熊本県産業技術センター
第7回技術普及講習会(組み込み技術)
「RTミドルウェアの概要と実習」

日時：2011年11月25日(金)　11:00～17:00
場所：熊本県産業技術センター 大会議室

RTミドルウェア講習会

11:00-11:30	第1部：RTミドルウェアの概略紹介
	担当：安藤 慶昭（産業技術総合研究所）
	概要: RTミドルウェア，RTコンポーネントの概要を説明します．また，Web上で自分の作品を公開できる仕組みについて紹介します．
11:30-12:00	第2部：RTミドルウェアの概略，導入方法の紹介
	担当：栗原眞二（産業技術総合研究所）
	概要: サンプルシステムを用いた概略紹介．RTミドルウェアの導入方法について紹介します．
13:00-13:45	第3部：RTミドルウェアを用いたシステム構築方法の紹介
	担当：坂本武志（株式会社グローバルアシスト）
	概要:RTコンポーネントを組み合わせてシステムを構築する方法について説明します．
14:00-14:45	第4部：RTコンポーネントの作り方
	担当：坂本武志（株式会社グローバルアシスト）
	概要:RTコンポーネントのテンプレート作成ツールRTCBuilderを用いたコンポーネントの設計と実装について説明します。
15:00-17:00	第5部：RTコンポーネント作成実習
	担当：栗原眞二（ 産業技術総合研究所）
	概要:参加者全員で実際にコンポーネントを作成してシステムを構築してみます．

第1部：RTミドルウェアの概略紹介

(独)産業技術総合研究所
知能システム研究部門
安藤慶昭

概要

- RTミドルウエアとは？
  - 目的、コンセプト
  - OpenRTM-aistについて
  - 基本機能
- RTコンポーネントのつくり方
  - プログラミングの流れ
  - 各機能の詳細

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.…)

