エージェントシステム
RTM/ROS相互運用
RTM第一回
2011/4/20

情報システム工学研究室
特任講師　吉海智晃

RTミドルウェアとは？

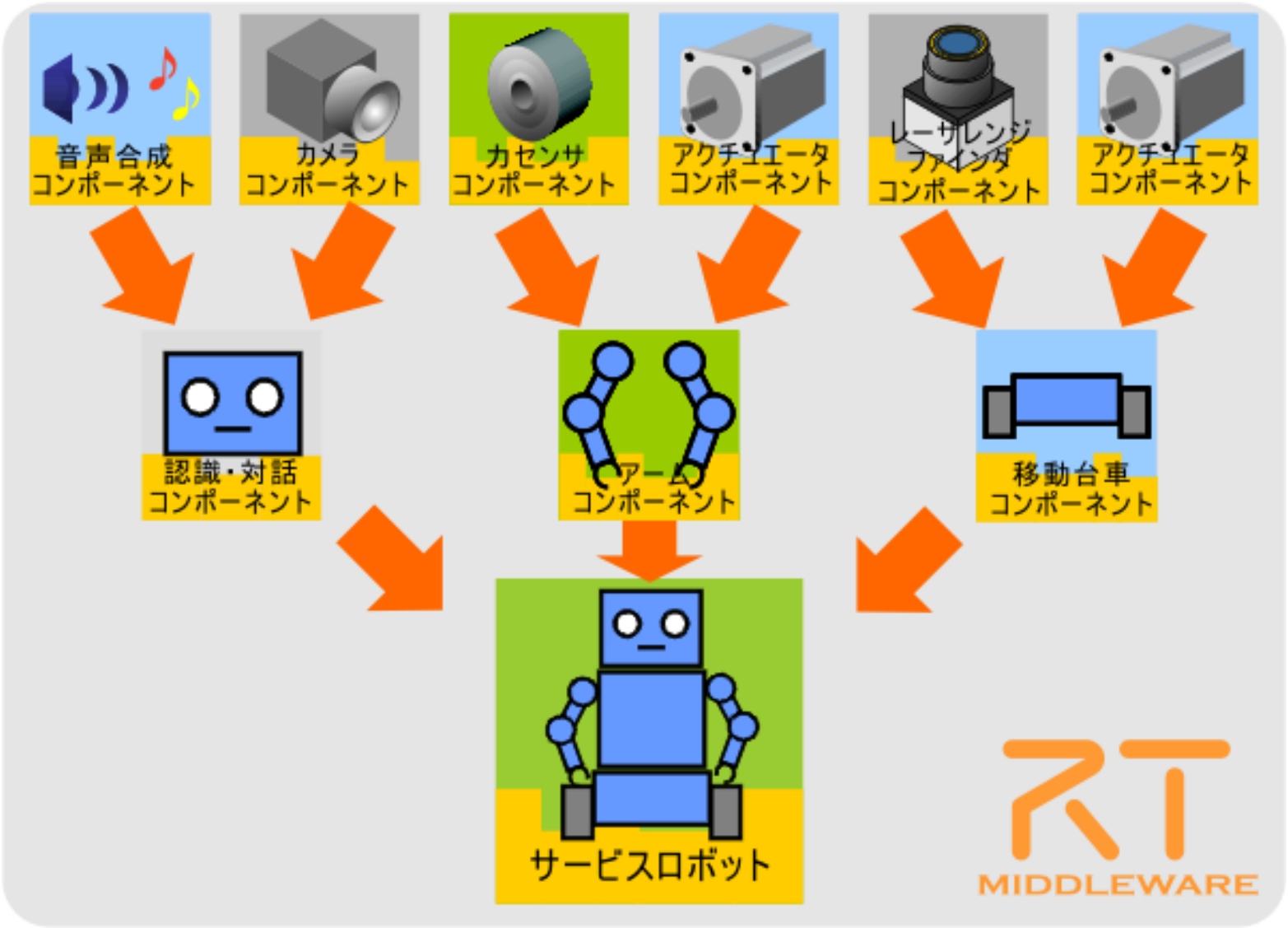
- 分散オブジェクト指向システムのための技術であるCORBAをベースにしたRT(Robot Technology)分野のアプリケーションの共通ミドルウェアを提供する枠組み

