2016年6月8日（水）
ROBOMECH2016 チュートリアル
RTミドルウェア講習会、RSNP講習会合同セッション

第１部（その２）：OpenRTM-aistおよびRTコンポーネントプログラミングの概要

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
安藤　慶昭

はじめに

- RTミドルウェアとは？
- ロボット新戦略
- ロボットソフトウェアプラットフォーム
- RTミドルウェアの特長
- ROSとの比較、動向
- RTミドルウェアとオープンソースコミュニティー
- RTコンポーネントの開発
- まとめ

RTミドルウェアとは？

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

従来のシステムでは…

Joystick

Joystick
software

Robot Arm
Control software

Robot Arm

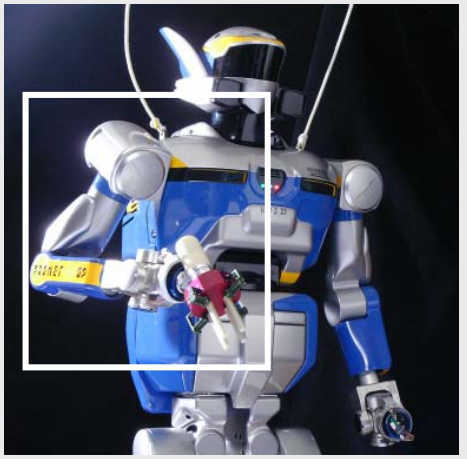
互換性のあるインターフェース同士は接続可能

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…

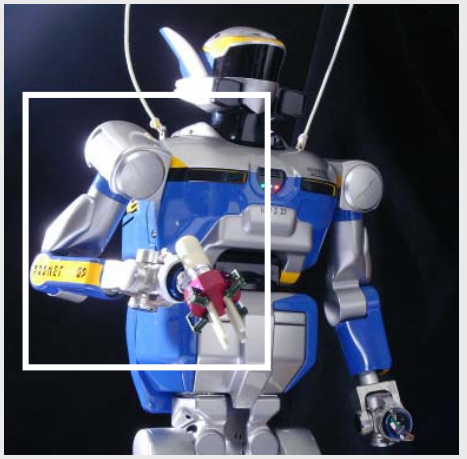
RTミドルウエアは別々に作られたソフトウエアモジュール同士を繋ぐための共通インターフェースを提供する

Joystick

Joystick software

compatible arm interfaces

Arm A Control software



Humanoid's Arm

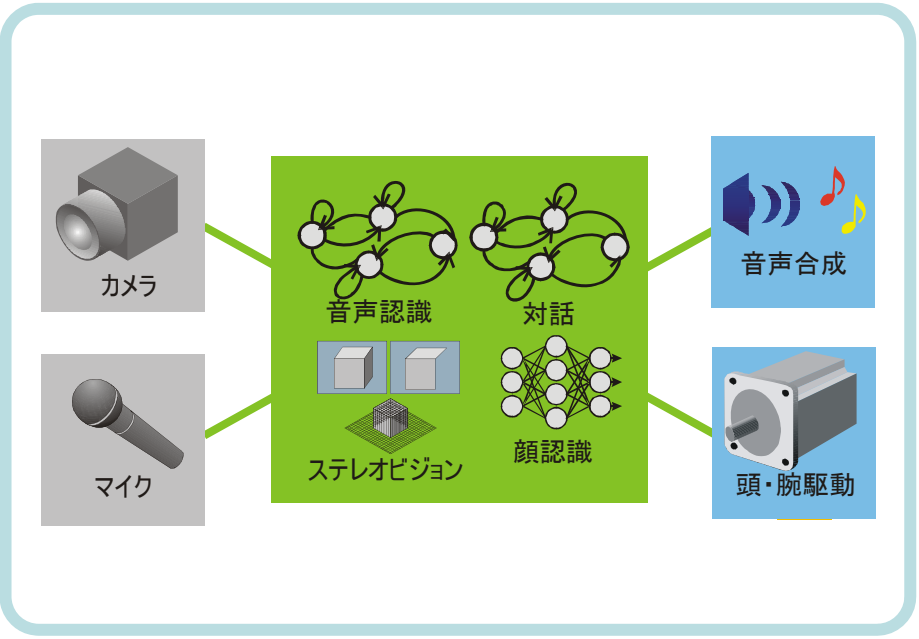
Arm B Control software

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

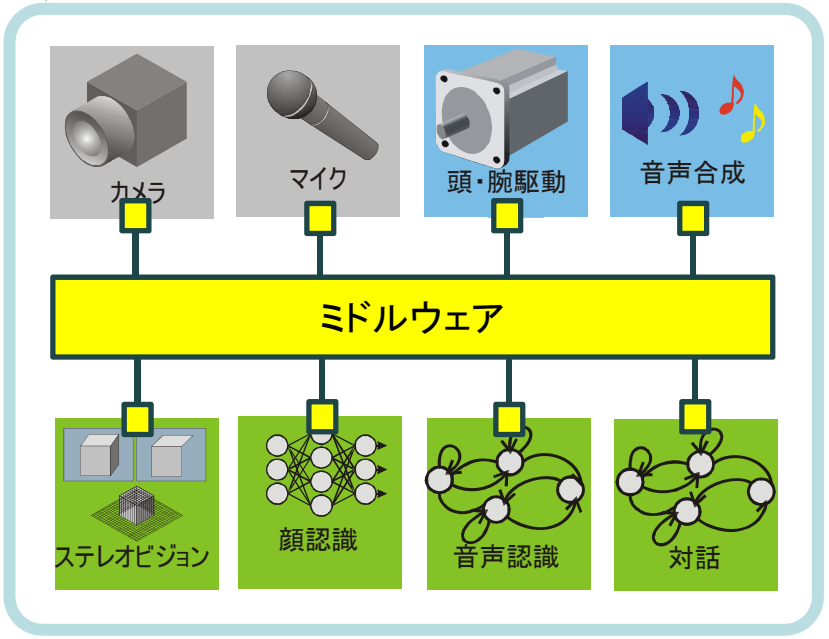
ロボットソフトウェア開発の方向

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

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

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：CORBA、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

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

RTコンポーネント化のメリット

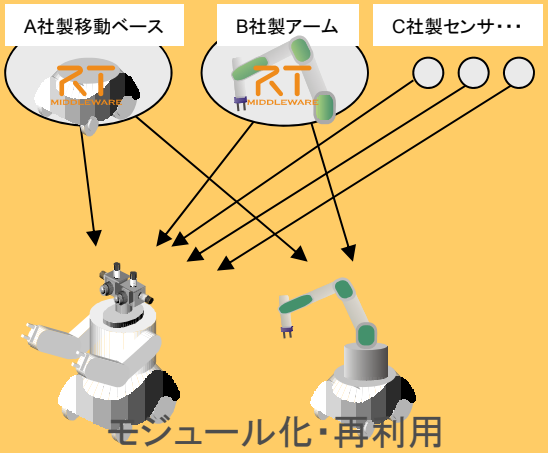
モジュール化のメリットに加えて

- 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

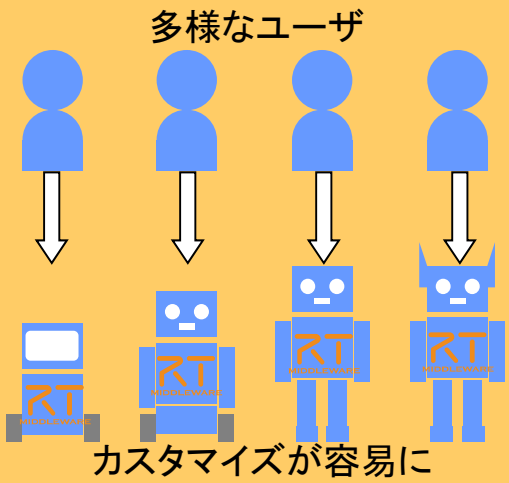


ロボットの低コスト化

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

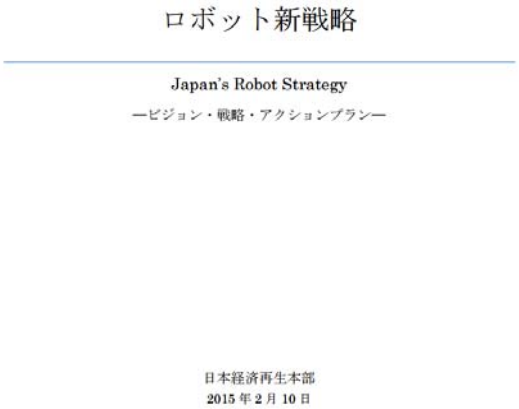
ロボット新戦略

ロボット新戦略(2015年2月)

- 日本経済再生本部決定文書
  - ロボット革命実現会議の取りまとめ

内容

- 第1部 総論
  - 現状分析
  - 革命実現のための方策（3つの柱）
- 第2部 アクションプラン
  - 分野横断事項
    - 協議会設置・次世代技術開発・標準化・人材育成等
  - 分野別事項
    - ものづくり・サービス・インフラ/災害対応/建設・農林水産/食品産業分野


