開発プロセスとRTコンポーネントのデバッグ・テスト手法

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
ロボットプラットフォーム研究チーム
黒瀬 竜一

目次

- 開発プロセスについて
- RTC のテスト方法
- デバッグ・テストツール紹介
- まとめ

開発プロセス

いきなりですが

開発プロセス

という言葉をご存知ですか？

製品の開発

- 製品開発に関する予備知識
  - 製品のフロー
    - 企画 → 開発 → 製造 → 運用・保守
    - その他にも、研究や販売なども必要
  - 製品の品質
    - 不具合の発生は、訴訟が発生するなどの可能性
    - 理想的な環境で1回動けば良いという研究レベルでは駄目
- 製品開発はどれくらいの人数がかかるのか
  - 数十人~数百人 (組み込み開発の例)
- 製品開発の期間はどれくらいかかるのか
  - 半年~2年くらい (組み込み開発の例)

開発で求められるもの

- 製品に求められるもの
  - 魅力的な機能
  - 高品質
  - 低価格 (開発費を安く、製造費を安く)
  - 早期の発売 (開発期間を短く)
- プロジェクトリーダーに求められるもの
  - 開発全体の計画
  - 計画に対する進捗管理
  - 他部署や顧客との折衝
  - 開発環境の調達・整備
  - 開発者のフォロー
  - 開発者として開発
  - 契約処理、納品処理
  - ・・・

PLの仕事は無限にあります

製品開発は何が難しいか

さて、あなたは製品開発のために、
30名規模のプロジェクトリーダー
を任されました。
どんな課題が発生するでしょうか？

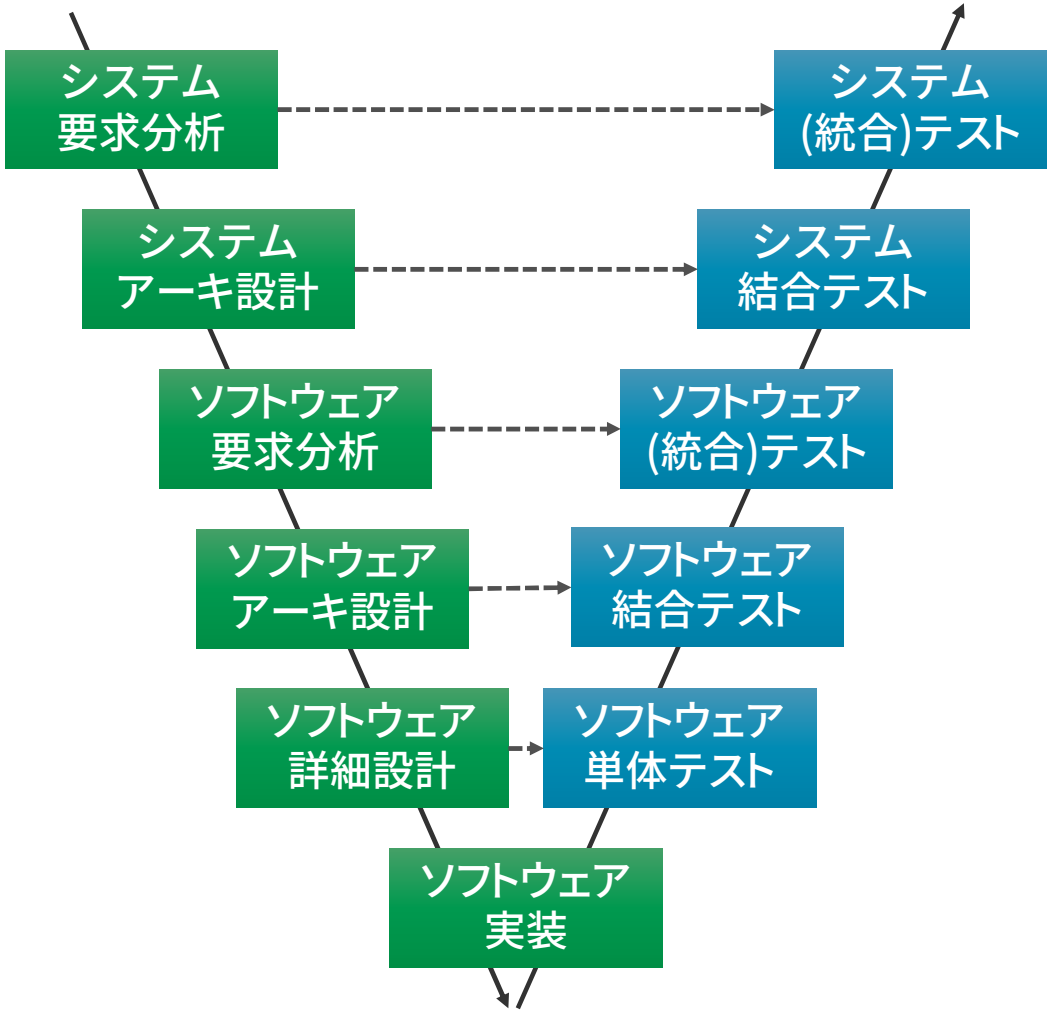
こんな事件が発生します

- 製品発売後に、変なメッセージがネットワークに出力されると言われる
  - 開発者が便利と思ってデバッグ用メッセージを垂れ流し
- 先週の確認では遅れ0日なのに、今週は10日遅れ
  - 1週間だから、遅れは最大5日じゃないの?