- 現在，本体に関しては産総研のグループを中心に開発が進められている
  - オフィシャルページ　http:// www.openrtm.org/
    - 現在の所，公式ページのドキュメント類は日/英/韓に対応
- NEDO次世代ロボット知能化技術開発プロジェクト（H20-H23)において，RTミドルウェアをコアとした知能化ソフトウェアモジュール群の開発が全国の企業・大学で行われている

RTMシステムの特徴

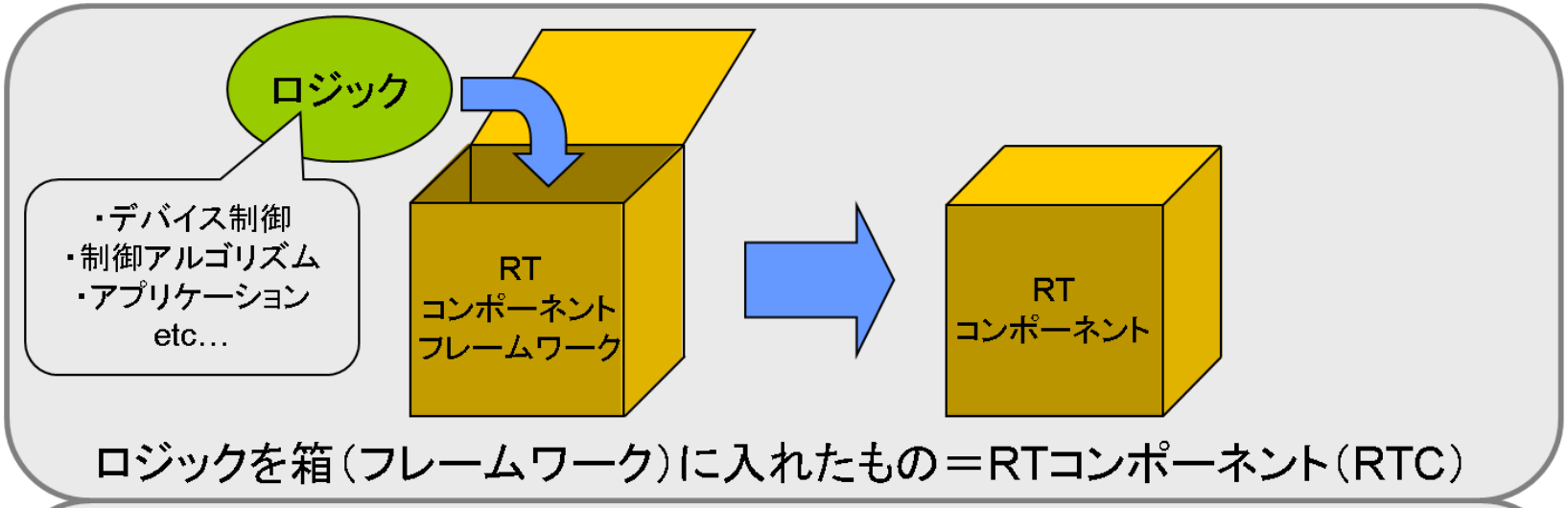
- プラットフォーム非依存，ネットワーク透過な分散オブジェクト環境
- 機能要素ごとのコンポーネント化
  - 再利用性の向上
  - システム堅牢性の向上
  - 接続構成組み換えによるシステム柔軟性の向上など
- OMG(Object Management Group, 国際的ソフトウェア標準化団体）による公式標準仕様になっている