ロボット新戦略

Japan's Robot Strategy

―ビジョン・戦略・アクションプラン―

日本経済再生本部
2015年2月10日

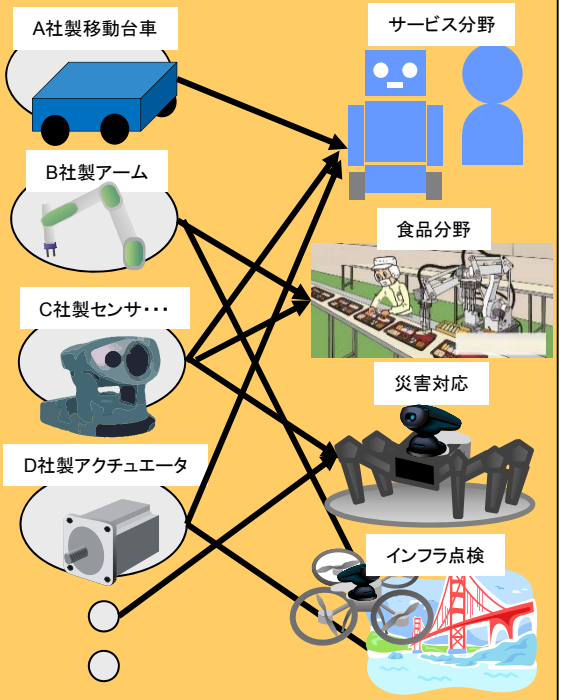
ロボット新戦略の重要性

関係各省庁も議論に参加し、内容の詳細に関与し合意したロボット戦略

→今後の施策に大きな影響

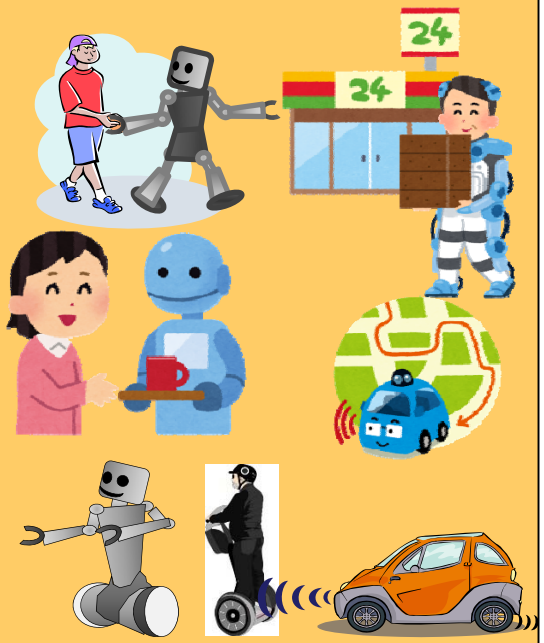
ロボット新戦略：3つの柱

ロボット創出力強化



多様な分野の要請に柔軟に対応できるロボット創出力強化とイノベーションを誘発する場の創出

ロボットの活用・普及



サービス、介護等様々な分野でロボットの開発・導入を進める環境整備の実施

世界を見据えた ロボット革命の展開・発展

Industry4.0やIndustrial Internet等世界的な潮流の中で、ものづくりを引き続きリードするロボット技術の世界展開

RTミドルウェアで

（ロボット創出力を）強化　（メーカーと地域を）つなぐ　（市場を）広げる

ロボット新戦略とRTミドルウェア（1）

p.13

第1部 総論 第2章 ロボット革命実現のための方策

第1節 ロボット創出力―日本のロボットを徹底して強化する

第3項 次世代に向けて

（2）国際的な展開を見据えた規格化・標準化

我が国おいて開発されたロボットの国際的な展開を見据えると、予め国際標準や規格を構築し、それに準拠する形で実用化を目指すことが重要である。

その際、今後、複数のロボットが連携しシステムとして機能することが必要となってくることを踏まえ、個々のデバイスのほか、ミドルウェア（ロボットOS）等のソフトウェア・インターフェース、通信等の機器間連携、そしてこれらの機能評価に関し、規格化・標準化に重点的に取り組む。

ロボット新戦略とRTミドルウェア（2）

p.23

第2部 アクションプラン―五カ年計画

第1章 分野横断的事項

第2節 次世代に向けた技術開発

（3） 開発すべき技術

④OS・ミドルウェア等

ロボットやロボットシステムを構成するためには、要素技術・部品・ロボットそのものを**OS・ミドルウェア等の基盤的なソフトウェアで統合**する必要がある。また、これにより**互換性、開発の生産性も高める**ことが可能となる。そうした技術について、現状では、

- （中略）使い勝手が良く一定程度**標準化されたOS・ミドルウェア・プログラミング言語**等を、将来の要素技術の**発展に対応させる**必要がある。
- 異なるOSのロボット同士が対話する場合、あるいは、ロボットに新たなモジュールを搭載する場合など、ロボット及びモジュールの**インターフェースを標準化する必要**がある。

等の課題が存在している。こうした課題に対して、動作環境を模擬するシミュレータや、シミュレータと連携可能なOS・ミドルウェア、また、**標準になり得る汎用的なOS・ミドルウェア等の研究開発が必要**と考えられる。

ロボット新戦略とRTミドルウェア（3）

pp.27-32

第2部 アクションプラン－ 五カ年計画

第1章 分野横断的事項

第3節 ロボット国際標準化への対応

第1項 ハードウェア／ソフトウェアのモジュール化を見据えた共通基盤

（ミドルウェア・ロボット OS）

（中略）ロボットを構成するハードウェア、ソフトウェアの機能要素をモジュール化・共通化することで、（中略）ロボットを低価格で構成することが可能となる。また、共通のソフトウェアプラットフォームを利用することで、（中略）ロボットインテグレータは対象毎の機能の実現に注力することができる。

（3） 2020 年に目指すべき姿

ロボットによる産業革命を日本主導で実現するためにも、（中略）ロボット制御の基幹部分であるロボット OS・ミドルウェアについても積極的な取組を進めていくことが重要である。（中略）様々な国際標準を日本主導で策定することで、（中略）日本のロボット産業が世界をリードすることが可能となる。

（中略）そのため、新たに設置される「ロボット革命イニシアティブ協議会」（第1項参照）が中心となって、必要な情報収集及び国際発信を行い、国際標準化の流れを主導していく。

ロボットソフトウェアプラットフォーム

ロボットプラットフォームの必要性

「ロボット新戦略」三つの柱

① 世界のロボットイノベーション拠点
「ロボット創出力の抜本的強化」
② 世界一のロボット利活用社会
「ショーケース（ロボットがある日常の実現）」
③ 世界をリードするロボット新時代への戦略

ロボット未活用領域

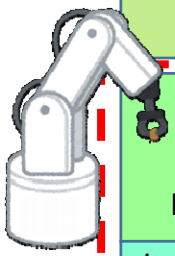
・ものづくりにおける未活用作業
・サービス産業（物流、バックヤード）
・生活支援、介護・医療
・災害対応、インフラ点検
・コミュニケーション、モビリティー等

- ユーザが求めるロボットをいち早く提供できる「Easy to use」なロボットの実現
- インテグレーションコストの削減によるイニシャルコストの低減
- 多様な分野への展開によりスケールメリットが働く産業横造

ロボットのプラットフォーム化によるロボット普及促進

■ プラットフォームロボットSWG（RRI・WG3に設置）
・プラットフォーム化が有効に働く分野について議論
・要求仕様や具体的なハードウェアプラットフォームを提案
・ソフトウェアプラットフォームとエコシステムを含めた具体的アクションプラン
を提言としてまとめた

ロボットソフトウェアプラットフォーム



4. SI費用

ティーチング、チューニング等顧客の要望に応じた作業
知能化研究・パッケージ化を進めPF化することで削減

2. ベース機能

ﾋﾟｯｸｱﾝﾄﾞﾌﾟﾚｰｽ、SLAM等用途別基本機能
MoveIt!, OepnCV等OSS活用or機能強化

1. ソフトウェア基盤

利用するHW・用途に応じて既存OS利用、
RTM・ROS等OSS活用or機能強化

3. 非機能要件

・テスト・検証・性能指標
・ドキュメンテーション
・教育・コミュニティー
・安全規格とのマッチング
・ライセンス・特許
・長期的な保守・運用体制

※OSS=オープンソースソフトウェア

ソフトウェアプラットフォーム

近年大規模かつ公共性の高いソフトウェアは企業がオープンソース開発に関与

コンパイラCLANGの例
（オープンソース）

ロボットのミドルウェア・基本機能ライブラリも公共性は高いが、OSS開発コミュニティーが未発達

プラットフォームとして強化すべき領域

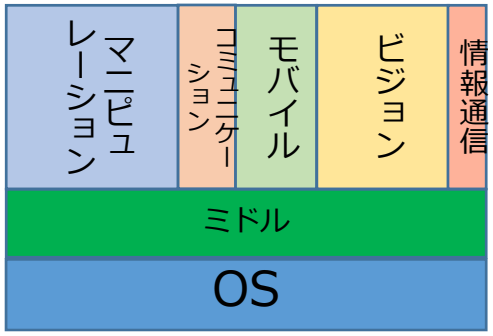
分野横断的機能開発 → 多様な参加者が必要
分野別機能開発 → 1社のみの開発は負担大
横展開可能な技術 → 情報共有の場が必要
市場が成熟するまではコンソーシアム方式が適当