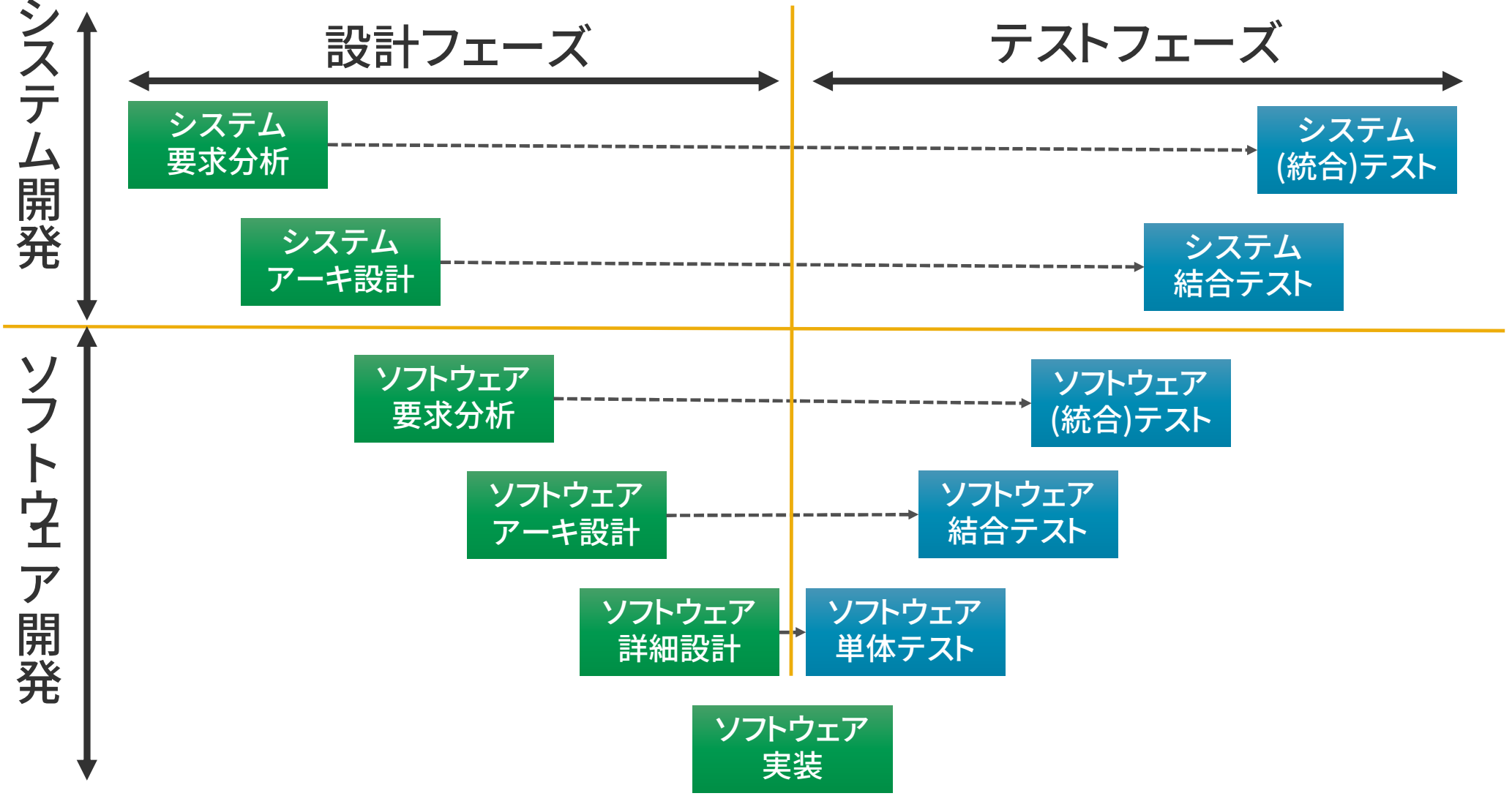
原因はプロジェクトのメンバーが、開発全体の流れと自身の役割 (成果物) を正しく認識していないこと

開発プロセスの概要

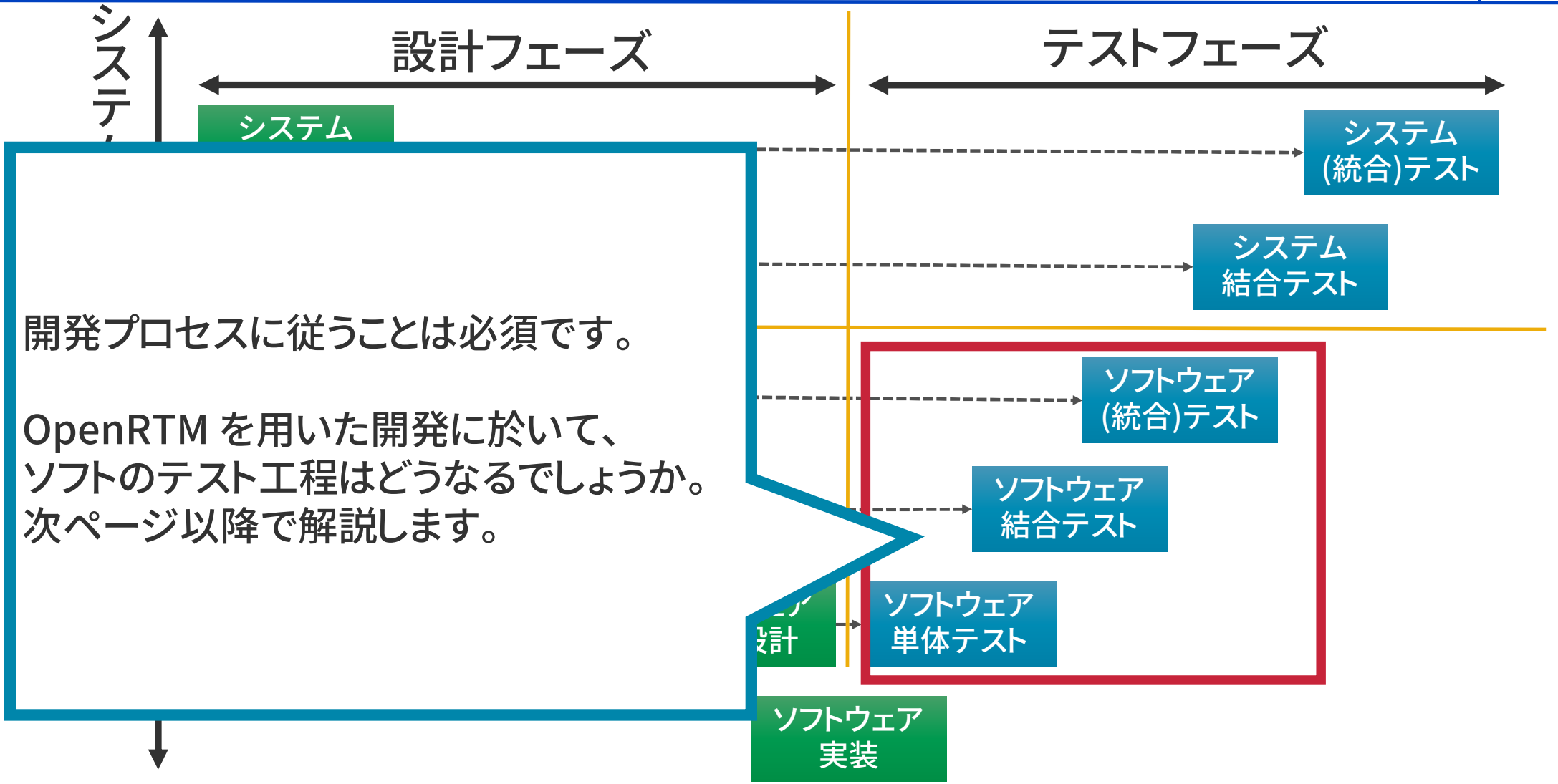
- 開発プロセスとは、
  - 開発の工程を構造化したもの
- 開発工程の基本
  - V字開発
- うれしいこと
  - 開発全体の流れを共有できる
  - 各工程の成果が明確になる
  - 開発者間で会話できる
  - 工程ごとに品質を確保できる
  - 計画がしやすい
  - 進捗が確認しやすい



