

エージェントシステム
RTM/ROS相互運用
RTM第二回
2011/4/27

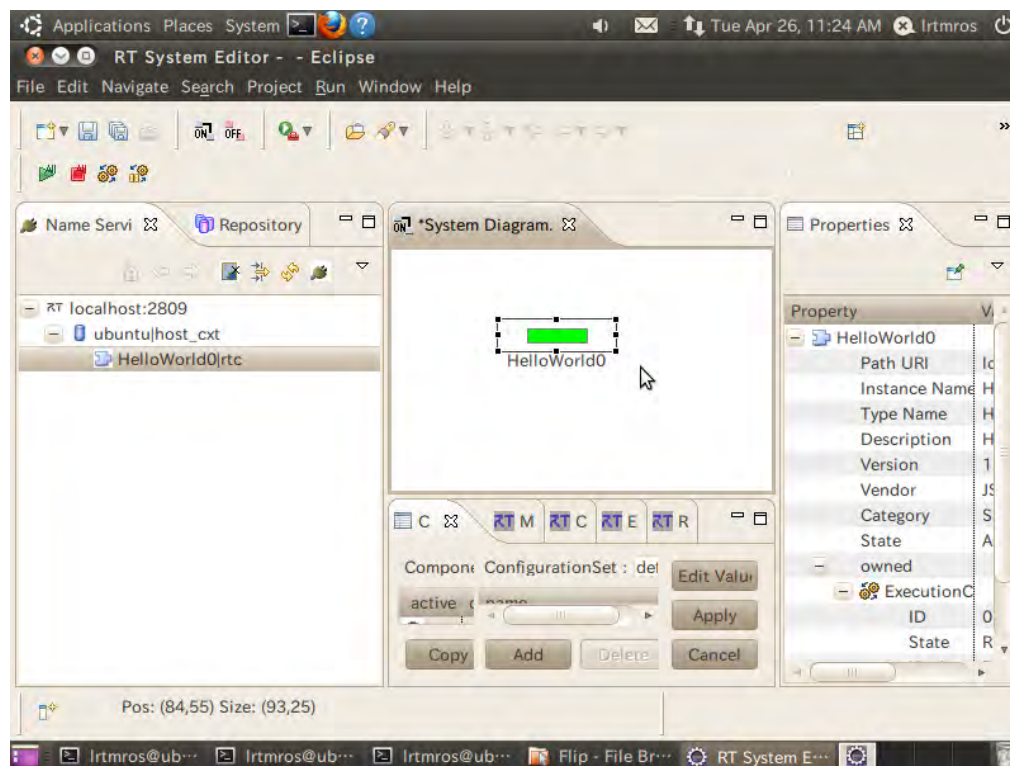
情報システム工学研究室
特任講師 吉海智晃

RTミドルウェアとは？-復習-

- 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

ハローワールドサンプル

- 起動時, アクティベート時に各々端末に特定の文字をプリントする
- アクティベート後は周期的に特定の文字をプリント

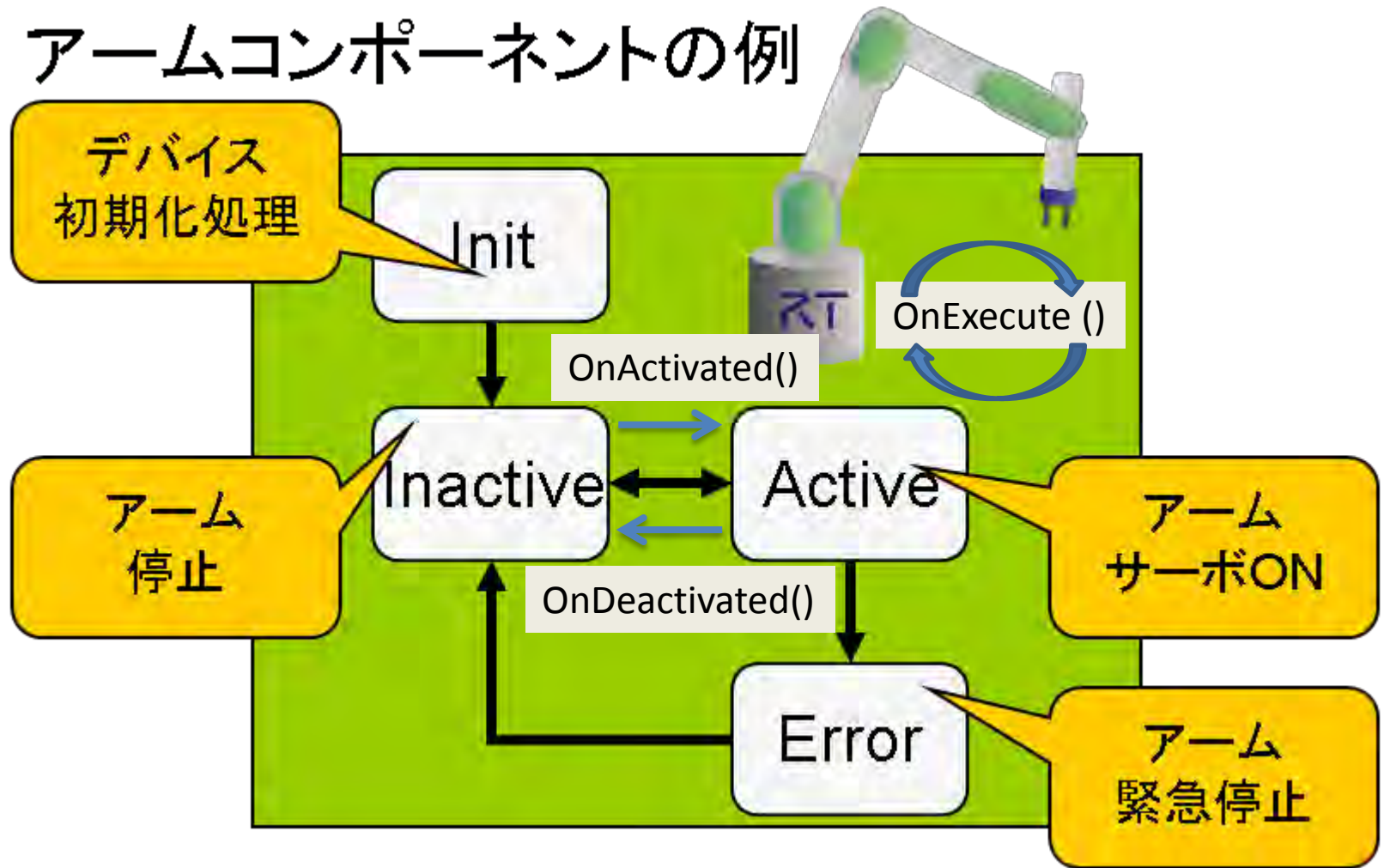


RTC Builderで雛形を作る

1. 「Basic」タブで下記項目を記入
 - Module name
 - Module vender
 - Module Category
 - Output Project
 - Execution rate
2. 「Activity」タブで下記のコールバックをONにする
 - OnInitialize
 - OnActivated
 - OnExecute
3. 「Language and Environment」タブで言語を選ぶ
 - C++ を選ぶ (Pythonで作りたい人はPythonで)
4. 「Basic」タブでCode Generationボタンを押して雛形作成

コンポーネントの状態遷移の復習

アームコンポーネントの例



コアロジックを記述

- Eclipseのworkspaceの下に指定したOutput Projectの名前のフォルダができているはず
- HelloWorld.cppを編集
 - OnInitialize() の中に

```
std::cerr << "Initializing Component " << std::endl;
```
 - OnActivated()の中に

```
std::cerr << "Activating Component " << std::endl;
```
 - OnExecute()の中に

```
std::cerr << "Hello World!" << std::endl;
```