ロボットソフトウェアプラットフォームコンソーシアム

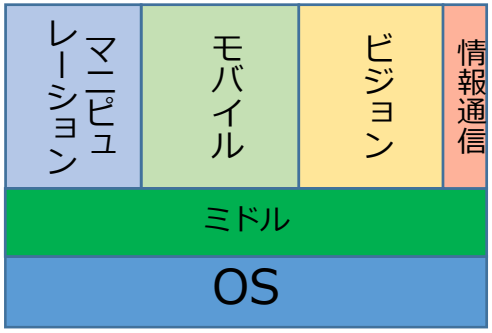
プレイヤー：ハードメーカー、ソフトベンダ、SIer、ユーザ

ソフトウェアプラットフォームの仕様

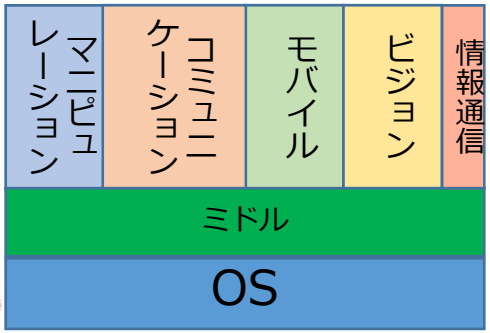
①ものづくり分野



②サービス分野（物流）



③生活支援分野



ソフトウェアプラットフォームの仕様

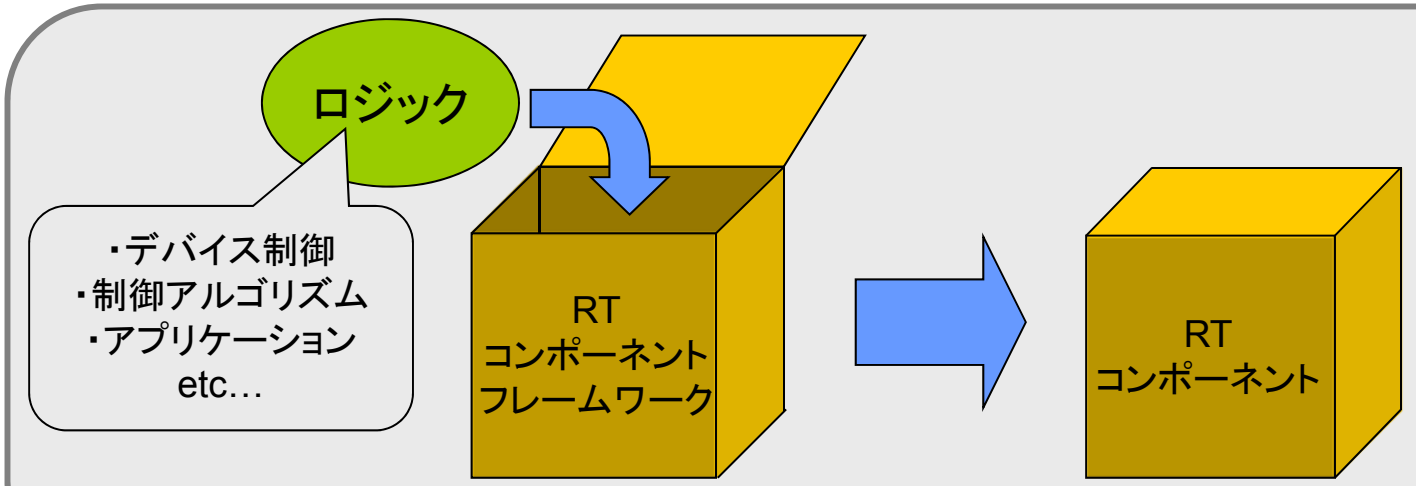
- 分野別ソフトウェアPF
  - A. マニピュレーション機能（①、②、③）
  - B. コミュニケーション機能（①、③）
  - C.モバイル機能（①、②、③）
  - D. ビジョンセンサ機能（ ①、②、③）
  - E. 情報通信機能（ ①、②、③）

  オープンソースを活用し品質向上と機能強化要求を満たすものが無ければ新規開発
- ミドルウェア層
  - コンポーネント指向開発 ---------- RTM実現済、ROS2で実現予定
    - 粒度：ｱﾙｺﾞﾘｽﾞﾑ・ﾃﾞﾊﾞｲｽﾚﾍﾞﾙ以上の細粒度
  - パッケージング機能 ---------- RTM要開発、ROS実現済
  - 分散システム開発 ---------- RTM・ROS要開発
    - ネットワーク透過、通信QoS制御
  - リアルタイム制御可能： RTM一部実現済、ROSは要開発
    - ﾏﾆﾋﾟｭﾚｰﾀ： 位置制御 5ms、力制御 1ms
    - 移動制御： 5～10ms が可能
    - モジュール化開発
  - 多言語開発可能 ---------- RTM一部実現済、ROSは要開発
    - C/C++、Python等スクリプト言語）
  - 多種OS対応 ---------- RTM要商用RTOS対応 ROS要Windows対応強化
    - Linux, Windows, リアルタイムOS（RTOS)等
- OS層
  - ハードリアルタイム実行可能
    - ﾌﾟﾘｴﾝﾌﾟﾃｨﾌﾞ・優先度ﾍﾞｰｽｽｹｼﾞｭｰﾘﾝｸﾞ
    - マルチタスク対応
    - 応答時間：100 μs程度以下
  - POSIX互換（非組込）またはTRON等組込OS
  - マルチコア対応
  - ネットワーク対応
  - 32bitまたは64bit

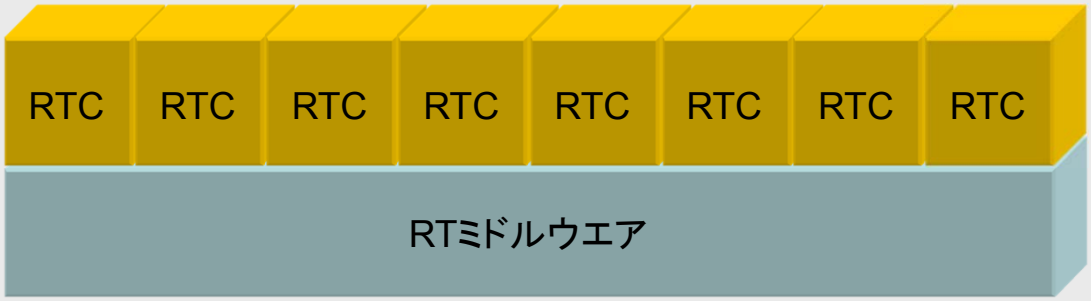
  ハードPFに応じて適切なものを選択

RTミドルウェアの特長

RTミドルウエアとRTコンポーネント



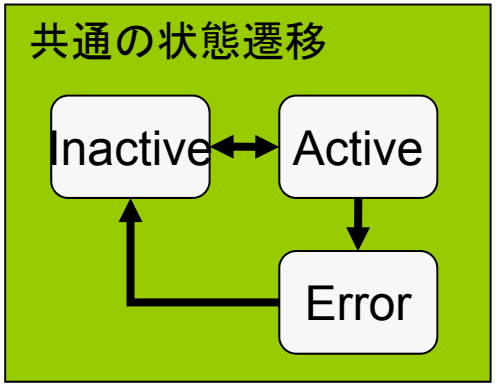
ロジックを箱（フレームワーク）に入れたもの＝RTコンポーネント（RTC）

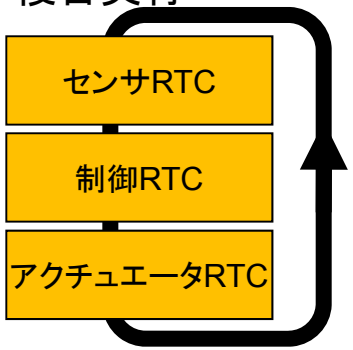


RTCの実行環境（OSのようなもの）＝RTミドルウエア（RTM）
※RTCはネットワーク上に分散可能

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

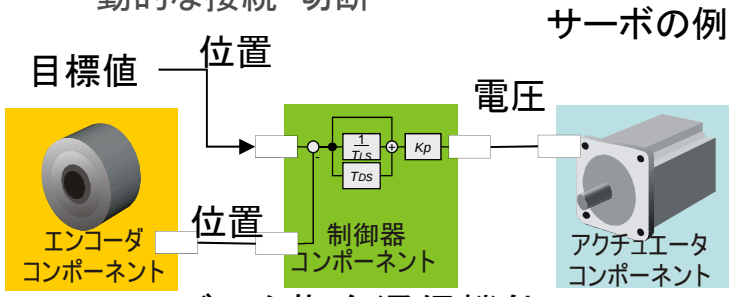




ライフサイクルの管理・コアロジックの実行

データポート

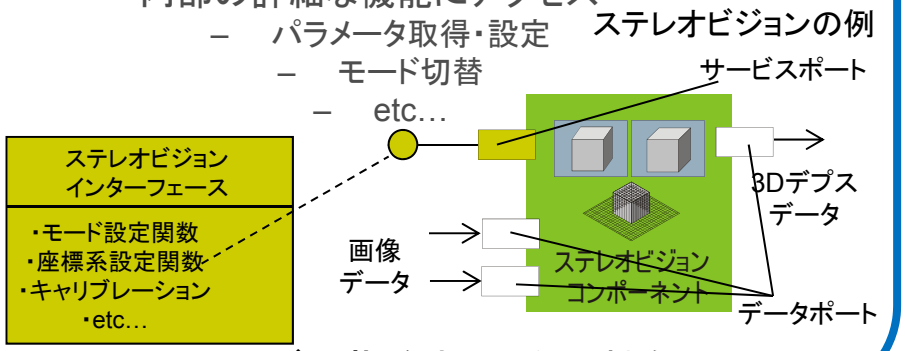
- データ指向ポート
- 連続的なデータの送受信
- 動的な接続・切断



データ指向通信機能

サービスポート

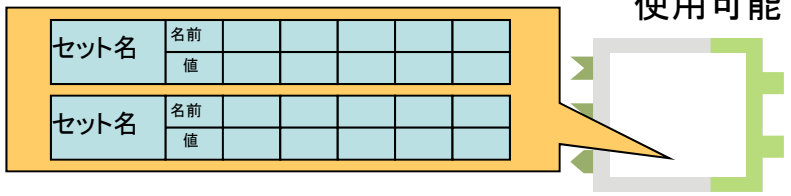
- 定義可能なインターフェースを持つ
- 内部の詳細な機能にアクセス
  - パラメータ取得・設定
  - モード切替
  - etc…



