Japan Robot Week 2016

2016年10月19日（水）
Japan Robot Week　RTM講習会

第１部：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?

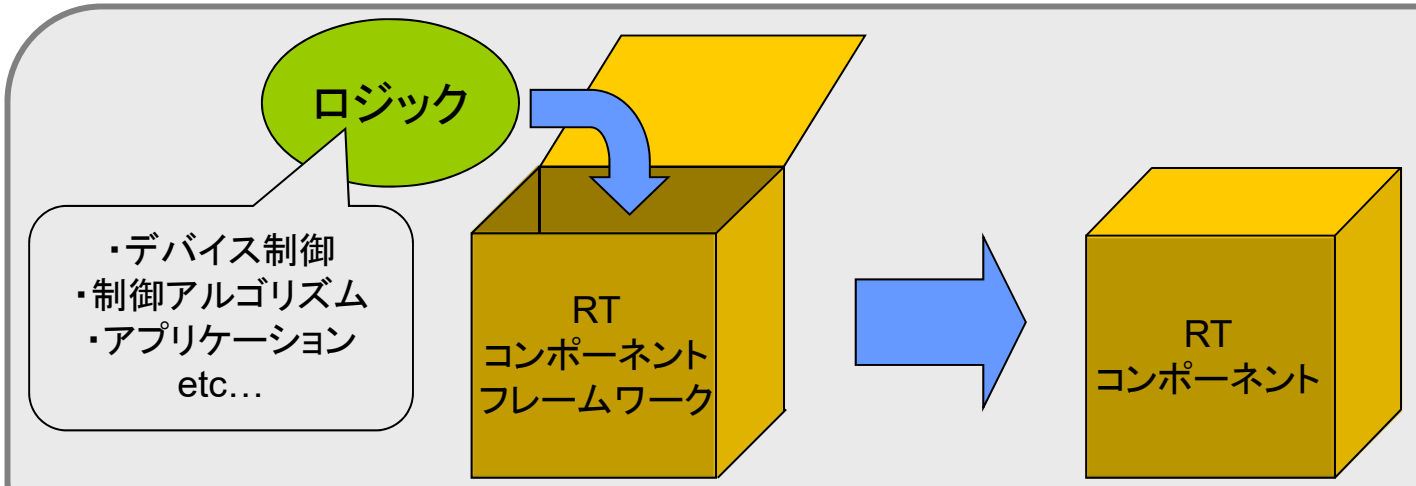
- RT = Robot Technology　cf. IT
  - ≠Real-time
  - 単体のロボットだけでなく、さまざまなロボット技術に基づく機能要素をも含む (センサ、アクチュエータ, 制御スキーム、アルゴリズム、etc.…)

