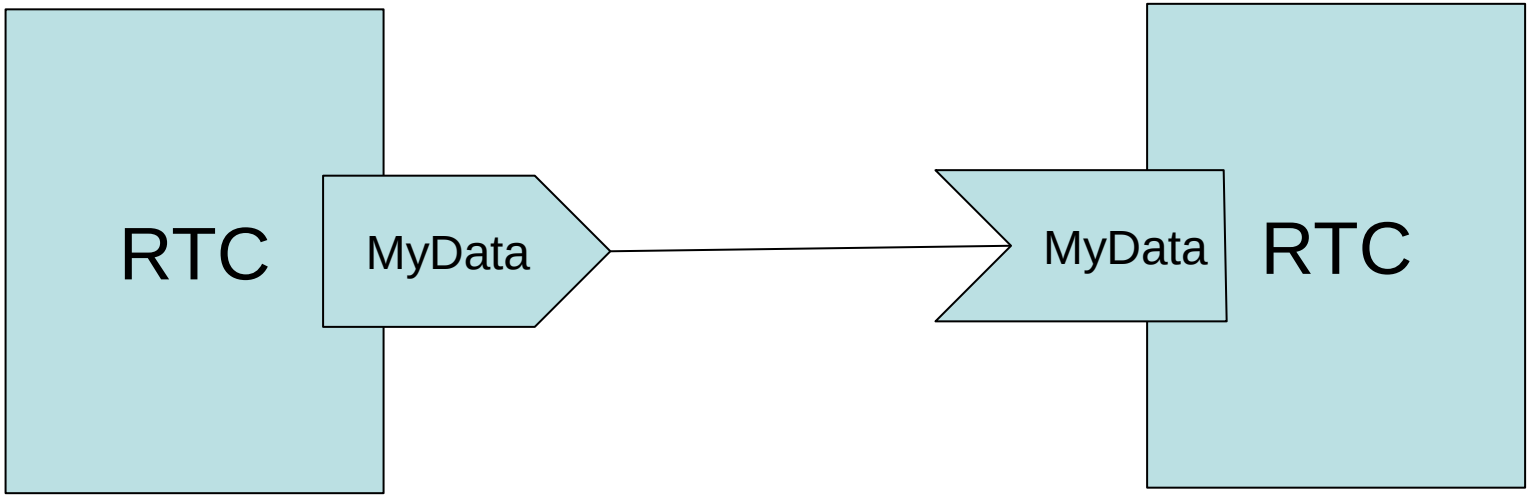
OpenRTM応用コース

Geoffrey Biggs
知能システム研究部門
産業技術総合研究所

独自データ型

- データポートでOpenRTMが提供する以外のデータ型を使う
- 例：自分のセンサーにふさわしいデータ型



お勧め

自分のデータ型を作る前に、既に似たようなデータ型が存在しないか確認してください

作り方の流れ

1. IDLファイルの作成
2. データポートの作成
3. コンポーネントのコンパイル

IDLファイルの作成

- データ型はIDLファイルで定義する
- 基本は C の struct と同様
- 例:

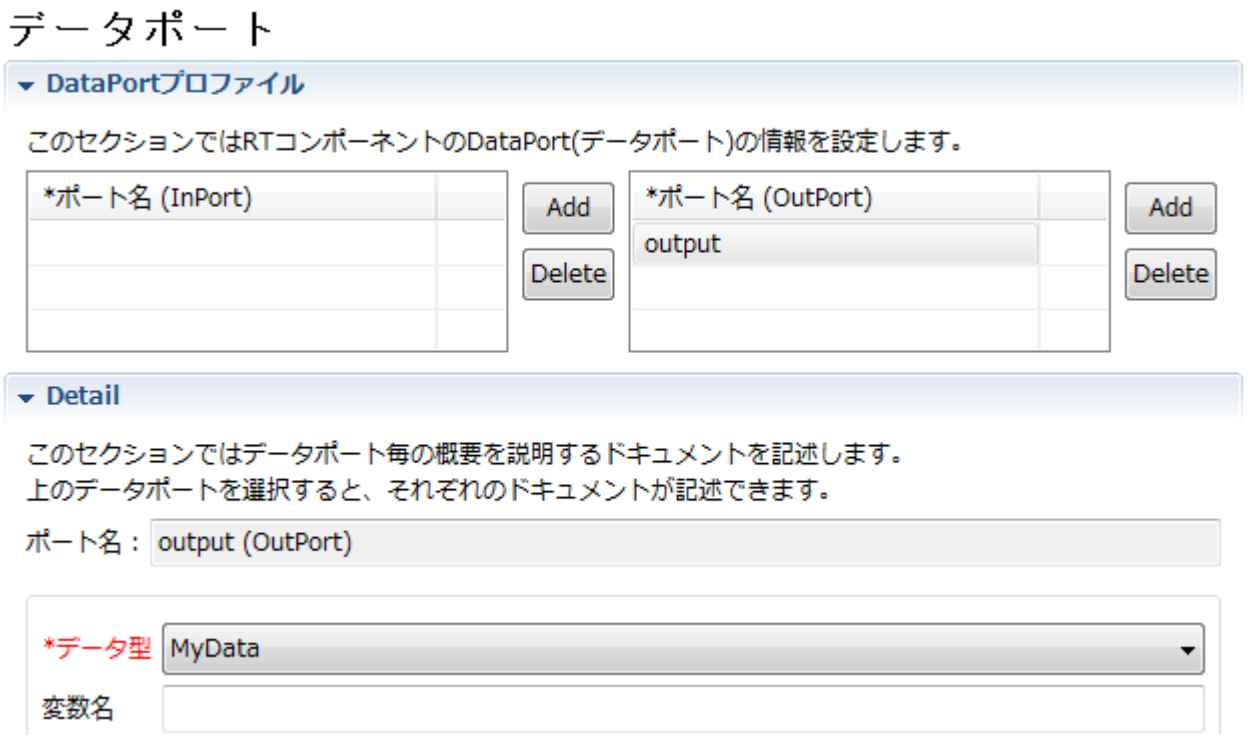
```
struct MyData {
    RTC::Time tm;
    short shortVariable;
    long longVariable;
    sequence<double> data;
};
```

IDL作成の注意

- メソッドは使えない
- OpenRTMが提供するデータ型も使える
  - OpenRTMのIDLファイルをincludeすることが必要

データポートの作成

- データポートのデータ型を独自データ型に設定する



コンポーネントのコンパイル

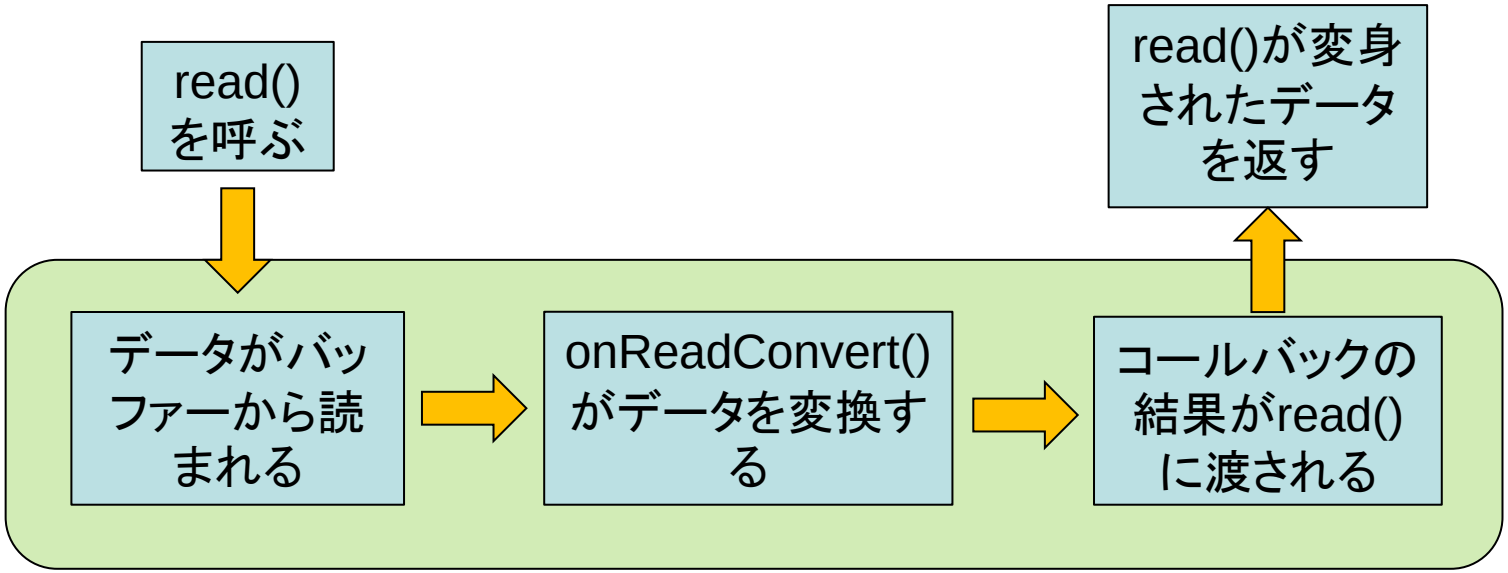
- IDLファイルからC++ファイルは自動的に作成される
- 再度コンパイルすると、コンポーネントのビルドを行える