V字開発の概要

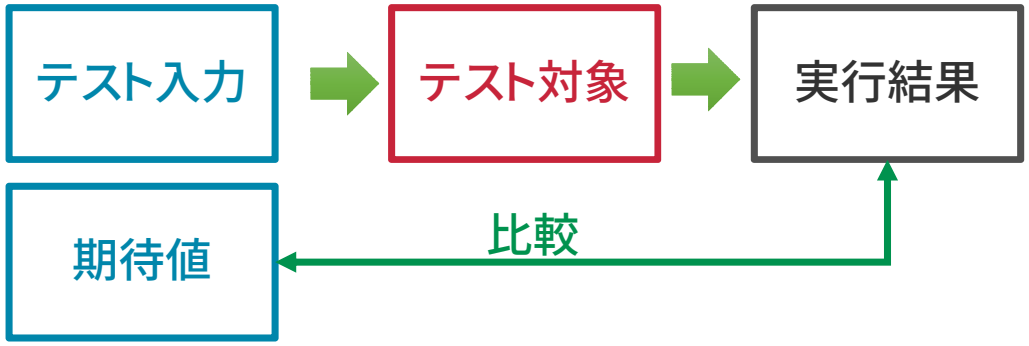

システム開発
ソフトウェア開発
設計フェーズ
テストフェーズ
システム
要求分析
システム
(統合)テスト
システム
アーキ設計
システム
結合テスト
ソフトウェア
要求分析
ソフトウェア
(統合)テスト
ソフトウェア
アーキ設計
ソフトウェア
結合テスト
ソフトウェア
詳細設計
ソフトウェア
単体テスト
ソフトウェア
実装

V字開発の概要


システム
設計フェーズ
テストフェーズ
システム
システム
(統合)テスト
システム
結合テスト
開発プロセスに従うことは必須です。
OpenRTM を用いた開発に於いて、
ソフトのテスト工程はどうなるでしょうか。
次ページ以降で解説します。
ソフトウェア
(統合)テスト
ソフトウェア
結合テスト
ソフトウェア
単体テスト
設計
ソフトウェア
実装

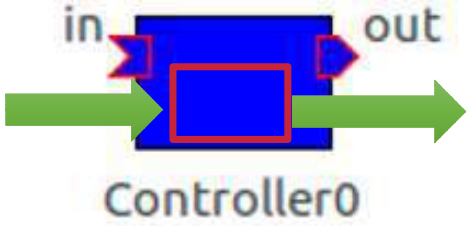
RTC のテスト方法 (1/2)

- テスト手順を守る
  - 手順
    1. テスト対象を決定
    2. テスト入力と期待値を決定
    3. テストを実行
    4. 実行結果と期待値を比較