産総研版RTミドルウエア

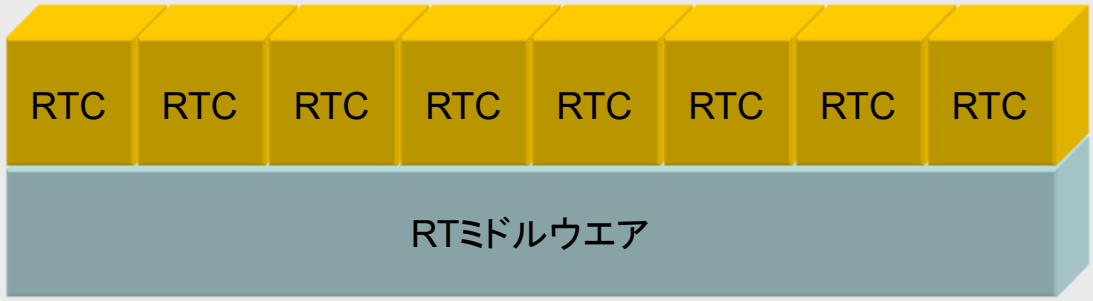
OpenRTM-aist

- 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

RTミドルウエアとRTコンポーネント



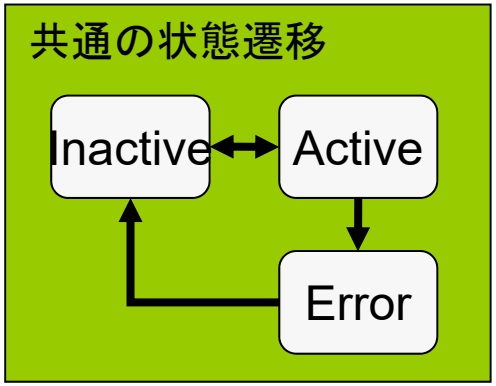
ロジックを箱（フレームワーク）に入れたもの＝RTコンポーネント（RTC）

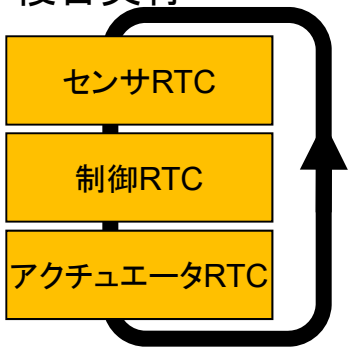


RTCの実行環境（OSのようなもの）＝RTミドルウエア（RTM）
※RTCはネットワーク上に分散可能

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

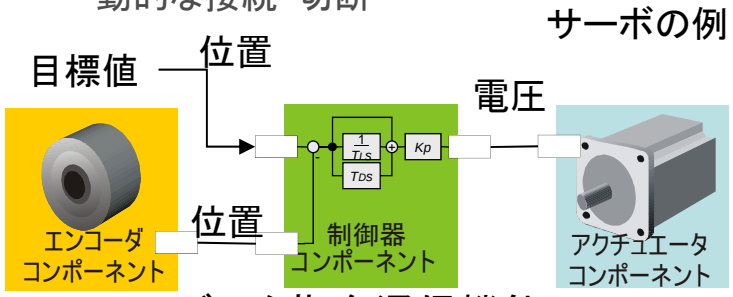




ライフサイクルの管理・コアロジックの実行

データポート

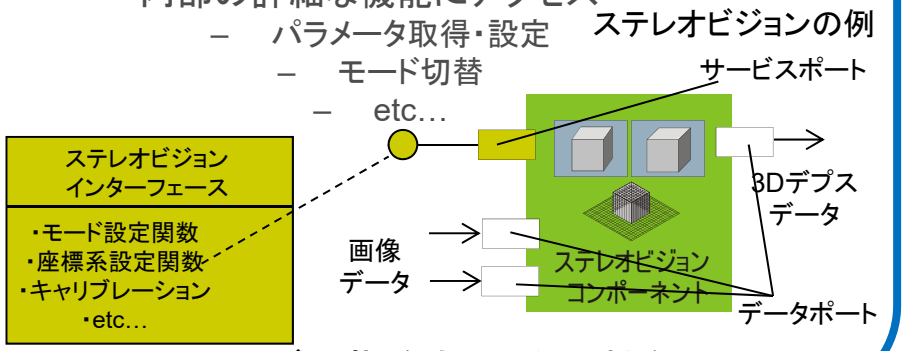
- データ指向ポート
- 連続的なデータの送受信
- 動的な接続・切断



データ指向通信機能

サービスポート

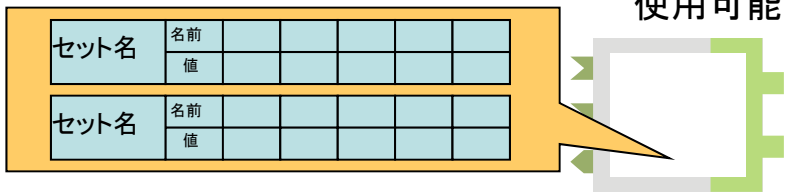
- 定義可能なインターフェースを持つ
- 内部の詳細な機能にアクセス
  - パラメータ取得・設定
  - モード切替
  - etc…