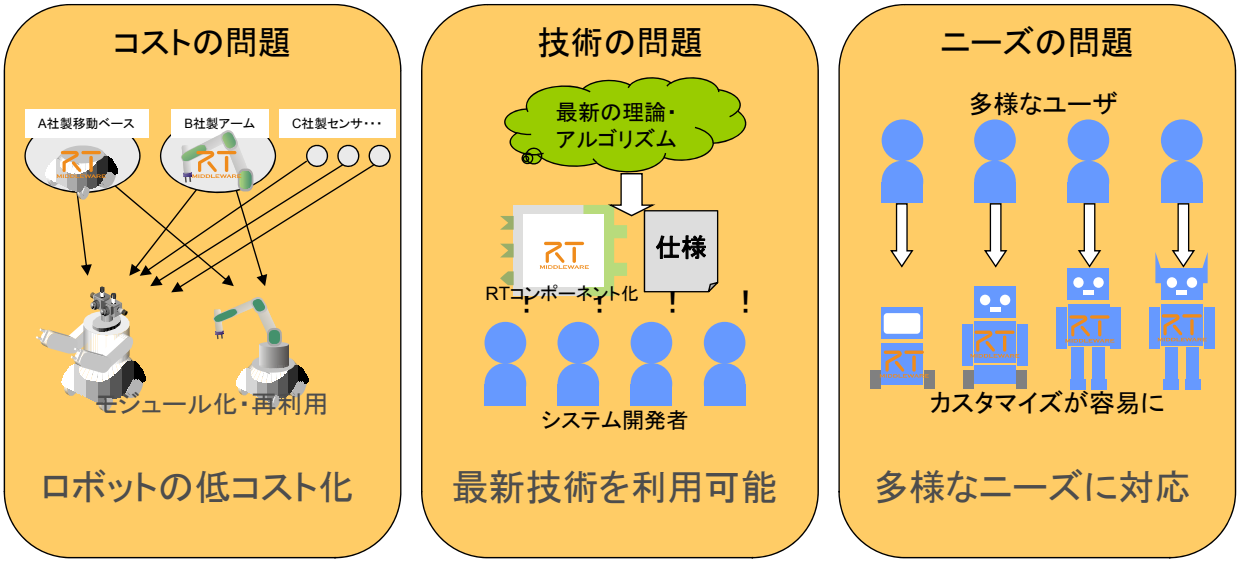
産総研版RTミドルウエア

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

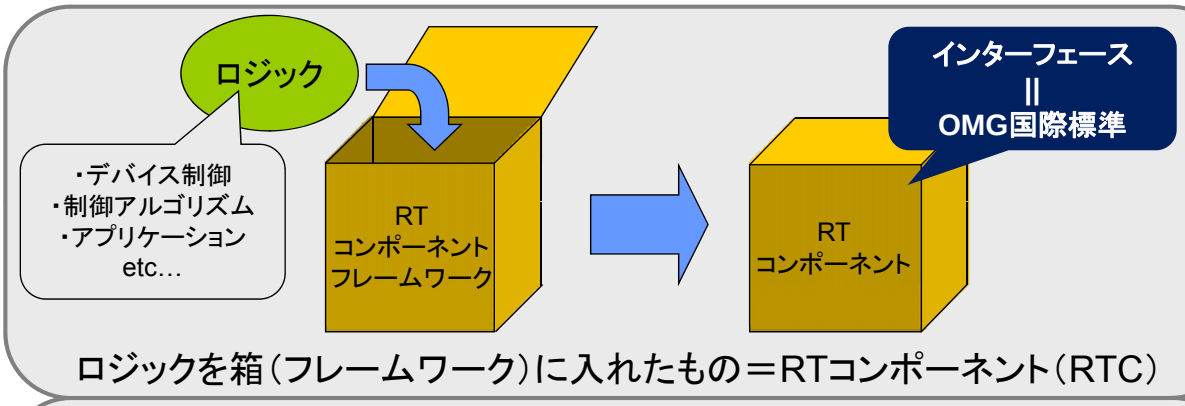
RTミドルウエアの目的

モジュール化による問題解決



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

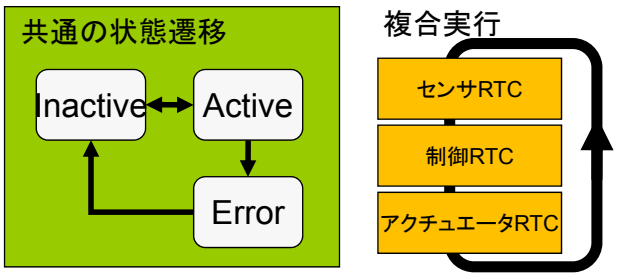


ロジックを箱（フレームワーク）に入れたもの＝RTコンポーネント（RTC）

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

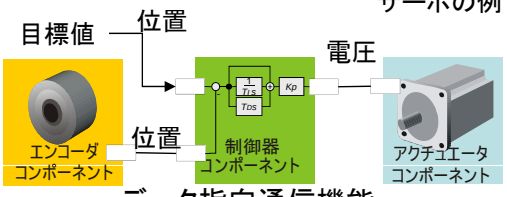
アクティビティ・実行コンテキスト



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

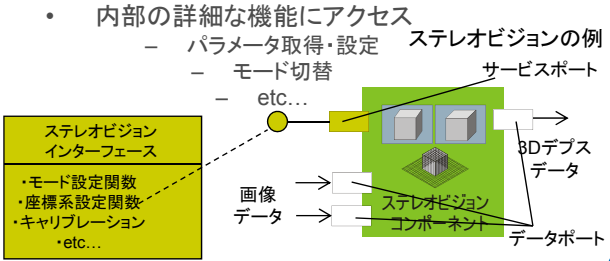
- データ指向ポート
- 連続的なデータの送受信
- 動的な接続・切断



データ指向通信機能

サービスポート

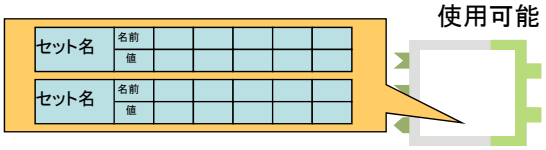
- 定義可能なインターフェースを持つ
- 内部の詳細な機能にアクセス
  - パラメータ取得・設定
  - モード切替
  - etc…