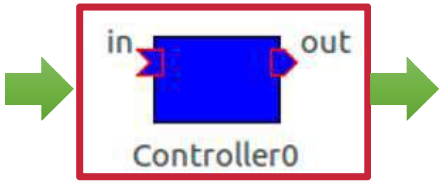


期待値を事前に用意しておくこと

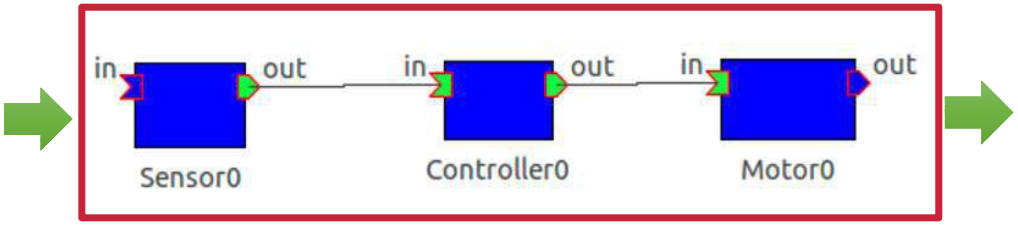
- テスト対象を順に広げる
  - 最初は中身をテスト



  - 次は RTC をテスト

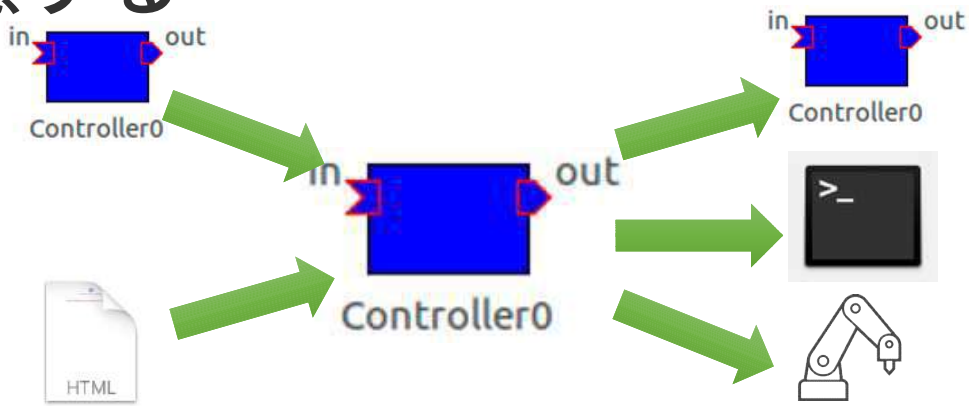


  - 最後は全体でテスト



RTC のテスト方法 (2/2)

- RTC の入出力をすべて考慮する
  - InPort/OutPort
  - コンソール
  - ファイル
  - デバイス (ロボット)
  - Ether, bluetooth,…



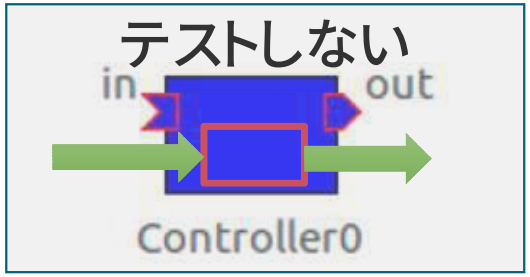
RTC の入出力は Port だけでない!!

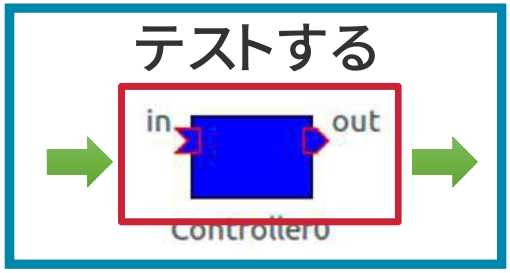
- テストを効果的に実行する
  - テストをしやすい設計をする
  - テストをツールで自動化する
  - 不具合の解析にツールの力を借りる

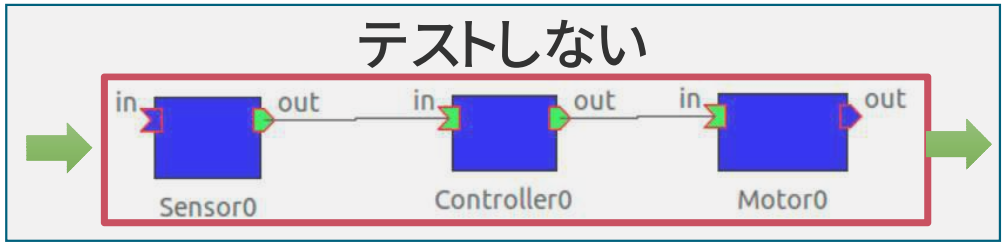
以降では、これらの手法を紹介します

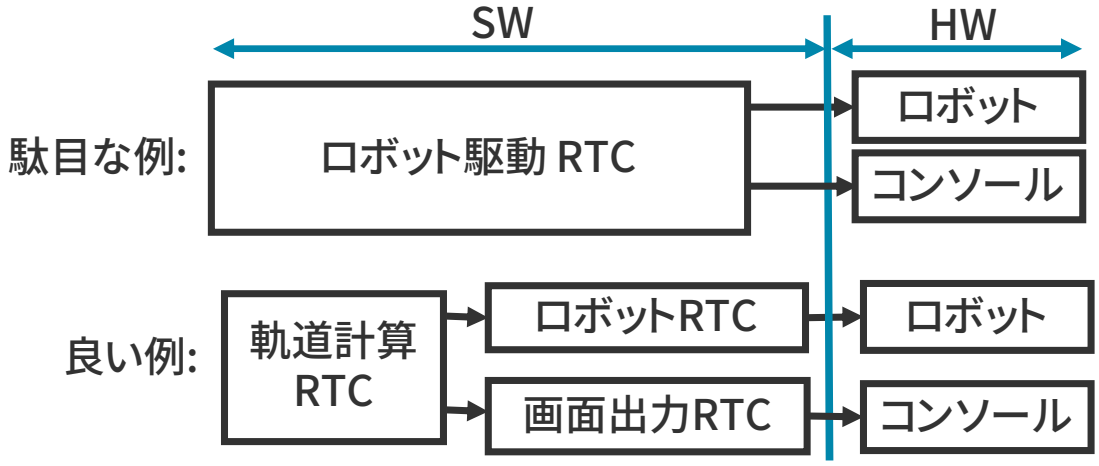
サマーキャンプ向けの戦略

- 作った RTC のみテスト
  - 有り物 RTC は動くと信じる
  - RTC 単体のテストのみ行う
- テスト項目を作りすぎない
  - RTCの仕様から考える
  - モード毎に1件くらいの粒度
- 論理部を分離する設計
  - 論理部と I/O部を別 RTCへ
  - テスト簡易化を狙い、論理部 RTC の I/O はPort のみ