サービス指向相互作用機能

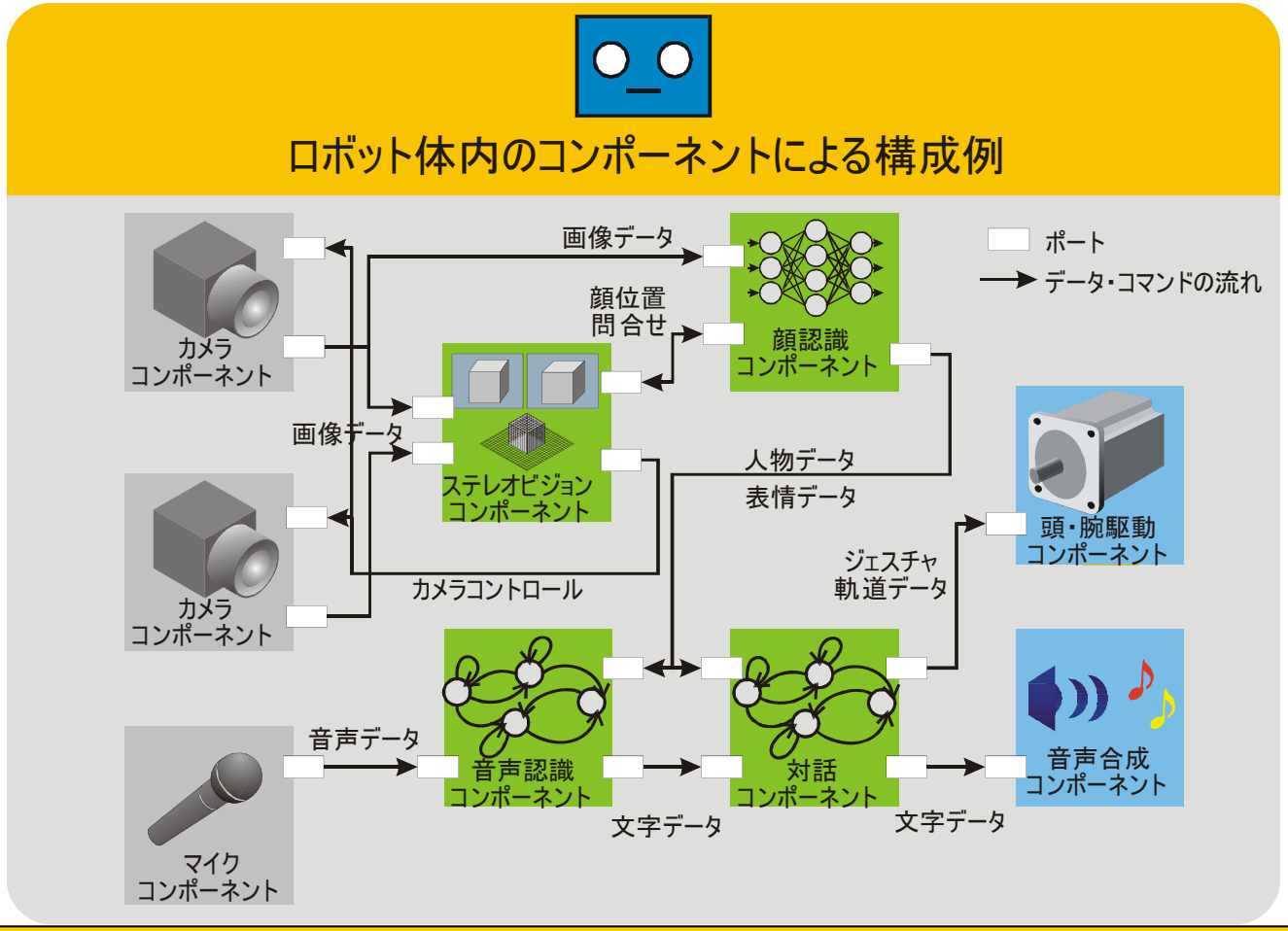
コンフィギュレーション

- 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



複数のセットを
動作時に
切り替えて
使用可能

RTCの分割と連携



（モジュール）情報の隠蔽と公開のルールが重要

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。

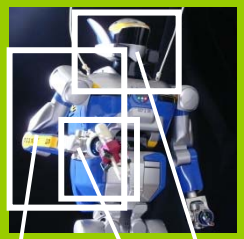
ロボットA

RTC | RTC
RTM
Solaris

ロボットB

RTC | RTC
RTM
FreeBSD

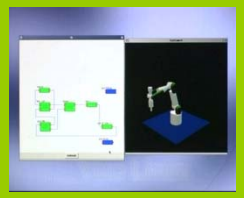
ロボットC



RTC | RTC | RTC
RTM
ARTLinux

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

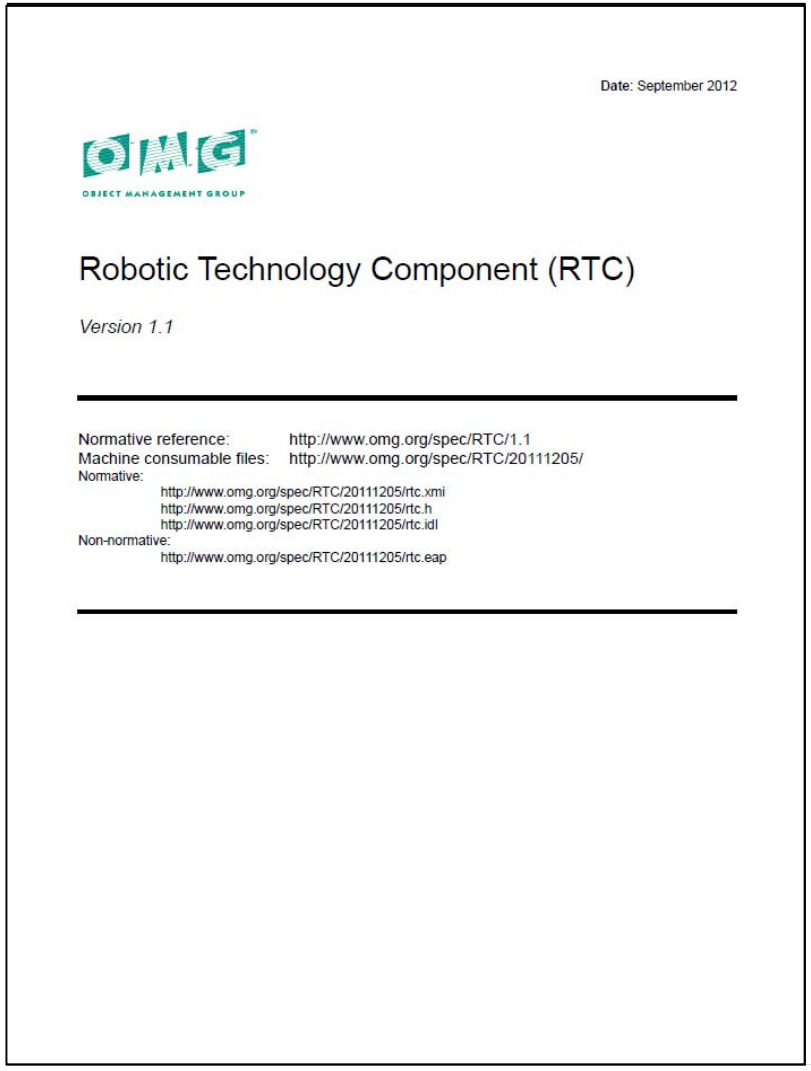
TRON
RTM
RTC | RTC

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。

OMG RTC 標準化

- 2005年9月
  RFP: Robot Technology Components (RTCs) 公開。
- 2006年2月
  Initial Response : PIM and PSM for RTComponent を執筆し提出
  提案者:AIST(日)、RTI(米)
- 2006年4月
  両者の提案を統合した仕様を提案
- 2006年9月
  ABにて承認、事実上の国際標準獲得
  FTFが組織され最終文書化開始
- 2007年8月
  FTFの最後の投票が終了
- 2007年9月
  ABにてFTFの結果を報告、承認
- 2008年4月
  OMG RTC標準仕様 ver.1.0公式リリース
- 2010年1月
  OpenRTM-aist-1.0リリース
- 2012年9月
  ver. 1.1改定
- 2014年12月
  FSM4RTC(FSM型RTCとデータポート標準) Beta1



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 RTC ファミリ

名称	ベンダ	特徴	互換性
OpenRTM-aist	AIST	C++, Python, Java	---
OpenRTM.NET	SEC	.NET(C#,VB,C++/CLI, F#, etc..)	◎
RTM on Android	SEC	Android版RTミドルウエア	◎
H-RTM	本田R&D	OpenRTM-aist互換、FSM型コンポーネントをサポート	◎
RTC-Lite	AIST	PIC, dsPIC上の実装	○(ブリッジ)
miniRTC, microRTC	SEC	CAN・ZigBee等を利用した組込用RTC実装	○(ブリッジ)
RTMSafety	SEC/AIST	機能安全認証 (IEC61508) capableなRTM実装，商用	○(ブリッジ)
RTC CANOpen	SIT, CiA	CANOpen-RTCマッピングを定めたCiA 標準	○(ブリッジ)
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	×

同一標準仕様に基づく多様な実装により
- 実装(製品)の継続性を保証
- 実装間での相互利用がより容易に

応用例（研究用プラットフォーム）

HRP-2（川田工業）

HRP-4（川田工業）

HIRO(川田工業）

ビュートローバーRTC/RTC-BT(VSTONE)