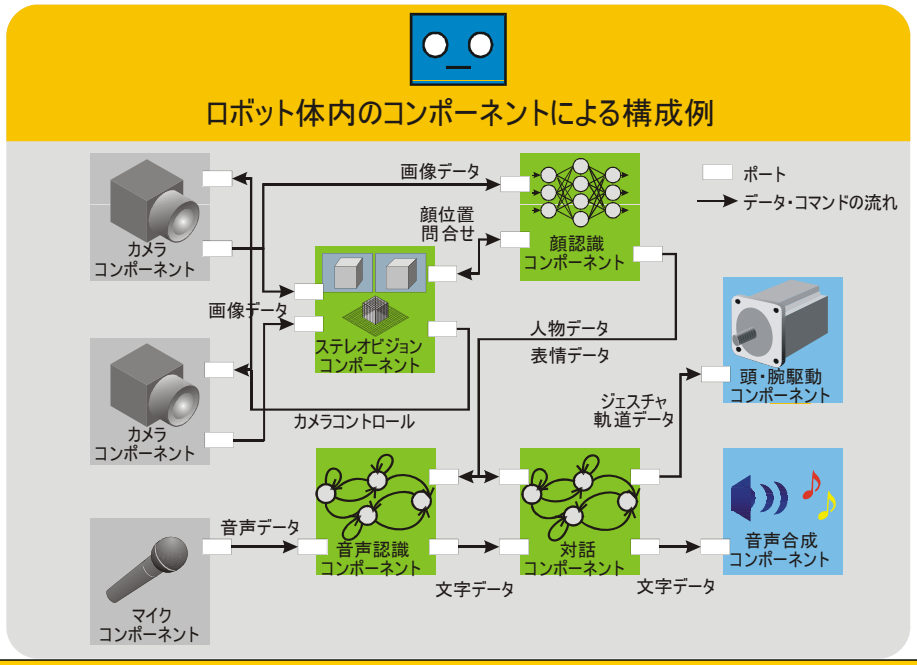
サービス指向相互作用機能

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

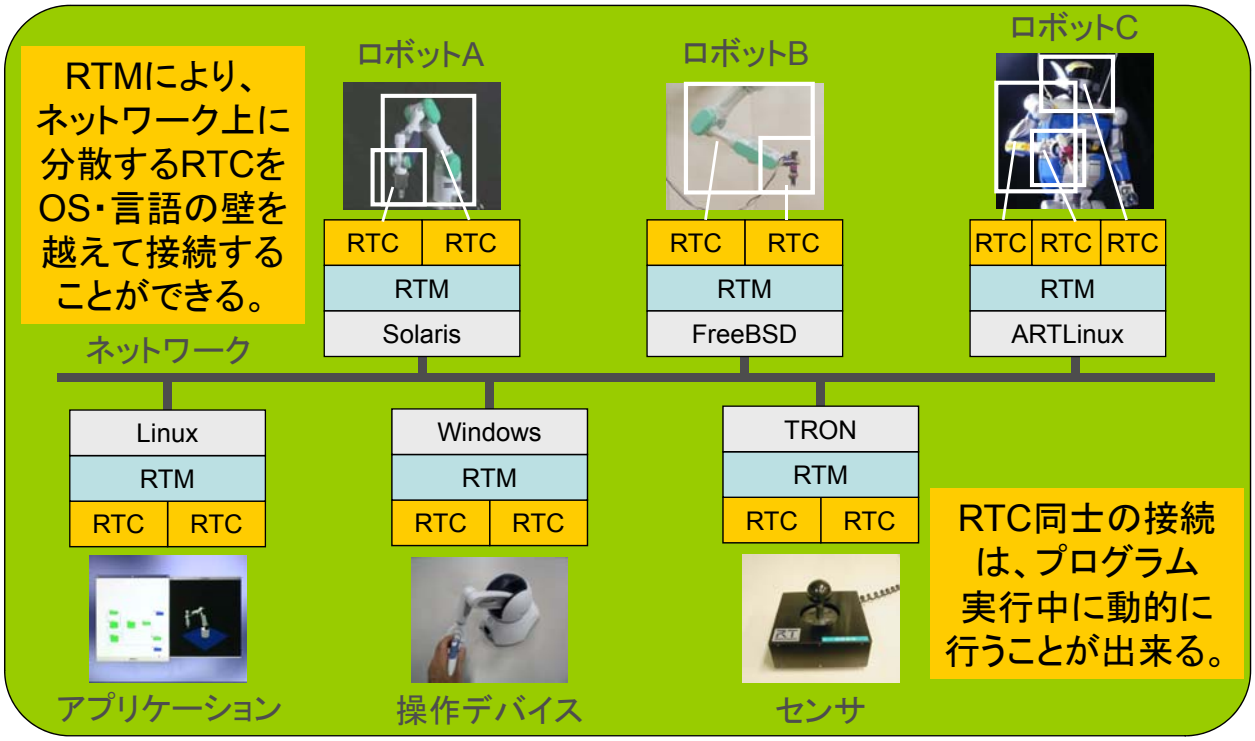


RTCの分割と連携



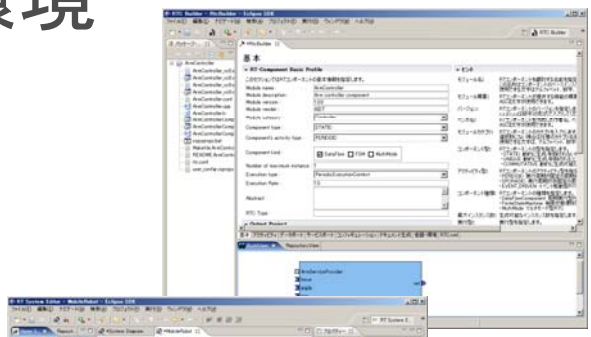
(モジュール)情報の隠蔽と公開のルールが重要

RTミドルウエアによる分散システム

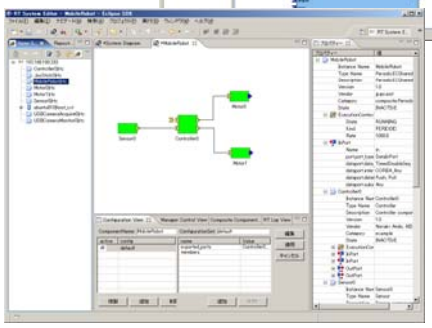


開発環境

- RTCBuilder (GUI版)
  - Eclipseプラグイン
  - RTコンポーネントのコードジェネレータ
  - GUI画面で必要事項を入力
  - C++, Python, Java, C#等のコードを自動生成
- RTSystemEditor
  - Eclipseプラグイン
  - ネットワーク上のすべてのコンポーネントの操作が可能
  - コンポーネントのON/OFF、パラメータの変更、状態監視
  - コンポーネント間の接続



RTCBuilder



RTSystemEditor

RTC・RTM統合開発環境の整備

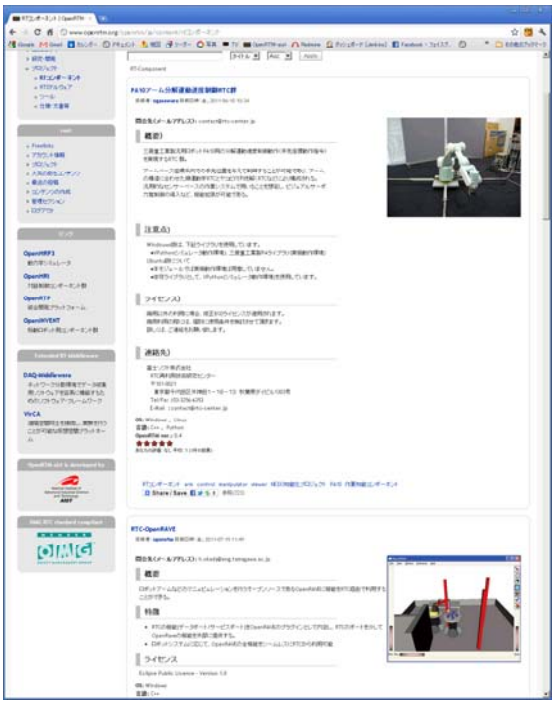
