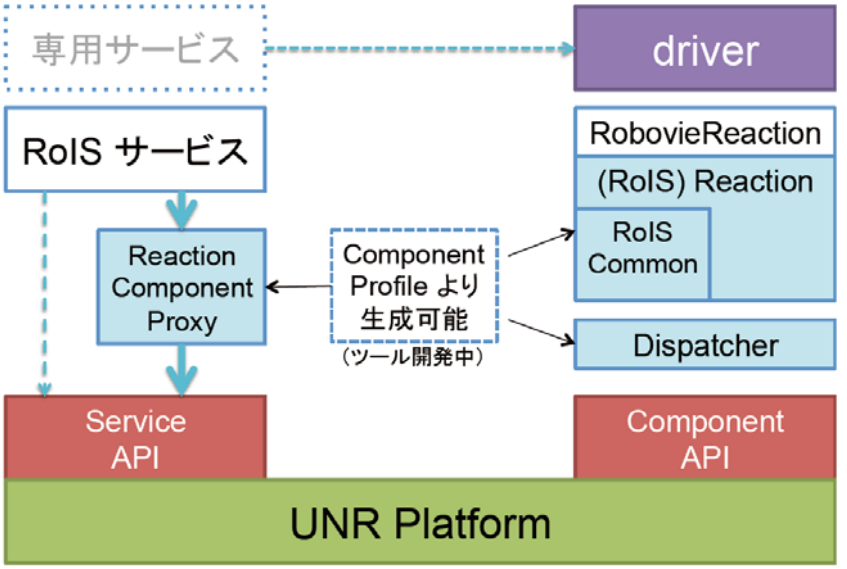


UNR Platform

- RoIS: ロボット対話サービスに必要な機能を標準化
- UNRプラットフォームは各種RoISサービスをクライアントの要求に応じて仲介
- ロボット側はRTMやROSなど何を利用してもよい

RoIS機能コンポーネント群

1）システム情報	9）音声認識
2）人検出	10）ジェスチャ認識
3）人位置検出	11）音声合成
4）個人同定	12）応答動作
5）顔検出	13）ナビゲーション
6）顔位置検出	14）追従
7）音検出	15）移動
8）音源位置検出	



Comparison of open frameworks

	Target	Open spec/source	Real-time	Language	OS	modularity	communication
OpenRTM-aist	Universal	○/○ OMG RTC	○	C++, C, Python, Java, .NET, Android	UNIX, Mac OS X, Windows, uITRON, QNX, VxWorks	○ CBSD	CORBA
ROS	Universal	△/○	○	C++, Python, Java, LISP, Matlab	Linux, (OS X, Windows)	× Free style	original protocol
OROCOS	Universal	△/○	○	C++, (scripting: Lua)	Linux, Windows, Etc..	○ CBSD	Ice, CORBA
OPRoS	Universal	○/○ OMG RTC	○	C++	Windows, Linux	○ CBSD	Original protocol
YARP	Humanoid/ Universal	△/○	×	C++, Java, Python, Lua, Matlab	Linux, Windows, Mac OS X	△ Free style	Original protocol
ORCA/ORCA2	Universal	△/○	○	C++, Python	RTLinux, Other	○ CBSD	CORBA, CURD, Ice
MSRS	Universal	△/△	×	.NET（C++,C#,VB,etc.）	Windows	△ SOA	DSS・SOAP

Comparison of open frameworks

	Target	Open spec/source	Real-time	Language	OS	modularity	communication
OpenRTM-aist	Universal	○/○ OMG RTC	○	C++, C, Python, Java, .NET, Android	UNIX, Mac OS X, Windows, uITRON, QNX, VxWorks	○ CBSD	CORBA
ORiN	FA robots	○/× ISO ORiN	×	C++	Windows	△ OOP	DCOM, CORBA
RSNP	Internet Service	○/× RSi RSNP	×	Java	Java VM	△ OOP	SOAP
UNR Platform	Internet Service	○/○ OMG RoIS/RLS	×	Java	Java VM	△ OOP	SOAP
PlayerStage	Mobile robot	△/○	○	C, C++, Tcl, LISP, Java, and Python	Linux, Etc..	△ PO	original protocol
OPEN-R	AIBO, SDR3X	○/×	×	C++	Linux	△ OOP	original protocol
Open Robot Controller Architecture	Universal	△/×	×	Java, Python	Linux, Etc..	△ OOP	HORB