OROCHI（アールティ）

応用例(実応用その他)

S-ONE:SCHAFT

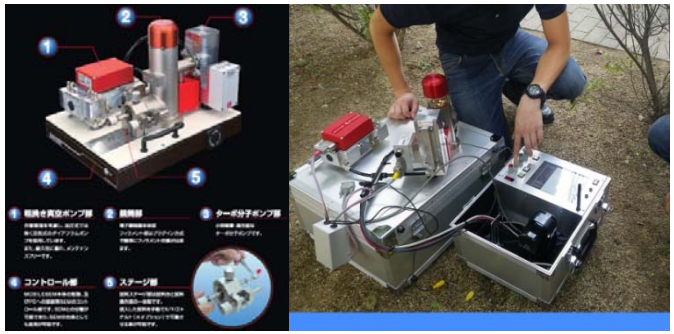
HRP-4C: AIST

TAIZOU: GRX

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

RAPUDA:Life Robotics



新日本電工他: Mobile SEM

新日本電工他: 小型ベクレルカウンタ

ROSとの比較、動向

ROS

ROSの良い点

- UNIXユーザに受けるツール（特にCUI）が豊富
- 独自のパッケージ管理システムを持つ
- ノード（コンポーネント）の質、量ともに十分
  - WGが直接品質を管理しているノードが多数
- ユーザ数が多い
- メーリングリストなどの議論がオープンで活発
- 英語のドキュメントが豊富

ROSの問題点

- Ubuntu以外の対応が今一つ
  - Windows対応はいろいろな人が試したがいまだに公式には含まれない
- コアライブラリの仕様が固まっていない
  - 他の言語で実装する際の妨げ
  - サードパーティー実装が出にくい
  - 品質を保証しづらい
- コンポーネントモデルがない

OpenRTM

OpenRTMの良い点

- 対応OS・言語の種類が多い
  - Windows、UNIX、uITRON、T-Kernel、VxWorks、QNX
  - Windowsでネイティブ動作する
- GUIツールがあるので初心者向き
- 仕様が標準化されている
  - OMGに参加すればだれでも変更可
  - サードパーティー実装が作りやすい
  - すでに10程度の実装あり
- コンポーネントモデルが明確
  - オブジェクト指向、UML・SysMLとの相性が良い
  - モデルベース開発
- IEC61508機能安全認証取得
  - RTMSafety

OpenRTMの問題点

- コンポーネントの数が少ない
- パッケージ管理システムがない
- CUIツールが少ない
  - UNIXユーザ受けしない
- ユーザ数が少ない
- 英語のドキュメントが少ない
- 知名度が低い
- 開発速度が遅い
  - 現在は開発者一人

新たなユースケースと新たな技術

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

RTミドルウェアと
オープンソースコミュニティー

RTミドルウエアの広がり

ダウンロード数

2012年2月現在

	2008年	2009年	2010年	2011年	2012年	合計
C++	4978	9136	12049	1851	253	28267
Python	728	1686	2387	566	55	5422
Java	643	1130	685	384	46	2888
Tool	3993	6306	3491	967	39	14796
All	10342	18258	18612	3768	393	51373

ユーザ数

タイプ	登録数
Webページユーザ	365 人
Webページアクセス	約 300 visit/day 約 1000 view/day
メーリングリスト	447 人
講習会	のべ 592 人+22人
利用組織（Google Map）	46 組織

プロジェクト登録数

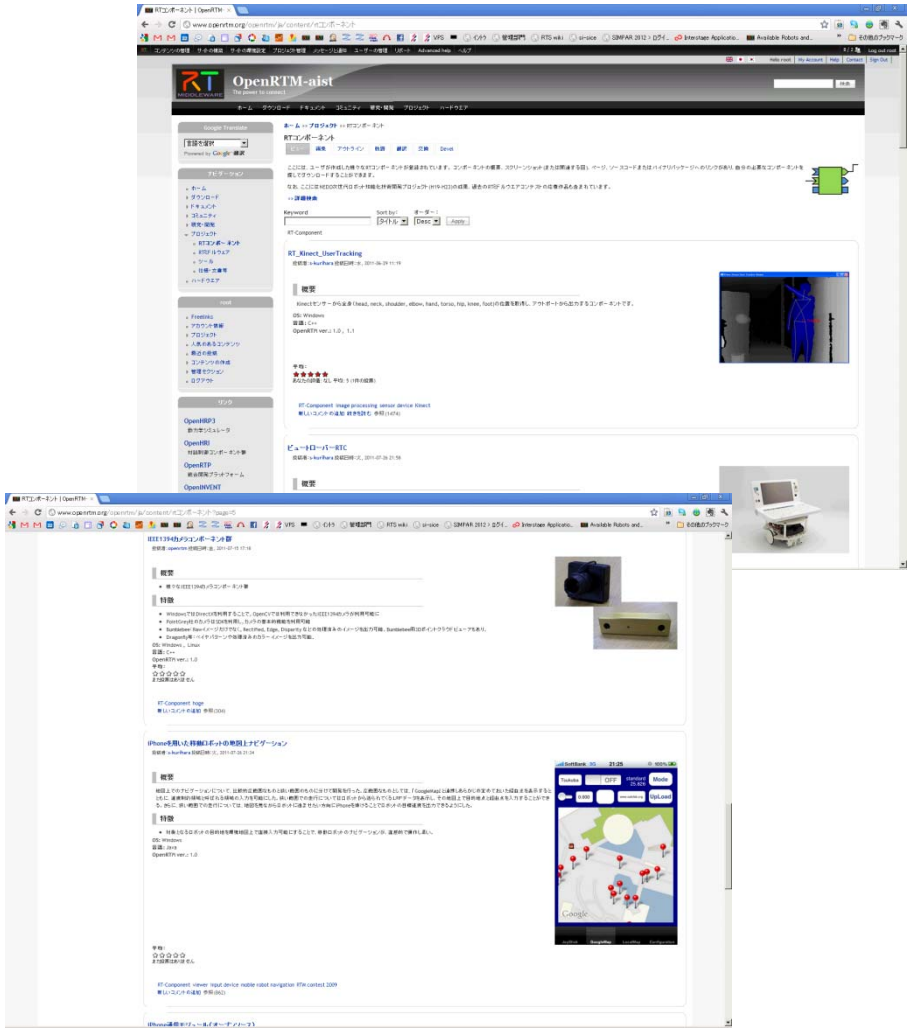
タイプ	登録数
RTコンポーネント群	287
RTミドルウエア	14
ツール	19
仕様・文書	4
ハードウエア	28

OMG RTC規格実装（11種類）

Name	Vendor	Feature
OpenRTM-aist	AIST	C++, Python, Java
OpenRTM.NET	SEC	.NET(C#,VB,C++/CLI, F#, etc..)
miniRTC, microRTC	SEC	CAN・ZigBee等を利用した組込用RTC実装
Dependable RTM	SEC/AIST	機能安全認証 (IEC61508) capableなRTM実装
RTC CANOpen	SIT, CiA	CANOpenのためのCiA (Can in automation) におけるRTC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

プロジェクトページ

- ユーザが自分の作品を登録
- 他のユーザの作ったRTCを探すことができる



タイプ	登録数
RTコンポーネント群	287
RTミドルウエア	14
ツール	19
仕様・文書	4
ハードウエア	28

サマーキャンプ

- 毎年夏に1週間開催
- 今年：8月1日～8月5日
- 募集人数：20名
- 場所：産総研つくばセンター
- 座学と実習を1週間行い、最後にそれぞれが成果を発表
- 産総研内のさくら館に宿泊しながら夜通し？コーディングを行う！