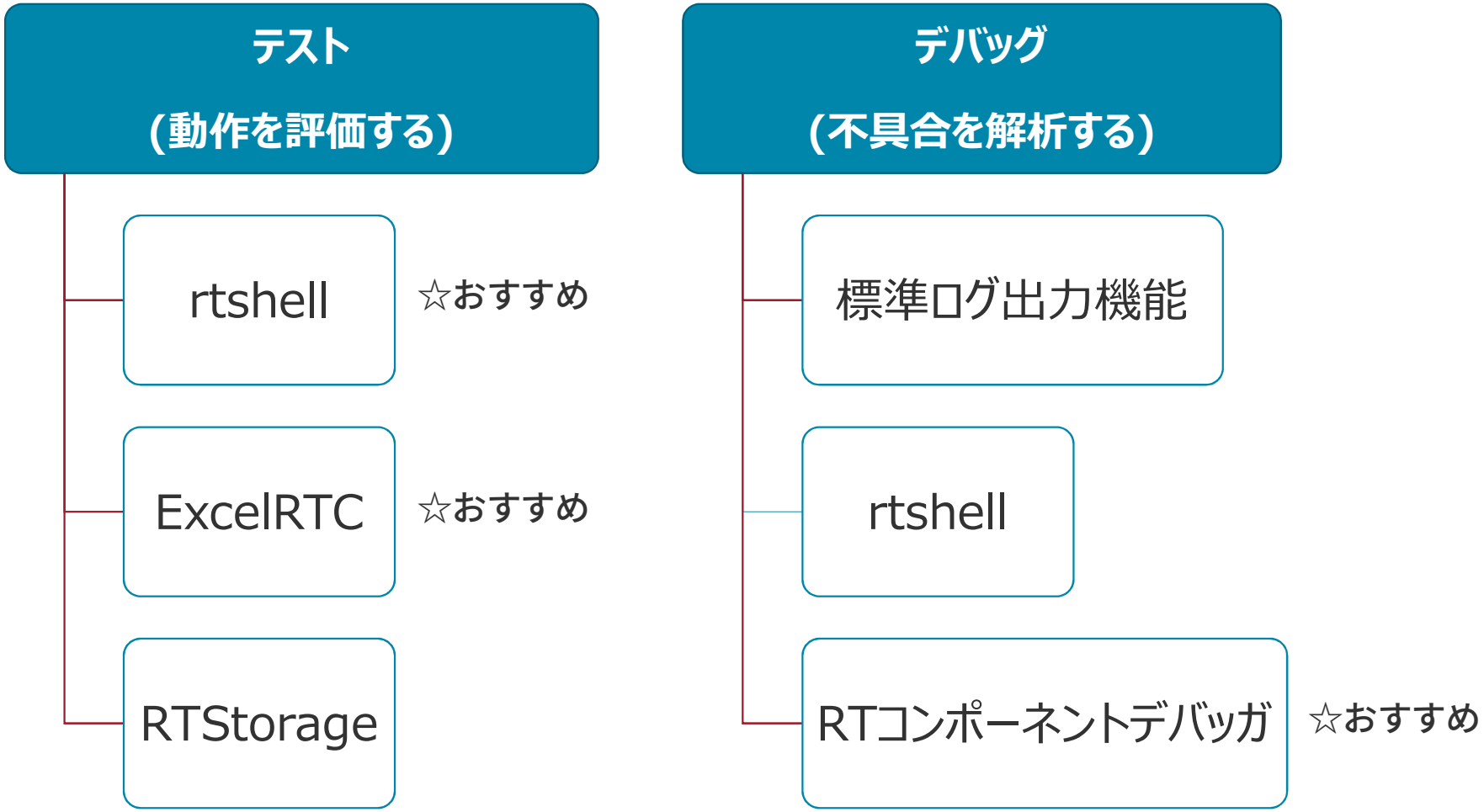








ツール紹介

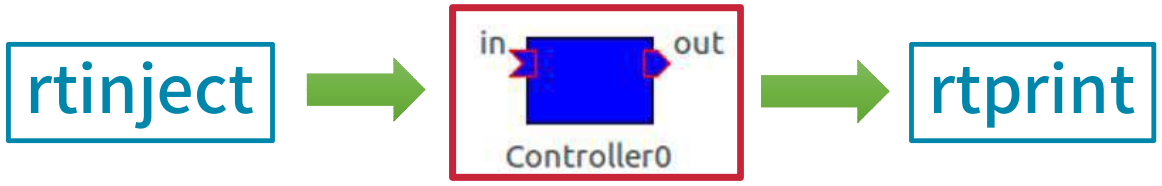

テスト
(動作を評価する)
rtshell
☆おすすめ
ExcelRTC
☆おすすめ
RTStorage
デバッグ
(不具合を解析する)
標準ログ出力機能
rtshell
RTコンポーネントデバッガ
☆おすすめ

rtshell (1/3)

- ツールの概要
  - もう知ってますよね。ジェフさんの資料を見よう!!
  - https://github.com/gbiggs/rtshell
- インストール方法
  - Windows: インストール不要
  - Linux: sudo pip install rtshell
- メリット
  - インストール不要・簡単
  - コマンド処理可能
  - RTC のテストに最適
- デメリット
  - 高速なログ出力はできない

rtshell (2/3)