コンパイル

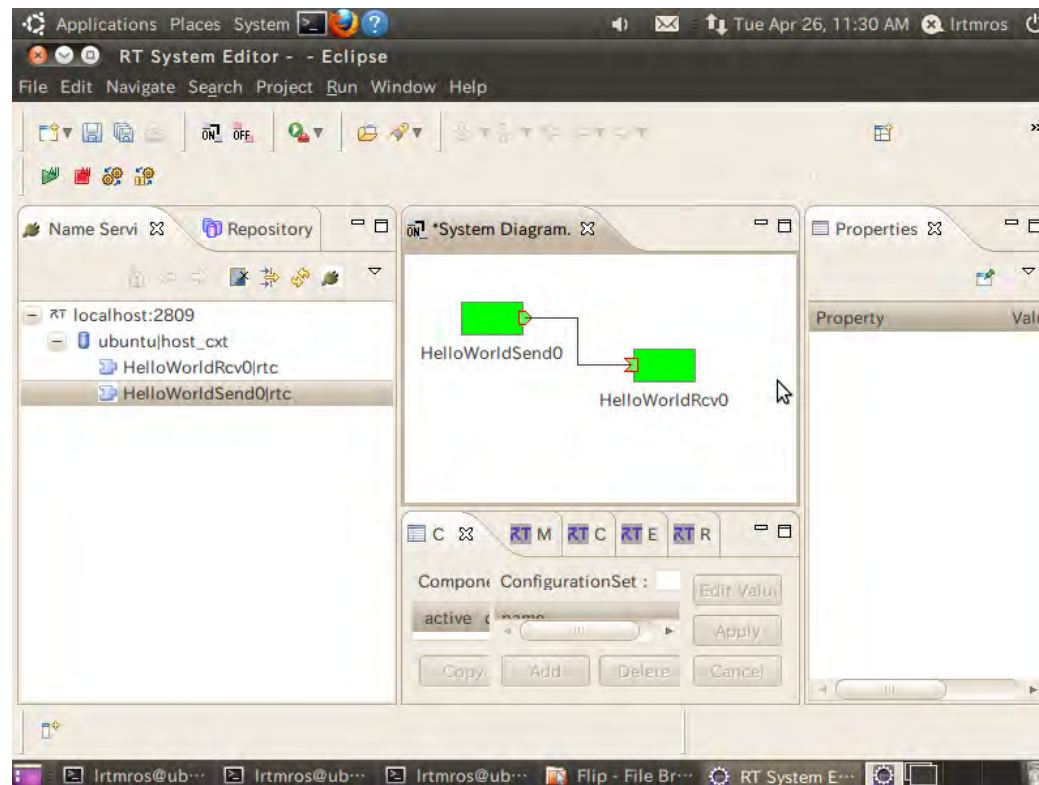
- Ubuntuの場合
 - \$ make -f Makefile.HelloWorld
- Windowsの場合
 - copyprops.bat をダブルクリック
 - 対応するバージョンの.slnをダブルクリックして, 「ビルド」
→「ソリューションのビルド」
- rtc.confを編集 (CORBAのネームサーバ情報, 命名規則, ロガー設定, 実行周期など)
 - corba.nameservers: localhost
 - naming.formats: %h.host_cxt/%n.rtc
 - logger.enable: NO
 - exec_cxt.periodic.rate: 1000

テスト

- ネームサーバを上げる(一度でも起動していればOK)
- 上記のrtc.confをコンポーネントと同じ場所においてコンポーネントを起動
 - Ubuntuの場合
\$./HelloWorldComp
 - Windowsの場合
HelloWorldComp.exeをダブルクリック等で実行
- RT System Editorでアクティベートする
 - rtc.confで実行周期を変えると挙動が変化することを確認