サービス指向相互作用機能

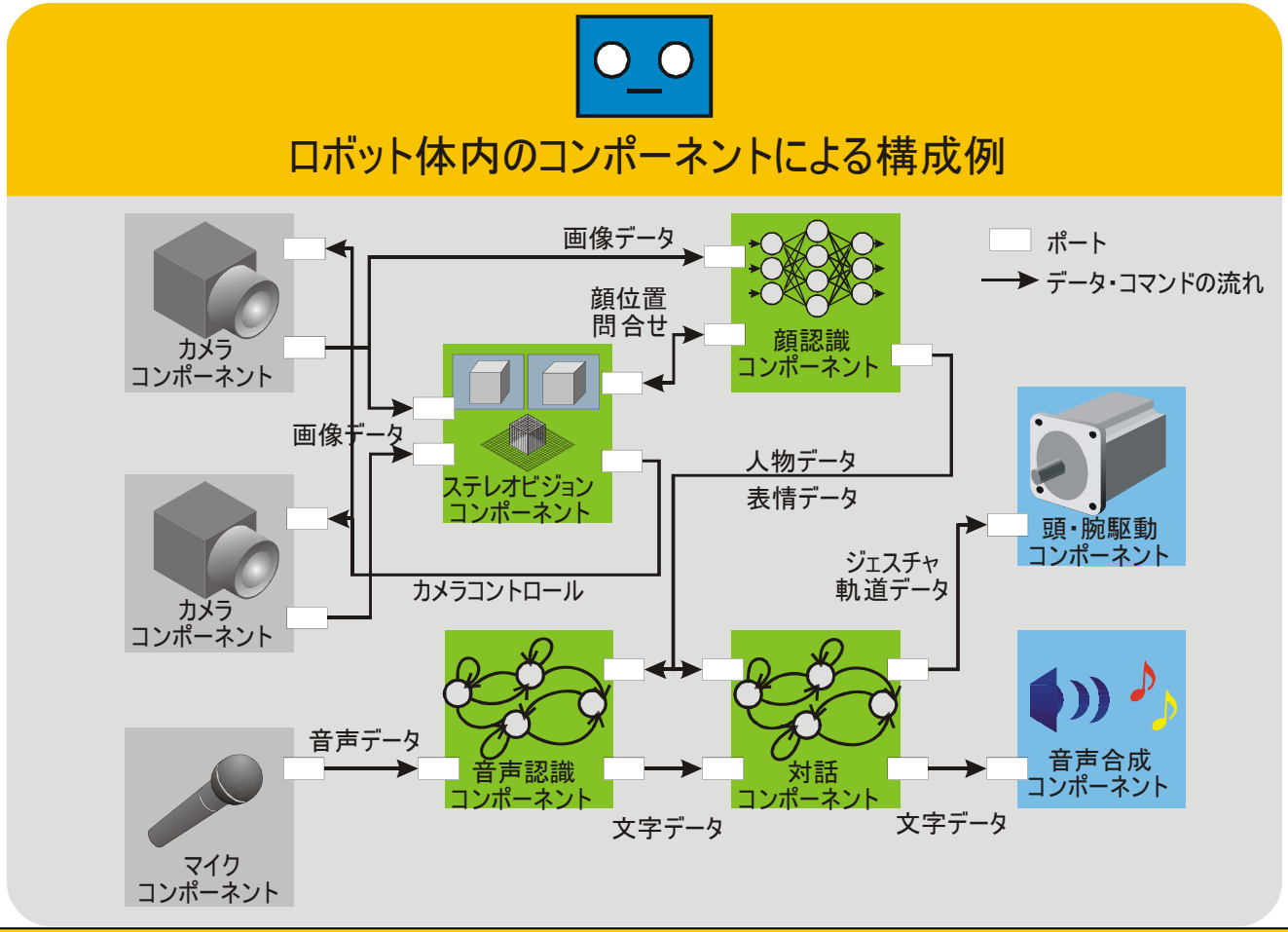
コンフィギュレーション

- 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

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

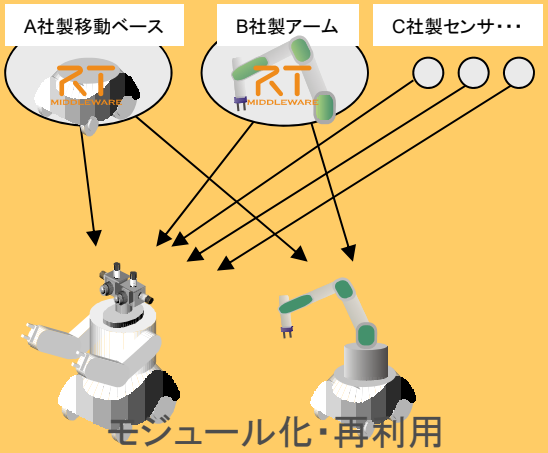
モジュール化のメリットに加えて

- 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

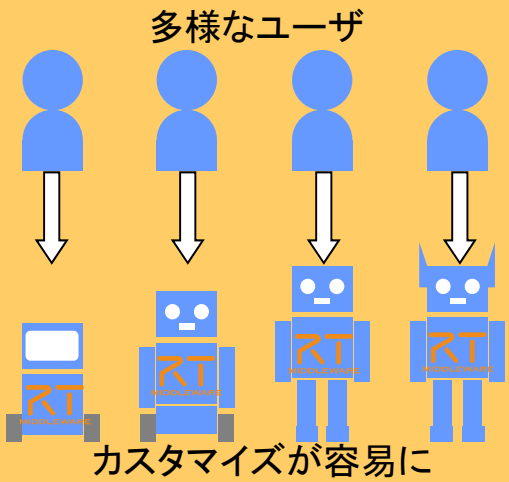


ロボットの低コスト化

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

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。

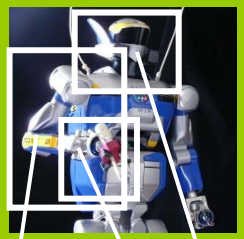
ロボットA

RTC | RTC
RTM
Solaris

ロボットB

RTC | RTC
RTM
FreeBSD

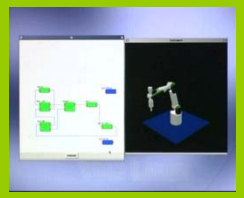
ロボットC



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

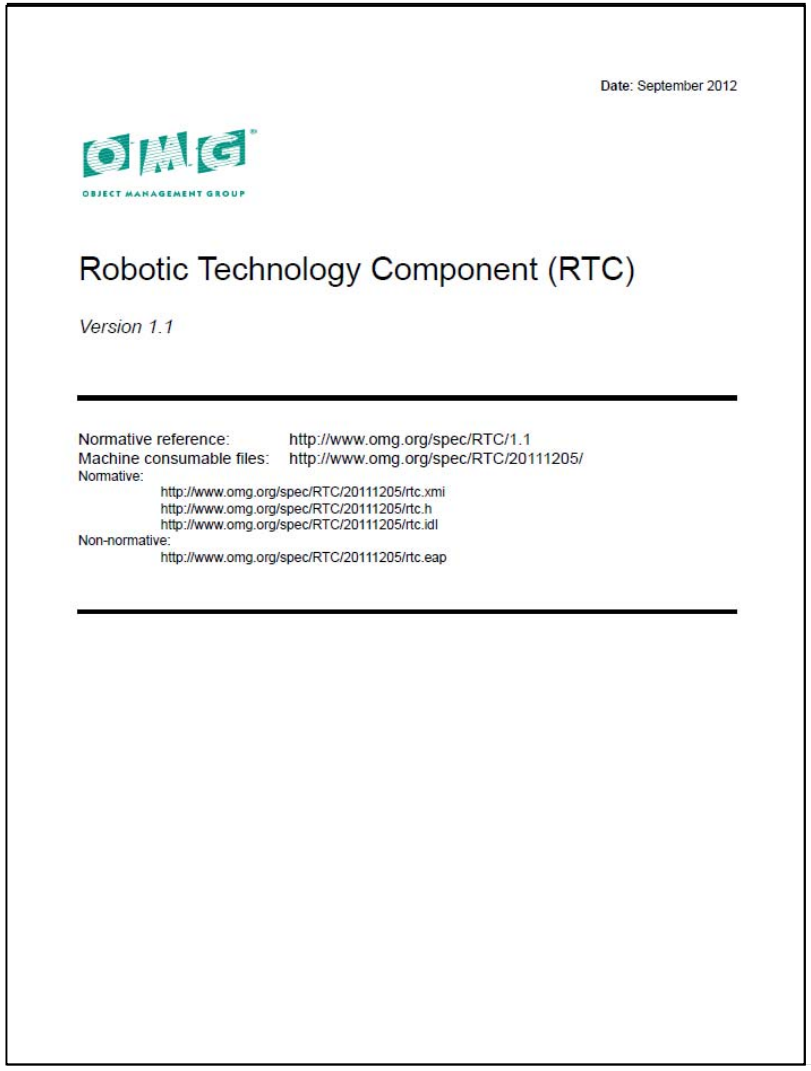
TRON
RTM
RTC | RTC

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

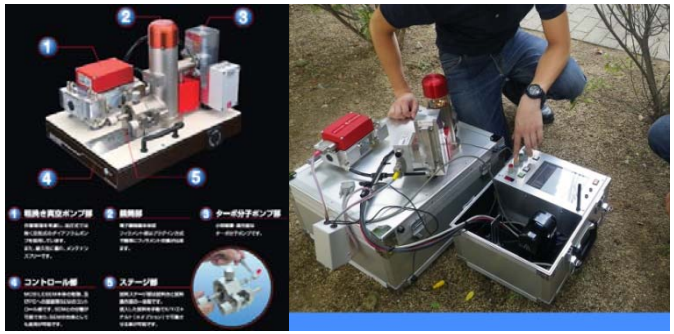
HRP-4C: AIST

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

RTMとROSの比較

ROS

- UNIXユーザに受けるツール（特にCUI）が豊富
  - →基本はLinuxのみサポート
- 独自のパッケージ管理システムを持つ
- 実装方法が比較的自由
  - コア部分の使用が固まっていない
  - コンポーネントモデルがない
- 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