終わりに

- ロボット新戦略とロボットOS・ミドルウェア
- RTミドルウェアの目的、アーキテクチャ、応用
- その他のロボットOS/ミドルウェアの動向

詳しくは…

検索 openrtm

本日の資料はopenrtm.orgに掲載します。
http://openrtm.org/openrtm/node/6023

レポート課題（1）

1. ロボット制御に必要な以下のプログラムを示せ

   a. 2自由度のアームの逆運動学を計算する以下の仕様の関数のPythonプログラムを作成し実行結果を示せ （20点）

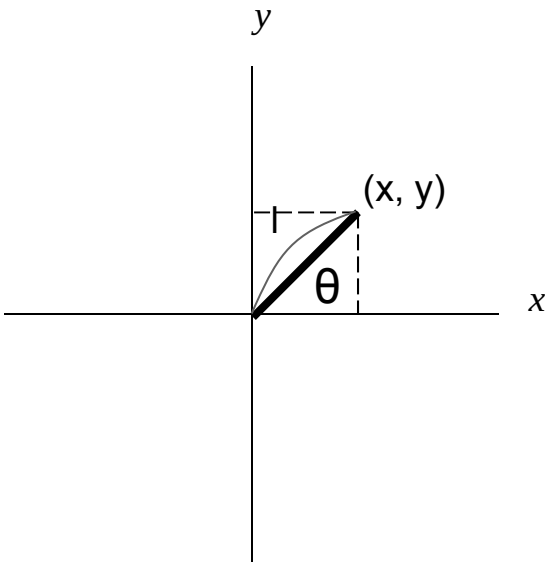
関数: th = invkinem(link, pos)
th: 2つの関節の角度[deg]
link: 2つのリンク長[m]
pos：手先位置[m]
引数、戻り値はいずれも要素数2の配列とする

   b. ジョイスティックの現在値から移動ロボットの車輪角速度を出力するPythonプログラムを作成し実行結果を示せ （20点）

関数: vl, vr = joy2vel (k, x, y)
k: 適当な係数
x, y: ジョイスティックの現在値
vl, vr：移動ロボットの左右車輪角速度 [rad/s]

ヒント：Pythonでのプログラミング

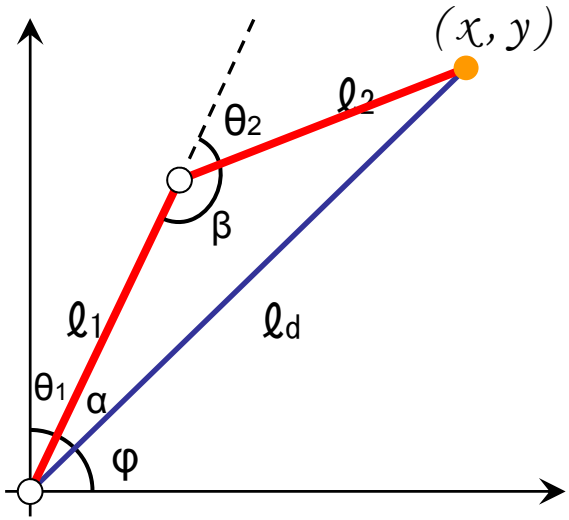
- 代入 (xに1を代入)
  - x=1
- 四則演算
  - ×→＊、÷→／
- 平方根(√)
  - math.sqrt(): 例 math.sqrt(x*x + y*y)
- 三角関数
  - math.sin(x), math.cos(x)
- arccos、arctan
  - math.acos(x), math.atan2(y,x)
- 円周率
  - math.pi
- ユークリッド距離
  - math.hypot(x, y) : 右図 $l = \sqrt{(x^2+y^2)}$