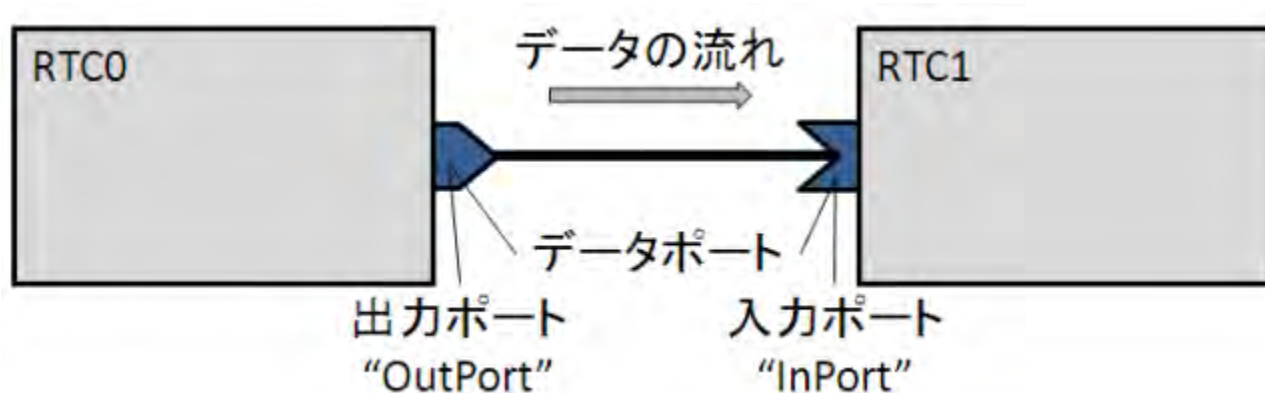
コンポーネント通信サンプル

- データポートによる連続的な通信
- HelloWorldコンポーネントを文字列を送るコンポーネントと受け取って表示するコンポーネントにわけ



データポートの復習

- 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



HelloWorldの時との違い

- 雛形作成時のデータポート作成の指定
 - 「データポート」タブでポートのポート名, 入出力属性, 通信に用いるデータの型(独自構造体も設定可能, 方法は, 第三回で紹介予定)を設定
- TimedString型のOutPortを持つHelloWorldSendとTimedString型のInPortを持つHelloWorldRecvを作る

データポートにまつわる操作(C++)

出力:

```
TimedString m_StringData;  
OutPort<TimedString> m_StringDataOut;  
  
m_StringData.data="test"  
m_StringDataOut.write() ←データ書出し
```

入力:

```
TimedString m_StringData;  
InPort<TimedString> m_StringDataIn;  
  
if(m_StringDataIn.isNew()){ ←新規データチェック  
    m_StringDataIn.read(); ←データ読み込み  
    std::cerr << m_sStringData.data << std::endl;  
}
```

データポートにまつわる操作(Python)

出力:

```
def onExecute(self, ec_id):  
    self.sendmsg.data = "HelloWorld"  
    OpenRTM_aist.setTimeStamp(self.sendmsg)  
    self._outport.write()  
    return RTC.RTC_OK
```

入力:

```
def onExecute(self, ec_id):  
    if self._inport.isNew():  
        rcvmsg = self._inport.read()  
        print rcvmsg.data  
    return RTC.RTC_OK
```

CUIによる雛形生成

- rtc-template

```
rtc-template -bcxx --module-name=HelloWorldSend  
--module-type='DataFlowComponent'  
--module-desc='HelloWorldSend'  
--module-version=1.0 --module-vendor='JSK'  
--module-category=example  
--module-comp-type=DataFlowComponent  
--module-act-type=SPORADIC  
--module-max-inst=10 --outport=SendMsg:TimedString
```