RTM

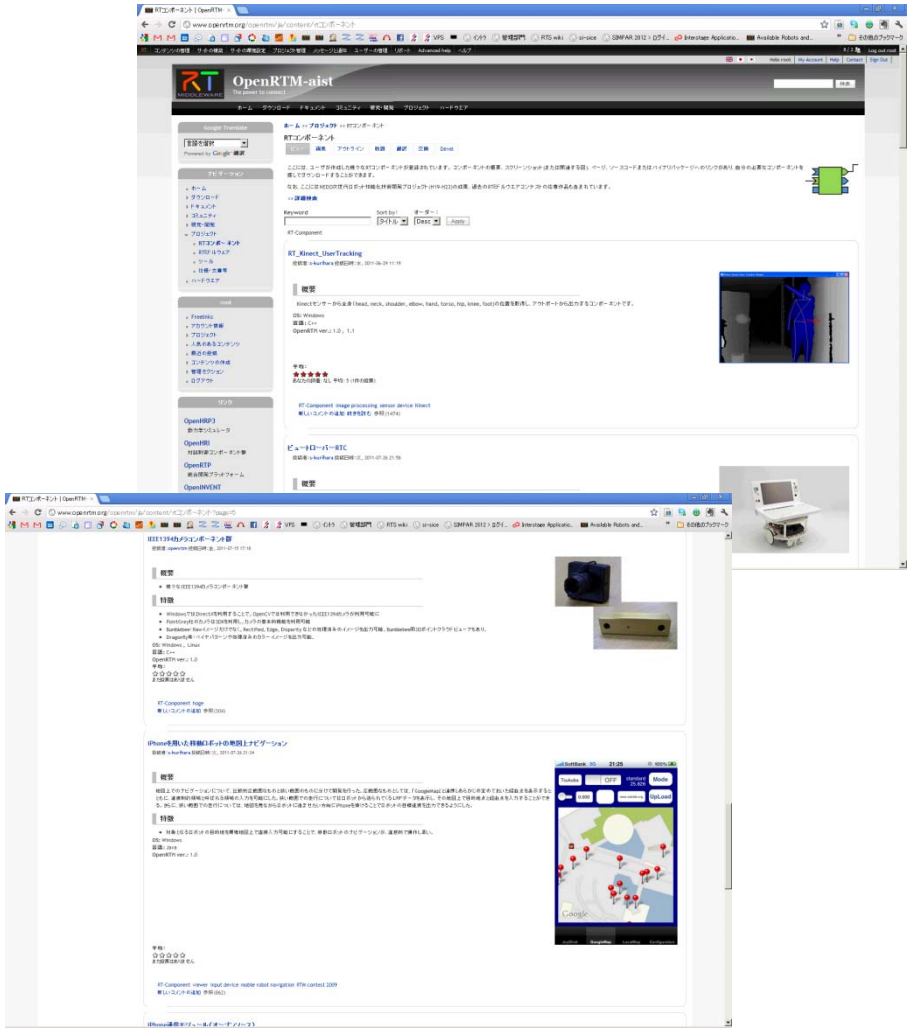
- 対応OS・言語の種類が多い
  - Windows、UNIX、uITRON、T-Kernel、VxWorks、QNX
  - Windowsでネイティブ動作する
- 日本製
  - 国外のユーザが少ない
  - 英語ドキュメントが少ない
- GUIツールがあるので初心者向き
- 仕様が標準化されている
  - OMGに参加すればだれでも変更可
  - サードパーティー実装が作りやすい
  - すでに10程度の実装あり
- コンポーネントモデルが明確
  - オブジェクト指向、UML・SysMLとの相性が良い
  - モデルベース開発
- IEC61508機能安全認証取得
  - RTMSafety