アプリケーションのイメージ

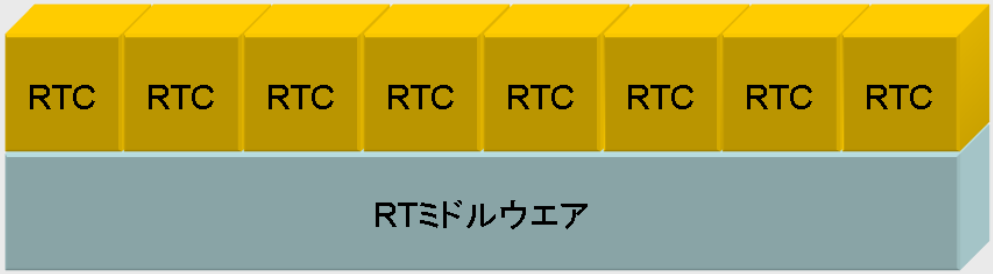


Official web ドキュメントより転載

個々のソフトウェア
コンポーネント（RTC）のイメージ



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

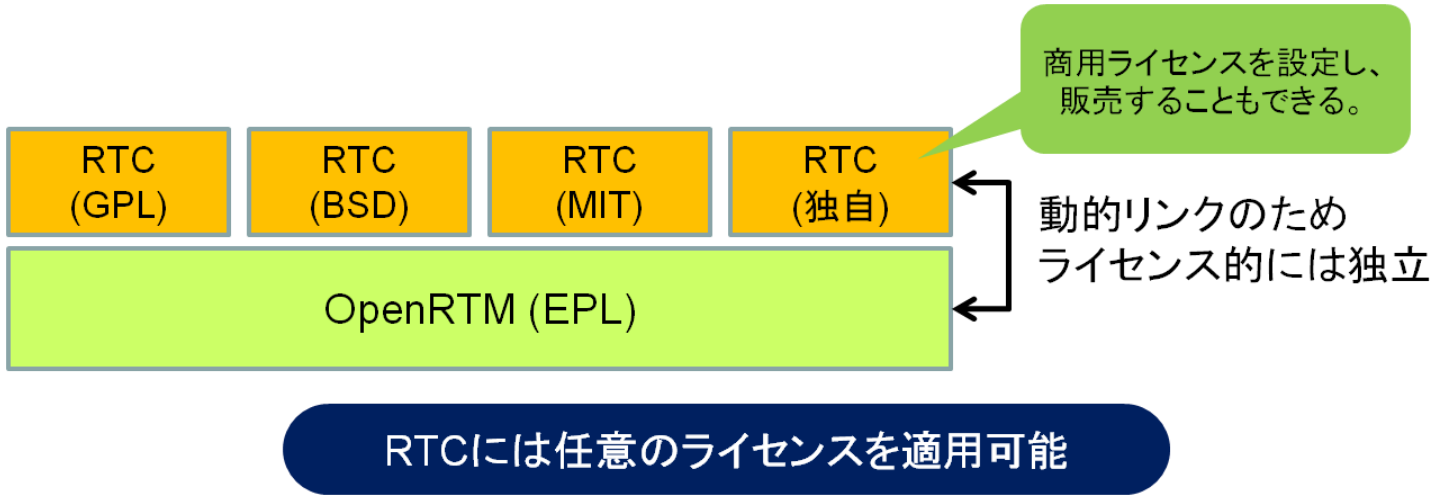
Official webドキュメントより転載

RTミドルウェアの利用例の一部

- OpenHRP3
  - 動力学シミュレータ，HRP-2, HIRO, HRP-4などで利用
- OpenINVENT
  - 車輪型移動ロボットの自律移動制御を行うソフトウエアモジュール
- OpenHRI
  - 対話制御コンポーネント群
- ヒューマノイド PALRO（パルロ）
  - 富士ソフト製小型ヒューマノイドロボット，富士ソフトから購入可能
- 小型組込画像処理モジュール
  - 富士通研究所開発，富士通九州ネットワークテクノロジーズから購入可能
- RTC-OpenRAVE
  - OpenRAVEのRTCインタフェース

OpenRTM-aist-1.0.0

- 産総研が研究・開発・配布を行っているRTC OMG標準のRTミドルウェア実装の一つ
- 2011年4月現在の最新版
  - EPL(Eclipse Public License)と個別契約のデュアルライセンス
  - 各RTコンポーネントにはこのライセンスは及ばないので，RTC開発者は任意のライセンスで配布・販売可能



Official webドキュメントより転載

開発環境・コミュニティなど

- マルチプラットフォーム
  - Windows, Linux各種ディストリビューション，MacOSなどに対応
  - C++, Python, Java でのRTC開発が可能
- 開発者メーリングリスト
  - openrtm-users@m.aist.go.jp
    - 400名以上が参加．開発者から直接回答がもらえる
- RTMコンテスト
  - 毎年，計測自動制御学会システムインテグレーション部会学術講演会の一部として開催されている

OpenRTM-aist-1.0.0の標準外部ツール

- RTCBuilder/RTSystem Editor
  - Eclipse上で動作するツール群
  - RTCBuilder: RTコンポーネント設計，コード生成のためのツール
  - RTSystemEditor: RTコンポーネントのグラフィカルな操作のためのツール（初期動作確認，単体開発には重要）
- rtcshell/rtsshell
  - RTコンポーネントのCUIインタフェース（多コンポーネントの組合せ，操作のために重要）