コールバック

- OpenRTMで複数のコールバックがある
  - データポートにデータが届いた
  - データポートにデータが書かれた
  - ポートがコネクトされた
  - バッファーがフルになった
  - …
- Python版にもある

例:データが届いたら変換する

- データが読まれるすぐ前に変換する
  - 単位変更など
- onReadConvert()を使う



コールバックの作り方

- ファンクタを実装する

```
class MyOnReadConvert
    : public RTC::OnReadConvert<int>
{
    public:
        virtual int operator()(int value)
        {
            int tmp;
            tmp.data = value.data / 1000.0;
            return tmp;
        }
};
```

コールバックの使い方

- ファンクタをポートに登録する

```
MyOnReadConvert on_read_convert();
inport.setOnReadConvert(&on_read_convert);
```

他のコールバック

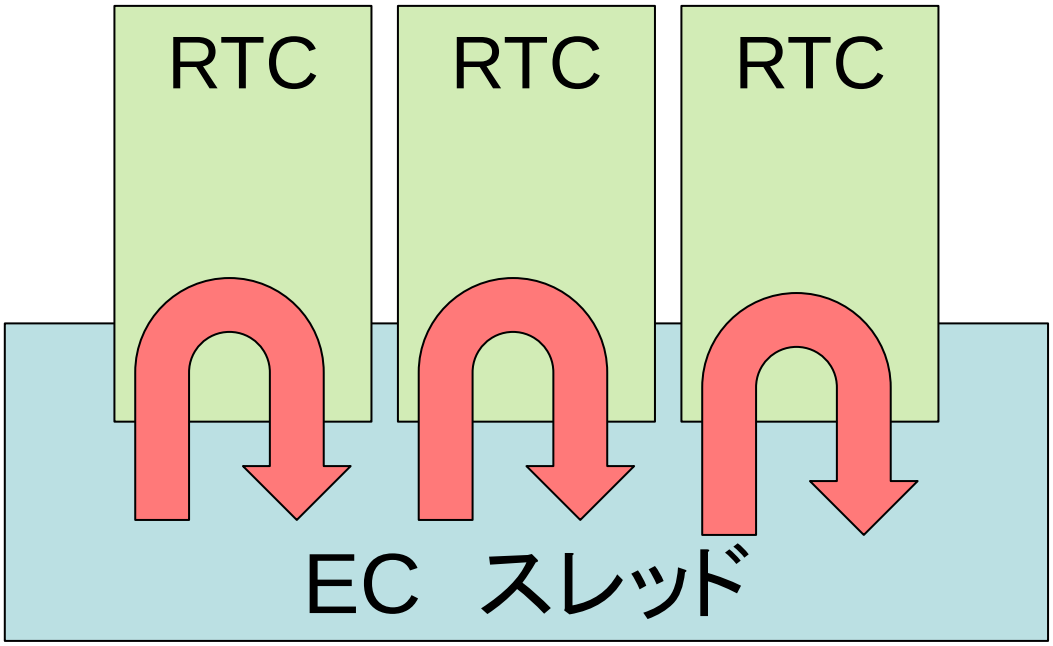
コールバック	意味
ON_CONNECT	InPortへの送信完了時
ON_DISCONNECT	InPortへの送信完了時
ON_RECEIVED	InPortへの送信完了時
ON_BUFFER_WRITE	バッファ書き込み時
ON_BUFFER_FULL	バッファフル時
OnRead	InPort の read() が呼び出された時
....	…