RTC設計・実装・デバッグ、RTMによるインテグレーション・デバッグまでを一貫して行うことができる統合開発環境をEclipse上に構築

OpenRTM-aist

- コンポーネントフレームワーク + ミドルウエアライブラリ
- コンポーネントインターフェース:
  - OMG Robotic Technology Component Specification ver1.0 準拠
- OS
  - 公式:FreeBSD, Linux, Windows, Mac OS X
  - 非公式:uITRON, T-Kernel, VxWorks
- 言語:
  - C++ (1.1.0-RC3), Python (1.1.0-RC1), Java (1.1.0-RC1)
  - .NET (implemented by SEC)
- CPU アーキテクチャ (動作実績):
  - i386, ARM9, PPC, SH4
  - PIC, dsPIC, H8 (RTC-Lite)
- ツール (Eclipse プラグイン)
  - テンプレートソースジェネレータ: rtc-template、RTCBuilder
  - システムインテグレーションツール: RTSystemEditor
  - その他
    - Pattern weaver for RT-Middleware (株式会社テクノロジックアートより発売中)

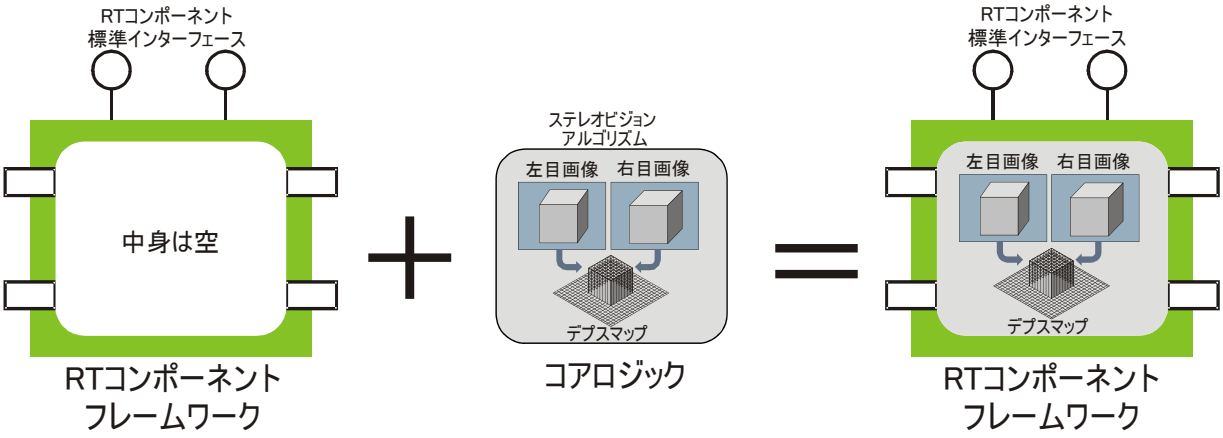
Webサイト (www.openrtm.org)