OpenRTM-aistによるシステム開発の流れ

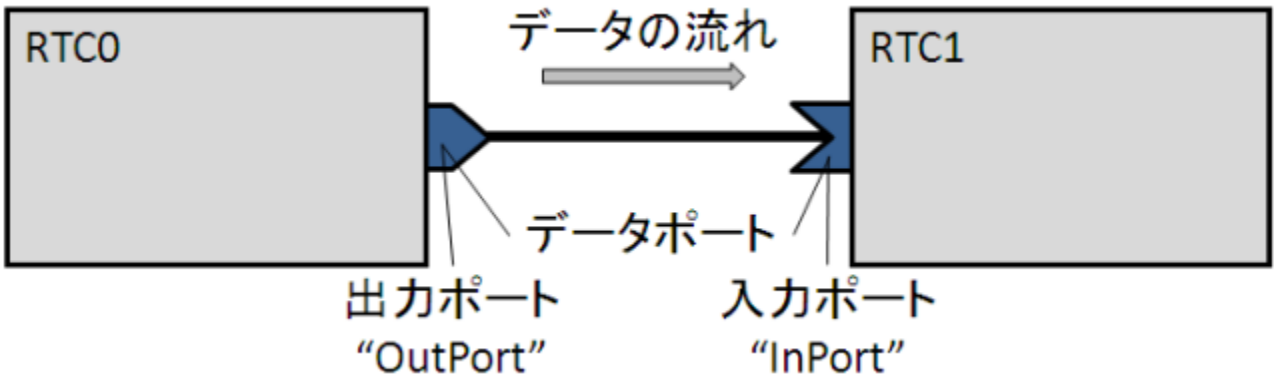
1. コンポーネントの仕様決定
2. コードジェネレータでコンポーネントの雛型を生成
3. コアロジックの部分を雛型の特定の場所に埋め込む
4. コンパイル，テスト，デバッグ
5. システムへの組み込み

1．コンポーネントの仕様決定

- RTコンポーネントのメタ情報
  - コンポーネントプロファイル
    - コンポーネント名，カテゴリ名，バージョン等
  - データポート
    - InPort・OutPort，ポート名，データ型
  - サービスポート
    - ポート名，サービスインターフェース（インターフェース名，プロバイダ・コンシューマ）
  - コンフィギュレーションインタフェース

データポート

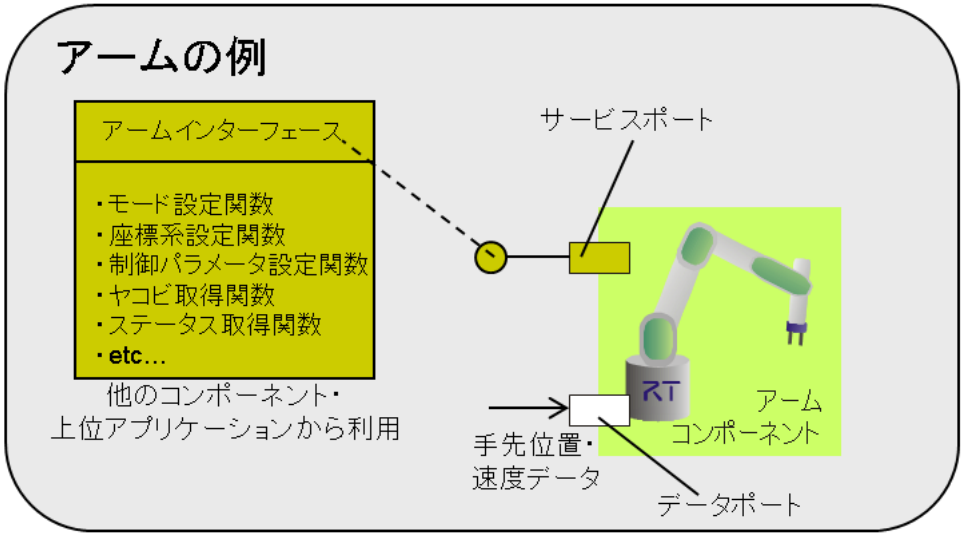
- InPort/OutPort: 連続的データ授受のための通信インタフェース，データは組込型だけでなく独自定義も可能
  - **データ入力ポート(InPort)** :
    外部からデータを受け取るポート
  - **データ出力ポート(OutPort)** :
    データを外部に送信するためのポート