RTミドルウエアコンテスト

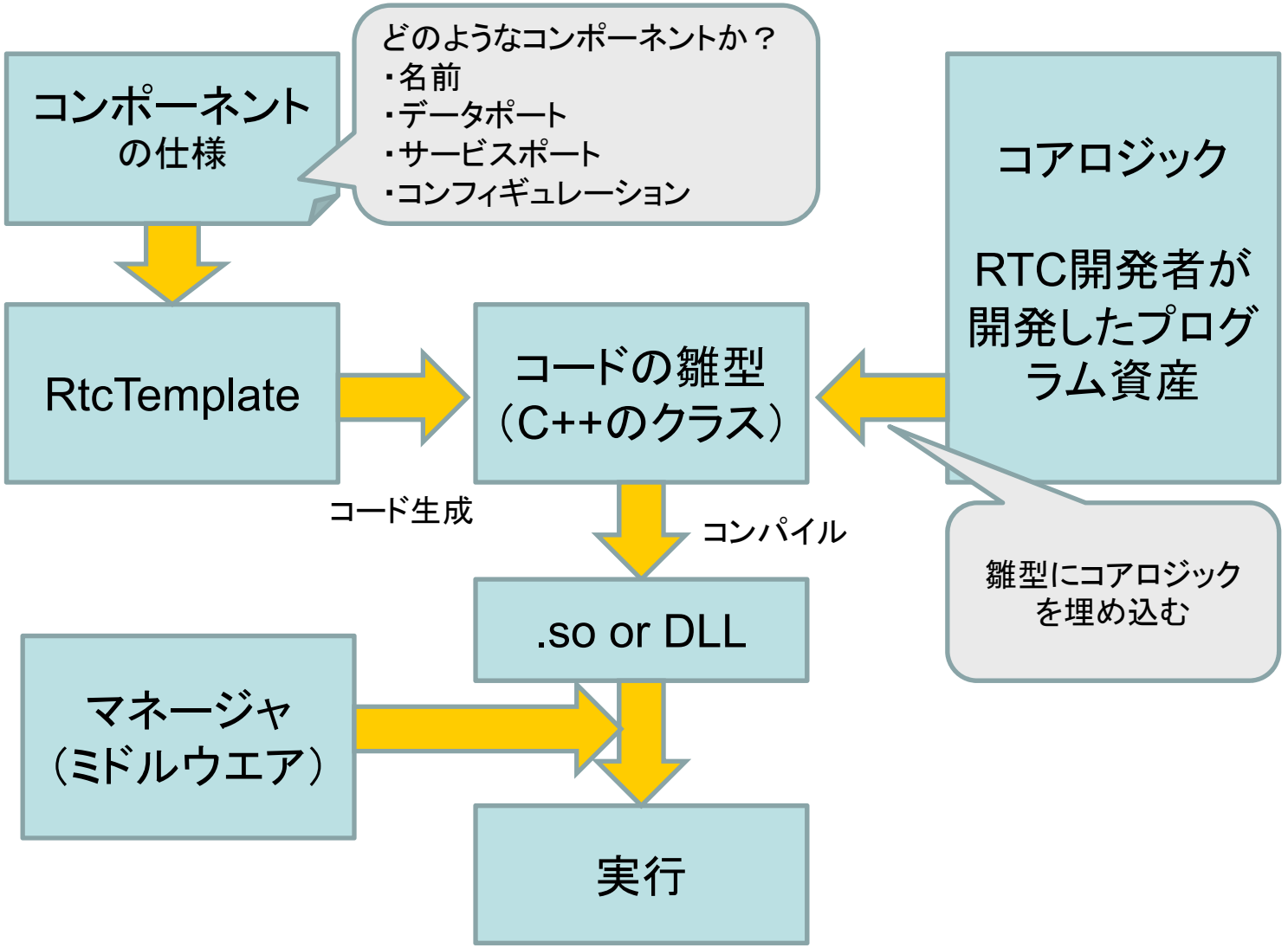
- SICE SI（計測自動制御学会 システムインテグレーション部門講演会）のセッションとして開催
  - 各種奨励賞・審査基準開示:5月頃
  - エントリー〆切:8月21日(SI2015締切)
  - ソフトウエア登録:10月ごろ
  - 講演原稿〆切:9月25日
  - オンライン審査:11月下旬～
  - 発表・授賞式:12月ごろ
- 2014年度実績
  - 応募数:20件
  - 計測自動制御学会学会RTミドルウエア賞（副賞10万円）
  - 奨励賞（賞品協賛）:2件
  - 奨励賞（団体協賛）:11件
  - 奨励賞（個人協賛）:7件
- 詳細はWebページ: openrtm.org
  - コミュニティー→イベント をご覧ください

提言

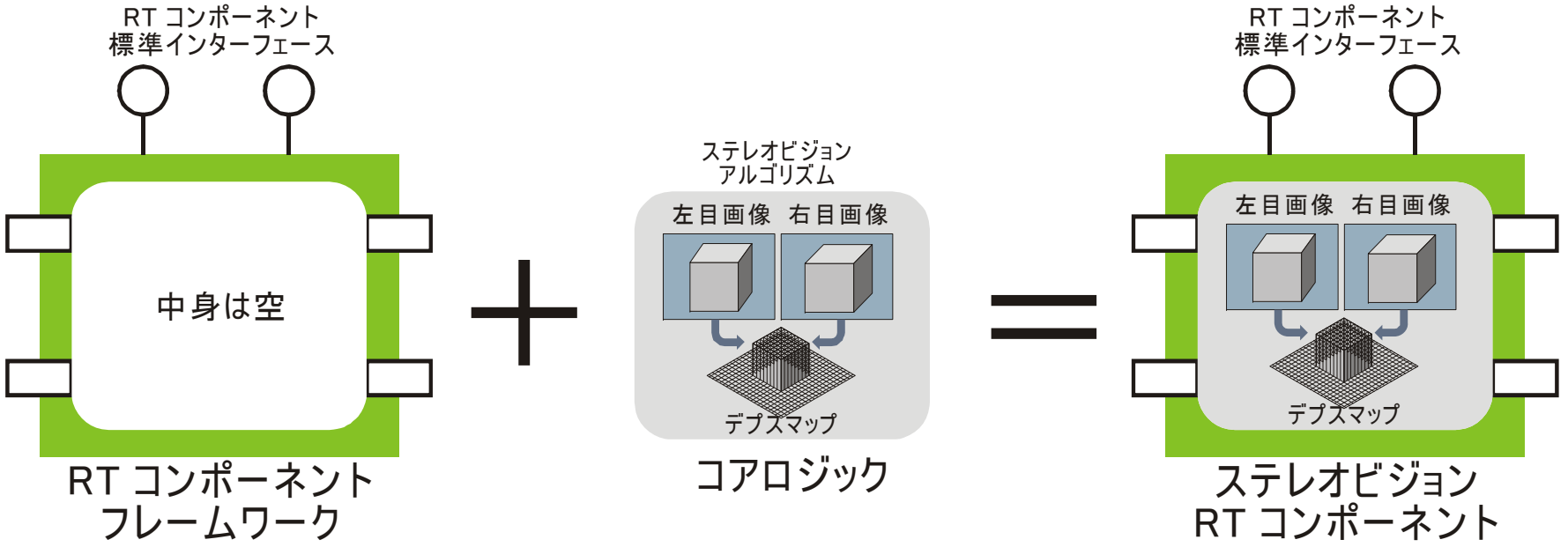
- 自前主義はやめよう！！
  - 書きたてのコードより、いろいろな人に何万回も実行されたコードのほうが動くコードである！！
  - 自分にとって本質的でない部分は任せて、本当にやりたい部分・やるべき部分のコードを書こう！！
  - 誰かがリリースしたプログラムは一度は動いたことがあるプログラムである！！
  - 人のコードを読むのが面倒だからと捨ててしまうのはもったいない！！
- オープンソースにコミットしよう！！
  - 臆せずMLやフォーラムで質問しよう！！
  - どんなに初歩的な質問でも他の人にとっては価値ある情報である。
  - 要望を積極的にあげよう！！
  - できればデバッグしてパッチを送ろう！