第三者による比較：http://ysuga.net/?p=146

RTミドルウェアと
オープンソースコミュニティー

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

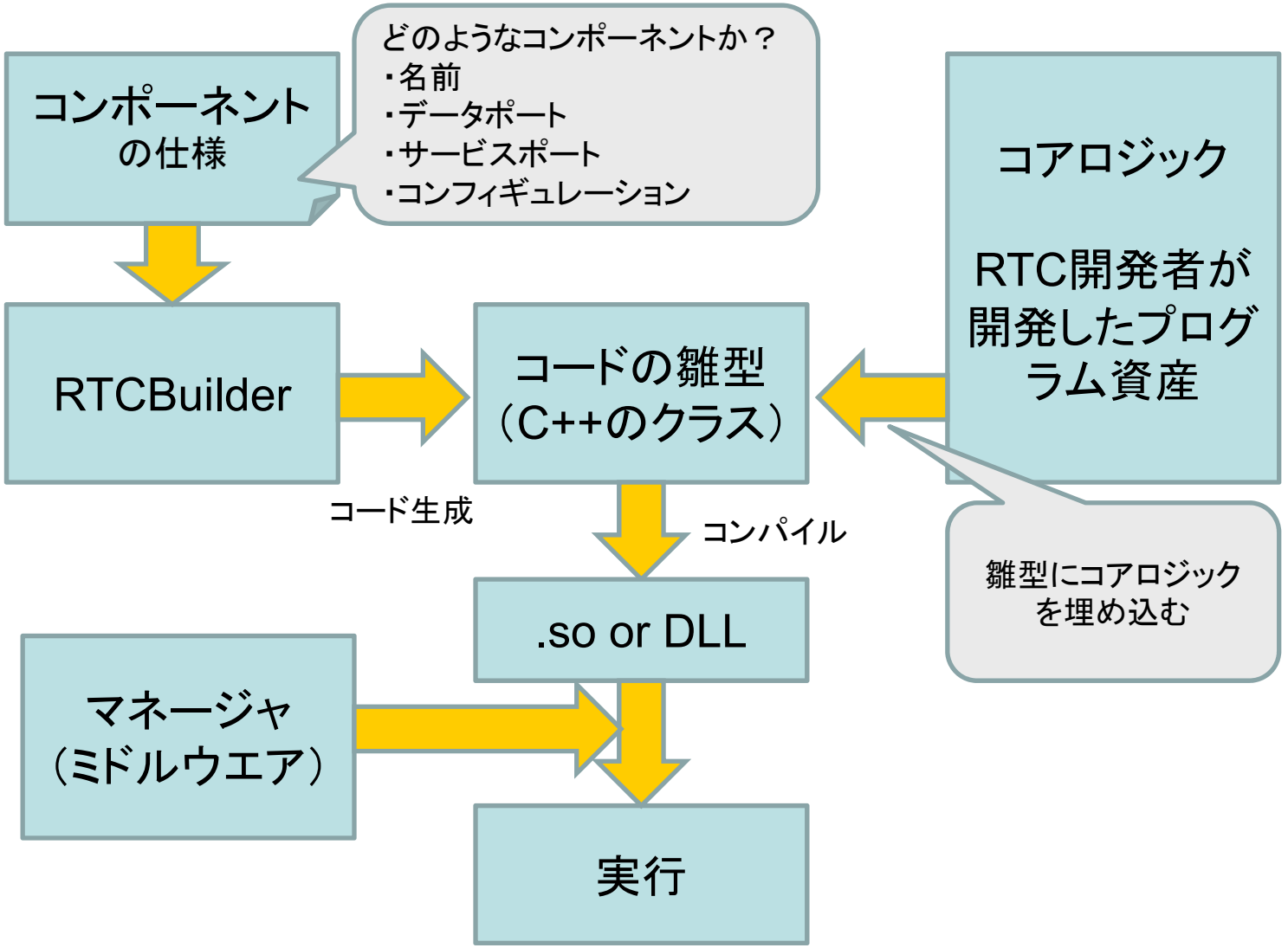
- SICE SI（計測自動制御学会 システムインテグレーション部門講演会）のセッションとして開催
  - 各種奨励賞・審査基準開示:5月頃
  - エントリー〆切:8月下旬 (SI2016締切)
  - ソフトウエア登録:10月ごろ
  - 講演原稿〆切:9月下旬
  - オンライン審査:11月下旬～
  - 発表・授賞式:12月15日
- 2015年度実績
  - 応募数:16件
  - 計測自動制御学会学会RTミドルウエア賞（副賞10万円）
  - 奨励賞:18件
- 詳細はWebページ: openrtm.org
  - コミュニティー→イベント をご覧ください

提言

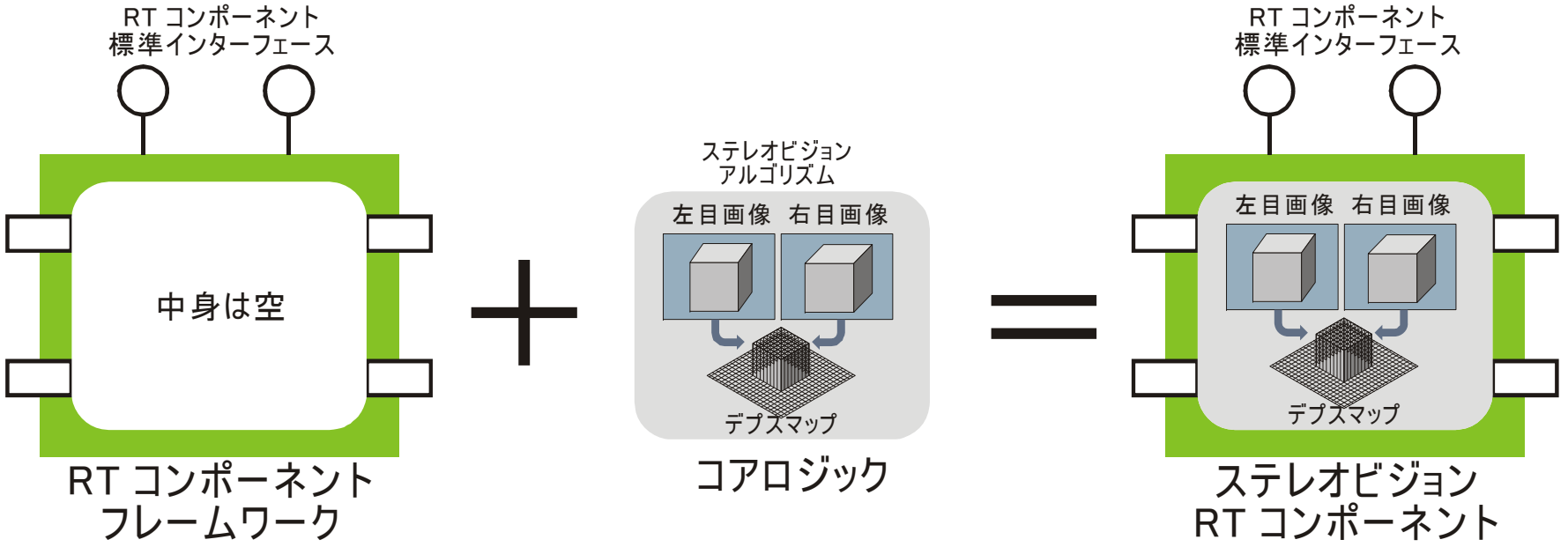
- 自前主義はやめよう！！
  - 書きたてのコードより、いろいろな人に何万回も実行されたコードのほうが動くコードである！！
  - 自分にとって本質的でない部分は任せて、本当にやりたい部分・やるべき部分のコードを書こう！！
  - 誰かがリリースしたプログラムは一度は動いたことがあるプログラムである！！
  - 人のコードを読むのが面倒だからと捨ててしまうのはもったいない！！
- オープンソースにコミットしよう！！
  - 臆せずMLやフォーラムで質問しよう！！
  - どんなに初歩的な質問でも他の人にとっては価値ある情報である。
  - 要望を積極的にあげよう！！
  - できればデバッグ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