- ダウンロード
  - OpenRTM-aist
  - ツール等
- ドキュメント
- コミュニティ
  - ML、フォーラム、講習会
- 研究・開発
- プロジェクト
  - 他のユーザがつくったRTCをダウンロードして使用可能
  - 誰でも自分のRTCを登録して公開可能



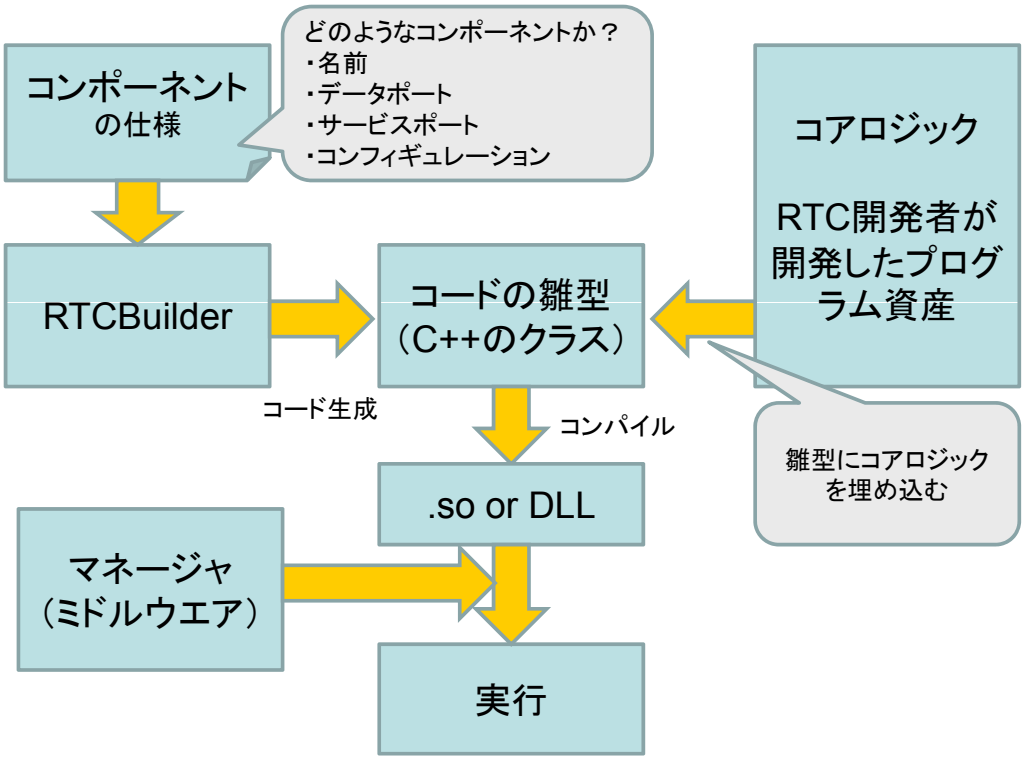
RTコンポーネントプログラミング

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

OpenRTMを使った開発の流れ



コード例

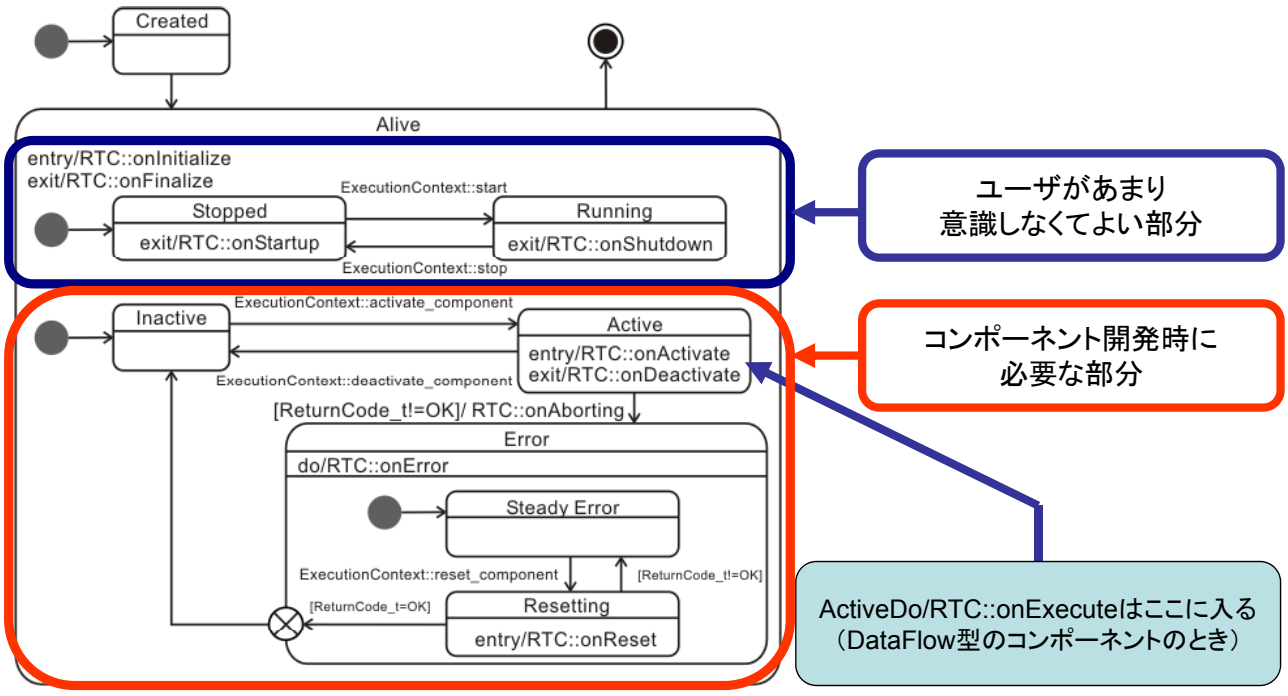
- 生成されたクラスのメンバー関数に必要な処理を記述
- 主要な関数
  - onExecute (周期実行)
- 処理
  - InPortから読む
  - OutPortへ書く
  - サービスを呼ぶ
  - コンフィギュレーションを読む

```
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

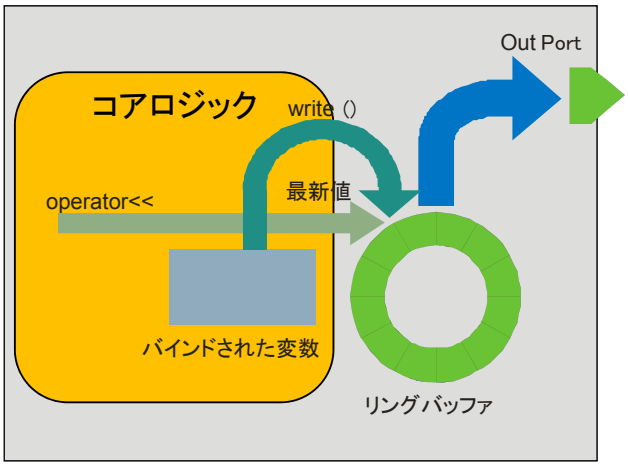
RTCの作成＝コールバック関数に処理を埋め込む