RTコンポーネントの開発

OpenRTMを使った開発の流れ

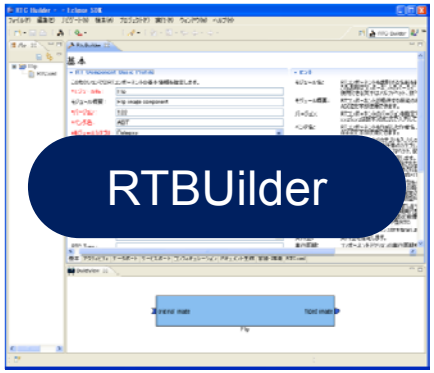


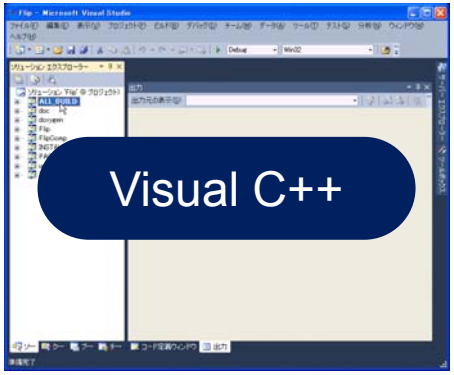
フレームワークとコアロジック



RTCフレームワーク＋コアロジック＝RTコンポーネント

コンポーネントの作成
（Windowsの場合）





コンポーネントの
仕様の入力

テンプレートコード
の生成

VCのプロジェクト
ファイルの生成

実装および
VCでコンパイル
実行ファイルの生成

コード例

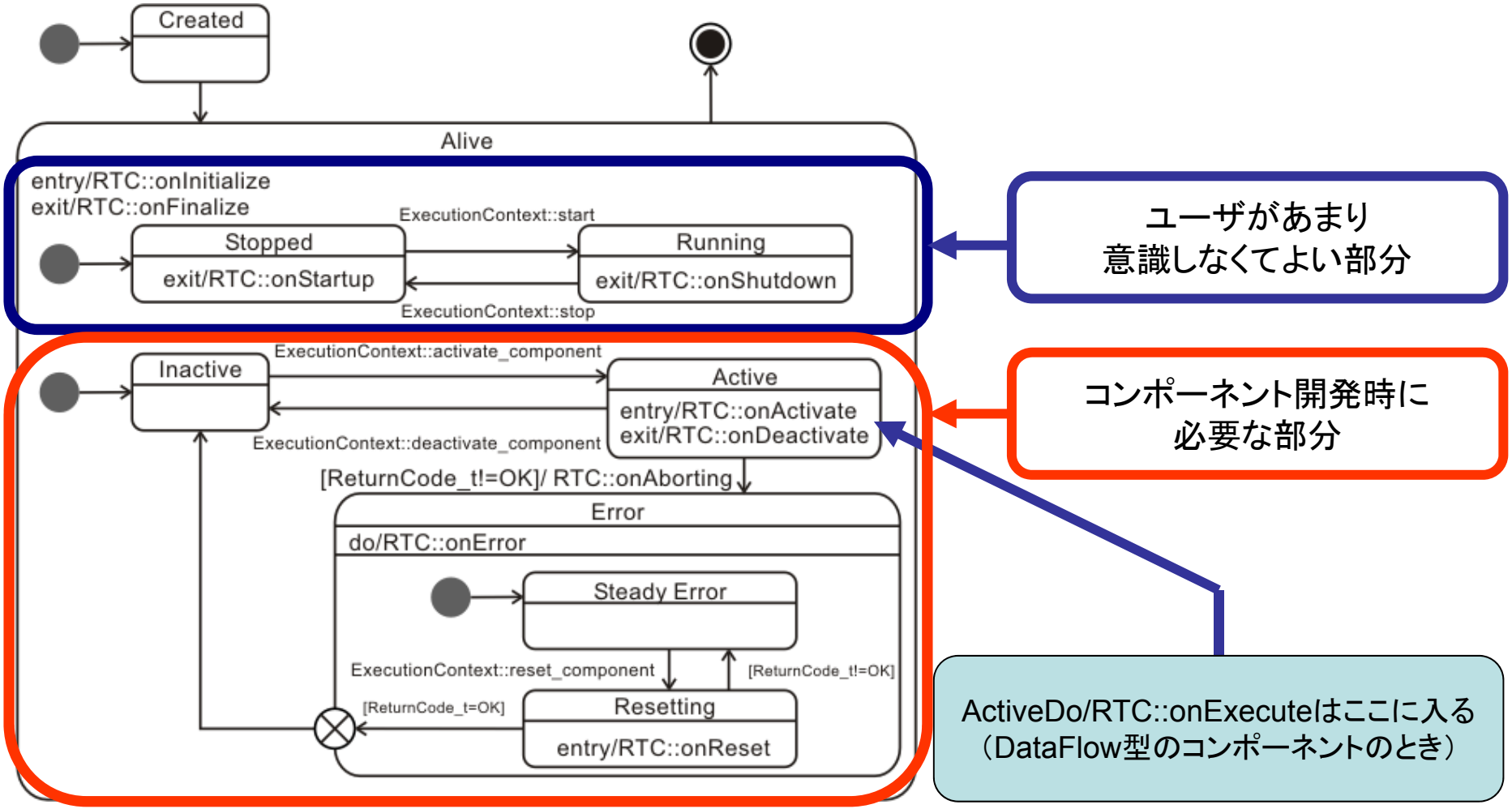
- 生成されたクラスのメンバー関数に必要な処理を記述
- 主要な関数
  - onExecute (周期実行)
- 処理
  - InPortから読む
  - OutPortへ書く
  - サービスを呼ぶ
  - コンフィギュレーションを読む

```
class MyComponent
 : public DataflowComponentBase
{
public:
  // 初期化時に実行したい処理
  virtual ReturnCode_t onInitialize()
  {
    if (mylogic.init())
      return RTC::RTC_OK;
    return RTC::RTC_ERROR;
  }

  // 周期的に実行したい処理
  virtual ReturnCode_t onExecute(RTC::UniqueId ec_id)
  {
    if (mylogic.do_someting())
      return RTC::RTC_OK;
    RTC::RTC_ERROR;
  }

private:
  MyLogic mylogic;
  // ポート等の宣言
  //   :
};
```

コンポーネント内の状態遷移



コールバック関数

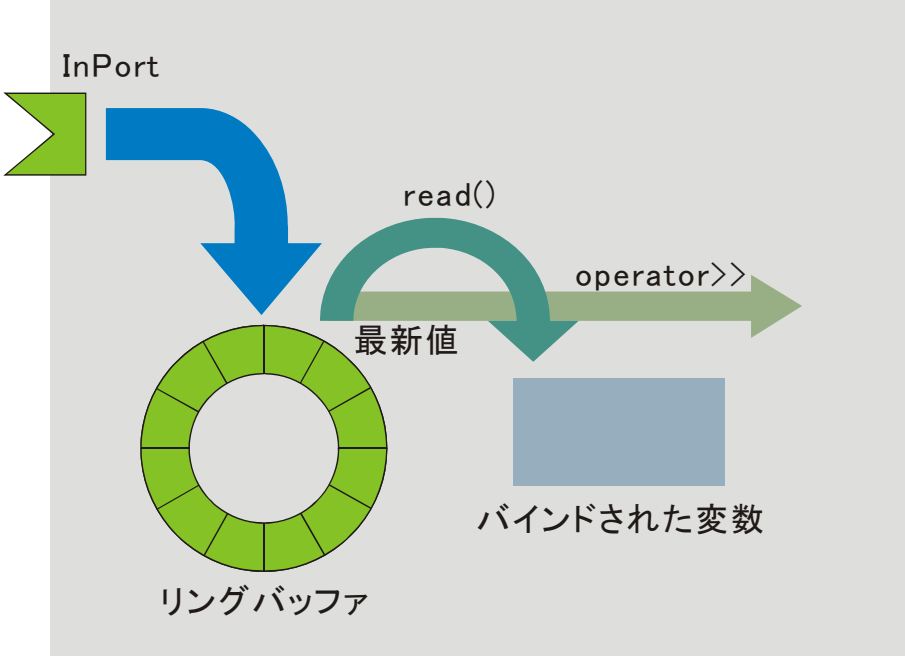
RTCの作成＝コールバック関数に処理を埋め込む

コールバック関数	処理
onInitialize	初期化処理
onActivated	アクティブ化されるとき1度だけ呼ばれる
onExecute	アクティブ状態時に周期的に呼ばれる
onDeactivated	非アクティブ化されるとき1度だけ呼ばれる
onAborting	ERROR状態に入る前に1度だけ呼ばれる
onReset	resetされる時に1度だけ呼ばれる
onError	ERROR状態のときに周期的に呼ばれる
onFinalize	終了時に1度だけ呼ばれる
onStateUpdate	onExecuteの後毎回呼ばれる
onRateChanged	ExecutionContextのrateが変更されたとき1度だけ呼ばれる
onStartup	ExecutionContextが実行を開始するとき1度だけ呼ばれる
onShutdown	ExecutionContextが実行を停止するとき1度だけ呼ばれる

とりあえずは
この5つの関数
を押さえて
おけばOK

InPort

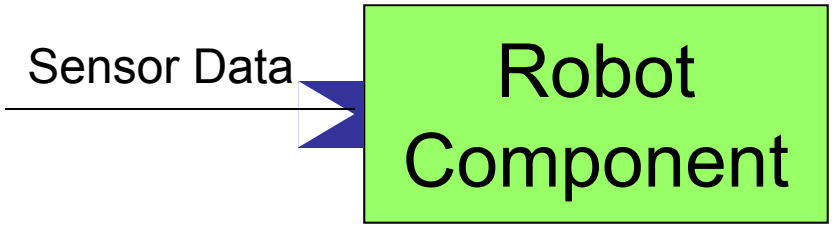
- InPortのテンプレート第2引数：バッファ
  - ユーザ定義のバッファが利用可能
- InPortのメソッド
  - read(): InPort バッファからバインドされた変数へ最新値を読み込む
  - >> : ある変数へ最新値を読み込む



基本的にOutPortと対になる

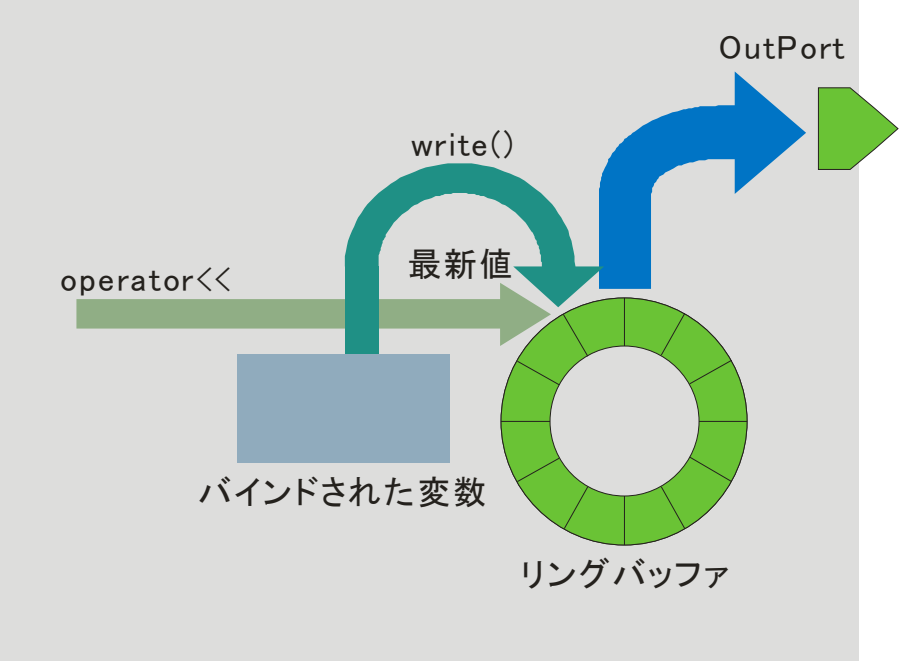
データポートの型を
同じにする必要あり

例



OutPort

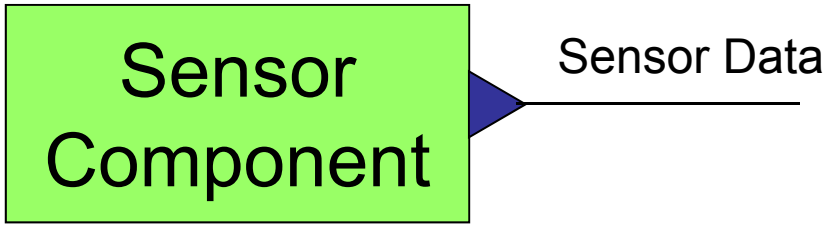
- OutPortのテンプレート第2引数：バッファ
  - ユーザ定義のバッファが利用可能
- OutPortのメソッド
  - write(): OutPort バッファへバインドされた変数の最新値として書き込む
  - >> : ある変数の内容を最新値としてリングバッファに書き込む