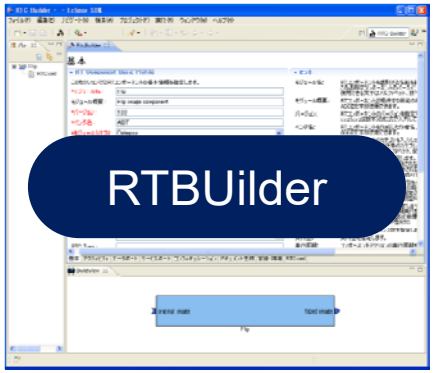


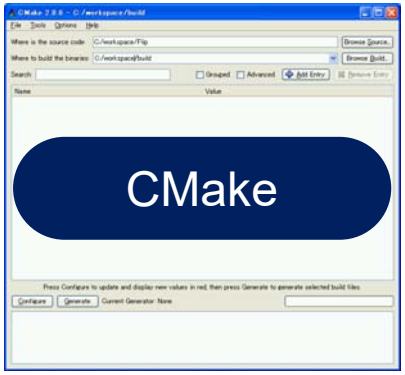
フレームワークとコアロジック

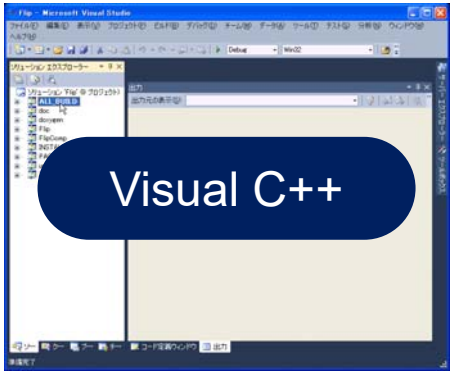


RTCフレームワーク＋コアロジック＝RTコンポーネント

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

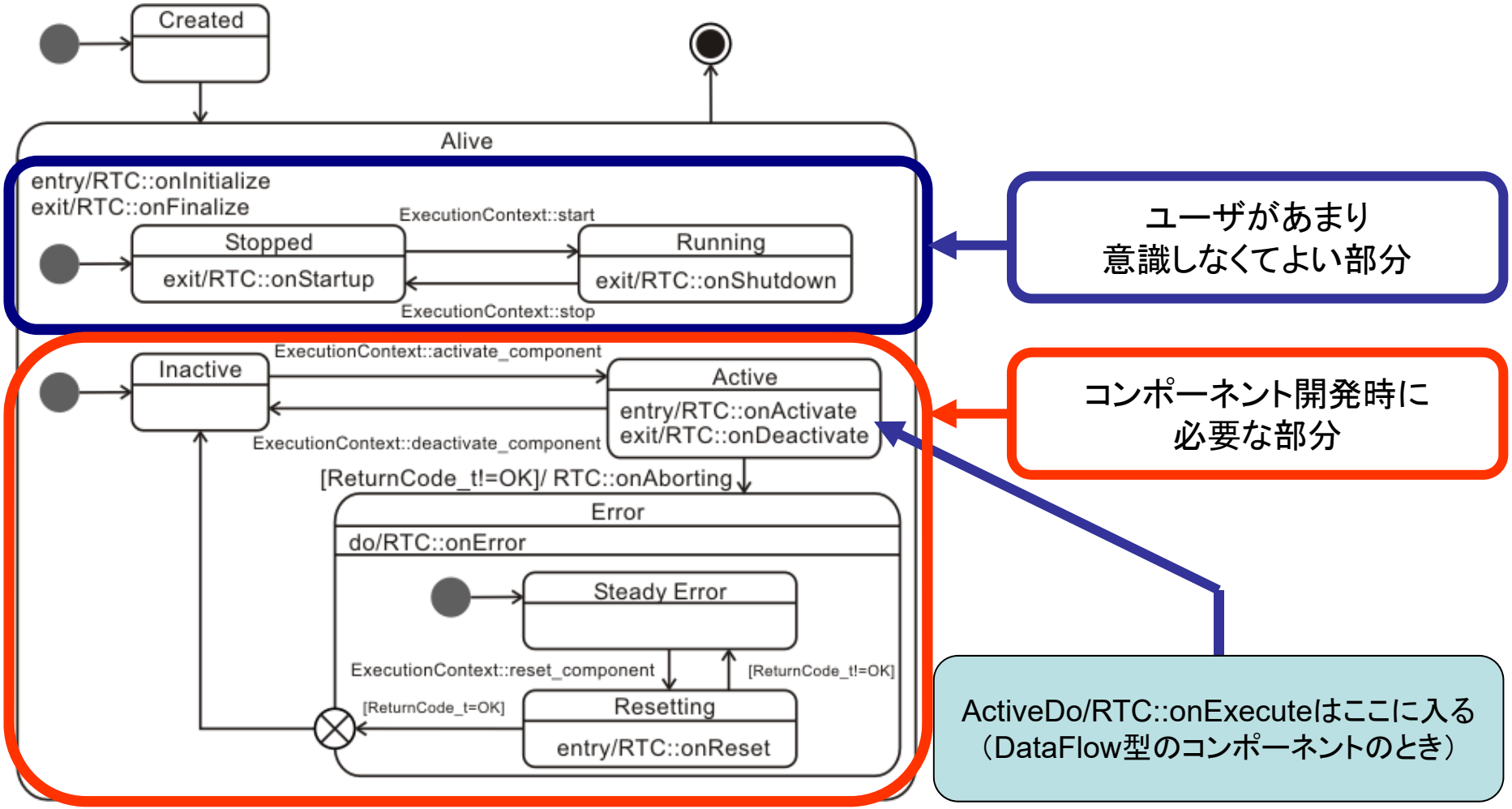
- 生成されたクラスのメンバー関数に必要な処理を記述
- 主要な関数
  - onExecute (周期実行)
- 処理
  - InPortから読む
  - OutPortへ書く
  - サービスを呼ぶ
  - コンフィギュレーションを読む

```