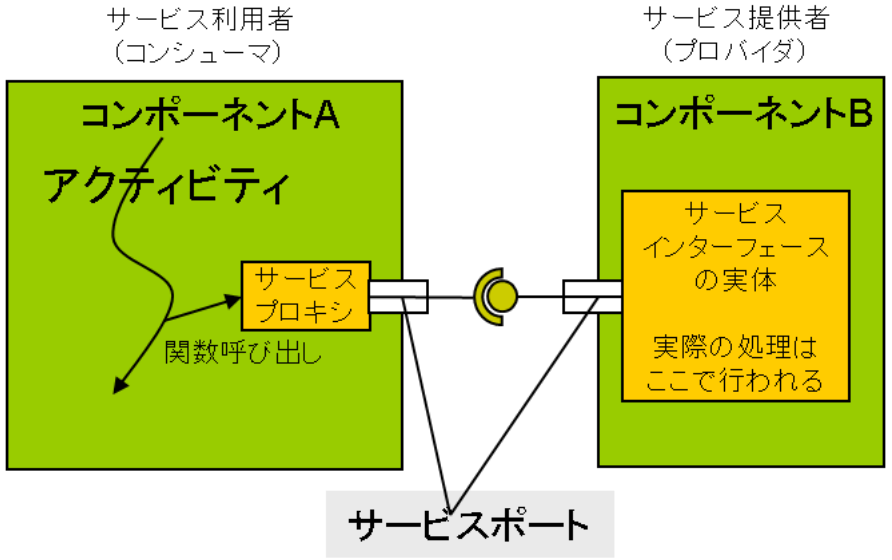


Official webドキュメントより転載

サービスポート

- 開発者定義の関数レベルコンポーネント間通信のための通信インタフェース
- 任意の数，種類のインタフェース作成が可能
  - **サービスプロバイダ** :サービス提供のためのインタフェース
  - **サービスコンシューマ** : サービス利用のためのインタフェース