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

データポートの仕組み(OutPort)

- データポート変数
  - OutPort変数とも呼ぶ
  - ポートにバインドされた変数
- OutPortの関数
  - 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バッファに書き込む



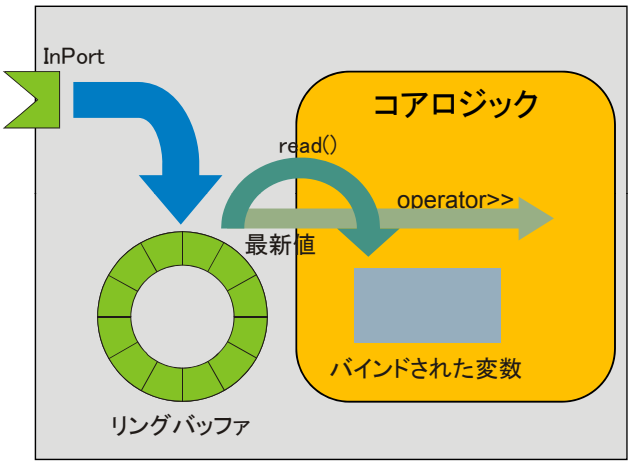
例

対になるInPortと型を同じにする必要あり

Sensor Component → Sensor Data

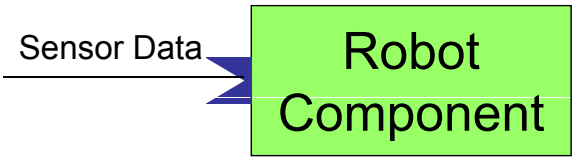
データポートの仕組み(InPort)

- データポート変数
  - InPort変数とも呼ぶ
  - ポートにバインドされた変数
- InPortの関数
  - isNew(): 新しいデータが来ているか調べる
  - 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



対になるOutPortと型を同じにする必要あり

例



データポート変数について

基本型の例