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

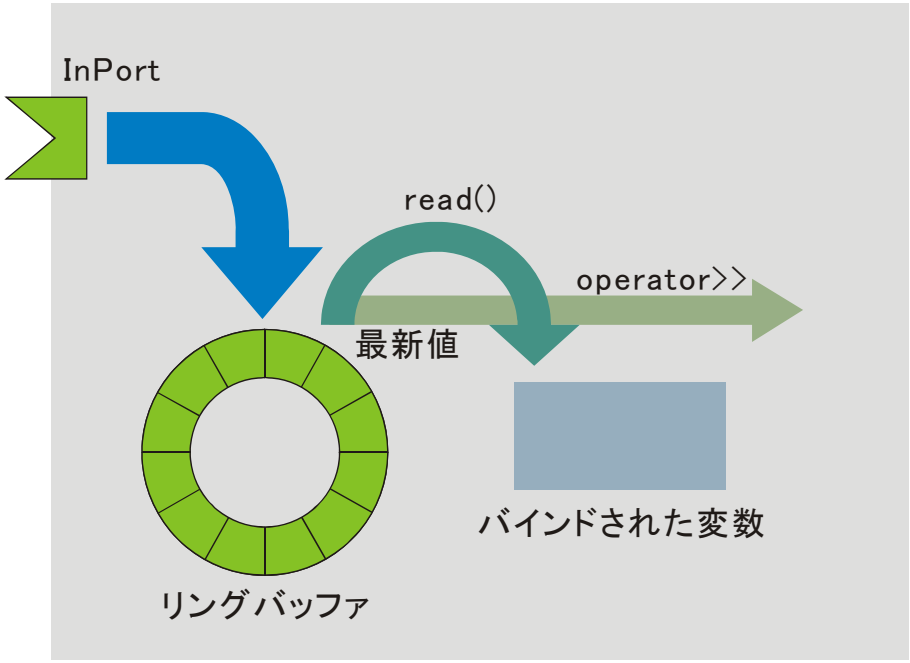
RTCの作成＝コールバック関数に処理を埋め込む

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

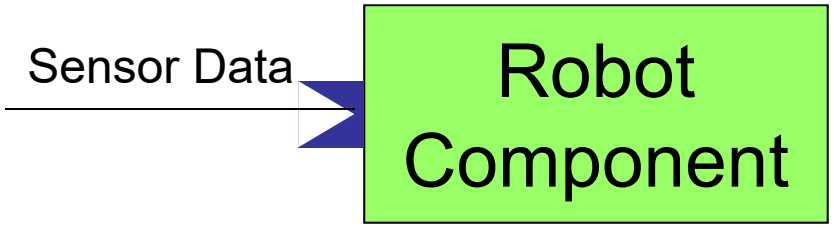
- InPortのテンプレート第2引数：バッファ
  - ユーザ定義のバッファが利用可能
- InPortのメソッド
  - read(): InPort バッファからバインドされた変数へ最新値を読み込む
  - >> : ある変数へ最新値を読み込む



基本的にOutPortと対になる

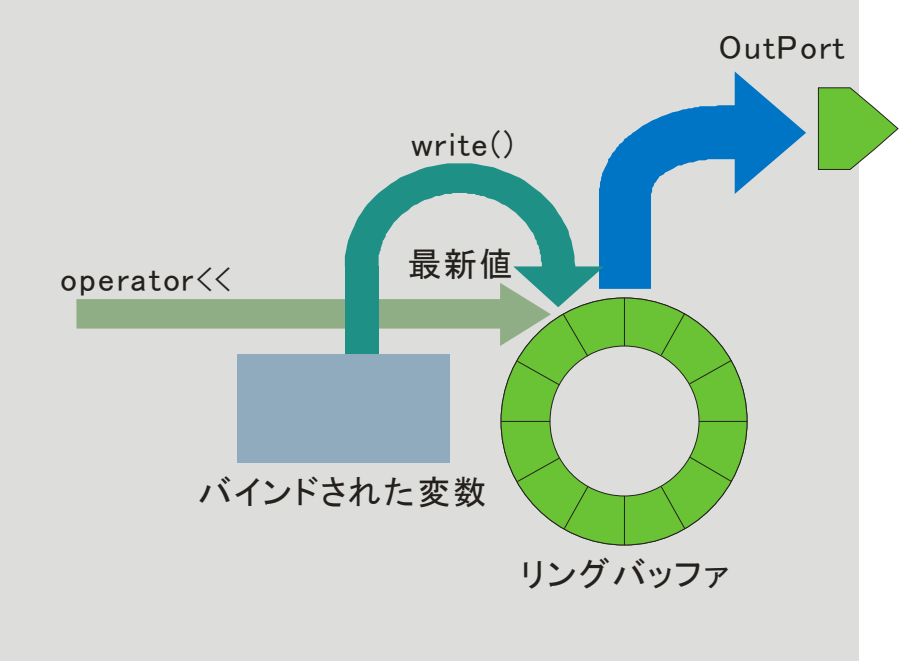
データポートの型を
同じにする必要あり

例



OutPort