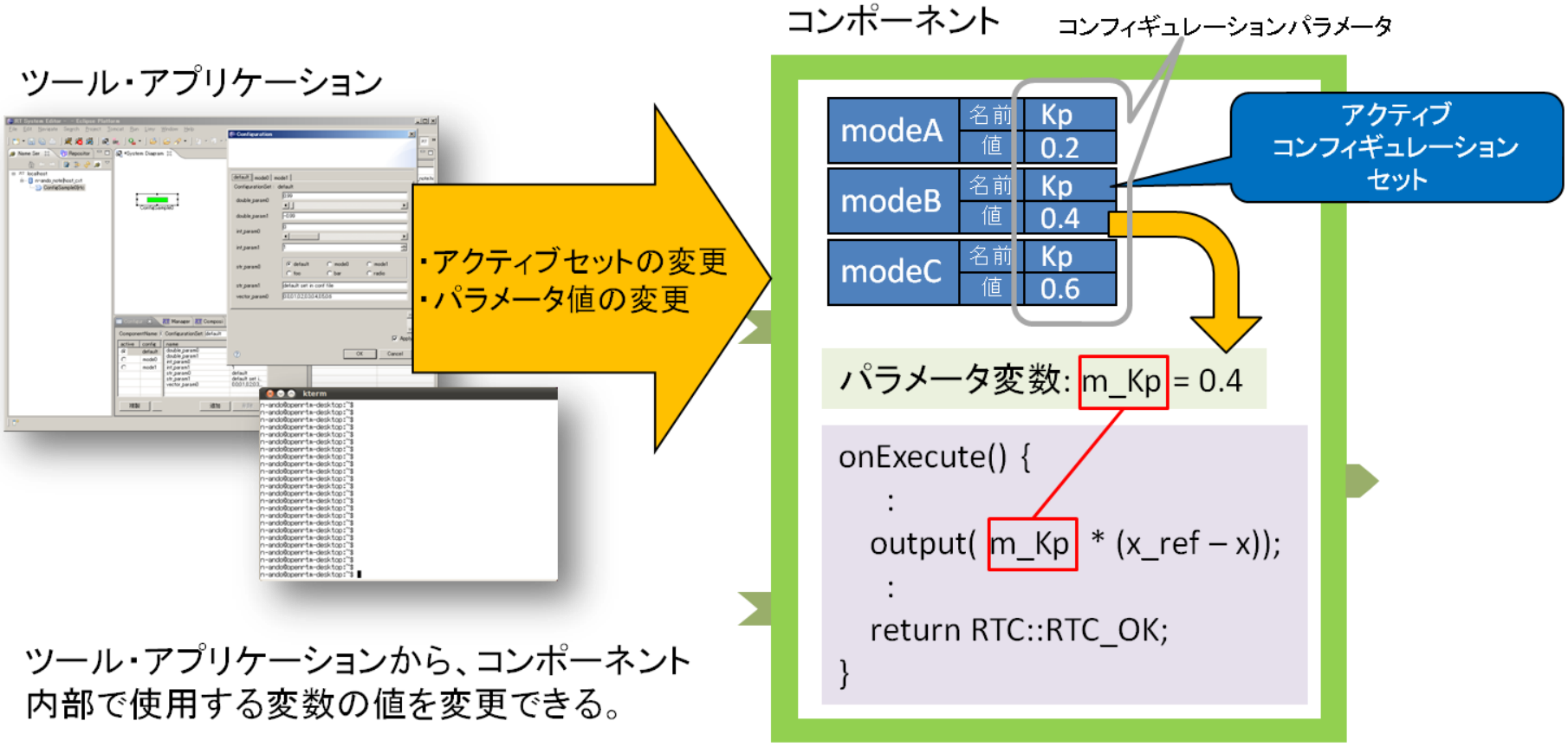




Official webドキュメントより転載

コンフィギュレーションインタフェース

- コンポーネントのコアロジックのパラメータを外部から参照・変更するための通信インタフェース



Official webドキュメントより転載

２．コードジェネレータでコンポーネントの雛型を生成

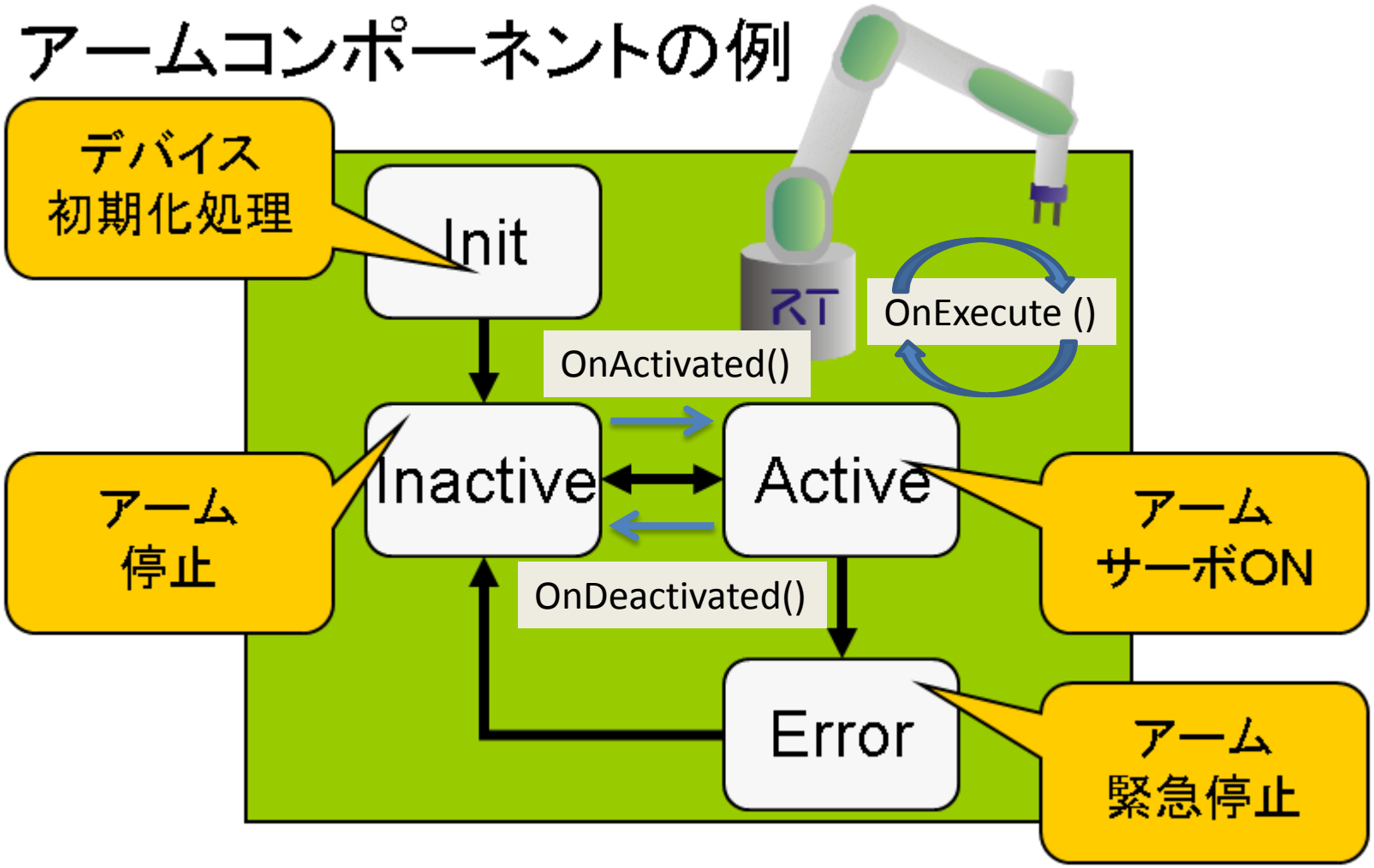
- GUIのコードジェネレータ(RTCBuilder)
  - C++(VS2005, VS2008対応)，Python, Javaなど
  - グラフィカルにパラメタを指定，コード生成