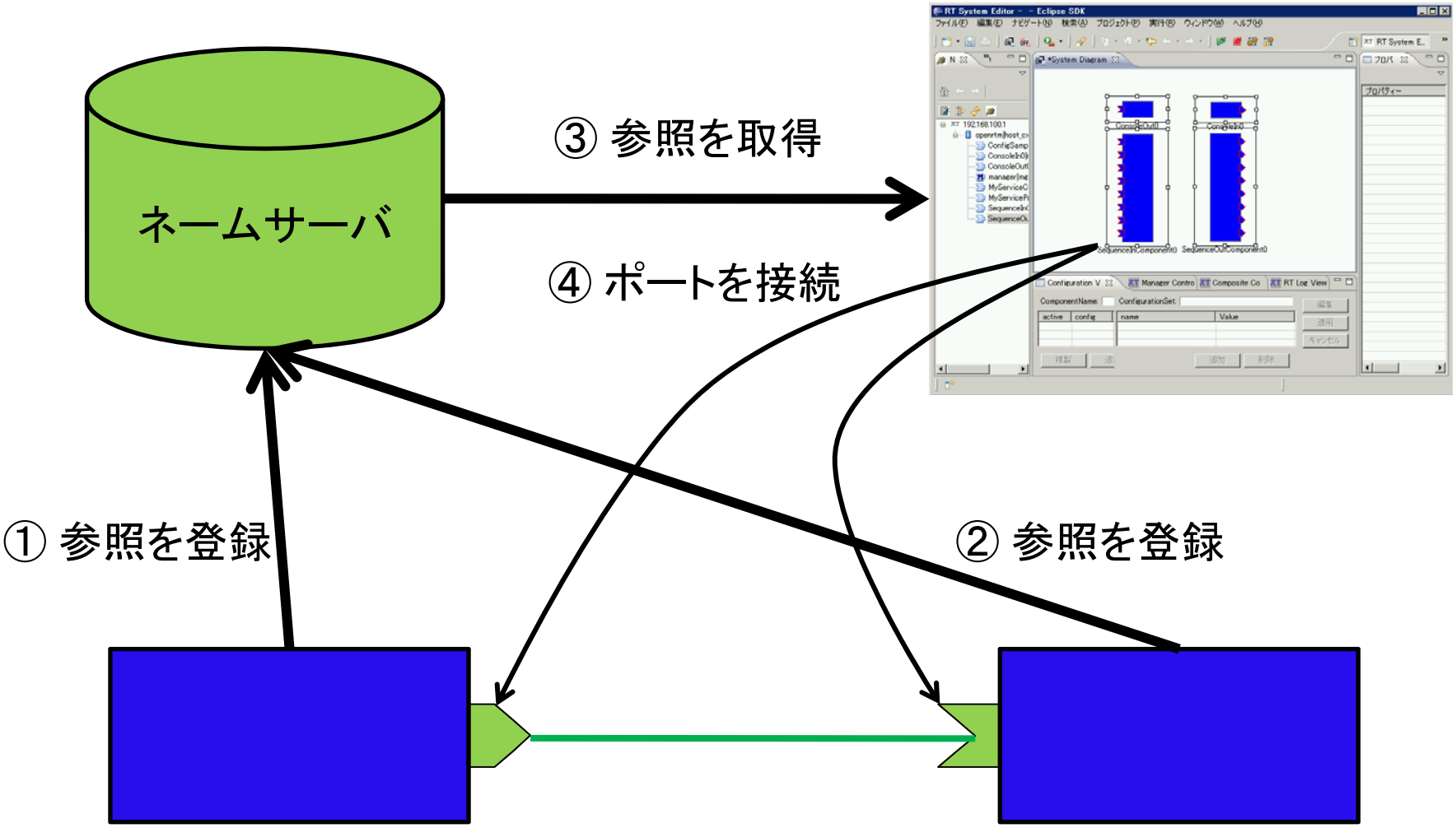
基本的にInPortと対になる

データポートの型を
同じにする必要あり

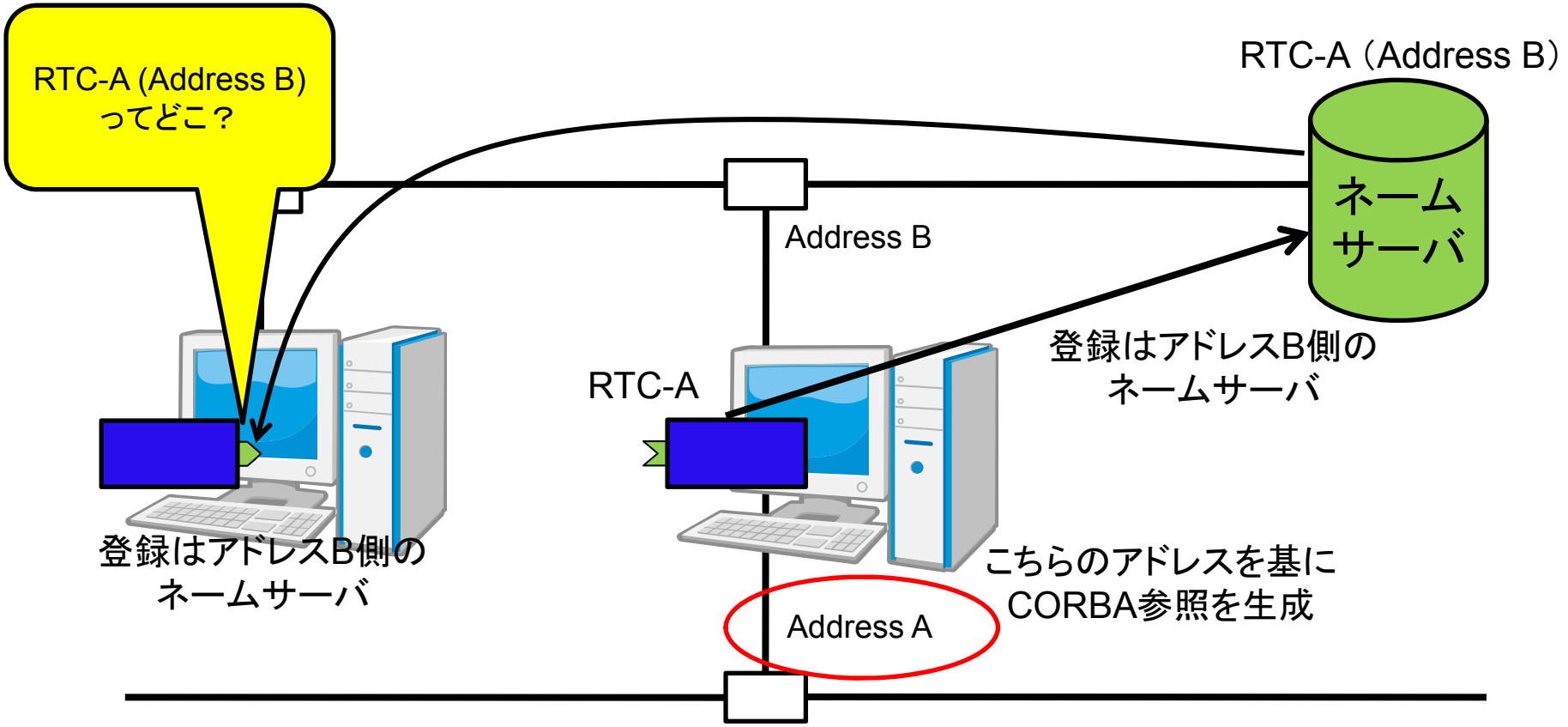
例



動作シーケンス



ネットワークインターフェースが2つある場合の注意

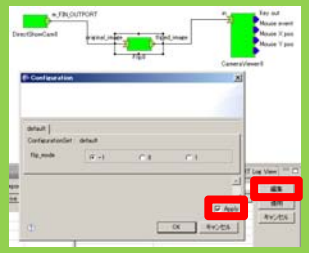
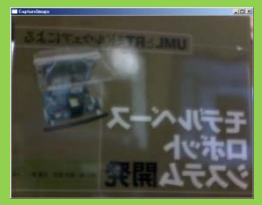
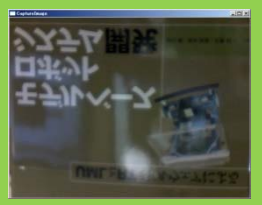
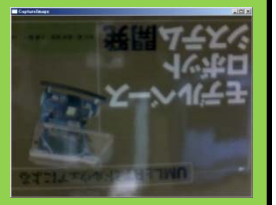


まとめ

- RTミドルウエアの概要
  - 基本概念
- ロボット新戦略とプラットフォーム
  - ロボット未活用領域の開拓
- ROSとの比較、動向
- RTコンポーネントの開発方法
  - 開発の流れ

本日の実習の流れ

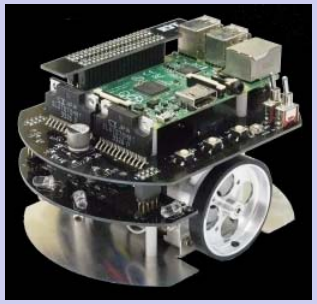
13:00 -14:30
RTコンポーネント作成
（Flipコンポーネント）
講師：安藤

RTM講習会はこの部屋（E204）のままで

RSNP講習会：E202へ移動

14:45 -16:45
プログラミング実習
講師：宮本信彦 氏 (産総研) 他


RaspberryPi マウス


LEGO Mindstorms EV3