

# RTShell 入門

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

黒瀬 竜一

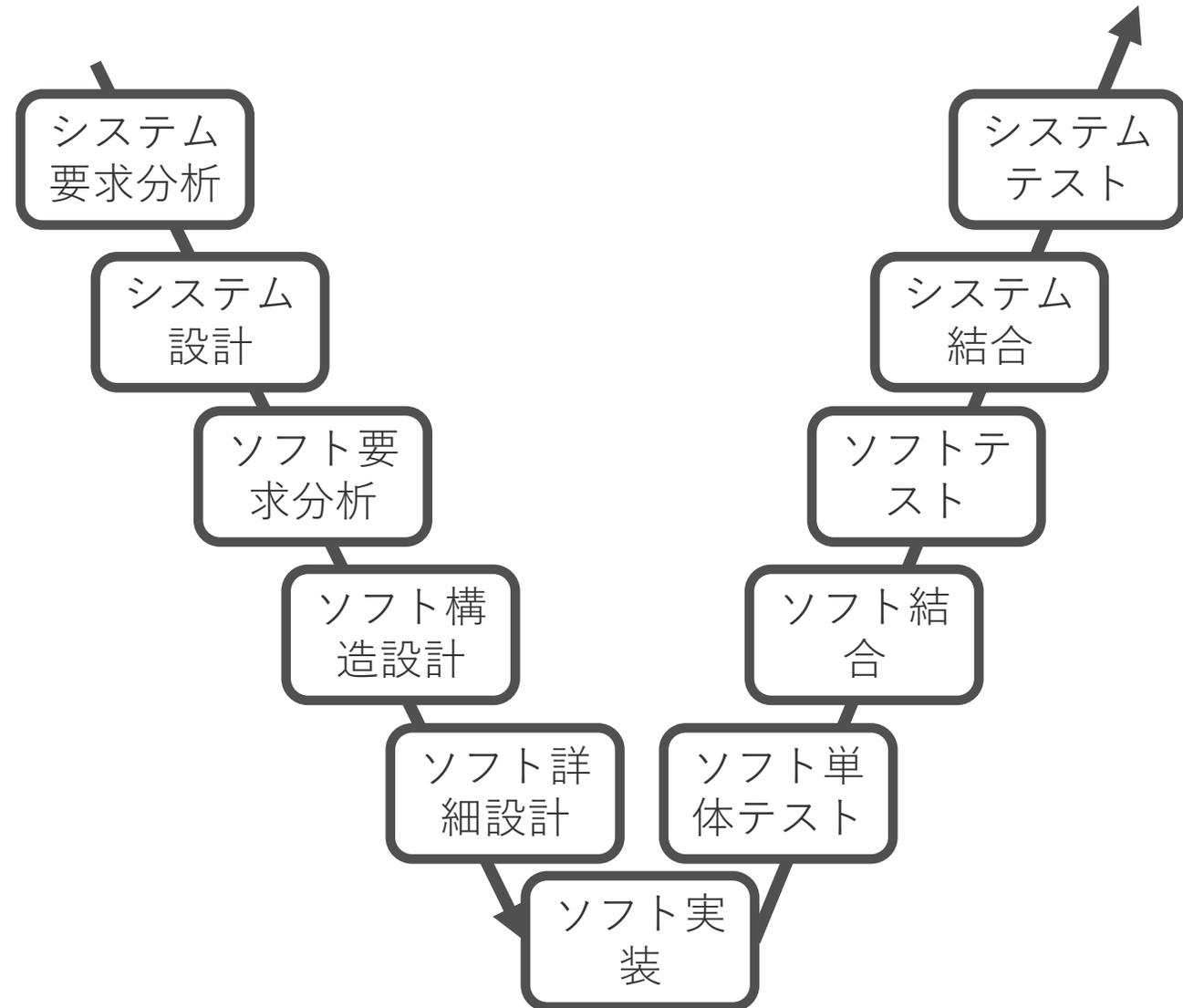
## □ 駄目なソフト開発

- 開発者を増員しても終わりが見えない
- バグを修正すると新しいバグが増える
- リリース後に許容できないバグが発覚する

...