```
struct TimedShort
 {
      Time tm;
      short data;
 };
```

- 基本型
  - tm : 時刻
  - data: データそのもの

OpenRTM-aistであらかじめ定義されているデータポート変数以外にも独自の型を定義使用することも可能。

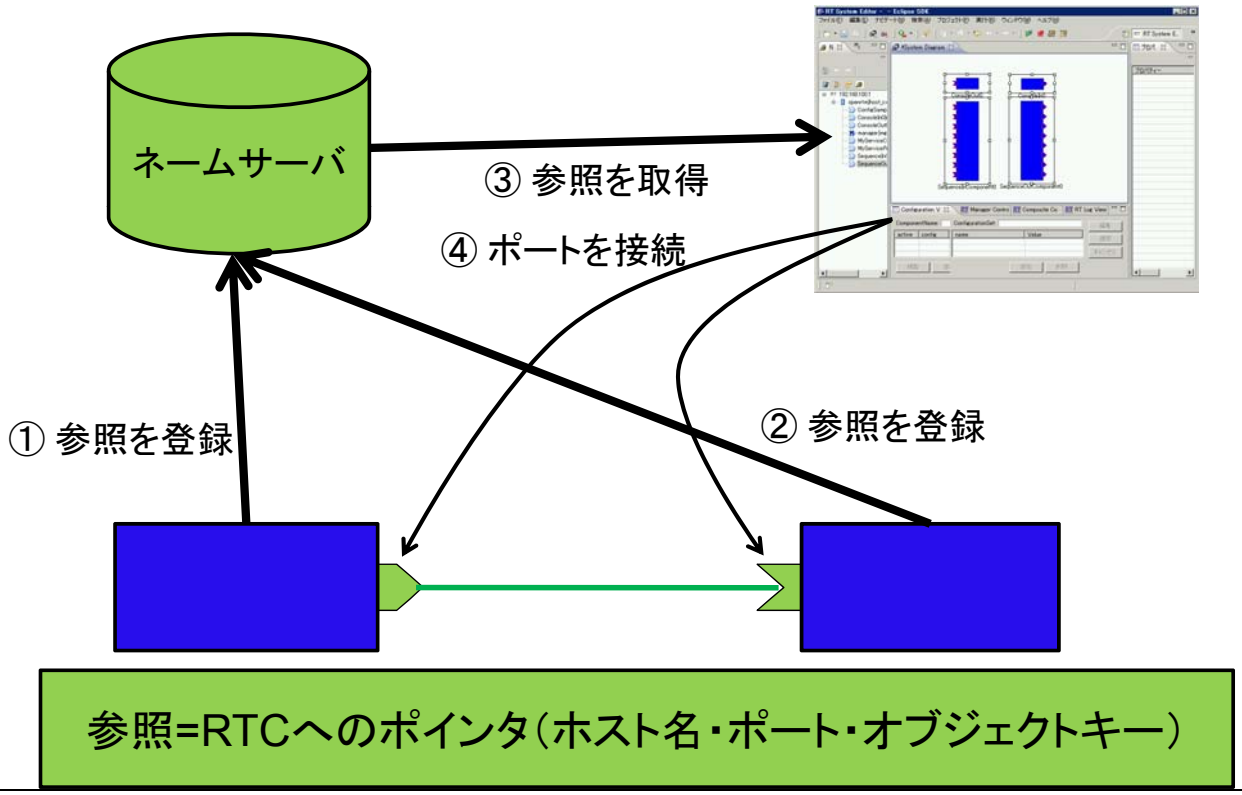
シーケンス型の例