ブラウザ上でプログラミング可能なサイト https://paiza.io/ を利用するとよい

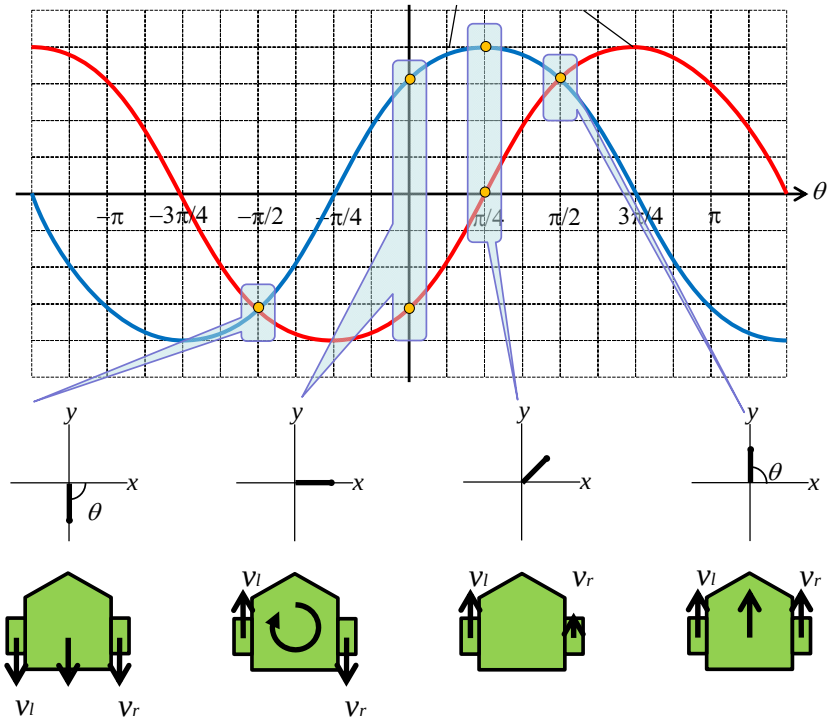
ヒント：a.2自由度アームの逆運動学関数

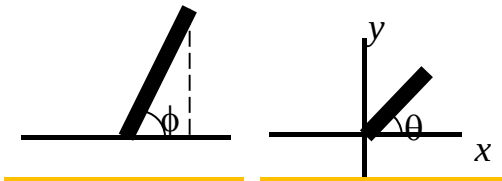
```
import math
def invkinem(link, pos):
    l1 = link[0]
    l2 = link[1]
    x = pos[0]
    y = pos[1]
    th[0] =
    th[1] =
    return th

link = (1.0, 1.0)
path = ((-1.0, 1.0), (-0.5, 1.0), (0.0, 1.0), (0.5, 1.0), (1.0,1.0))
for pos in path:
    print invkinem(link, pos)
```



ヒント：b.ジョイスティックから車輪速度への変換

```
import math
def joy2vel(k, pos):
    th = math.atan2(pos[1], pos[0])
    v =
    vl =
    vr =
    return (vl, vr)

pos = ((0.0,1.0), (1.0,1.0), (1.0, 0.0))
for  p in pos:
    print joy2vel(1.0, p)
```



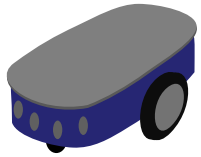


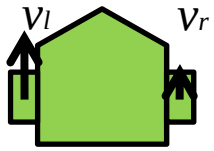
横から見る
↓
どれだけ倒したか

上から見る
↓
どの方向へ倒したか

x, yの値が出力される

指令値
左右の車輪の（角）速度





このタイプ（差動駆動型2輪移動ロボット）は
左右の車輪の速度を与えることで
ロボットを任意の場所へ移動させる

左右の車輪の速度を入力する

レポート課題（2）

2. ミドルウエアを利用した以下のサンプルプログラムを示せ。言語はC++とする。
   a. ロボットミドルウエアを一つ選び、データの送信を行う手順・方法を調べ説明せよ。結果として、コメントを付したソースコード（完全である必要はないが、データ送信に必要な最低限の部分を示すこと。例えばOpenRTMであればヘッダとonInitialize, onExecute関数部分。）を添付せよ。

   b. 同様に、データの受信を行う手順・方法を調べ説明せよ。結果として、コメントを付したソースコード（完全である必要はないが、データ受信に必要な最低限の部分を示すこと。例えばOpenRTMであればヘッダとonInitialize, onExecute関数部分。）を添付せよ。

コードに付記されたコメントを重視します。

3. 授業の感想

解答

- レポート提出期限後に資料掲載サイトに解答を掲載します。

http://openrtm.org/openrtm/node/6023