## □ ソフト開発で重要なこと

- 目標品質の達成
- 十分な作業の効率化

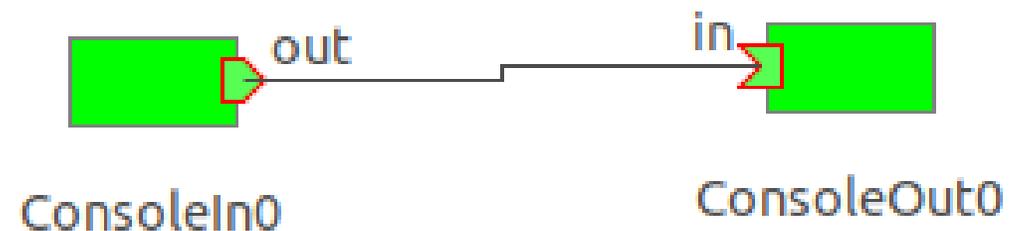
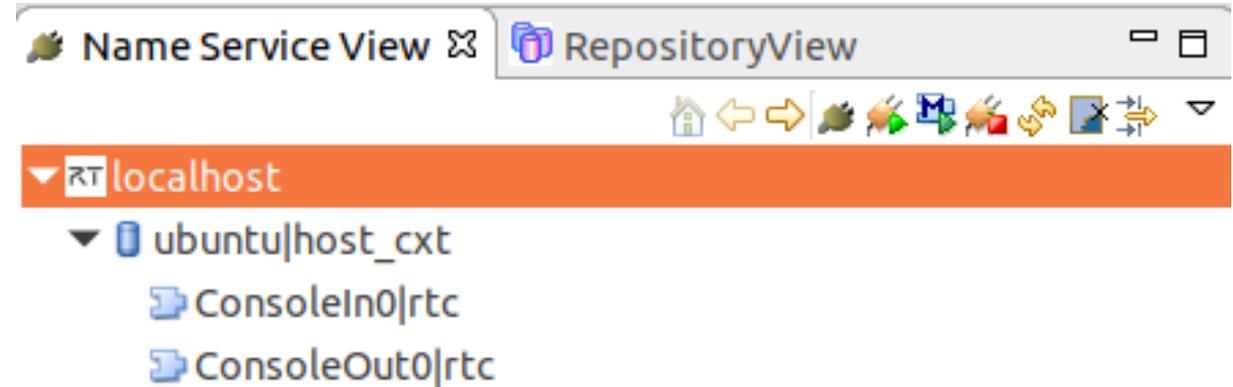


- RT System Editor (RTSE) 同様に RTC を操作できる
  - RTC のアクティベート、ポート接続、…
- システム管理にも使える
  - 複数の RTC に対する動作をサポート
- テスト・デバッグにも使える
  - テストデータを RTC に送信、RTC の出力保存、…
- コマンドラインで動作する
  - GUI よりも自動化が容易！

## まずは OpenRTM の基本（講習会の内容）を復習

### □ RTC を動作する

1. ConsoleOutComp の起動
2. ConsoleInComp の起動
3. RT System Editor (OpenRTP) の起動
4. ネームサーバーの起動
5. RTC のポート接続
6. RTC のアクティベート



## rtls：コンポーネントを確認するコマンド

### □ コマンドの実行

#### 1. コマンドライン環境の起動

##### □ [Windows]

コマンドプロンプト  
or PowerShell or WSL

##### □ [Linux]

任意のターミナル

#### 2. 以下を実行

□ `rtls localhost`

□ `rtls localhost/xxx.cxt/`

□ `rtls -R localhost`

 kuro@DESKTOP-3FT7447: ~

```
kuro@DESKTOP-3FT7447:~$ rtls
kuro@DESKTOP-3FT7447:~$ rtls localhost
DESKTOP-3FT7447.host_cxt/
kuro@DESKTOP-3FT7447:~$ rtls localhost/DESKTOP-3FT7447.host_cxt/
ConsoleIn0.rtc ConsoleOut0.rtc
kuro@DESKTOP-3FT7447:~$
```