- おすすめの使い方1: スクリプトによる自動化
  - テスト入力を rtijnect
  - 結果の取得をrtpirnt



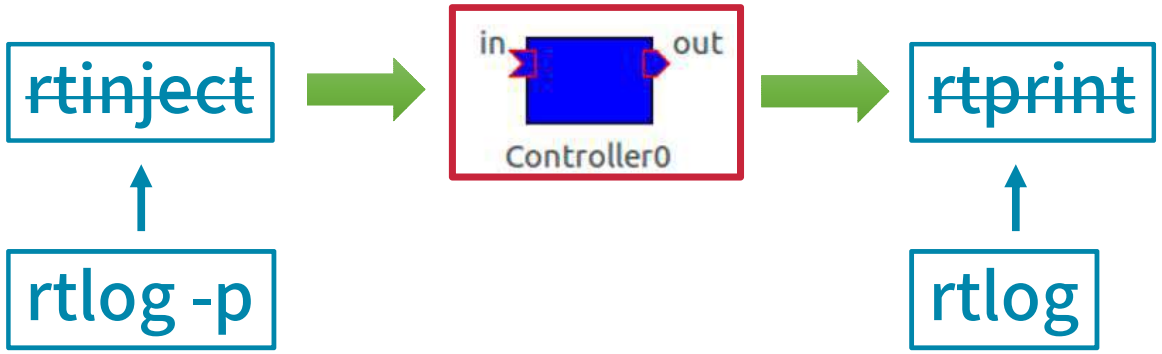
- テストをスクリプトで自動化する
  1. test.sh を作成
  2. chmod +x test.sh
  3. ./test.sh で実行
  4. test.log を確認

このサンプルは Linux 用のもの。
Windows で行う場合は、.sh
の代わりに.batを作る。

```
#! /bin/sh

# コンポーネントの有効化
rtact localhost/Controller0

# RTC にテスト入力 (10)
rtinject localhost/Controller.rtc:in –c ‘RTC.TimedLong({time}, 10)’

# RTC から10秒間の結果を取得し、test.log に保存
rtprint localhost/Controller.rtc:out -t 10 > test.log
```

rtshell (3/3)

- おすすめの使い方2: ログ保存 & ログ再生
  - コマンド rtlog の機能
    - ポートの出力をファイルに保存
    - ファイルに保存した内容をポートに入力
    - 保存した内容の表示
  - 前ページのコマンドたちを rtlog に置き換えることもできる
    - 複雑な入力 (例えばゲームパッドの操作) などの記録にも使える

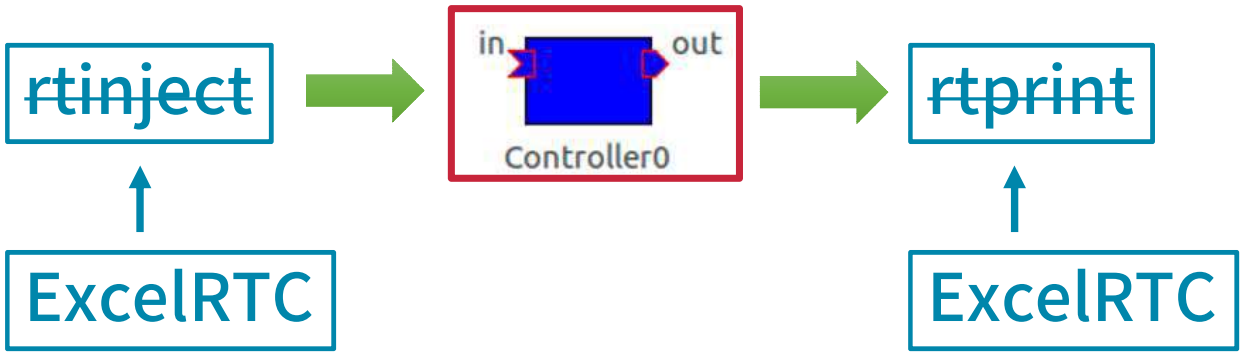


ExcelRTC (1/2)

- ツールの概要
  - もうしってますね。宮本さん資料をみよう!!
- インストール方法
  - 宮本さん配布の USB を使うだけ
- メリット
  - インストール不要・簡単
  - GUIのみで操作
  - RTC のテストに最適
- デメリット
  - Windows 限定
  - テスト完全自動化できない

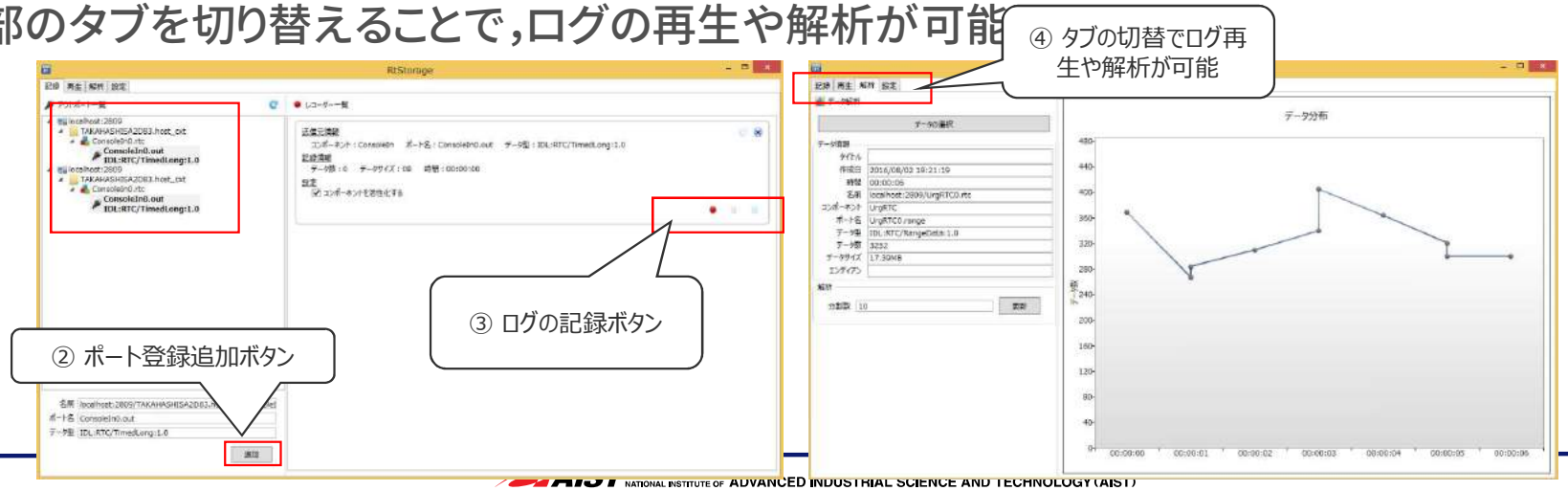
ExcelRTC (2/2)