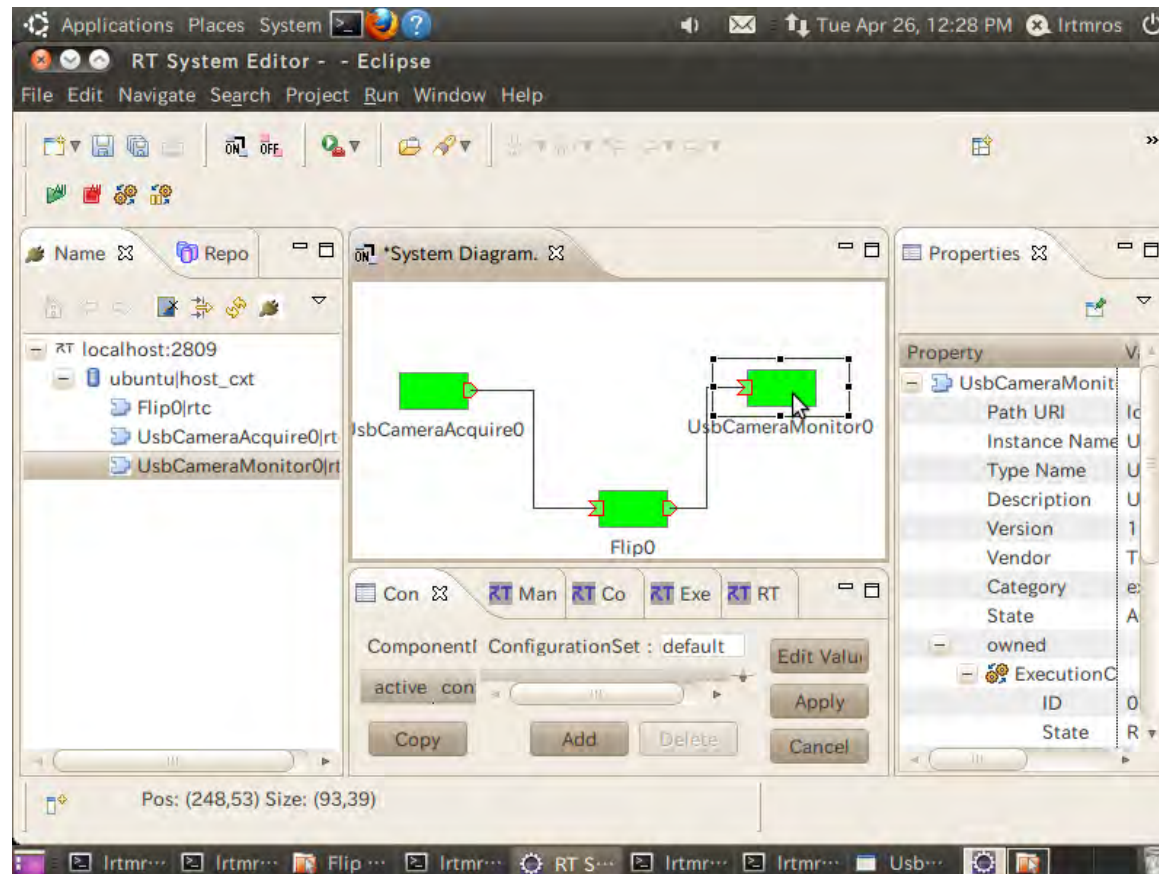
ただし, -bpythonは最新版に対応していないようなので要注意

テスト

- C++版とPython版の接続
- Windows版とUbuntu版の接続

画像処理サンプル

- USBカメラキャプチャコンポーネント
- USBカメラモニタコンポーネント
- OpenCV画像処理コンポーネント(Flipする)



画像入出力コンポーネント

- Windows版
 - C++のexampleとして付属. OpenCV1.0用であることに注意
- Ubuntu版
 - サンプルを, 講義ページのWIKIに掲載

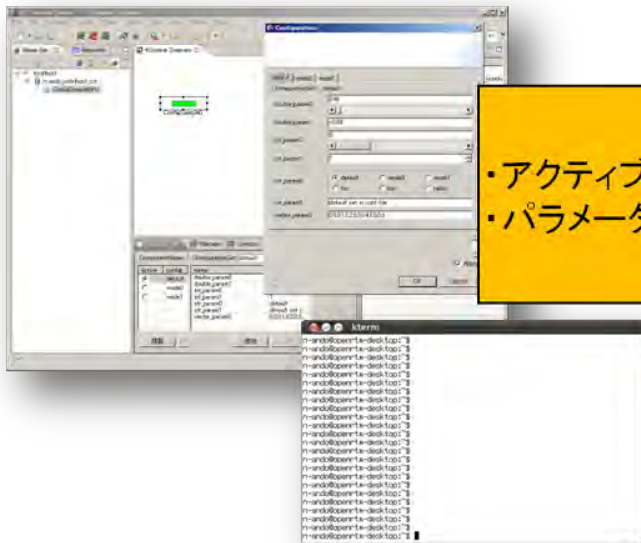
Flipのパラメータを外部から適時修正する

- コンフィギュレーションインタフェースを利用することで実現可能
- RTSystemEditorから直接変数をいじることができるようになる.

コンフィギュレーションインタフェースの復習

- 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

ツール・アプリケーション



・アクティブセットの変更
・パラメータ値の変更

コンポーネント

コンフィギュレーションパラメータ

modeA	名前	Kp
	値	0.2
modeB	名前	Kp
	値	0.4
modeC	名前	Kp
	値	0.6

アクティブ
コンフィギュレーション
セット

パラメータ変数: $m_Kp = 0.4$

```
onExecute() {  
    :  
    output(  $m\_Kp * (x\_ref - x)$  );  
    :  
    return RTC::RTC_OK;  
}
```