注意：コマンドが存在しない場合は `rtshell` のインストールミス

## Windows

1. 空ファイル rtc.bat を作る
  - 空テキストのリネーム
2. 中身を書き保存する

```
dir
```

3. rtc.bat を実行する
  - コマンドラインに rtc.bat をドラッグアンドドロップも可
  - ダブルクリック実行は一瞬で終了するので pause を追加

## Linux

1. rtc.sh という名前でファイルを作る
2. 中身を書き保存

```
#!/bin/sh  
ls
```

3. 必要に応じて権限付与する
  - chmod +x rtc.sh
4. rtc.sh を実行する
  - ./rtc.sh
  - sh -x rtc.sh

ここで RTC と OpenRTP は終了してください。

その後に RTC を再度起動してください

- ConsoleInComp
- ConsoleOutComp

□ スクリプトにより自動ポート接続・アクティベート

□ 赤字は自分の環境に置き換えること

□ 実行後に RTSE を起動して実習1と同じになったか確認

## Windows

## Linux

```
rem コンポーネント確認
rtls -R localhost/
rtcat -l localhost/%H%.host_cxt/ConsoleIn0.rtc
rtcat -l localhost/%H%.host_cxt/ConsoleOut0.rtc

rem ポート接続
rtcon localhost/%H%.host_cxt/ConsoleIn0.rtc:out ^
    localhost/%H%.host_cxt/ConsoleOut0.rtc:in

rem 動作開始
rtact localhost/%H%.host_cxt/ConsoleIn0.rtc ^
    localhost/%H%.host_cxt/ConsoleOut0.rtc
```

```
#!/bin/sh -x

# コンポーネント確認
rtls -R localhost/
rtcat -l localhost/${H}.host_cxt/ConsoleIn0.rtc
rtcat -l localhost/${H}.host_cxt/ConsoleOut0.rtc

# ポート接続
rtcon localhost/${H}.host_cxt/ConsoleIn0.rtc:out¥
    localhost/${H}.host_cxt/ConsoleOut0.rtc:in

# 動作開始
rtact localhost/${H}.host_cxt/ConsoleIn0.rtc¥
    localhost/${H}.host_cxt/ConsoleOut0.rtc
```

- システム環境の保存
  - 演習4の状態にしてください
    - RTSE でポート接続
  - rtcryo
- システム環境の復元
  - rtresurrect
- システムの実行・停止
  - rtstart
  - rtstop
  - rtteardown

RTSE も環境保存と復元ができます。活用しましょう。

```
# システムを保存
rtcryo localhost -o env.rtsys
echo "wait to exit"
read (Windows の場合は pause)
```

```
# システムを復元
rtresurrect env.rtsys
```

```
# システム を 起動
rtstart env.rtsys
```

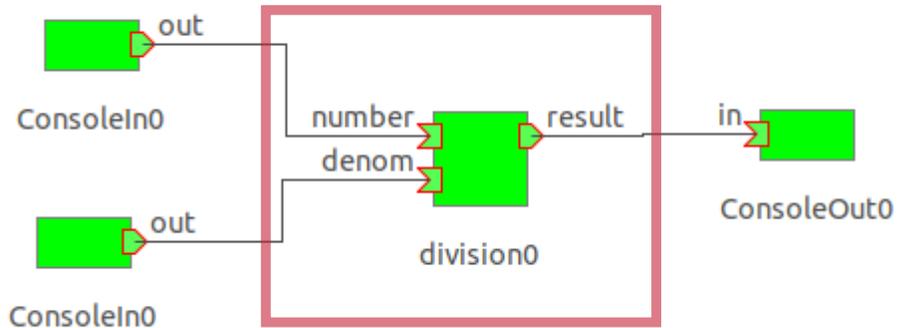
```
echo "Running ..."
read (Windows の場合は pause)
```

```
# システム を 停止
rtstop env.rtsys
```

```
# システムの接続を削除
rtteardown env.rtsys
```