- おすすめの使い方: テストの自動化
  - お気づきの通り、ExcelRTC を入出力ツールに使用可能
  - rtshell と違い、GUI で操作できるのでわかりやすい
  - RTCの起動などは、rtshellと違い、システムエディターでやる必要あり (ワンボタンでテスト自動化まではできない)



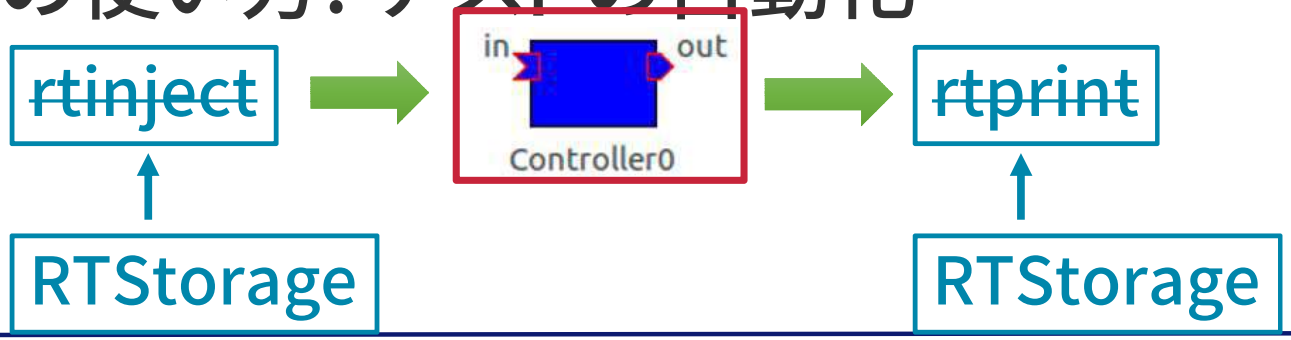
RTStorage (1/2)

- ツールの概要
  - GUIからポートの出力データ保存ができます
  - GUIから保存したデータをポートに入力できます
- 使い方
  1. RtStorage.exe を実行
  2. 実行中の RTC が表示されるので記録したいポートのペアを選択し，追加ボタンを押下
  3. レコード一覧に表示されたボタンで，ログの記録，一時停止，停止操作ができる
  4. 上部のタブを切り替えることで，ログの再生や解析が可能



RTStorage (2/2)

- インストール方法
  - https://github.com/zoetrope/RtStorage
  - インストーラはあると書かれているが実際はない
  - Visual Studio でビルドし、使用する
- メリット
  - インストール不要・簡単
  - GUIのみで操作
- デメリット
  - Windows 限定
  - テスト完全自動化できない
- おすすめの使い方: テストの自動化



OpenRTM-aist 標準ログ機能 (1/3)

ツールの概要

1.ソースコードに出力処理を埋め込む

```
RTC_INFO(("Hello, Summer Camp!"));

if(err) {
  RTC_ERR(("Not registered: %d", errno));
}
```

2.実行中にコンソールで出力を確認できる

便利機能

- ログレベルに応じた出力
- ファイルにも出力可能

ログレベル	想定される用途
FATAL	動作継続不可能な異常
ERROR	動作に影響のある異常
WARN	動作に影響の無い異常
INFO	処理上の重要な情報
NORMAL	処理上の情報
DEBUG	デバッグ用の情報
TRACE	デバッグ用の処理箇所特定情報
VERBOSE	デバッグ用の冗長な情報

OpenRTM-aist 標準ログ機能 (2/3)

- インストール方法
  - インストール不要
- 使い方
  1. ソースコードにログ出力コードを書く
     - 例 C++の場合: RTC_ERR((“エラーが発生しました”))
     - 例 Pythonの場合: self.log.RTC_ERR(“エラーが発生しました”)
  2. ログレベルを設定して実行する
     - rtc.conf を変更する

rtc.conf

```
logger.enable: YES
logger.file_name: stdout
logger.log_level: NORMAL
```

     - または、起動時にオプションをつける

```
xxRTCComp.exe -o "logger.enable:YES" -o "logger.file_name:stdout" -o "logger.log_level:NORMAL"
```

OpenRTM-aist 標準ログ機能 (3/3)

- メリット
  - RTC 内部の状態を出力可能
  - ソースコード通りの出力
- デメリット
  - 出力処理は軽くない
  - ソースコードが汚れる
- おすすめの使い方
  - ケース1: 適切な箇所に ERROR レベルの出力を出す
    - 未実装部分の処理
    - エラー処理 (普段実行されない例外処理など)
    - 設計上は動くはずがない処理 (default: など)
  - ケース2: ファイルに出しながら、リアルタイムにも見る
    - bash 上で tailf –f xxx.log を使う

RTコンポーネントデバッガ (1/2)

- ツールの概要
  - GUIでツールのポートの入出力監視、値の変更ができる
  - グラフ表示など解析しやすい
- インストール
  - 以下URLからアーカイブをダウンロード
    - http://www.sec.co.jp/robot/download_tool.html
  - 解凍したフォルダ内でスクリプトを実行
    - 【Windows】コマンドプロンプトで下記を実行