独自Execution context

- OpenRTMが提供しないECを使う
  - 外部の信号によって実行する（CORBAではない場合）
  - 異なるリアルタイム環境で実行する
  - 等

ECの構造

- ECはRTCの関数を呼び出すクラス
- ECのsvc()メソッドでRTCを呼び出す



実行管理

- ECのsvc()はいつ呼び出すか？
  - 自分のアプリケーションによって
- 周期的（規定）
- 外部CORBAインターフェースに信号が受信される
- データポートにデータが到着する時
  - （特に応用）

ECの作り方（1）

- RTC::ExecutionContextBaseクラスを拡張する
- svc()メソッドでRTCを呼び出すロジックを定義する
- 自由に他の支援するメソッドを書く
  - CORBAインターフェースの実装
  - FIFOに聞く

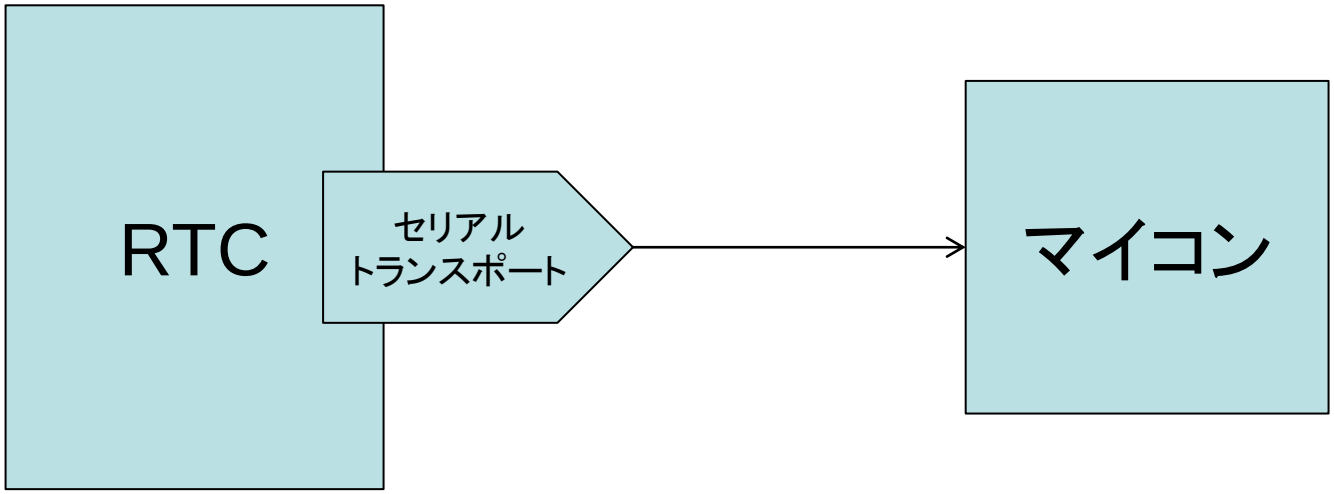
ECの作り方(2)

- Init()メソッドを定義する
  - 例:MyECInit(RTC::Manager* mgr){}
- ECはモジュールで提供する
- Init()メソッドはマネージャがモジュールをロードする時に呼び出す

```
void MyECInit(RTC::Manager* mgr) {
    RTC::Manager::instance().registerECFactory(
        “MyEC",
        RTC::ECCreate<OpenRTM::MyEC>,
        RTC::ECDelete<OpenRTM::MyEC>);
}
```

独自トランスポート

- OpenRTMが提供しないトランスポートを使う
  - 違うプロトコルを使う
  - リアルタイム通信を使う
  - データをセリアルでマイコンに送る
  - 等



トランスポートの構造

- コネクション管理
  - ポートを接続する時に
  - RTC の API に従う
- データ管理
  - コンポーネントがデータを書く・読む時
  - 実装は自由