- CUIのコードジェネレータ(rtctemplate)
  - C++(gcc)，Pythonなど
  - 引数でパラメタを指定，コード生成

３．コアロジックの部分を雛型の特定の場所に埋め込む

- RTコンポーネントは，通常ある特別な基底クラスを継承した一つのクラスとして実装される
- RTコンポーネントにさせたい処理は，その基底クラスのメンバ関数をオーバーライドする
- コンポーネントの状態遷移ごとのアクションに対応する関数の実装をする
  - 各アクションに対応したメンバ関数をオーバーライドし，関数の中身を記述
    - OnActivated()　：　アクティベート時の挙動の記述
    - OnExecute()　：　周期実行時の挙動の記述
    - OnDeactivated()　：　ディアクティベート時の挙動の記述

コンポーネントの状態遷移の例



Official webドキュメントより転載

４．コンパイル，テスト，デバッグ

- コンパイルはコードジェネレータで生成されたMakefile / VS用のソリューションファイルを利用して行う
- 単体テストとデバッグはRtcSystemEditorを利用するのが簡便
  - スタンドアロン型（.exe）で作り，各々実行
  - 接続とコンポーネントの動作開始をRtcSystemEditorで試す

コンパイル後の実行準備

- 実行形式の各コンポーネントの実行には，コンフィギュレーションファイル（rtc.conf）が必要！
  - rtc.conf

    corba.nameservers: ネームサーバ名 : ポート番号

    naming.formats: %n.rtc

    logger.log_level: PARANOID

- CORBAのネーミングサービスを起動
  - rtm-naming ポート番号

OpenRTM-aist-1.0.0本体のインストール

- インストールパッケージ（Win, Unix)
  - 公式ホームページ

    http://www.openrtm.org/openrtm/ja/

    の「ナビゲーション」→「ダウンロード」にてソース，バイナリが配布されている

- Ubuntu10.04 及びWindowsに関しては，詳しくは講義wiki「RTM_Install」参照

OpenRTM-aist-1.0.0の標準外部ツールのインストール

- コードジェネレータRTCBuilder
- RTC操作用GUI　RTSystemEditor
- Ubuntu10.04 及びWindowsに関しては，講義wikiページの「RTM_Install」参照

インストール後の動作確認

1. ネームサーバの立ち上げ
2. Eclipseを起動して，RTSystemEditrorを開く
3. TkJoystickComp.pyの立ち上げ
4. TkMobileRobotSimulator.pyの立ち上げ
5. TkMobileRobotsSimulatorで「Create」を押す
6. RTSystemEditrorで2つのコンポーネントを接続
7. RTSystemEditrorでコンポーネントをアクティベートする

詳しくは講義wiki「RTM_Example」参照

次回までの宿題

- 講義ホームページを参照して，OpenRTM本体及び，RTCBuilder, RTSystemEditorをインストールし，サンプルが動くことを確認する.

- 次回には回答として，Ubuntu版のVMWareイメージを配布する予定