```
struct TimedShortSeq
 {
      Time tm;
      sequence<short> data;
 };
```

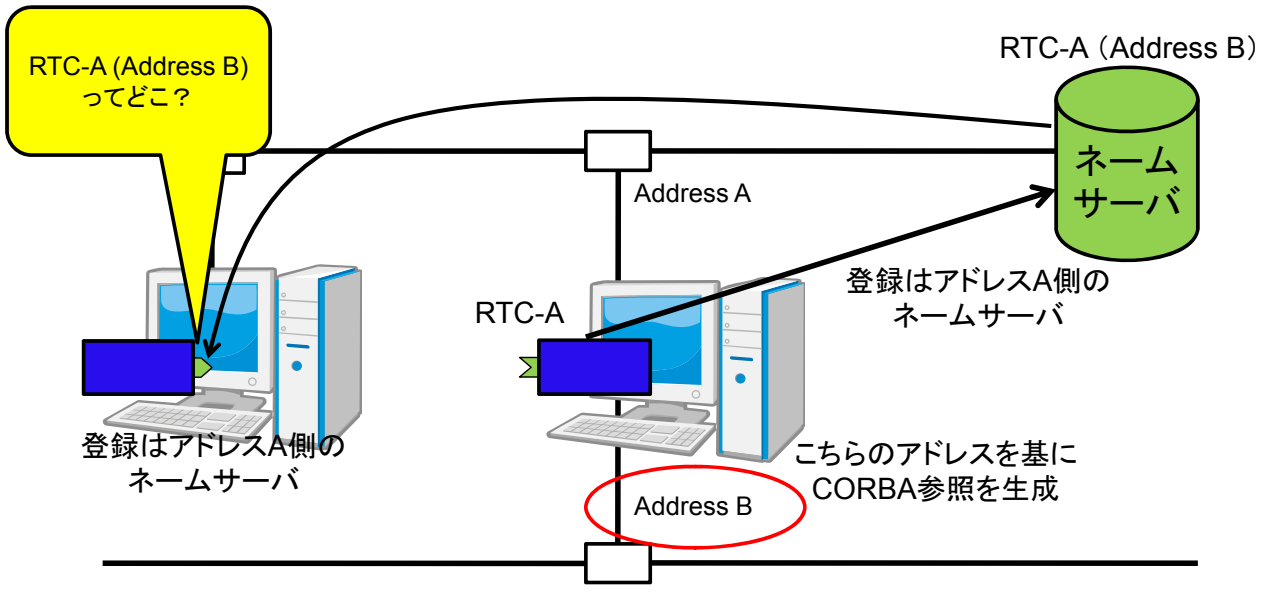
- シーケンス型
  - data[i]: 添え字によるアクセス
  - data.length(i): 長さiを確保
  - data.length(): 長さを取得

- データを入れるときにはあらかじめ長さをセットしなければならない。
- CORBAのシーケンス型そのもの
- 今後変更される可能性あり

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



rtc.confについて

RT Component起動時の登録先NamingServiceや、登録情報などについて記述するファイル

記述例：

corba.nameservers: localhost:9876

naming.formats: SimpleComponent/%n.rtc

（詳細な記述方法は etc/rtc.conf.sample を参照）

以下のようにすると、コンポーネント起動時に読み込まれる

./ConsoleInComp –f rtc.conf

ネーミングサービス設定

corba.nameservers	host_name:port_numberで指定、デフォルトポートは2809(omniORBのデフォルト)、複数指定可能
naming.formats	%h.host_cxt/%n.rtc →host.host_cxt/MyComp.rtc 複数指定可能、0.2.0互換にしたければ、 %h.host_cxt/%M.mgr_cxt/%c.cat_cxt/%m.mod_cxt/%n.rtc
naming.update.enable	“YES” or “NO”: ネーミングサービスへの登録の自動アップデート。コンポーネント起動後にネームサービスが起動したときに、再度名前を登録する。
naming.update.interval	アップデートの周期[s]。デフォルトは10秒。
timer.enable	“YES” or “NO”: マネージャタイマ有効・無効。naming.updateを使用するには有効でなければならない
timer.tick	タイマの分解能[s]。デフォルトは100ms。

必須の項目　　必須でないOption設定

ログ設定

logger.enable	“YES” or “NO”: ログ出力を有効・無効
logger.file_name	ログファイル名。 %h：ホスト名、%M:マネージャ名,%p：プロセスID 使用可
logger.date_format	日付フォーマット。strftime(3)の表記法に準拠。 デフォルト：%b %d %H:%M:%S → Apr 24 01:02:04
logger.log_level	ログレベル： SILENT, ERROR, WARN, NORMAL, INFO, DEBUG, TRACE, VERBOSE, PARANOID SILENT：何も出力しない PARANOID：全て出力する ※以前はRTC内で使えましたが、現在はまだ使えません。

必須の項目　必須でないOption設定

その他

corba.endpoints	IP_Addr:Port で指定：NICが複数あるとき、ORBをどちらでlistenさせるかを指定。Portを指定しない場合でも”:”が必要。 例 “corba.endpoints: 192.168.0.12:” NICが2つある場合必ず指定。 (指定しなくても偶然正常に動作することもあるが念のため。) 使いたいNICに割り当てられているIPアドレス
corba.args	CORBAに対する引数。詳細はomniORBのマニュアル参照。
[カテゴリ名]. [コンポーネント名]. config_file または [カテゴリ名]. [インスタンス名]. config_file	コンポーネントの設定ファイル •カテゴリ名：manipulator, •コンポーネント名：myarm, •インスタンス名myarm0,1,2,… の場合 manipulator.myarm.config_file: arm.conf manipulator.myarm0.config_file: arm0.conf のように指定可能

必須の項目　必須でないOption設定

まとめ

- ロボット用ミドルウエア：OpenRTM
  - ロボットに適した共通フレームワークの提供
  - OMG国際標準
  - 多様な実装、多様な言語、OSに対応
- RTコンポーネントの作り方
  - フレームワークとコアロジック
  - コールバック関数に実装
  - InPortから読み込み、処理してOutPortへ書き込み