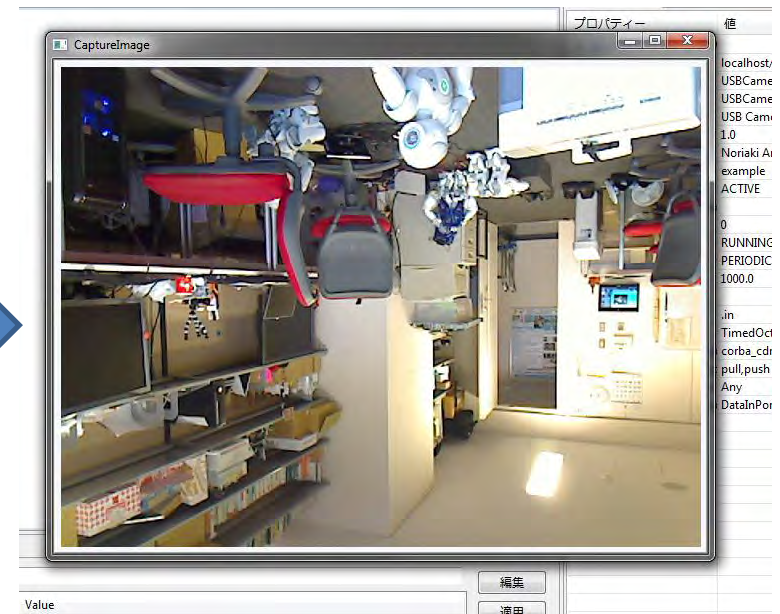
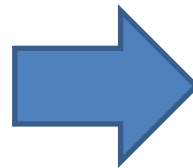
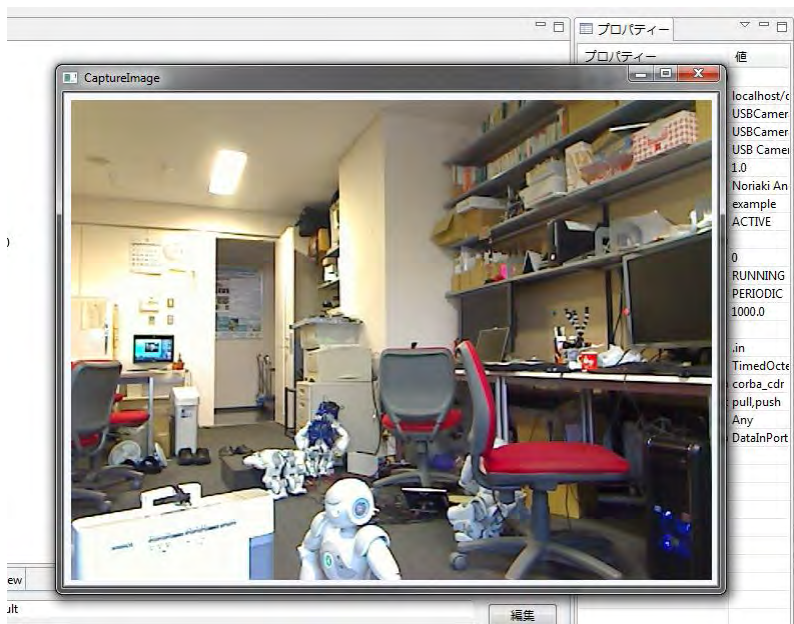
ツール・アプリケーションから、コンポーネント
内部で使用する変数の値を変更できる。

C++版でのコンフィギュレーション インタフェースの実装

1. `Static_const char* xx_spec[]` 中での記述
“`conf.default.flipMode`”, “1”,
のように`conf.default`.変数名とする
2. `onInitialize()` 中での記述
`bindParameter(“flipMode”, m_flipMode, “1”);`
のようにして, メンバ変数との対応付けを行う
3. `onExecute()` 中での記述
周期実行の際に`m_flipMode`の値に応じた
処理を書くことで外部からの制御が可能になる

テスト

- 接続してコンフィギュレーションを変更することにより処理が即時に変わることを確認



次回までの宿題

- 基本課題

- 講義Wikiの画像処理サンプルを参考にして, USBカメラでキャプチャした画像に対してFlip以外の処理を行って, それを表示する

- 応用課題

- マルチプラットフォーム環境を用意し(例えば, Windows+VMWare Ubuntuなど), 一方でキャプチャした画像に対して他方で画像処理を行うコンポーネントを書いてみよう