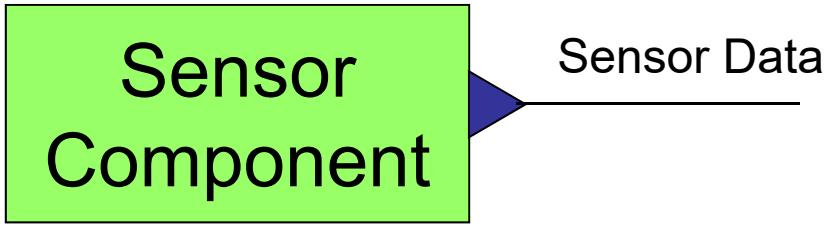
- OutPortのテンプレート第2引数：バッファ
  - ユーザ定義のバッファが利用可能
- OutPortのメソッド
  - write(): OutPort バッファへバインドされた変数の最新値として書き込む
  - >> : ある変数の内容を最新値としてリングバッファに書き込む



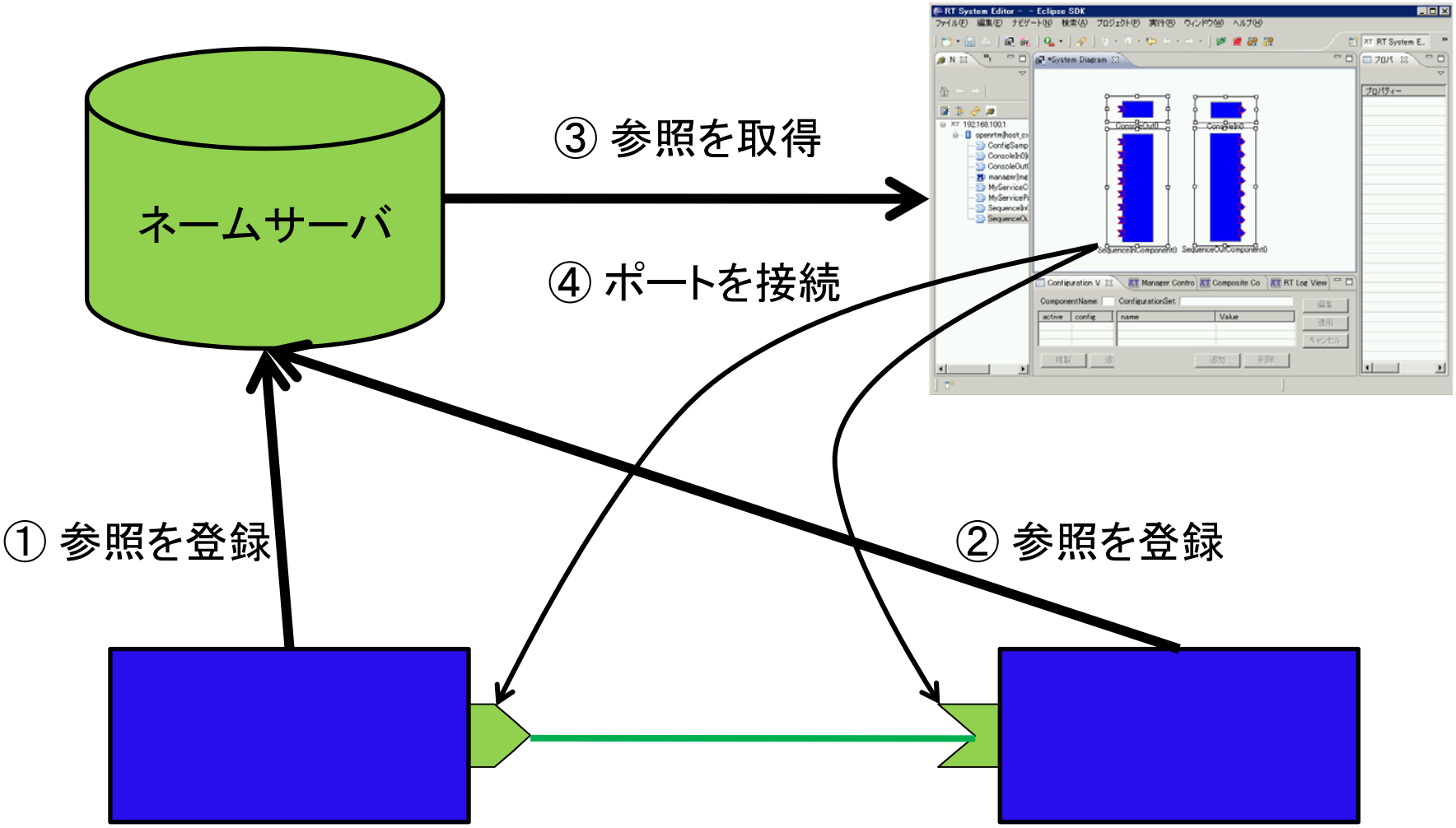
基本的にInPortと対になる

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



まとめ

- RTミドルウエアの概要
  - 基本概念・メリット
  - 標準化
  - ROSとの比較、動向
- RTミドルウェアコミュニティー
  - サマーキャンプ
  - RTMコンテスト
- RTコンポーネントの開発方法
  - 開発の流れ