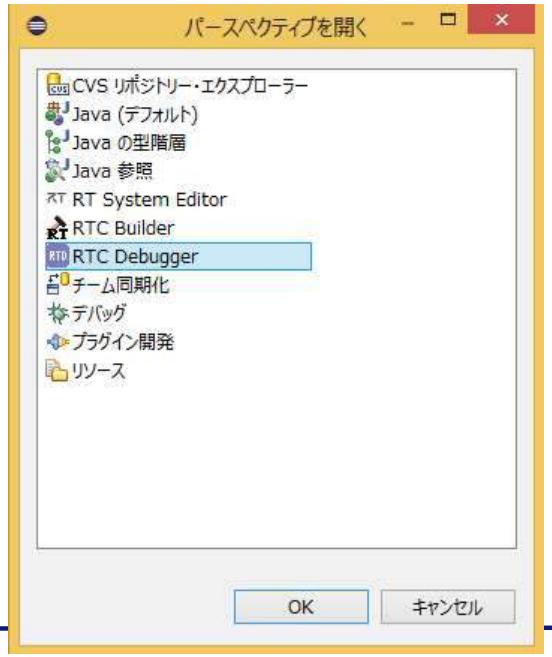
      install_1.1.0.20120522.bat
    - 【Linux】シェルで下記を実行

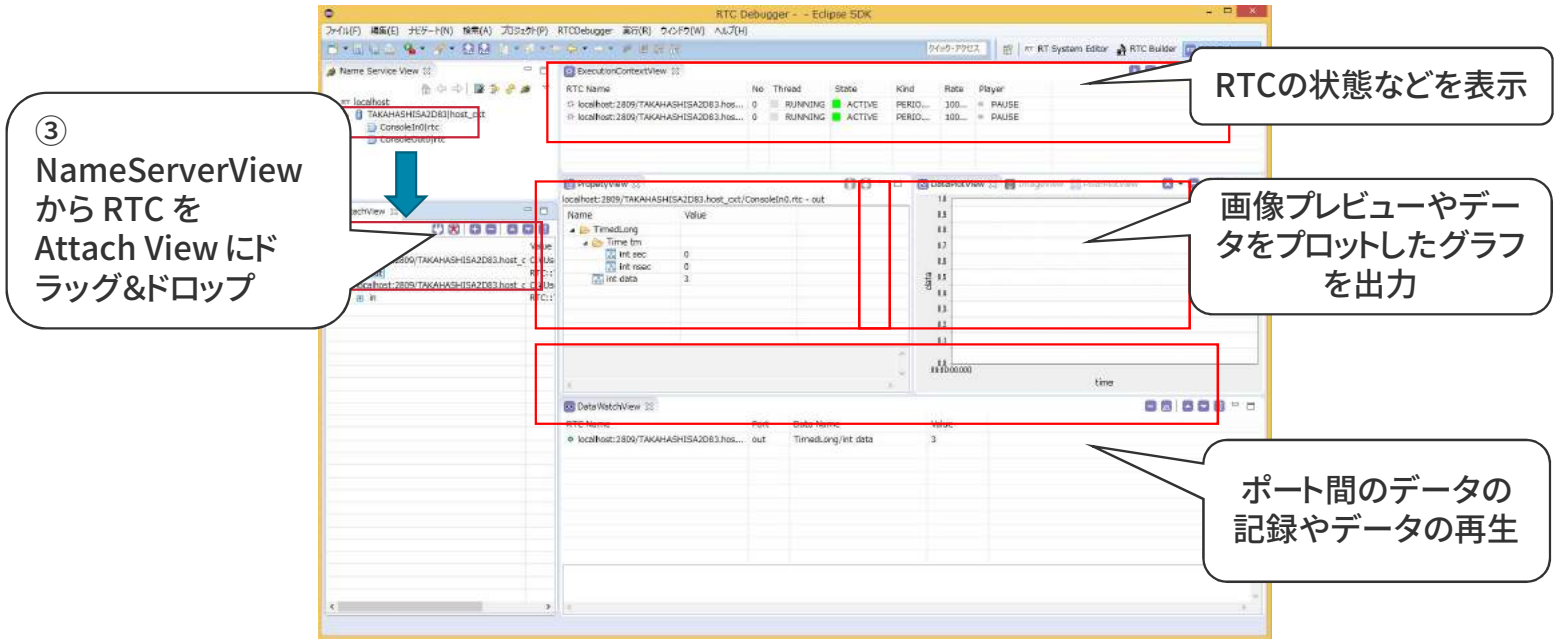
      chmod a+x install_1.1.0.20120522.sh && ./install_1.1.0.20120522.sh
  - 生成されたフォルダー一式（org.openrtp.debugger_1.x.x.yyyymmdd）をOpenRTPのインストールされたフォルダの plugins 以下にコピー
    - 【Windows】（例）C:¥Program Files¥OpenRTM-aist¥1.1.2¥utils¥OpenRTP¥plugins
    - 【Linux】（例）/usr/share/openrtm-1.1/eclipse/plugins/

RTコンポーネントデバッガ (1/2)

使い方 (※ 多彩な機能は同梱マニュアル参照)

1. OpenRTP 起動
2. メニュータブ:”ウィンドウ”→”パースペクティブを開く”→”その他”→”RTC Debugger” を選択
3. 画面左上の NameServerView から RTC を 左下のAttach View にドラッグ&ドロップする





まとめ

- 作成した RTC はテストしましょう
  - サマーキャンプは短期間なので、自作部分に限定しましょう
  - テストしやすい形に設計しましょう
- RTC のテストは手順に従いましょう
  - サマーキャンプは短期間なので、テスト項目を絞りましょう
- RTC のテストはツールを十分活用して行いましょう
  - ソースコード修正するたびにテストできるように、テストは(半)自動化しましょう
- デバッグもツールを十分活用して行いましょう