## □ RTC Divisionをテストする

- まずは手動で実行
- 自動実行のために Division 以外を終了



## □ スクリプトを作成・実行

- Division 以外を終了し、実行
- 早くできた人は、RTCを修正してみましょう

# 位置の移動

```
WORKDIR=/localhost/`hostname`.host_cxt/  
(Windows の場合は set WORKDIR=/localhost/...)
```

# テスト結果の確認

```
rtprint ${WORKDIR}division0.rtc:result -n 100
```

# テストデータの送信

```
rtinject ${WORKDIR}division0.rtc:number -c  
'RTC.TimedLong({time}, 10)'  
rtinject ${WORKDIR}division0.rtc:denom -c  
'RTC.TimedLong({time}, 2)'
```

```
rtinject ${WORKDIR}division0.rtc:number -c  
'RTC.TimedLong({time}, 20)'  
rtinject ${WORKDIR}division0.rtc:denom -c  
'RTC.TimedLong({time}, 0)'
```

## □ RTC 接続後のシステム動作の保存と再現

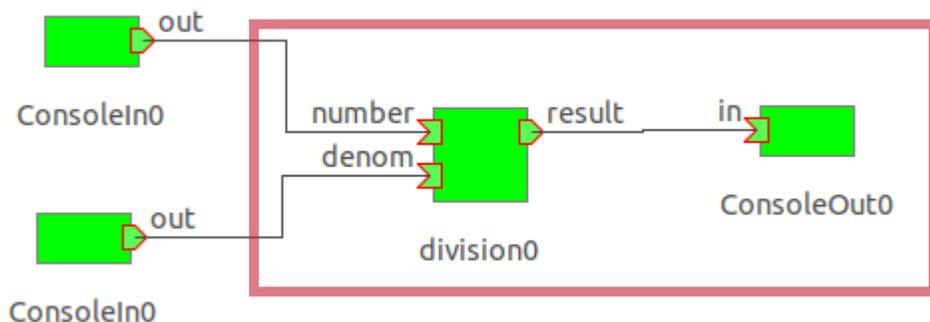
- rtlog 保存

- rtlog -p で再生

## □ システム動作のテスト

- センサーの情報再現

- 人間の操作ログ再現



```
# 位置の移動
```

```
WORKDIR=/localhost/`hostname`.host_cxt/  
(Windows の場合は set WORKDIR=…)
```

```
# 10秒間の入力データを保存
```

```
rtlog -f test.log -t 10 ${WORKDIR}division0.rtc:number  
${WORKDIR}division0.rtc:denom
```

```
read
```

```
# (Windows の場合は pause)
```

```
# ここでConsoleIn から入力いれる
```

```
# rtlog を終了する
```

```
# 出力データの再生
```

```
rtlog -p -f test.log ${WORKDIR}division0.rtc:number  
${WORKDIR}division0.rtc:denom
```

- rtact
- rtcat
- ~~□ rtcwd~~
- rtcheck
- rtcomp
- rtcon
- rtconf
- rtcprof
- rtcryo
- rtdeact
- rtdel
- rtdis
- rtdoc
- rtexit
- rtfind
- rtfsm
- rtinject
- rtlog
- rtls
- rtmgr
- rtprint
- rtpwd
- rtreset
- rtresurrect
- restart
- rtstodot
- rtstop
- rtteardown
- rtlog
- rtwatch

- コマンドのヘルプを見よう
  - 例： `rtls --help`
- rtshell のチュートリアルを見よう
  - <http://openrtm.org/openrtm/ja/node/5014/>
  - <http://openrtm.org/openrtm/ja/node/5015/>
  - <http://www.youtube.com/playlist?list=PLE06F481CC7089B9A>
- Github で質問しよう
  - <https://github.com/OpenRTM/rtshell>
  - Issue 発行で質問や要望を受け付けています
  - もちろんプルリクエストも受け付けています！

- RTShell による **作業の自動化** は重要です
  - ヒューマンエラーを排除しましょう
  - 繰り返し作業は PC にやらせましょう
  
- サマーキャンプでも活用してください
  - まずはスクリプトを一つ作ってみんなで共有しましょう
  - コンポーネント追加ごとにスクリプトを拡張しましょう
  - テストごとに別のスクリプトにするなどの工夫はお任せ
  
- 今回のような **自動化** はソフト開発で常に重要です
  - 今回に限らず、常に自動化する癖をつけましょう

- 下記に本資料のスク립トサンプルがあります。
  - [https://github.com/r-kurose/rtm\\_summer\\_camp\\_2019](https://github.com/r-kurose/rtm_summer_camp_2019)